BasicsNotes

https://samples.openweathermap.org/data/2.5/forecast/daily?
lat=35&lon=139&cnt=10&appid=b1b15e88fa797225412429c1c50c122a1
"homepage": "https://appbaseio-apps.github.io/airbeds",
npm cache clean --force
npm cache clean --force
https://github.com/tifa2UP/Netly
office email id=mukesh.singh@alphavarna.in
www.aPlayersServicesDrive.com
As@8802442
url-https://aplayersservicesdrive.wordpress.com/
github:https://github.com/manavgupta08/codeshare
cross browser testing
https://app.crossbrowsertesting.com
https://github.com/atlassian/react-beautiful-dnd
https://burnsout.pipedrive.com/deal/1#
@babel error=>
npm install --save-exact @babel/runtime@7.0.0-beta.55
create new card in dnd
https://www.smoothterminal.com/articles/adding-drag-and-drop-to-react
https://github.com/kutlugsahin/smooth-dnd-demo
npm install react-popper@0.9.1-alpha.1
[contact-form-7 id="222" title="Contact form 1"]
Password- PQMMH%XkxW#lSlH1O1W
Domain-aplayersservices.com
webhost-https://aplayers.000webhostapp.com/#content
https://medium.com/@sunnykay/testing-react-applications-a-tutorial-setup-f87591b97295

ReactQuestions

Q1. How React works? How Virtual-DOM works in React?

Ans: React creates a virtual DOM. When state changes in a component it firstly runs a “diffing” algorithm, which identifies what has changed in the virtual DOM. The second step is reconciliation, where it updates the DOM with the results of diff. The HTML DOM is always tree-structured — which is allowed by the structure of HTML document. Since we are more and more pushed towards dynamic web apps, we need to modify the DOM tree incessantly and a lot. And this is a real performance and development pain. The Virtual DOM is an abstraction of the HTML DOM. It is lightweight and detached from the browser-specific implementation details. It is not invented by React but it uses it and provides it for free. ReactElements lives in the virtual DOM. They make the basic nodes here. Once we defined the elements, ReactElements can be render into the "real" DOM. Whenever a ReactComponent is changing the state, diff algorithm in React runs and identifies what has changed. And then it updates the DOM with the results of diff. The point is - it’s done faster than it would be in the regular DOM.

Q2. What is JSX?

JSX is a syntax extension to JavaScript and comes with the full power of JavaScript. JSX produces React “elements”. You can embed any JavaScript expression in JSX by wrapping it in curly braces. After compilation, JSX expressions become regular JavaScript objects. This means that you can use JSX inside of if statements and for loops, assign it to variables, accept it as arguments, and return it from functions. Eventhough React does not require JSX, it is the recommended way of describing our UI in React app.

```
const element = (
'<'h1>
Hello, world!
'<'/h1>
);
```

Q3. What is React.createClass?

React.createClass allows us to generate component "classes." But with ES6, React allows us to implement component classes that use ES6 JavaScript classes. The end result is the same, we have a component class. But the style is different. And one is using a "custom" JavaScript

class system (createClass) while the other is using a "native" JavaScript class system. When using React's createClass() method.
Using an ES6 class to write the same component is a little different. Instead of using a method from the react library, we extend an ES6 class that the library defines, Component.

Q4. What is ReactDOM and what is the difference between ReactDOM and React?

As the name implies, ReactDOM is the glue between React and the DOM. Often, we will only use it for one single thing: mounting with ReactDOM. Another useful feature of ReactDOM is ReactDOM.findDOMNode() which we can use to gain direct access to a DOM element. For everything else, there's React. We use React to define and create our elements, for lifecycle hooks, etc.

Q5. What are the differences between a class component and functional component?

Class components allows us to use additional features such as local state and lifecycle hooks. Also, to enable our component to have direct access to our store and thus holds state. When our component just receives props and renders them to the page, this is a 'stateless component', for which a pure function can be used. These are also called dumb components or presentational components. ex. Booklist component is functional components and are stateless.

```
import React from "react";
const Booklist = books => (
{books.map(({ title, author }) =>
{title}-{author}
)}
'</ul>
)
```

Q6. What is the difference between state and props?

The state is a data structure that starts with a default value when a Component mounts. It may be mutated across time, mostly as a result of user events. Props are a Component's configuration. Props are how components talk to each other. They are received from above component and immutable as far as the Component receiving them is concerned. A Component cannot change its props, but it is responsible for putting together the props of its child Components. Props do not have to just be data — callback functions may be passed in as props. There is also the case that we can have default props so that props are set even if a parent component doesn't pass props down.
Props and State do similar things but are used in different ways. The majority of our components will probably be stateless. Props are used to pass data from parent to child or by the component itself. They are immutable and thus will not be changed. State is used for mutable data, or data that will change. This is particularly useful for user input.

Q7. What are controlled components?

In HTML, form elements such as "input", "textarea", and "select" typically maintain their own state and update it based on user input. When a user submits a form the values from the aforementioned elements are sent with the form. With React it works differently. The component containing the form will keep track of the value of the input in it's state and will re-render the component each time the callback function e.g. onChange is fired as the state will be updated. A form element whose value is controlled by React in this way is called a "controlled component". With a controlled component, every state mutation will have an associated handler function. This makes it straightforward to modify or validate user input.

Q8. What is a higher order component?

A higher-order component (HOC) is an advanced technique in React for reusing component logic. HOCs are not part of the React API. They are a pattern that emerges from React's compositional nature. A higher-order component is a function that takes a component and returns a new component. HOC's allow you to reuse code, logic and bootstrap abstraction. HOCs are common in third-party React libraries. The most common is probably Redux's connect function. Beyond simply sharing utility libraries and simple composition, HOCs are the best way to share behavior between React Components. If you find yourself writing a lot of code in different places that does the same thing, you may be able to refactor that code into a reusable HOC.

Q9. What is create-react-app?

We don't need to install or configure tools like Webpack or Babel. They are preconfigured and hidden so that we can focus on the code. We can install easily just like any other node modules. Then it is just one command to start the React project.
It includes everything we need to build a React app:

- React, JSX, ES6, and Flow syntax support.
- Language extras beyond ES6 like the object spread operator.
- Autoprefixed CSS, so you don't need -webkit- or other prefixes.
- A fast interactive unit test runner with built-in support for coverage reporting.
- A live development server that warns about common mistakes.
- A build script to bundle JS, CSS, and images for production, with hashes and sourcemaps.

Q3. What is PureComponent? When to use PureComponent over Component?

PureComponent is exactly the same as Component except that it handles the shouldComponentUpdate method for us. When props or state changes, PureComponent will do a shallow comparison on both props and state. Component on the other hand won't compare current props and state to next out of the box. Thus, the component will re-render by default whenever shouldComponentUpdate is called. When comparing previous props and state to next, a shallow comparison will check that primitives have the same value (eg, 1 equals 1 or that true equals true) and that the references are the same between more complex javascript values like objects and arrays. It is good to prefer PureComponent over Component whenever we never mutate our objects.

Q10. How Virtual-DOM is more efficient than Dirty checking?

In React, each of our components have a state. This state is like an observable. Essentially, React knows when to re-render the scene because it is able to observe when this data changes. Dirty checking is slower than observables because we must poll the data at a regular interval and check all of the values in the data structure recursively. By comparison, setting a value on the state will signal to a listener that some state has changed, so React can simply listen for change events on the state and queue up re-rendering. The virtual DOM is used for efficient re-rendering of the DOM. This isn't really related to dirty checking your data. We could re-render using a virtual DOM with or without dirty checking. In fact, the diff algorithm is a dirty checker itself. We aim to re-render the virtual tree only when the state changes. So using an observable to check if the state has changed is an efficient way to prevent unnecessary re-renders, which would cause lots of unnecessary tree diffs. If nothing has changed, we do nothing.

11. Is setState() is async? Why is setState() in React Async instead of Sync?

setState() actions are asynchronous and are batched for performance gains. This is explained in documentation as below. setState() does not immediately mutate this.state but creates a pending state transition. Accessing this.state after calling this method can potentially return the existing value. There is no guarantee of synchronous operation of calls to setState and calls may be batched for performance gains. This is because setState alters the state and causes rerendering. This can be an expensive operation and making it synchronous might leave the browser unresponsive. Thus the setState calls are asynchronous as well as batched for better UI experience and performance.

Q12. Is setState() is async? Why is setState() in React Async instead of Sync?

setState() actions are asynchronous and are batched for performance gains. This is explained in documentation as below. setState() does not immediately mutate this.state but creates a pending state transition. Accessing this.state after calling this method can potentially return the existing value. There is no guarantee of synchronous operation of calls to setState and calls may be batched for performance gains. This is because setState alters the state and causes rerendering. This can be an expensive operation and making it synchronous might leave the browser unresponsive. Thus the setState calls are asynchronous as well as batched for better UI experience and performance.

Q13. What is render() in React? And explain its purpose?

Each React component must have a render() mandatorily. It returns a single React element which is the representation of the native DOM component. If more than one HTML element needs to be rendered, then they must be grouped together inside one enclosing tag such as "form", "group" etc. This function must be kept pure i.e., it must return the same result each time it is invoked.

Q14. What are controlled and uncontrolled components in React?

This relates to stateful DOM components (form elements) and the difference: A Controlled Component is one that takes its current value through props and notifies changes through callbacks like onChange. A parent component "controls" it by handling the callback and managing its own state and passing the new values as props to the controlled component. You could also call this a "dumb component". A Uncontrolled Component is one that stores its own state internally, and you query the DOM using a ref to find its current value when you need it. This is a bit more like traditional HTML. In most (or all) cases we should use controlled components.

Q15. What is the second argument that can optionally be passed to setState and what is its purpose?

A callback function which will be invoked when setState has finished and the component is re-rendered. Since the setState is asynchronous, which is why it takes in a second callback function. With this function, we can do what we want immediately after state has been updated.

Q16. What is the difference between React Native and React?

React is a JavaScript library, supporting both front end web and being run on the server, for building user interfaces and web applications. On the other hand, React Native is a mobile framework that compiles to native app components, allowing us to build native mobile applications (iOS, Android, and Windows) in JavaScript that allows us to use ReactJS to build our components, and implements ReactJS under the hood. With React Native it is possible to mimic the behavior of the native app in JavaScript and at the end, we will get platform specific code as the output. We may even mix the native code with the JavaScript if we need to optimize our application further.

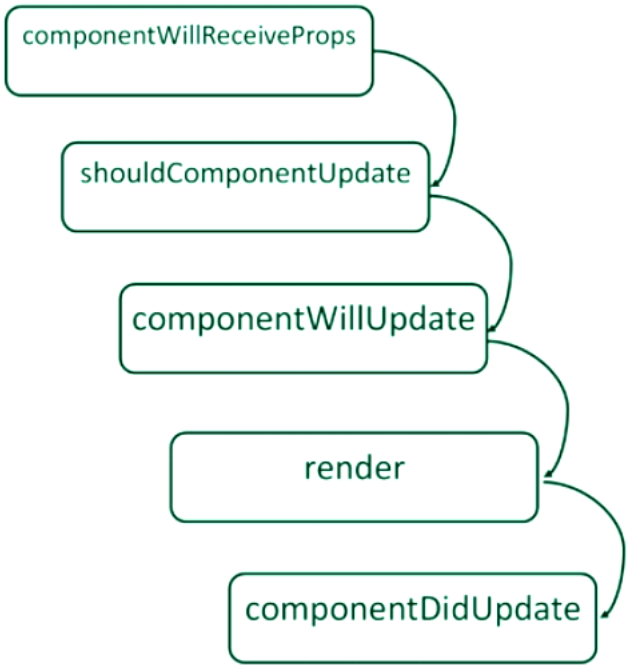
Q17. What is React.cloneElement? And the difference with this.props.children?

React.cloneElement clone and return a new React element using using the passed element as the starting point. The resulting element will have the original element's props with the new props merged in shallowly. New children will replace existing

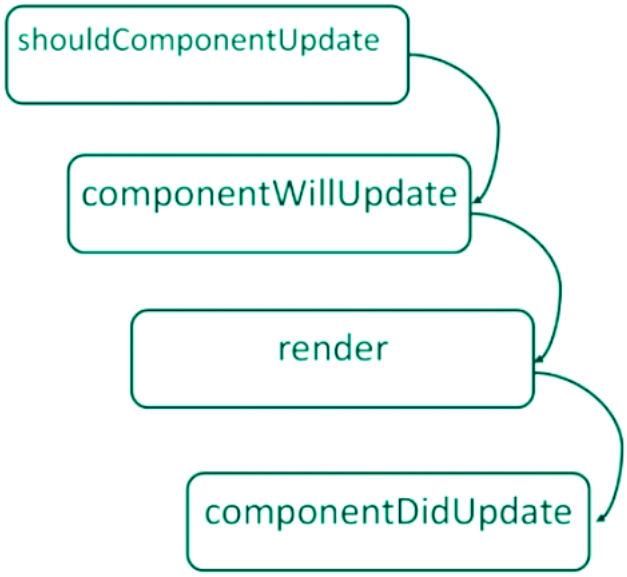
children. key and ref from the original element will be preserved. React.cloneElement only works if our child is a single React element. For almost everything "'this.props.children'" is the better solution. Cloning is useful in some more advanced scenarios, where a parent send in an element and the child component needs to change some props on that element or add things like ref for accessing the actual DOM element.

Q18.React component's lifecycle

Looks like we need more control over the stages that a component goes through. The process where all these stages are involved is called the component's lifecycle and every React component goes through it. React provides several methods that notify us when certain stage of this process occurs. These methods are called the component's lifecycle methods and they are invoked in a predictable order.
Basically all the React component's lifecyle methods can be split in four phases: initialization, mounting, updating and unmounting. Let's take a closer look at each one of them.
InitializationThe initialization phase is where we define defaults and initial values for this.props and this.state by implementing getDefaultProps() and getInitialState() respectively. The getDefaultProps() method is called once and cached — shared across instances — when the class is created, before any instance of the component are created, hence we can't rely on this.props here. This method returns an object which properties values will be set on this.props if that prop is not specified by the parent component.
The **getInitialState()** method is also invoked once, right before the mounting phase. The return value of this method will be used as initial value of this.state and should be an object.**Mounting**Mounting is the process that occurs when a component is being inserted into the DOM. This phase has two methods that we can hook up with: componentWillMount() and componentDidMount().
The componentWillMount() method is the first called in this phase. It's invoked once and immediately before the initial rendering occurs, hence before React inserts the component into the DOM. It's very important to note that calling this.setState() within this method will not trigger a re-render. If we add the next code to the Counter component we will see that the method is called after getInitialState() and before render().
The componentDidMount() is the second invoked in this phase, just once and immediately after React inserts the component into the DOM. Now the updated DOM is available for access, which means that this method is the best place for initializing other Javascript libraries that need access to the DOM and for data fetching operations.
UpdatingThere are also methods that will allow us to execute code relative to when a component's state or properties get updated. These methods are part of the updating phase and are called in the following order:
1.When receiving new props from the parent:



2.When the state changes via this.setState():



The first one is componentWillReceiveProps(), invoked when a component is receiving new props. We can use this method as an opportunity to react to a prop transition before the render() method is called. Calling this.setState() within this function will not trigger an additional re-render, and we can access the old props via this.props.
The shouldComponentUpdate() method allows us to decide whether the next component's state should trigger a re-render or not. This method returns a boolean value, which by default is true. But we can return false and the next methods won't be called:
The componentWillUpdate() method is called immediately before rendering, when new props or state are being received. We can use this as an opportunity to perform preparation before an updates occurs, however is not allowed to use this.setState().

- 1.prevProps: the previous properties object
- 2.prevState: the previous state object

A common case for this method is when we are using a third-party library that needs the rendered DOM to perform its job**Unmounting**
In this phase React provide us with only one method:**componentWillUnmount()**
It is called immediately before the component is unmounted from the DOM. We can use it to perform any cleanup we might need, such as invalidating timers or cleaning up any DOM elements that were created in componentDidMount()/componentDidUpdate().

Compare and contrast setState() and forceUpdate(). What is the significance of each, and when would you use one or the other? In addition, how might you use any data layer with React (like Backbone, Ember, or Redux)? What are the systemic requirements of doing so?

setState() called within a Component will tell React to trigger the proper re-rendering. It will also invoke the lifecycle methods, and those methods' control on the rendering process. For example, if **setState()** is called within **componentWillMount(),** the state update will be synchronous, and the Component will only render once. If called within **componentWillReceiveProps(),** there will not be an additional render. It will also not render if **shouldComponentUpdate()** returns false. forceUpdate(), on the other hand, completely overrules the rendering process and queues up a new render for React to display to the screen; it will also not call shouldComponentUpdate().

In most cases, you should use setState() unless your code is setup to need a bypass for shouldComponentUpdate(). As per the React docs, if your render() method reads from something other than this.props or this.state, such as if your Component has access to a shared variable that holds a Backbone Model, Redux Store, Ember Model, etc. In these cases, you'll need to likely use forceUpdate(). It's worth noting that you should never mutate this.state directly.

Q19: List some of the major advantages of React.

- It increases the application's performance
- It can be used on client and server side
- Code's readability increases, because of JSX.
- It is easy to integrate with other frameworks such as Angular, Meteor etc
- Using React, writing UI test cases become extremely easy
- React uses virtual DOM which is JavaScript object.
- This will improve apps performance
- Component and Data patterns improve readability.

Q20: What are the limitations of React?

- React is just a library, not a full-blown framework
- Its library is very large and takes time to understand
- It can be little difficult for the novice programmers to understand
- Coding gets complex as it uses inline templating and JSX

Q21: How is React different from Angular?

TOPIC	REACT	ANGULAR
ARCHITECTURE	Only the View	Complete MVC
RENDERING	Server side rendering	Client side rendering
DOM	Uses virtual DOM	Uses real DOM
DATA BINDING	One-way data binding	Two-way data binding
DEBUGGING	Compile time debugging	Run time debugging

Q22: What is arrow function in React? How is it used?

Arrow functions are more of brief syntax for writing the function expression. They are also called 'fat arrow' (=>) the functions. These functions allow to bind the context of the components properly since in ES6 auto binding is not available by default. Arrow functions are mostly useful while working with the higher order functions.

Q23: What is an event in React?

In React, events are the triggered reactions to specific actions like mouse hover, mouse click, key press, etc. Handling these events are similar to handling events in DOM elements. But there are some syntactical differences like:
Events are named using camel case instead of just using the lowercase.
Events are passed as functions instead of strings.
The event argument contains a set of properties, which are specific to an event. Each event type contains its own properties and behavior which can be accessed via its event handler only.

Q24: What are synthetic events in React?

Synthetic events are the objects which act as a cross-browser wrapper around the browser's native event. They combine the behavior of different browsers into one API. This is done to make sure that the events show consistent properties across different browsers.

Q25: List some of the cases when you should use Refs.

- When you need to manage focus, select text or media playback
- To trigger imperative animations
- Integrate with third-party DOM libraries

Q26: What can you do with HOC?

- Code reuse, logic and bootstrap abstraction
- Render High jacking
- State abstraction and manipulation
- Props manipulation

Q27: What is the significance of keys in React?

Keys are used for identifying unique Virtual DOM Elements with their corresponding data driving the UI. They help React to optimize the rendering by recycling all the existing elements in the DOM. These keys must be a unique number or string, using which React JS just reorders the elements instead of re-rendering them. This leads to increase in application's performance.

Q28: What is React Router?

React Router is a powerful routing library built on top of React, which helps in adding new screens and flows to the application. This keeps the URL in sync with data that's being displayed on the web page. It maintains a standardized structure and behavior and is used for developing single page web applications. React Router has a simple API.

Q29:What is the difference between React Native and React?

This is another one of the common React interview questions. It all starts with the fact that React is a JavaScript library and it supports both front-end web and being run on the server and it is commonly used for building user interfaces and web applications. React Native, however, is a mobile framework that compiles to native app components. It allows us to build native mobile applications for Windows, Android, and iOS in JavaScript while we can use ReactJS to build our components. With React Native, we can mimic the behavior of the native app in JavaScript and get a platform-specific code as the output. It is also possible to mix the native code with the JavaScript if we need to optimize our application further.

30: How is flux different from redux?

FLUX	REDUX
Flux is a container for application state and logic that are	Redux is a container for JavaScript apps.

registered to callbacks.

It is an observer pattern that is modified to fit React.	It offers interesting features such as writing applications, testing in different environments such as a server, client, etc.
It is a fancy name given to observer pattern and Facebook has developed tools to aid the implementation of these patterns.	In redux, actions can be functions or promises.
Flux supports actions that are simple JavaScript objects.	Redux is the first choice for web developers because it offers live code editing.

Q31: What are the Controlled component and uncontrolled component.

Controlled component is more advisable to use as it is easier to implement forms in it. In this, form data are handled by React components. A controlled input accepts values as props and callbacks to change that value.
The uncontrolled component is a substitute for controlled components. Here form data is handled by DOM itself. In uncomfortable components, the ref can be used to get the form values from DOM.

Q32: What is reconciliation in React?

It's React's process of re-rendering its tree of UI components.

Q33: What is Babel and how is it often used in the context of a React application?

Babel is a JS compiler, and is often used for things like translating JSX into JS, or translating ES6 JS code into ES5 for broader browser support.

Q34: What is the difference between a Presentational component and a Container component?

Presentational components are concerned with how things look. They generally receive data and callbacks exclusively via props. These components rarely have their own state, but when they do it generally concerns UI state, as opposed to data state.
Container components are more concerned with how things work. These components provide the data and behavior to presentational or other container components. They call Flux actions and provide these as callbacks to the presentational components. They are also often stateful as they serve as data sources.

CallBacks

CallBack
First.jsx

```
import React, { Component } from ‘react’;
import Second from './seconds';

class First extends Component{
constructor(props){
super();
this.state={ourData : ""}
}

_onClickChange= mydata =>{
this.setState({ourData : mydata})
this.props.refered(mydata)
}
render(){{
return(
'<'Second refer = {this._onClickChange} />

) } }

export default First;
```

Second.jsx

```
import React, { Component } from ‘react’;
import Thired from './tired’;

class Second extends Component{
constructor(props){
super();
this.state={ourData : ""}
}
```

```
_onClickChange= mydata =>{
this.setState({ourData : mydata})
this.props.refer(mydata)
}
render(){
return{(
'<'Thired refers = {this._onClickChange} />
) } }

export default Second;
```

Thired.jsx

```
import React, { Component } from ‘react’;

class Thired extends Component{
constructor(props){
super();
this.state=(data:'Thired ka Props'}
}

changeText(){
var mydata = this.state.data
this.props.refers(mydata)
}
render(){
return(

'<'button onClick={()=>this.changeText()}>CallBack'<'/button>
) } }

export default Thired;
```

CallBack.jsx

```
import React, { Component } from ‘react’;
import First from ‘./first’;

class CallBack extends Component{
constructor(props){
super();
this.state={ourData : ""}
}

_onClickChange= mydata =>{
this.setState({ourData : mydata})
}
render(){
return(
'<'First refered = {this._onClickChange} />

{this.state.ourData}


) } }

export default CallBack;
```

SetInterval

Axios

setInterval.jsx

```
import React, { Component } from ‘react’;
import axios from ‘axios';

class Setinterval extends Component {
constructor(props){
super(props);
```

```
this.state ={
persons: [],
}
}

getData(){
this.timeriD = setinterval(
() => this.tick()},
1000
)
}

tick() {
// this.setState({
axios.get('https://jsonplaceholder.typicode.com/users')
.then(res => {
const persons =res.data;
this.setState({ persons });
}
console.log('updating',this.state.person)
}

componentDidMount(){
this.getData();
}

// componentWillUnmount() {
// clearinterval(this.timeriD);

render() {
return( '<'div>
'<'p>Axios Setinterval'<'/p>
{ this.state.persons.map(person => '<'li key={person.id}>
{person.name}'<'/li>)}
'<'/div>
)
}
}
```

RenderData

Render data Greeting.jsx

```
import React from 'react';

function UserGreeting(props) {
return '<'h1>Welcome back!'<'/h1>;
}

function GuestGreeting(props) {
return '<'h1>Please sign up.'<'/h1>; }

export function Greeting(props) {
const isLoggedin = props.isLoggedin;
if (isLoggedin) {
return '<'UserGreeting >
}
return '<'GuestGreeting>
}

export function LoginButton(props) {
return (
'<'button onClick={props.onClick}>
Login
'<'/button>
)
}
```

```
export function LogoutButton(props) {
return (
'<'button onClick={props.onClick}>
Logout
'<'/button>
)
}
```

If Else.jsx

```
import React, { Component } from ‘react’;
import { Greeting, LoginButton,LogoutButton } from './greeting.js'

class RenderConditional extends Component {
constructor(props) {
super(props);
this.handleLoginClick = this.handleLoginClick.bind(this);
this.handleLogoutClick = this.handleLogoutClick.bind(this);

this.state ={
isLoggedin: false
}

handleLoginClick() {
this.setState({isLoggedin: true});
}

handleLogoutClick() {
this.setState({isLoggedin: false});
}
render() {
const isLoggedin = this.state.isLoggedin;
let button;
if (isLoggedin) {
button = '<'LogoutButton onClick={this.handleLogoutClick} />;
else {
button = '<'LoginButton onClick={this.handleLoginClick} />;
}
return (
'<'div> '<'Greeting isLoggedin={isLoggedin} />
{button}
'<'/div> )
}
}

export default RenderConditional;
```

Ternary ConditionWarning.jsx

```
import React, { Component } from ‘react’;

export function WarningBanner(props) {
if (!props.warn) {
return null;
}
return (
'<'div className="warning">
Warning!
'<'/div> )
}
```

TernaryCondition.jsx

```
import React, { Component } from ‘react’;
import { WarningBanner } from './warning'

class TernaryCondition extends Component {
constructor(props) {
super(props);
this.state = {showWarning: true}
this.handleToggleClick = this.handleToggleClick.bind(this);
```

```
}

handleToggleClick() {
this.setState(prevState => ({
showWarning: !prevState.showWarning
));
}
render() {
return (
'<'div>
'<'WarningBanner warn={this.state.showWarning} />
'<'button onClick={this.handleToggleClick}>
{this.state.showWarning ? ‘Hide’ : ‘Show'}
'<'/button>
'<'/div>
)
}
}

export default TernaryCondition;
```

Event

Event

event.jsx

```
import React ,{ Component }from ‘react’;
import ReactDOM from ‘react-dom';

class Toggle extends React.Component {
constructor(props) {
super(props);
this.state = {isToggleOn: true};
this.handleClick = this.handleClick.bind(this);
}

handleClick() {
this.setState(prevState => ({
isToggleOn: !prevState.isToggleOn
)
}
render() {
return (
'<'button onClick={this.handleClick}>
{this.state.isToggleOn ? 'ON' : 'OFF'}
'<'/button>
)
}
}

export default Toggle;
```

Select

Select

select.jsx

```
import React ,{ Component }from ‘react’;
import ReactDOM from ‘react-dom';

class Toggle extends React.Component {
constructor(props) {
super(props);
this.state = {isToggleOn: true};
this.handleClick = this.handleClick.bind(this);
}
```

```
handleClick() {
this.setState(prevState => ({
isToggleOn: !prevState.isToggleOn
)
}

render() {
return (
'<'button onClick={this.handleClick}>
{this.state.isToggleOn ? 'ON' : 'OFF'}
'<'/button>
)
}
}

export default Toggle;
```

PassFunction

Pass Function As Child

two.jsx

```
import React from 'react';
export const Amount = ({ amount, onIncrement,
onDecrement }) => (
'<'div>
'<'span>Us Dollar: {amount} '<'/span>
'<'button type="button" onClick={onincrement}>
+ {console.log('increment'}}
'<'/button>
'<'button type="button" onClick={onDecrement}>
{console.log('decrement'}}
'<'/button>
'<'/div>
)

export const Euro = ({ amount }) => '<'p>Euro: {amount * 0.86}'<'/p>;
export const Pound =({ amount }) => '<'p>Pound: {amount * 0.76}'<'/p>;

export default Amount;
```

passFunctions.jsx

```
import {Amount,Euro,Pound} from './two'

class PassFunction extends Component {
constructor(props) {
super(props);
this.state ={
amount: 0,
}

onincrement = () => {
this.setState(state => ({ amount: state.amount + 1 }));

onDecrement =() => {
this.setState(state => ({ amount: state.amount - 1 }));

render() {
return (
'<'div>
'<'Amount
amount={this.state.amount}
onincrement={this.onIncrement}
onDecrement={this.onDecrement}
>
'<'Euro amount={this.state.amount} />
```

```
'<'Pound amount={this.state.amount} />
'<'/div>
)
}
}

export default PassFunction;
```

Refs

Refs

child.jsx

```
import React, { Component } from 'react';

class Child extends React.Component {
constructor(props) {
super(props)
this.state ={
text:'child'
}

changeText=()=> {
this.setState({ text:'child text changed'})
}

render() {
return (
'<'div>
{this.state.text}
'<'/div>
)
}
}

export default Child;
```

refsMain.jsx

```
import React, { Component } from 'react';
import Child from './child';

class Ref extends Component {
constructor(props) {
super(props)
this.state = {}
}

accessChild=()=> {
this.refs.child.changeText()
}

render() {
return (
'<'div>Parent Component '<'button onClick={this.accessChild} >change'<'/button>
'<'Child ref="child" />
'<'/div>
)
}
}

export default Ref;
```

CheckBox

Checkbox checkbox.jsx

```
import React, { Component } from ‘react’;
class Checkbox extends Component {
state ={
isChecked: false,
}
toggleCheckboxChange = (} => {
const { handleCheckboxChange, label } = this.props;
this.setState({
isChecked :!this.state.isChecked
}
handleCheckboxChange(label);
})
render() {
const { label } = this.props;
const { isChecked } = this.state;
return (
'<'div className="checkbox"'>'
'<'label'>'
'<'input
type="checkbox"
value={label}
checked={isChecked}
onChange={this.toggleCheckboxChange}
'>'
{label}
) }
}

export default Checkbox;
```

mainCheckbox.jsx

```
import React, { Component } from ‘react’;
import Checkbox from './checkbox';
const items =[ ‘One’, ‘Two', ‘Three’, ];

class CheckBox extends Component {
componentWillMount(){
this.selectedCheckboxes = new Set();
}

toggleCheckbox = label => {
if (this.selectedCheckboxes.has(label)) {
this.selectedCheckboxes.delete(label);
}
else {
this.selectedCheckboxes.add(label);
} }

handleFormSubmit = e => {
e.preventDefault();
for (const checkbox of this.selectedCheckboxes) {
console.log(checkbox, ‘is selected.');
} }

createCheckbox = label => (
'<'Checkbox label={label} handleCheckboxChange= {this.toggleCheckbox} key={label} '>'
)

createCheckboxes =() => (
items.map(this.createCheckbox)
)
render() {
return (
'<'form onSubmit={this.handleFormSubmit}'>'
{this.createCheckboxes()}
'<'button type="submit">Save'<'/button'>'
'<'/form>
)
}
```

```
}
```

Splice

Splice

splice.jsx

```
import React, { Component } from 'react';

class Splice extends Component {
constructor() {
super();
this.state ={
components: []
}
addNewElement(element, selectedindex) {
if (selectedindex =="last") {
this.state.components.push(element);
felse {
this.state.components.splice(selectedindex, 0, element);
}
this.setState({ components: this.state.components });
}
render() {
let input, option;
// log out state
console.log(this.state.components);
return (
'<'div>
'<'h3>{this.state.components.join(", "}}'<'/h3>
'<'input placeholder="enter element"
ref={node => {
input = node;
'>'
'<'select
className={this.state.components.length ?"" : "hidden"}
ref={node => {
option = node;
'>'
'<'option value="" disabled selected>insert at index...'<'/option>
{this.state.components.map((component, index) =>
'<'option>{index}
'<'option>last
'<'/select>
'<'button onClick={() => {
this.addNewElement(input.value, option.value); '>'
Click '<'/button>
'<'/div>
)
}
}

export default Splice;
```

Curd

Curd

curd.jsx

```
import React, { Component } from 'react';

class Curd extends Component {
constructor(){
super();
this.state={
isEdit:false,
```

```
id:'',
mockData:[{id:'1',title:' Apple’ isEdit:false,date:new Date()},
{id:'2',title:'Milk',isEdit:false,date:new Date()}},
{id:'3',title:'Mango',isEdit:false,date:new Date()}},
{id:'4',title:'Papaya', isEdit:false,date:new Date()},]
}
}

onAdd=(evt)=>{
evt.preventDefault();
this.setState({
mockData:[...this.state.mockData, {
title:evt.target.item.value,
date:new Date(),
isEdit:false
]
}
evt.target.item.value=""
}

onDelete(id){
this.setState({
mockData:this.state.mockData.filter(item=>{
if(item.id != id){
return item
}
}
}
}

onEdit (){
this.setState({
isEdit:true,
id:arguments[0],
title:arguments([1]
}
}

onUpdate = (evt) => {
evt.preventDefault();
this.setState({
mockData: this.state.mockData.map(item => {
if (item.id === this.state.id) {
item['title’] = evt.target.updates.value;
return item;
}
return item
}
})
this.setState({isEdit:false})
}

onUpdateHandle(){
if(this.state.isEdit){
return '<'form onSubmit=(this.onUpdate}>
'<'input type="text" name="updates" defaultValue=({this.state.title} />
'<'button>update'<'/button>
'<'/form>
}
}
render(){
return(
'<'div>
{this.onUpdateHandle(}}
'<'form onSubmit={this.onAdd}>
'<'input type="text" name="item"/>
'<'button>Add'<'/button>
'<'/form>
{this.state.mockData.map(({items)=>(
'<'div key={items.id}>{items.title}
```

```
'<'button onClick={this.onDelete.bind(this.items.id)}>Delete'<'/button>
'<'button onClick={this.onEdit.bind(this, items.id, items.title)} >Edit'<'/button>
'<'/div>
'<'/div>
)
}
}

export default Curd;
```

MernCurd

MernCurd
DB.js

```
module.exports = {
DB: ''mongodb://localhost:27017/reactcrud'
}
```

bussinesModal.js

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;
// Define collection and schema for Business

let Business = new Schema({
person_name: {
type: String
business_name: {
type: String
business_gst_number: {
type: Number
}
collection: 'business'
);

module.exports = mongoose.model{'Business', Business);
```

bussinesRoutes.js

```
const express = require('express');
const businessRoutes = express.Router();
// Require Business model in our routes module
let Business = require('./business.model');

// Defined store route
businessRoutes.route('/add').post(function (req, res) {
let business = new Business(req.body);
business.save()
.then(business => {
res.status(200).json({'business': 'business in added successfully'});
)

.catch(err => {
res.status(400).send("unable to save to database");
);
);

// Defined get data(index or listing) route
businessRoutes.route('/").get(function (req, res) {
Business.find(function(err, businesses){

if(err){
console.log(err);
}
else {
res.json(businesses);
}
);
```

```
businessRoutes.route(‘/edit/:id').get(function (req, res) {
let id = req.params.id;
Business.findByld{id, function (err, business){
res.json(business);
);

businessRoutes.route(‘/update/:id').post(function (req, res) {
Business.findByld(req.params.id, function(err, business) {
if (!business)
res.status(404).send("data is not found");
else {
business.person_name =req.body.person_name;
business.business_name =req.body.business_name;
business.business_gst_number =req.body.business_gst_number;
business.save().then(business => {
res.json('Update complete’);
)
.catch(err => {
res.status(400).send("unable to update the database");
);
}
);

businessRoutes.route(‘/delete/:id').get(function (req, res) {
Business.findByldAndRemove({_id: req.params.id}, function(err, business){
if(err) res.json(err);
else res.json(‘Successfully removed’);
);
);

module.exports = businessRoutes;
```

server.js

```
const express = require(‘express');
const app = express();
const bodyParser = require(‘body-parser');
const PORT = 4000;
const cors = require('cors');
const mongoose = require(‘mongoose’);
const config = require('./DB.js');
const businessRoute = require('./business.route');
mongoose.Promise = global.Promise;
mongoose.connect(config.DB, { useNewUrlParser: true }).then(
() => {console.log(‘Database is connected’) },
err => { console.log(‘Can not connect to the database'+ err)}
)
app.use(cors(});
app.use(bodyParser.urlencoded({extended: true}));
app.use(bodyParser.json();
app.use('/business', businessRoute);
app.listen(PORT, function(){
console.log(‘Server is running on Port:',PORT);
);
```

itemActions.js

```
import { GET_ITEMS, ADD_ITEM, DELETE_ITEM, ITEMS_LOADING } from ‘./types'
import axios from ‘axios'

export const getitems =() => dispatch => {
dispatch(setitemsLoading()});
axios
.get('./api/items')
.then(res =>
dispatch({
type: GET_ITEMS,
payload: res.data
)
```

```
}

export const deleteItem = (id) => dispatch => {
axios
.delete(*/api/items/'$'{id}).then(res =>
dispatch({
type:DELETE_ITEM,
payload: id
)
}

export const additem = (item) => dispatch => {
axios
-post('/api/business/add', item)
.then(res =>
dispatch({
type: ADD_ITEM,
payload: res.data
)
}

export const setitemsLoading = () => {
return {
type: ITEMS_LOADING
}
}
```

types.js

```
export const GET_ITEMS ='GET_ITEMS'
export const ADD_ITEM =‘ADD_ITEM'
export const DELETE_ITEM = ‘DELETE_ITEM'
export const ITEMS_LOADING = 'ITEMS_LOADING'

//getting json placehilder data
export const FETCH_POSTS ='FETCH_POSTS';
export const NEW_POST = 'NEW_POST';
export const UPDATE = ‘UPDATE’;
```

itemReducers.js

```
import { GET_ITEMS, ADD_ITEM, DELETE_ITEM, ITEMS_LOADING } from'_.factions/types'
const initialState={
items:[],
loading: false
}

export default function(state=initialState, action){
switch(action.type){
case GET_ITEMS:
return {
..-State,
items: action.payload,
loading: false
}
case DELETE_ITEM:
return {
..-State,
items: state.items.filter{item => item._id !== action.payload)
}

case ADD_ITEM:
return {
..-State,
items: [action.payload, ...state.items]
}

case ITEMS_LOADING:
return {
..-State,
```

```
loading: true
}

default:
return state;
}
}
```

indexMern.js

```
import React, { Component } from ‘react’;
import axios from ‘axios';
import TableRow from ‘./TableRow';

export default class Index extends Component {
constructor(props) {
super(props);
this.state = {business: []};
}

componentDidMount(){
axios.get(‘http://localhost:4000/business')
.then(response => {
this.setState({ business: response.data });
}
.catch(function (error) {
console.log(error);
}
}
tabRow(){
return this.state.business.map(function(object, i){
return ;
}
render() {
return (
'<'div> '<'h3>Business List'<'/h3> '<'table style={{ marginTop: 20 }}> '<'thead> '<'tr> '<'th>Person '<'th>Business '<'th>GST
Number '<'th colSpan="2">Action '<'/tr> '<'/thead> '<'thody> { this.tabRow() } '<'/tbody> '<'/table> '<'/div> )
}
}
```

store.js

```
import { createStore, applyMiddleware, compose } from ‘redux’;
import thunk from ‘redux-thunk’;
import rootReducer from './reducers';

const initialState = {};
const middleWare = [thunk];
const store = createStore(
rootReducer,
initialState,
compose(applyMiddleware(...middleWare),
)
)

export default store;
```

create.jsx

```
import React, { Component } from ‘react’;
import axios from ‘axios';

export default class Create extends Component {
constructor(props) {
super(props);
this.state ={
person_name: '',
business_name:'',
business_gst_number:''
}
}
```

```
onChangePersonName=(e)=> {
this.setState({
person_name: e.target.value,
);
}

onChangeBusinessName=(e)=> {
this.setState({
business_name: e.target.value
)
}

onChangeGstNumber=(e)=> {
this.setState({
business_gst_number: e.target.value
)
}

onSubmit=(e)=> {
e.preventDefault();
const obj ={
person_name: this.state.person_name,
business_name: this.state.business_name,
business_gst_number:
this.state.business_gst_number
axios.post('http://localhost:4000/business/add', obj)
.then(res => console.log(res.data));
this.setState({
person_name: '',
business_name:'',
business_gst_number:''
)
}
render() {
return (
'<div> '<'h3 align="center">Add New Business'<'/h3> '<'form onSubmit={this.onSubmit}> '<'div> '<'label>Person Name:
'<'/label> '<'input type="text" value={this.state.person_name} onChange={this.onChangePersonName} /> '<'/div> '<'div>
'<'label>Business Name: '<'/label> '<'input type="text" value= {this.state.business_name} onChange=
{this.onChangeBusinessName} /> '<'/div> '<'div> '<'label>GST Number: '<'/label> '<'input type="text" value=
{this.state.business_gst_number} onChange= {this.onChangeGstNumber} /> '<'/div> '<'div> '<'input type="submit"
value="Register Business" /> '<'/div> '<'/form> '<'/div> )
}
}
```

edit.jsx

```
import React, { Component } from 'react';
import axios from 'axios';

export default class Edit extends Component {
constructor(props) {
super(props);
this.state ={
person_name: '',
business_name:'',
business_gst_number:''
}
}

componentDidMount() {
axios.get('http://localhost:4000/business/edit/'+this.props.match.params.id)
.then(response => {
this.setState({
person_name: response.data.person_name,
business_name: response.data.business_name,
business_gst_number: response.data.business_gst_number });
)
.catch(function (error) {
console.log(error);
```

```
)
}

onChangePersonName=(e)=> {
this.setState({
person_name: e.target.value
);
}

onChangeBusinessName=(e)=> {
this.setState({
business_name: e.target.value
)
}

onChangeGstNumber=(e)=> {
this.setState({
business_gst_number: e.target.value
)
}

onSubmit=(e)=> {
e.preventDefault();
const obj ={
person_name: this.state.person_name,
business_name: this.state.business_name,
business_gst_number: this.state.business_gst_number
axios.post('http://localhost:4000/business/update/'+this.props.match.params.id, obj)
.then(res => console.log(res.data));
this.props.history.push('/index');
}
render() {
return (
'<'div> '<'h3 align="center">Update Business'<'/h3> '<'form onSubmit={this.onSubmit}> '<'div> '<'label>Person Name:
'<'/label> '<'input type="text" value=(this.state.person_name} onChange={this.onChangePersonName} /> '<'/div> '<'div>
'<'label>Business Name: '<'/label> '<'input type="text" value={this.state.business_name} onChange=
({this.onChangeBusinessName} /> '<'/div> '<'div> '<'label>GST Number: '<'/label> '<'input type="text" value=
{this.state.business_gst_number} onChange=({this.onChangeGstNumber} /> '<'/div> '<'div> '<'input type="submit"
value="Update Business" /> '<'/div> '<'/form> '<'/div> )
}
}
```

table.jsx

```
import React, { Component } from 'react';
import { Link } from 'react-router-dom';
import axios from 'axios';

class TableRow extends Component {
constructor(props) {
super(props);
}

delete=()=> {
axios.get ('http://localhost:4000/business/delete/'+this.props.obj. _id)
.then(console.log('Deleted'})
-catch(err => console.log(err))
}
render() {
return (
'<'tr>
'<'td>
{this.props.obj.person_name}
'<'/td>
'<'td>
{this.props.obj.business_name} '<'/td>
'<'td>
{this.props.obj.business_gst_number} '<'/td>
'<'td>
'<'Link to={"/edit/"+this.props.obj._id}>Edit'<'/Link> '<'/td>
```

```
'<'td>
'<'button onClick={this.delete}>Delete'<'/button> '<'/td>
'<'/tr>
)
}
}

export default TableRow;
```

indexComponents.jsx

```
import React, { Component } from ‘react’;
import axios from ‘axios';
import TableRow from ‘./TableRow';

export default class Index extends Component {
constructor(props) {
super(props);
this.state = {business: []};
}

componentDidMount(){
axios.get(‘http://localhost:4000/business')
.then(response => {
this.setState({ business: response.data });
}
.catch(function (error) {
console.log(error);
}
}
tabRow(){
return this.state.business.map(function(object, i){
return '<'TableRow obj={object} key={{i} />;
}
render() {
return (
'<'div>
'<'h3>Business List'<'/h3>
'<'table style={{ marginTop: 20 }}>
'<'thead>
'<'tr>
'<'th>Person'<'/th>
'<'th>Business'<'/th>
'<'th>GST Number'<'/th>
'<'th colSpan="2">Action'<'/th>
'<'/tr>
'<'/thead>
'<'thody>
{ this.tabRow() }
'<'/tbody>
'<'/table>
'<'/div>
)
}
}
```

App.jsx

```
import { BrowserRouter as Router, Switch, Route, Link } from ‘react-router-dom';
import { Provider } from ‘react-redux';
import store from './store’;

import Create from './components/create’;
import Edit from './components/edit';
import Index from './components/index';

class App extends Component {
render() {
return (
'<'Provider store={store}>
```

```
'<'Router>
'<'div>
'<'nav>
'<'Link to={'/'} >React CRUD Example'<'/Link>
'<'tr>
'<'th>'<'Link to={'/'} >Home'<'/Link>'<'/th>
'<'th>'<'Link to={'/create'}>Create'<'/Link>'<'/th>
'<'th>'<'Link to={'/index'}>Index'<'/Link>'<'/th>
'<'/tr>
'<'/nav>
'<'Switch>
'<'Route exact path='/create' component={ Create } />
'<'Route path="/edit/:id' component={ Edit } />
'<'Route path="/index' component={ Index } />
'<'/Switch>
'<'/div>
'<'/Router>
'<'/Provider>
)
}
}

export default App;
```

EditDummyData

EditDummyData

dummyData.jsx

```
[
{
"id": 1,
"firstname": John",
"lastname": "Doe",
“types
“typeld":"3",
“typeName":"Email"
“amailreminder":"03:45"
{
"id": 2,
"firstname": "Bruno",
"lastname": "Mars",
'typeld":"3",
“typeName":"Email"
“amailreminder":"03:45"
{
"id": 3,
"firstname": John",
"lastname": "Doe",
'typeld":"3",
“typeName":"Email"
“emailreminder":"03:45"
}
]
```

editDummyData.jsx

```
import React from "react";
import idGenerator from "react-id-generator";
import DummyJsonData from './dummy.json'

export default class DummyData extends React.Component {
state ={
employees: [],
firstname: "",
lastname: "",
id: 0,
create: true
```

```
componentDidMount() {
const emps = DummyJsonData
this.setState({
employees: emps.map(e => {
return {
firstname: e.firstname,
lastname: e.lastname,
id: idGenerator()
)
);
}

handleChange = e => {
const name = e.target.name;
this.setState({ [name]: e.target.value });

handleCreateEmployee = () => {
if (this.state.employees) {
this.setState({
employees: [
...this.state.employees,
{
firstname: this.state. firstname,
lastname: this.state.lastname,
id: idGenerator()
}
]
);
felse {
this.setState({
employees: [
{
firstname: this.state. firstname,
lastname: this.state.lastname,
id: idGenerator()
}
]
);
}
this.setState({ firstname: "", lastname: "" });

handleEdit =e => {
const employee = this.state.employees.find(function(emp) {
if (emp.id === e.target.id) {
return emp;
}
this.setState({
firstname: employee.firstname,
lastname: employee.lastname,
id: employee.id,
create: false
); handleDelete =e => {
this.setState({
employees: this.state.employees.filter(function(emp) {
if (emp.id !== e.target.id) return emp;
)
);

handleUpdateEmployee = () => {
const employee = {
firstname: this.state. firstname,
lastname: this.state.lastname,
id: this.state.id

const employeesupdated = this.state.employees.map(emp => {
if (emp.id === this.state.id) {
return employee;
} else return emp;
);
```

```
this.setState((prevStae, props) => ({
employees: employeesupdated,
create: true,
firstname: "",
lastname: ""
)

render() {
const create =this.state.create ? "Save" : "Update";
const { employees } = this.state;
const inputIsEmpty = this.state.firstname ===""  | | this.state.lastname ===""  ? true : false;
return (
'<'div>
'<'input style={{ width: 120 }} type="text" placeholder="Enter Firstname" onChange=(this.handleChange} name="firstname"
value={this.state.firstname} >
'<'input style={{ width: 120 }} type="text" placeholder="Enter Firstname" onChange=(this.handleChange} name="lastname"
value={this.state.lastname} >
'<'button style={{ width: 150 }} disabled={inputIsEmpty} onClick={ this.state.create ? this.handleCreateEmployee :
this.handleUpdateEmployee } >
{create}
'<'/button>


'<'table border="1" style=({ width: 400, paddingTop: 5 }}>
'<'thead>
'<'tr>
'<'th>First Name
'<'th>Last Name
'<'th>Edit
'<'th>Delete
'<'/tr>
'<'/thead>
'<'thody>
{employees.map((emp, i) => {
return (
'<'tr key={i}>
'<'td>{emp.firstname}
'<'td>{emp.lastname}
'<'td>
'<'button onClick={this.handleEdit} id={emp.id}>
Edit '<'/button>
'<'/td>
'<'td>
'<'button onClick={this.handleDelete} id={emp.id}>
Delete '<'/button>
'<'/td>
'<'/tr>
)
'<'/tbody>
'<'/table>
'<'/div>
)
}
}
```


Sort

sort.jsx

```
class StoreViews extends Component {
state ={
store_views: [],
sort: {
key: null,
direction: ‘desc’,
}
}
```

```
componentDidMount() {
const apiUrl = ‘https://jsonplaceholder.typicode.com/users';
axios.get(apiUrl).then(res => {
const store_views =res.data;
this.setState({ store_views });
console.log(store_views, "store data")
}
-catch(error => {
throw(error);
);

compareBy = (key) => {
const direction = this.state.sort.key ? (this.state.sort.direction === 'asc' ? ‘desc’: ‘asc’) : ‘desc’;
const sortedData = this.state.store_views.sort((a, b) => {
if (key === 'accountName’) {
const nameA = a.accountName.toUpperCase();
const nameB = b.accountName.toUpperCase();
if (nameA < nameB) {
return -1;
}
if (nameA > nameB) {
return 1;
}
// names must be equal
return G;
else {
return a.contractValue - b.contractValue;
}
);
if (direction === 'desc’') {
sortedData.reverse();
}
this.setState({
store_views: sortedData,
sort: {
key,
direction,
}
);
}

render(){{
const { classes } = this.props;
return (
'<'div>
'<'AppBar position = "static">
'<'Typography variant = "h5" align = "center" className={classes.typographyText}>Store Views'<'/Typography>
'<'Link to="/" className={classes.linkStyles2}>
'<'Button style = {{color:‘white'}}>Home'<'/Button>
'<'/Link>
'<'Link to = "/clientsView" className={classes.linkStyles}>
'<'Button style = {{color:'white'}}>Client Views'<'/Button>
'<'/Link>
'<'/AppBar>
'<'Paper>
'<'Grid item xs={12}>
'<'Table className={classes.table}>
'<'TableRow style={{backgroundColor:‘lightblue'}}>
'<'StyledTablecell>
'<'div onClick={() => this.compareBy(‘id'}} className={classes.headerText}>
'<'i className="material-icons" style={{fontSize:'13px'}}>unfold_more'<'/i>Store ID
'<'/div>
'<'/StyledTableCell>
'<'StyledTablecell align="left">
'<'div onClick={() => this.compareBy(‘name'}} className={classes.headerText}>
'<'i className="material-icons" style={{fontSize:'13px'}}>unfold_more'<'/i>Name
'<'/div>
'<'/StyledTableCell>
'<'StyledTablecell align="left">
'<'div onClick={() => this.compareBy(‘phone'}} className={classes.headerText}>
```

```
'<'i className="material-icons" style={{fontSize:'13px'}}>unfold_more'<'/i>Phone Number
'<'/div>
'<'/StyledTableCell>
'<'StyledTablecell align="left">
'<'div onClick={() => this.compareBy(‘username'}} className={classes.headerText}>
'<'i className="material-icons" style={{fontSize:'13px'}}>unfold_more'<'/i>Address
'<'/div>
'<'/StyledTableCell>
'<'StyledTablecell align="left">
'<'div onClick={() => this.compareBy(‘email'}} className={classes.headerText}>
'<'i className="material-icons" style={{fontSize:'13px'}}>unfold_more'<'/i>Region
'<'/div>
'<'/StyledTableCell>
'<'/TableRow>
'<'TableBody>
{this.state.store_views.map(items => (
'<'StyledTableRow key={items.name}>
'<'StyledTableCell component="th" scope="row">
{items.id}
'<'/StyledTableCell>
'<'StyledTableCell align="left">{items.name}'<'/StyledTableCell>
'<'StyledTableCell align="left">{items.phone}'<'/StyledTableCell>
'<'StyledTableCell align="left">{items.address.street}'<'/StyledTablecell>
'<'StyledTableCell align="left">{items.address.city}'<'/StyledTableCell>
'<'/StyledTableRow>
'<'/TableBody>
'<'/Table>
'<'/Grid>
'<'/Paper>
'<'/div>
)
}
}

export default (withStyles(styles)(StoreViews));
```

ModalProps

modalProps.jsx

```
import React, { Component } from ‘react’;
import List from ‘@material-ui/core/List';
import Listitem from '@material-ui/core/Listitem’;
import Button from ‘@material-ui/core/Button’;
import Paper from '@material-ui/core/Paper'’;
import Previews from './previews';
import ViewDetails from ‘./view_details'
import data from './courses.json'

class LandingPage extends Component {
constructor(){
super();
this.state={
data:data,
sortBy:data,
anchorEl:false,
direction:{
percentage:‘asc',
name:'desc’,
durations:‘asc’,
}
}
}

handleClose =() => {
this.setState({ anchorEl: null });
render() {
return (
'<'div>
'<'List>
{data.map((value,index) => (
```

```
'<'div key ={index}>
'<'Paper elevation=(1} style=({marginBottom:10}}>
'<'ListiItem style={{fontSize:10}}>
'<'div className="col-sm-1">
'<'div className="live-text">{value.text}'<'/div>
'<'/div>
'<'div className="col-sm-6" style={{fontSize:14}}>
'<'div style={{fontWeight:'bold'}}>{value.name} '<'/div>
'<'div style={{fontWeight:'bold'}}>{value.course}'<'/div>
'<'div>{value.exame}'<'/div>
'<'div>'<'b>Course Duration: '<'/b>{value.durations}'<'/div>
'<'/div>
'<'div>
'<'Button style={{outline:'none'}}>
'<'div style={{marginTop:'15px'}}>{value.time}
'<'/div>
'<'i className="material-icons" style={{marginTop:15}}>account_circle'<'/i
> '<'/Button>
'<'/div>
'<'div style={{marginLeft:245}}>
'<'Previews data={value}/>
'<'ViewDetails data ={value}/>
'<'/div>
'<'/Listitem>
'<'/Paper>
'<'/div>
'<'/List>
'<'/div>
)
}
}

export default LandingPage;
```

previews.jsx

```
import React, { Component } from ‘react’;
import Button from ‘@material-ui/core/Button’;
import Dialog from ‘@material-ui/core/Dialog';
import Chip from '@material-ui/core/Chip';
import Paper from '@material-ui/core/Paper'’;
import TextField from ‘@material-ui/core/TextField';

class Previews extends Component {
state ={
open: false,
items:this.props.data,
name:'',
course:'',
durations:'',
course_admin:''

handleClickOpen =() => {
this.setState({ open: true });

handleClose =() => {
this.setState({ open: false, });

render() {
return (
'<'div>
'<'Chip label="Edit"
color="primary"
onClick={this.handleClickOpen}
variant="outlined"
style=(toggleBtn}
>
'<'Dialog
open={this.state.open}
onClose={this.handleClose}
```

```
aria-labelledby="alert-dialog-title"
aria-describedby="alert-dialog-description"
>
'<'Paper elevation={4} style={{borderBottom:'1px solid #e0e0e0'}}>
'<'div style={cardDetails}>Make changes
'<'button onClick={this.handleClose} style=(clearBtn}>
'<'i className="material-icons" style={{color:'gray'}}>clear'<'/i>
'<'/button>
'<'/div>
'<'/Paper>
'<'form noValidate style={{backgroundColor:'#F4F7F8'}}>
'<'div className="container">
'<'div className="row">
'<'div className="col-sm-6" style={{marginTop:16}}>
'<'label>Course Name'<'/label>
'<'TextField
name="course"
variant="outlined"
fullWidth margin="dense"
value={this.props.data.name}
inputProps={{
required: true,
minLength: 3
>
'<'/div>
'<'div className="col-sm-6" style={{marginTop:16}}>
'<'label>Course'<'/label>
'<'TextField
name="course"
variant="outlined"
fullWidth margin="dense"
value={this.props.data.course}
inputProps={{
required: true,
minLength: 3
>
'<'/div>
'<'/div>
'<'div className="row">
'<'div className="col-sm-12">
Goals '<'/div>
'<'div className="col-sm-12">
'<'/div>
'<'div className="col-sm-6">
'<'label>Course Duration'<'/label>
'<'TextField
name="course"
variant="outlined"
fullWidth margin="dense"
value={this.props.data.durations}
inputProps={{
required: true,
minLength: 3
>
'<'/div>
'<'div className="col-sm-6">
'<'label>Course Admin'<'/label>
'<'TextField
name="course"
variant="outlined"
fullWidth margin="dense"
value={this.props.data.course_admin}
inputProps={{
required: true,
minLength: 3
>
'<'/div>
'<'div className="col-sm-4">
'<'label>Course Icon'<'/label>
'<'input
```

```
accept="image/*"
id="text-button-file"
multiple
type="file"
>
'<'label htmlFor="contained-button-file">
'<'/label>
'<'/div>
'<'div className="col-sm-12">
'<'Button
variant="contained"
color="primary"
onClick={this.handleClose}
style={{float:'right', marginTop:'-15px',marginRight:'15px'}}
>
submit
'<'/Button>
'<'/div>
'<'/div>
'<'/div>
'<'/form>
'<'/Dialog>
'<'/div>
)
}
}

export default Previews
```

view_details.jsx

```
import React, { Component } from ‘react’;
import Dialog from ‘@material-ui/core/Dialog';
import Chip from '@material-ui/core/Chip';
import Paper from '@material-ui/core/Paper'’;
import TextField from ‘@material-ui/core/TextField';

class ViewDetails extends Component {
state ={
open: false,
items:this.props.data,
handleClickOpen =() => {
this.setState({ open: true });
handleClose =() => {
this.setState({ open: false });

render() {
return (
'<'div>
'<'Chip
label="Details"
color="primary"
onClick={this.handleClickOpen}
variant="outlined"
style=(toggleBtn}
>
'<'Dialog
open={this.state.open}
onClose={this.handleClose}
aria-labelledby="alert-dialog-title"
aria-describedby="alert-dialog-description"
>
'<'Paper elevation={4} style=({borderBottom:'1px solid #e0e0e0'}}>
'<'div style={cardDetails}>Details
'<'button onClick={this.handleClose} style=(clearBtn}>
'<'i className="material-icons" style={{color:'gray'}}>clear'<'/i>
'<'/button>
'<'/div>
'<'/Paper>
'<'div className="container"
```

```
style={{backgroundColor:'#F4F7F8'}}
>
'<'div className="row">
'<'div className="col-sm-6" style={{marginTop:10}}>
'<'label>'<'b>Course Name'<'/b>'<'/label>
'<'div>{this.props.data.name}'<'/div>
'<'/div>
'<'div className="col-sm-6"
style={{marginTop:10}}
>
'<'label>'<'b>Course'<'/b>'<'/label>
'<'div>{this.props.data.course}'<'/div>
'<'/div>
'<'/div>
'<'div className="row">
'<'div className="col-sm-12"
style={{marginTop:10}}
>
'<'label>'<'b>Course Duration'<'/b>'<'/label>
'<'div>{this.props.data.durations}'<'/div>
'<'/div>
'<'div className="col-sm-12"
style={{marginTop:10}}
>
'<'b>Goals'<'/b>
'<'/div>
'<'div className="col-sm-12"
style={{marginTop:10}}
>
'<'/div>
'<'div className="col-sm-6"
style={{marginTop:10,marginBottom:17}}
>
'<'label>'<'b>Course Admin'<'/b>'<'/label>
'<'TextField
name="course"
variant="outlined"
fullWidth margin="dense"
value={this.props.data.course_admin}
inputProps={{
required: true,
minLength: 3
>
'<'/div>
'<'/div>
'<'/div>
'<'/Dialog>
'<'/div>
)
}
}

export default ViewDetails
```


stteper.jsx

```
import React, { Component } from ‘react’;
import _ from ‘lodash';
import { withStyles } from ‘@material-ui/core/styles’;
import { Theme } from ‘@material-ui/core/styles/createMuiTheme';
import { Grid, Typography, WithStyles, createStyles, } from ‘@material-ui/core’;
import Paper from '@material-ui/core/Paper’’;
import { AccessTime as Backlcon, } from ‘@material-ui/icons';
import PlanCard from './roadMap.json'

class CoursePlaneRoadMap extends Component {
constructor(props){
super(props);
```

```
this.state ={
value:[],
image : true,

this.handleLeft = this.handleLeft.bind(this)
this.handleRight = this.handleRight.bind(this)
}

handleLeft() {
let { classes } = this.props;
this.state.value.push(
'<'Grid container>
'<'Grid item xs={2}>
'<'img alt="no content" className={classes.img} src="/assets/img/coursemapplane/leftline.png" />'<'/Grid> '<'Grid item xs=
{1}> '<'img alt="no content" className={S{classes.mark} {classes.markimg}'} src="/assets/img/coursemapplane/mark.png"
/>'<'/Grid>
'<'Grid item xs={5}>
'<'Paper elevation={5} className={classes.leftCard}>
{PlanCard.map({(value, index) => (
'<'div key ={index}>
'<'Typography variant="body2" className={classes.headerText}>{value.toptitle}'<'/Typography>

'<'Typography variant="body2">'<'Backicon />{value.durations}'<'img alt="no content" style={{width:"30px",
height:"20px"}}
src="/assets/img/coursemapplane/Goallcon.png" />{value.icons}'<'/Typography>
'<'Typography variant="body1" className={classes.assignmentText}>{value.assigntest}'<'/Typography>
'<'Typography variant="body1">{value.notes}'<'/Typography>
'<'/div>
'<'/Paper>
'<'/Grid>
'<'/Grid>
)

this.setState(
{image:!this.state.image}
)
}

handleRight() {
let { classes } = this.props;
this.state.value.push(
'<'Grid container>
'<'Grid item xs={1}>
'<'img alt="no content" className={classes.img} src="/assets/img/coursemapplane/rightline.png" />
'<'/Grid>
'<'Grid item xs={3}>
'<'Typography variant="body2" className={classes.subHeaderText}>-2 Sub Milestones'<'/Typography>
'<'Grid item xs={1}>
'<'img alt="no content" className={"S{classes.markpanel}'} src="/assets/img/coursemapplane/panel.png" />
'<'/Grid>
'<'Paper elevation=(5} className={classes.rightCard}>
{PlanCard.map({(value, index) => (
'<'div key ={index}>
'<'Typography variant="body2" className={classes.testHeader}>{value.title}'<'/Typography>
'<'Typography variant="body1" className={classes.testDate}>{value.startdate}'<'/Typography>
'<'Typography variant="body2">{value.questions}'<'/Typography>
'<'/div>
'<'/Paper>
'<'Grid item xs={2}>
'<'img alt="no content" className={classes.markpanels} src="/assets/img/coursemapplane/panel.png" />
'<'/Grid>
'<'Paper elevation={5} className={classes.rightCards}>
{PlanCard.map({(value, index) => (
'<'div key ={index}>
'<'Typography variant="body2" className={classes.testHeader}>{value.title}'<'/Typography>
'<'Typography variant="body1" className={classes.testDate}>{value.startdate}'<'/Typography>
'<'Typography variant="body2">{value.questions}'<'/Typography>
'<'/div>
'<'/Paper>
'<'/Grid>
```

```
'</Grid>
)

this.setState(
{image:!this.state.image}
)
}

render(){
const { classes } = this.props;
let { value, image } = this.state;
let button;
if (image) {
button = '<'button onClick={this.handleLeft}>Add'</'button>;
}
else {
button = '<'button onClick={this.handleRight}>Add'</'button>;
}

return{(
'<'Grid container
spacing=(0}
direction="column"
>
'<'Grid item xs={10} className={classes.root}>
'<'div>
{button}
'</'div>
'<'div>
'<'Grid container>
'<'Grid item xs={1}>
'<'img alt="no content" className={classes.start} src="/assets/img/coursemapplane/station.png" />'<'/Grid>
'<'Grid item xs={6} className={classes.startText}>
'<'Typography variant="body2">+2 Sub Milestones'<'/Typography>'<'/Grid>
'<'/Grid>
{value.map((val:any) => {
return '<'div>{val}'<'/div>
'<'/div>
'<'img alt="no content" className={classes.goalcourse} src="/assets/img/coursemapplane/goal.png" />
'<'Typography variant="h6" className={classes.goalText}>Goal Complete'<'/Typography>
'<'/Grid>
'<'/Grid>
)
}

export default CoursePlaneRoadMap;
```


Search

data.json

```
[{
"id": 0,
"name": "Vernon Dunham",
"company": "Qualcore",
"email": "vernon.dunham@qualcore.com"
},
{
"id": 1,
"name": "Dori Neal",
"company": "Sunopia",
"email": "dori.neal@sunopia.com"
},
{
"id": 2,
"name": "Rico Muldoon",
"company": "Airconix",
"email": "rico.muldoon@airconix.com"
}]
```

Search.jsx

```
import React, { Component } from 'react'
import users from './data.json'

class Search extends Component {
constructor(props) {
super(props);
this.state = {
searchString: "",
users: []
};
}

componentDidMount() {
this.setState({
users: users
});
this.refs.search.focus();
}

handleChange = () => {
this.setState({
searchString: this.refs.search.value
});
}

render() {
let users = this.state.users;
let search = this.state.searchString.trim().toLowerCase();

if (search.length > 0) {
users = users.filter(function(user) {
return user.name.toLowerCase().match(search);
});
}

return (
'<'div>
'<'h3>React - simple search'</'h3>
'<'div>
'<'input type="text" value={this.state.searchString} ref="search"
onChange={this.handleChange} placeholder="type name here" />
'<'ul>
{users.map(l => {
return (
'<'li>
{l.name} '<'a href="#">{l.email}'</'a>
'</'li>
);
})}
'</'ul>
'</'div>
'</'div>
); } }
export default Search
```


controll uncontroll *Uncontrolled component , the component owns the state.*
Controlled component , the component does not own the state.

```
class ControllUncontroll extends Component{
state={
username: "",
password: ""
}

handleChange = (key, value) => {
this.setState({
[key]: value
});
```

```
};

handleSubmit = (e) => {
e.preventDefault()
alert(JSON.stringify(this.state))
}

render() {
return (
'<'form onSubmit={this.handleSubmit}>
'<'input
value={this.state.username}
onChange={e => this.handleChange("username", e.target.value)}
style={{ fontSize: 24 }} placeholder="type your email" />

'<'input
value={this.state.password}
onChange={e => this.handleChange("password", e.target.value)}
style={{ fontSize: 24}} placeholder="type your password" />

'<'button type="submit" style={{ fontSize: 24 }}>Submit'<'/button>
'<'/form>
);
}
}

export default ControllUncontroll;
```

Filter

filter

```
import React, { Component } from 'react';
import {connect} from 'react-redux'
import './App.css';
import countries from './countries.json'

class App extends Component {
constructor(){
super()
this.state={
arrForComp:countries.arr,
searchText:''
}
}

handleChange=(e) =>{
let {searchText,form} = this.props

let newArr = countries.arr.filter( item =>
item.toLowerCase().includes(e.target.value.toLowerCase()) )

this.setState({
arrForComp:newArr
})
// searchText(e.target.value)
}

render() {
console.log(this.state.arrForComp)
return (
'<'div className="App">
'<'input type="text" onChange={e=>this.handleChange(e)}/>

'<'div>
{this.state.arrForComp.map((data)=>
'<'h5 style={{padding:'30px',}}>{data}'<'/h5>)}
'<'/div>
'<'/div>
);
}
```

```
}

const mapStatetoProps = state =>({
form: state.form
})

const mapDispatchToProps = dispatch =>({
searchText : data =>{
dispatch({type:'SEARCH',data})
}
})

export default connect(mapStatetoProps,mapDispatchToProps) (App);
```


filter*Now your component has been mounted and ready, next React lifecycle method componentDidMount() comes in play. Unlike the render() method, componentDidMount() allows the use of setState(). Calling the setState() here will update state and cause another rendering but it will happen before the browser updates the UI. This is to ensure that the user will not see any UI updates with the double rendering.*

As the name suggests it handles the rendering of your component to the UI. It happens during the mounting and updating of your component.

React requires that your render() is pure. Pure functions are those that do not have any side-effects and will always return the same output when the same inputs are passed. This means that you can not setState() within a render().

```
import React, { Component } from 'react';
import axios from 'axios';

class ApiIntegration extends Component{
state={
apiData:[],
}

componentDidMount(){
axios.get('https://jsonplaceholder.typicode.com/posts')
.then(res => {
this.setState({ apiData:res.data })
})
}

render(){
return(
'<'div>
{ this.state.apiData.map(data =>
{ return '<'li>{data.title}'<'/li>})}
'<'/div>
)
}
}

export default ApiIntegration;
```


Counter.jsx

```
import React, { Component } from "react";

class Counter extends Component {
componentDidUpdate(prevProps, prevState) {
console.log("Previous Props", prevProps);
console.log("Previous States", prevState);
if (prevProps.counter.value === this.props.counter.value) {
console.log("didn't updated!!!");
}
}

componentWillUnmount() {
console.log("Component - Unmount");
}

getBadgeClasses() {
let classes = "badge m-2 badge-";
classes +=this.props.counter.value === 0 ? "warning" : this.props.counter.value < 0 ? "danger"
```

```
: this.props.counter.value > 0 ? "primary" : 0;
return classes;
}

formatCount() {
const { value } = this.props.counter;
return value === 0 ? "Zero" : value;
}
render() {
console.log("Counter - Rendered");
return (
'<'div>
'<'h4>Counter# {this.props.counter.id}'<'/h4>
'<'button
onClick={() => this.props.onDecrement(this.props.counter)}
>
- '<'/button>
'<'span className={this.getBadgeClasses()}> {this.formatCount()}'<'/span>
'<'button
onClick={() => this.props.onIncrement(this.props.counter)}
>
+ '<'/button>
'<'button
onClick={() => this.props.onDelete(this.props.counter.id)}
>
Delete '<'/button>
'<'/div>
);
}
}

export default Counter;
```

counters.jsx

```
import React, { Component } from "react";
import Counter from "./component_did_update";

class Counters extends Component {
render() {
// 'Object destructuring' applied
const { counters, onDecrement, onIncrement, onDelete } = this.props;
return (
'<'div>
{counters.map(counter => (
'<'Counter
key={counter.id}
onDecrement={onDecrement}
onIncrement={onIncrement}
onDelete={onDelete}
counter={counter}
/>
))}
'<'/div>
);
}
}

export default Counters;
```

MainCounter.jsx

```
import React, { Component } from "react";
import Counters from "./counters";

class MainCounter extends Component {
state = {
name: "Counters",
counters: [
{ id: 1, value: 4 },
```

```
{ id: 2, value: 0 },
{ id: 3, value: 0 },
{ id: 4, value: 0 }
]
};

render() {
return (
'<'React.Fragment>
'<'div
totalCounters={this.state.counters.filter(c => c.value > 0).length}
onReset={this.handleReset}
/>
'<'main>
'<'Counters
counters={this.state.counters}
onDecrement={this.handleDecrement}
onIncrement={this.handleIncrement}
onDelete={this.handleDelete}
/>
'<'/main>
'<'/React.Fragment>
);
}

handleDecrement = counter => {
const counters = [...this.state.counters];
const index = counters.indexOf(counter);
counters[index].value--;
this.setState({ counters });
};

handleIncrement = counter => {
const counters = [...this.state.counters];
const index = counters.indexOf(counter);
counters[index].value++;
this.setState({ counters });
};

handleDelete = counterId => {
const counters = this.state.counters.filter(c => c.id !== counterId);
this.setState({ counters });
};

handleReset = () => {
const counters = this.state.counters.map(c => {
c.value = 0;
return c;
});
this.setState({ counters });
};
}

export default MainCounter;
```

GetDerivedStateFromProps

getDerivedStateFromProps.jsx

```
import React from 'react';

class GetDerivedStateFromProps extends React.Component {
state = {
list : []
}

static getDerivedStateFromProps(props, preState){
console.log('getDerivedStateFromProps');
if(!props.add){
return {list:[]};
}
```

```
else {
return null;
}
}

addRow=()=> {
this.setState({
list: [ Math.abs(Math.random()*100),...this.state.list ]
})
}

render(){
console.log('children render');
return(
'<'div>
{this.state.list.map((val)=>(
'<'div key={val.toFixed(5)}>{val}'<'/div>
))}
'<'button disabled={!this.props.add} onClick={this.addRow}>Generate Number'<'/button>
'<'/div>
);
}
}

export default GetDerivedStateFromProps;
```


getSnapshotBeforeUpdate.jsx *replace componentWillUpdate().*

It is called right before the DOM is updated. The value that is returned from getSnapshotBeforeUpdate() is passed on to componentDidUpdate().

Keep in mind that this method should also be used rarely or not used at all.

Resizing the window during an async rendering is a good use-case of when the getSnapshotBeforeUpdate() used.

```
import React, { Component, Fragment } from "react";

class GetSnapshotBeforeUpdate extends Component {
constructor(props) {
super(props);
this.state = {
t: 0,
messages: []
};
this.chatRef = React.createRef();
}

componentDidMount() {
this.timerID = setInterval(() => this.addMessage(), 500);
}

addMessage() {
let { messages, t } = this.state;
if (messages.length < 1000) {
const newMessage = 2019-07-12T17:22:16.953Z message t;
messages = [...messages, newMessage];
t++;

this.setState({ messages, t });
}
}

getSnapshotBeforeUpdate(prevProps, prevState) {
const { current } = this.chatRef;
const isScrolledToBottom =
current.scrollTop + current.offsetHeight >= current.scrollHeight;
return { isScrolledToBottom };
}

// Recieve the snapshot and check if the user is scrolled to the bottom of the log
```

```
componentDidUpdate(prevProps, prevState, snapshot) {
const { isScrolledToBottom } = snapshot;
if (snapshot.isScrolledToBottom) {
this.chatRef.current.scrollTop = this.chatRef.current.scrollHeight;
}
}

renderMessage(msg, i) {
return
- {msg}
;
}

render() {
return (
'<'Fragment>
'<'h1>Message Log'<'/h1>
'<'div ref={this.chatRef} className="log">
'<'ul>
{this.state.messages.map((msg, i) => {
return this.renderMessage(msg, i);
})}
'<'/ul>
'<'/div>
'<'/Fragment>
);
}
}

export default GetSnapshotBeforeUpdate;
```