Simple step by step guide React with Redux

In this demo, in order to understand the importance of redux we will create a web application in two ways:

1. Web Application without Redux (Normal working with the use of State objects)
2. Web Application with Redux (Using Store and reducers)

The goal of this application is to display a join button along with a message reading “Would you like to join our team at Ironhack?.” Once the user clicks on the button, the message changes to “Thanks for joining us!” So first let’s start by building the application without Redux. Later on, we will look into Redux implementation. As we work through both designs, note the differences.

Web Application without Redux

1. Create a component called NewComp with the following code and export the component to the main component. The component NewComp.js reads the following code:

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

import { connect } from "react-redux";

class NewComp extends Component {

  constructor(props) {

super(props);



this.state = {
   message: "Would you like to join our team at Ironhack?"

};

  }

  Buttonchange = () => {

this.setState({

   message: "Thanks for joining us!"

});

  };



  render() {

return (

   <div className="App">

            <h3 style={this.styles}>{this.state.message}</h3>

            <button onClick={this.Buttonchange}>Join now</button>

   </div>

);

  }

}
```

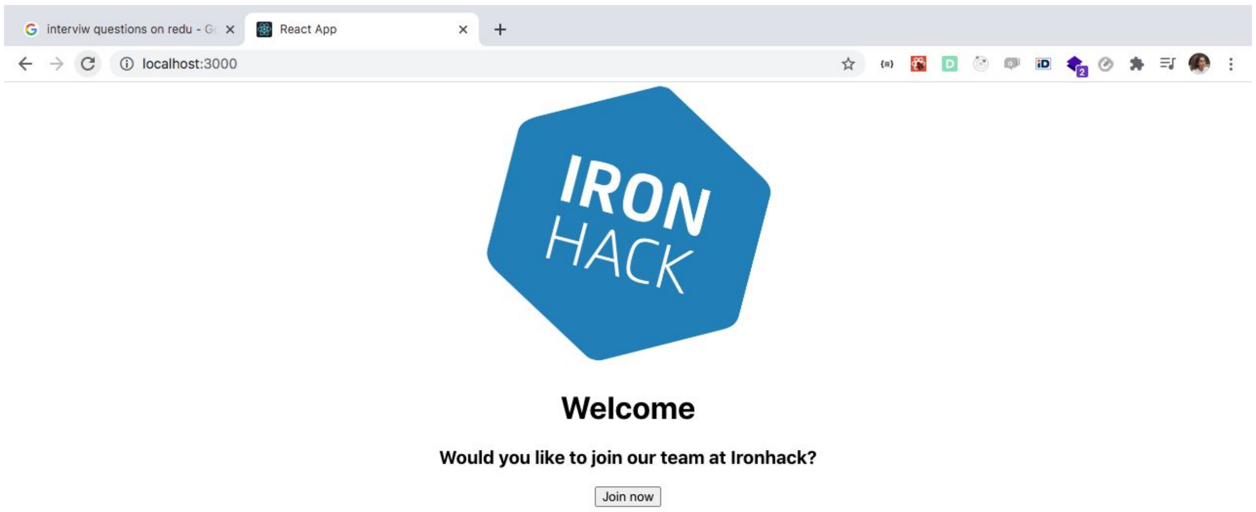
- In this component, we first initialize the state with the message set as “Would you like to join our team at Ironhack?”
- We then set the style of text using a styles keyword
- After setting the style, we create an arrow function that changes the state of the message to “Thank you for joining us!”
- Finally, we define a render method that renders the state on-screen along with the button

2. Import the component NewComp into the main component App.js. The App.js main component reads the following code:

```
import React from "react";

import "./App.css";

import NewComp from "./Components/NewComp";

class App extends React.Component {

  styles = {

fontStyle: "bold",

color: "teal"

  };

  render() {

return (

   <div className="App">

            <h1 style={this.styles}> Welcome </h1>

            <NewComp />
  </div>

);

 }

}

export default App;
```

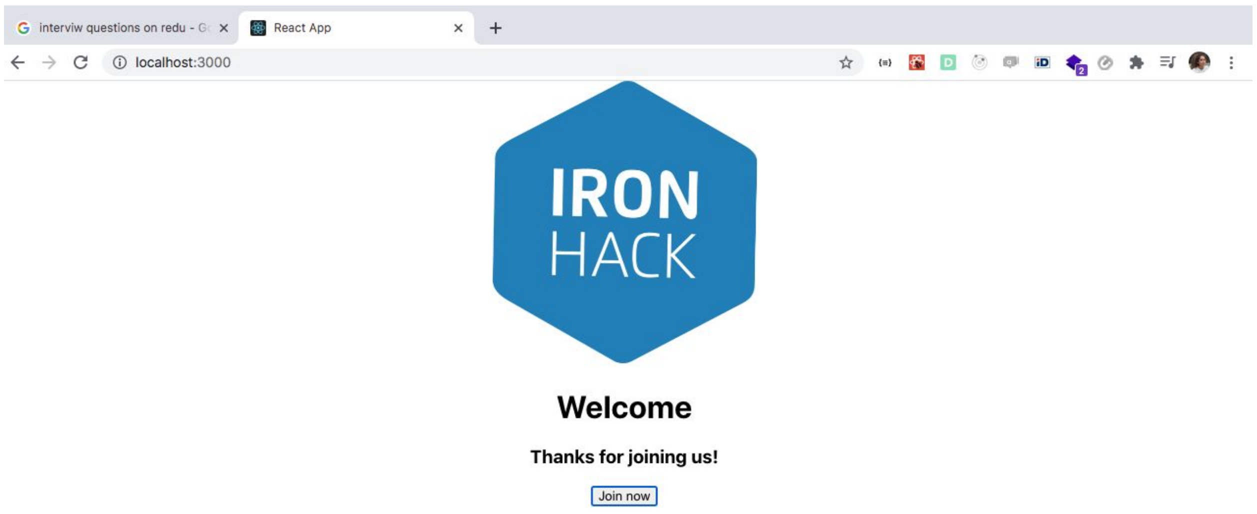
- Create the main component App.js
- Import NewComp into this file
- Define the styles for text
- Define the render() method that returns Welcome on the screen along with the content of NewComp that we imported before

3. The application, when run, should like this:



Welcome

Would you like to join our team at Ironhack?



Web Application with Redux

1. Create a store in the index.js file

import React from "react";

import ReactDOM from "react-dom";

import "./index.css";

import App from "./App";

```
import * as serviceWorker from "./serviceWorker";

import { Provider } from "react-redux";

import { createStore } from "redux";

import reducer from "./Store/Reducer";

const store = createStore(reducer);

ReactDOM.render(

  <Provider store={store}>

<App />

  </Provider>,

  document.getElementById("root")

);

serviceWorker.unregister();
```

- Import the necessary modules needed for implementing Redux in a React application
- Create a constant store with “reducer” as the function parameter
- Set the Provider to store and enclose the App component within it

2. Create a component called NewComp.js with the following code:

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

import { connect } from "react-redux";
```

```
class NewComp extends Component {

  render() {

return (

   <div className="App">

            <h3 style={this.styles}>{this.props.message}</h3>

            <button onClick={this.props.Buttonchange}>Subscribe</button>

   </div>

);

  }

}



const mapStatetoProps = state => {

  return {

message: state.message

  };

};



const mapDispatchToProps = dispatch => {
  return {

Buttonchange: () => dispatch({ type: "Message_change" })

  };

};

export default connect(

  mapStatetoProps,

  mapDispatchToProps

)(NewComp);
```

- Define the styles for the text
- Define the render() method that returns the text on the screen along with the Subscribe button
- As the name suggests, the mapStatetoProps() method maps the state to Props so the component can access it

3. Create a folder called Store and a JavaScript file called Reducer.js within it with the following code:

```
const initialState = {

  Message: “Would you like to join our team at ironhack?”

};



const reducer = (state = initialState, action) => {
  const newState = { ...state };

  if (action.type === "Message_change")

newState.message = "Thanks for joining us!";

  return newState;

};

export default reducer;
```

- Define a constant initialState and set the message
- Once the message is set, define a reducer arrow function which changes the state when it encounters a relevant action type

4. Finally, import the NewComp component in the main component App.js.

```
import React from "react";

import "./App.css";

import NewComp from "./Components/NewComp";

class App extends React.Component {

  styles = {

fontStyle: "bold",

color: "teal"

  };

  render() {
```

```
return (

   <div className="App">

            <h1 style={this.styles}> Welcome </h1>

            <NewComp />

   </div>

);

  }

}

export default App;
```

- This component will be very similar to the app component that we implemented in the previous method
- We set the styles for the on-screen text
- The render method returns the Welcome text, along with the contents of the NewComp component that we imported in this file

There’s practically no noticeable difference between the output of both implementations. The difference, however, is in how each implementation manages the states in the web application.

We have now created a React web application using both methods. React with Redux is more effective with larger React applications and should not be used for small or simple React web applications.

In case you have any questions related to this demo, feel free to contact me at Shadan.behzadian@gmail.com