**Lesson 04 Demo 03**

**React with Redux with Employee app**

**Objective:** To demonstrate the react with redux to store employee details, view employee details as well delete employee information from store.

**Tools required:** Node JS and React JS

**Prerequisites:** HTML, CSS, JavaScript ES5/ES6, Basic React Concept

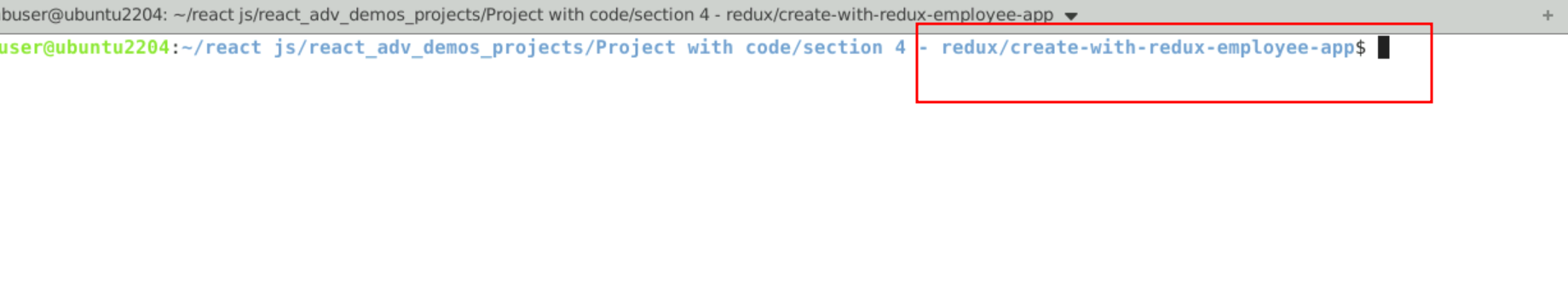
**Note** : All react js project already created with version 18.x with Sample App.js file

**Steps to be followed:**

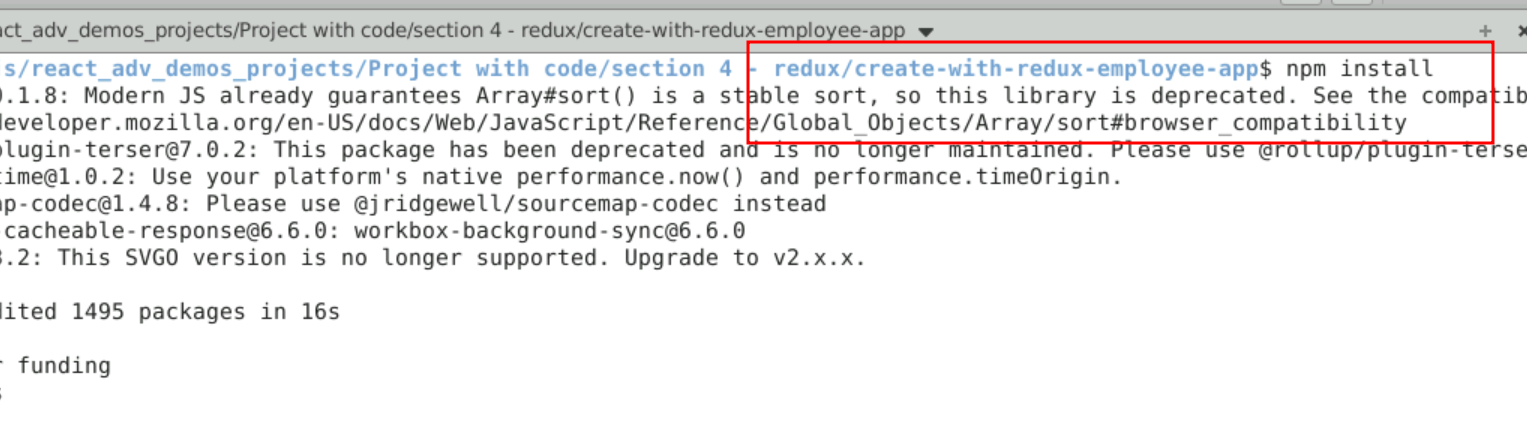
1. Set up for react js project
2. Create reducer, store and configuration for store in index.js function.
3. Now create user defined component as DisplayEmployee, AddEmployee, DeleteEmployee.
4. In App.js file import all user defined component and test the application.

**Step 1: Set up for react js project**

1. Open a terminal window inside a React JS pre-created project ie **create-with-redux-employee-app**

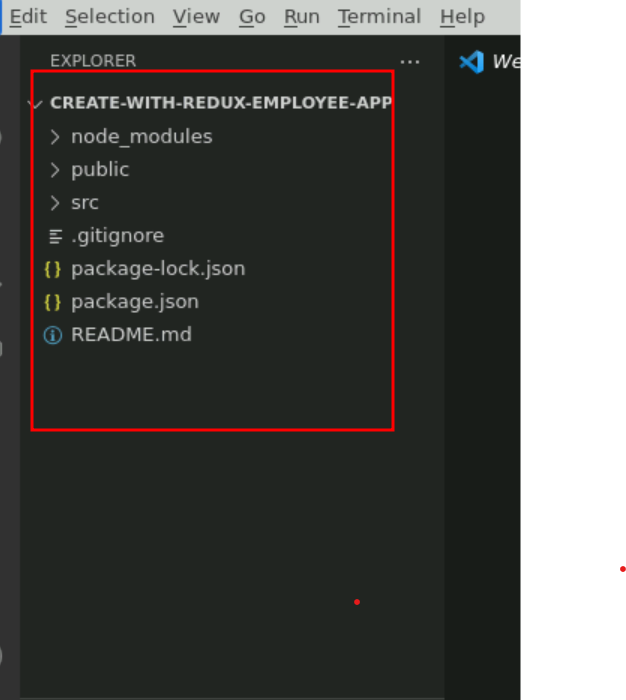


1. Now you need to run the command as **npm install.** This command helps us to installed all required dependencies mention in package.json file in local machine in the form of node\_module folder.

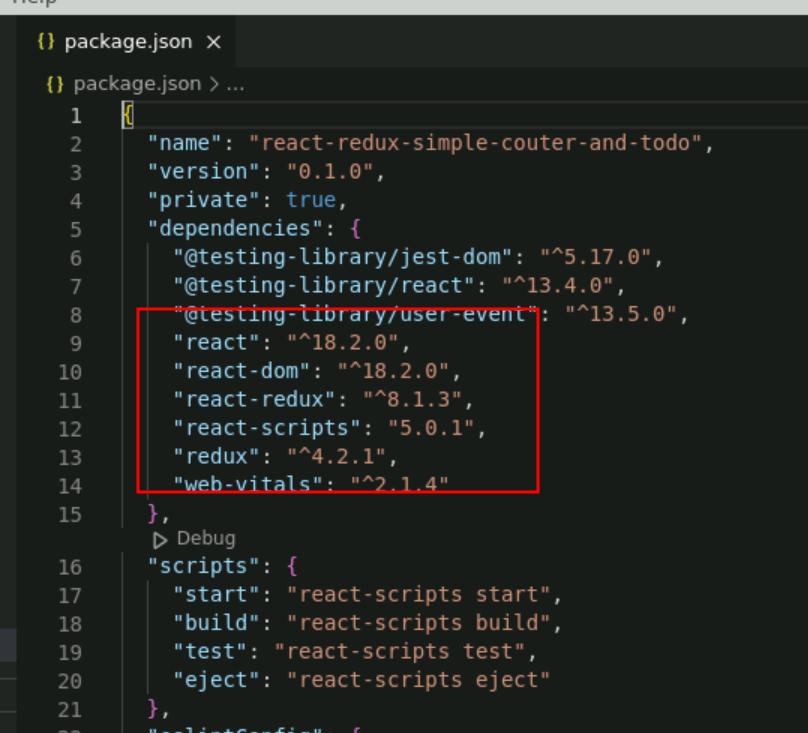


1. Now open **react-redux-employee-app** folder in VS Code Editor

Note: short cut to open write **code .**

****

1.4 now open package.json file and view external dependencies.

****

**Step 2:** Create reducer, store and configuration for store in index.js function.

2.1 create employeeReducer.js file. Which contains normal JavaScript function which takes two parameter ie state and action. Base upon action it will do some operation on state variable. And these variable as consider as global variable if we configure with store.

let initialState = {

employees:[

{id:100,name:"Ravi",age:21},

{id:101,name:"Ramesh",age:22}

]

}

function employeeReducer(state=initialState,action){

// base upon action we can do the changes on state variable.

// which is consider as global state

console.log(action);

if(action.type=="ADD\_EMPLOYEE"){

return {

...state,employees:state.employees.concat(action.payload)

//...state,employees:[...state.employees,action.payload]

}

}

if(action.type=="DELETE\_EMPLOYEE"){

return {

...state,employees: state.employees.filter(e=> e.id !== action.payload)

}

}

return state;

}

export default employeeReducer;

**2.2** We need to create react store with help of redux library. Here store connect with reducer to make state variable as global variable.

employeeStore.js

import { legacy\_createStore as createStore} from 'redux';

import reducer from './employeeReducer';

const store = createStore(reducer);

export default store;

**2.3 Now we need to configure store the index.js file. Inside this file we need to take the help of Provider part of react-redux and connect to store. Provider must apply for whole application so wrap for App root components.**

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';

**import store from './employeeStore'**

**import { Provider } from 'react-redux';**

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

<React.StrictMode>

**<Provider store={store}>**

<App />

**</Provider>**

</React.StrictMode>

);

// If you want to start measuring performance in your app, pass a function

// to log results (for example: reportWebVitals(console.log))

// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals

reportWebVitals();

**Step 3 :** Now create user defined component as DisplayEmployee, AddEmployee, DeleteEmployee..

.

3.1 create DisplayEmployee.js file which is responsible to access global state variable with help of useSelector hook and responsible to display the records in table format.

import { useSelector } from "react-redux";

import DeleteEmployee from "./DeleteEmployee";

function DisplayEmployee() {

let employees = useSelector(gs=>gs.employees);

let employeeRecord = employees.map((e,index)=>

<tr key={index}>

<td>{e.id}</td>

<td>{e.name}</td>

<td>{e.age}</td>

<td><DeleteEmployee id={e.id}/></td>

</tr>

)

return(

<div>

<h2>Employee Details</h2>

<table border="1">

<thead>

<th>Id</th>

<th>Name</th>

<th>Age</th>

<th>Delete</th>

</thead>

<tbody>

{employeeRecord}

</tbody>

</table>

</div>

)

}

export default DisplayEmployee;

3.2 create addEmloyee.js file. Which is responsible to add the employee details with the of useDispatch hook.

import { useState } from "react";

import { useDispatch, useSelector } from "react-redux";

function AddEmployee() {

//useDispatch is pre defined hook

// which help to pass action and payload (data)

// to reducer.

let [employee,setEmployee]=useState({id:0,name:"",age:0.0})

let dispatch = useDispatch();

let employees = useSelector(gs=>gs.employees);

let addEmployee = (event)=> {

event.preventDefault();

//console.log(employee)

let result = employees.find(e=>e.id==employee.id);

if(result==undefined){

dispatch({type:"ADD\_EMPLOYEE",payload:employee})

}else {

alert("Employee id must be unique")

}

setEmployee({id:0,name:"",age:0.0})

}

return(

<div>

<h2>Add Employee</h2>

<form onSubmit={addEmployee}>

<label>Id</label>

<input type="number" name="employee.id" value={employee.id} onChange={(event)=> {

setEmployee(e=>{return {...e,id:event.target.value}})

}}/><br/>

<label>Name</label>

<input type="text" name="employee.name" value={employee.name} onChange={(event)=> {

setEmployee(e=>{return {...e,name:event.target.value}})

}}/> <br/>

<label>Age</label>

<input type="number" name="employee.age" value={employee.age} onChange={(event)=> {

setEmployee(e=>{return {...e,age:event.target.value}})

}}/><br/>

<input type="submit" value="Add Employee" />

<input type="reset" value="reset"/>

</form>

</div>

)

}

export default AddEmployee;

3.3 Now create DeleteEmployee.js file. Which is responsible to delete the employee details using employee id with help of dispatch function.

import { useDispatch } from "react-redux";

function DeleteEmployee({id}){

let dispatch = useDispatch();

return(

<div>

<td><input type="button" value="Delete"

onClick={()=>dispatch({type:"DELETE\_EMPLOYEE",payload:id})}/></td>

</div>

)

}

export default DeleteEmployee;

**Step 4.** In App.js file import all user defined component and test the application.

**4.1** In App.js file import AddEmployee and DisplayEmployee component to test the application.

import AddEmployee from './AddEmployee';

import './App.css';

import DisplayEmployee from './DisplayEmployee';

function App() {

return (

<div>

<AddEmployee></AddEmployee>

<DisplayEmployee></DisplayEmployee>

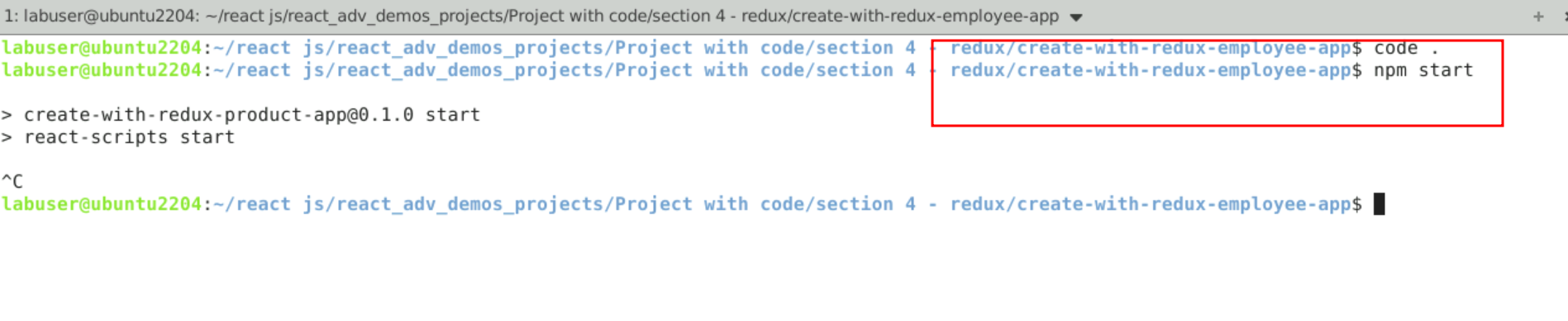
</div>

);

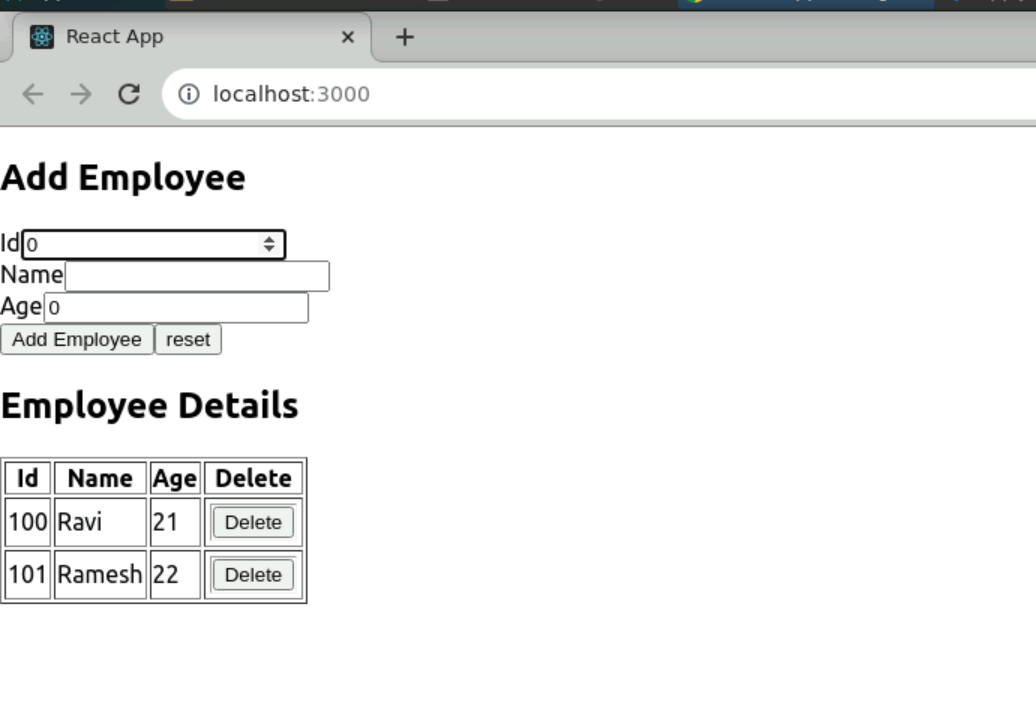
}

export default App;

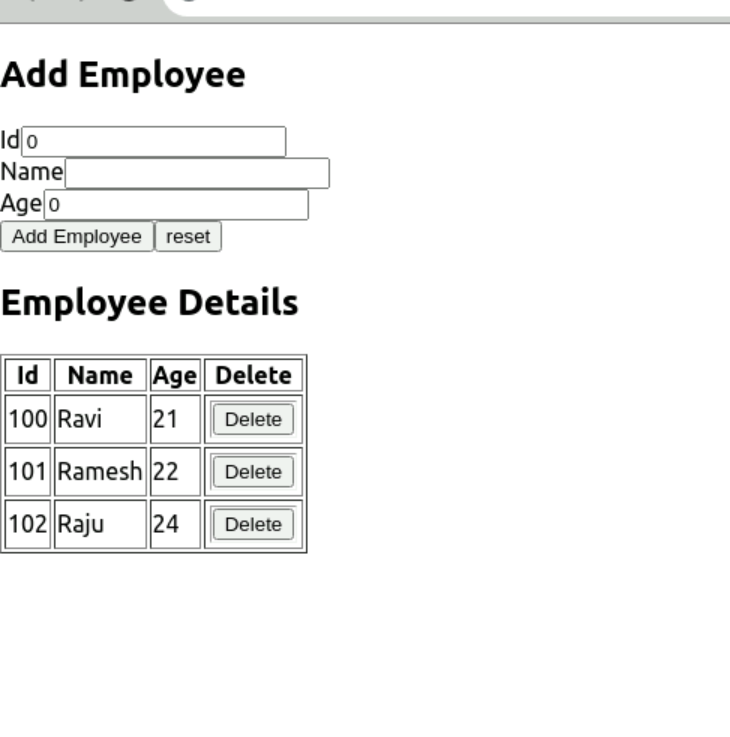
**4.2 Now we run the application using npm start**

****

**4.3** Now you can view the output on browser. You can view all employee details, you can add as well as delete the employee information.



After added new employee



After deleted employee info

