* Container in turns calls the component page.
* Declare a function in fat arrow notation
* Don’t forget to use return inside the fat arrow functions wherever is necessary in case of using curly braces for body
* return state.map(course => {
* return course.id === action.course.id ? action.course : course;
* Store= rsa ; a={catr)

return createStore(

rootReducer,

initialState,

composeEnhancers(applyMiddleware(thunk, reduxImmutableStateInvariant()))

);

const composeEnhancers =

window.\_\_REDUX\_DEVTOOLS\_EXTENSION\_COMPOSE\_\_ || compose;

* Rpc

Switch

<Route exact path="/" component={HomePage} />

* LNT
* <Link to="about" className="btn btn-primary btn-lg">
* Learn More
* </Link>
* Setting state object of object

this.setState({ employee: { ...this.state.employee, name: value } });

DATE PICKER:

**import** React **from** "react";

**import** DatePicker **from** "react-datepicker";

**import** "react-datepicker/dist/react-datepicker.css";

*// CSS Modules, react-datepicker-cssmodules.css*

*// import 'react-datepicker/dist/react-datepicker-cssmodules.css';*

class Example extends React.Component {

  constructor(props) {

    super(props);

    this.state **=** {

      startDate**:** **new** Date()

    };

    this.handleChange **=** this.handleChange.bind(this);

  }

  handleChange(date) {

    this.setState({

      startDate**:** date

    });

  }

  render() {

**return** (

**<**DatePicker

        selected**=**{this.state.startDate}

        onChange**=**{this.handleChange}

dateFormat="yyyy/MM/dd"

placeholderText="Click to select a date"

name=”Date of birth ”

**/>**

    );

  }

}

--------------------------------------------------------------------------------------------------------------------------------------------------------

apiServer.js

/\*

This uses json-server, but with the module approach: https://github.com/typicode/json-server#module

Downside: You can't pass the json-server command line options.

Instead, can override some defaults by passing a config object to jsonServer.defaults();

You have to check the source code to set some items.

Examples:

Validation/Customization: https://github.com/typicode/json-server/issues/266

Delay: https://github.com/typicode/json-server/issues/534

ID: https://github.com/typicode/json-server/issues/613#issuecomment-325393041

Relevant source code: https://github.com/typicode/json-server/blob/master/src/cli/run.js

\*/

/\* eslint-disable no-console \*/

const jsonServer = require("json-server");

const server = jsonServer.create();

const path = require("path");

const router = jsonServer.router(path.join(\_\_dirname, "db.json"));

// Can pass a limited number of options to this to override (some) defaults. See https://github.com/typicode/json-server#api

const middlewares = jsonServer.defaults({

// Display json-server's built in homepage when json-server starts.

static: "node\_modules/json-server/dist"

});

// Set default middlewares (logger, static, cors and no-cache)

server.use(middlewares);

// To handle POST, PUT and PATCH you need to use a body-parser. Using JSON Server's bodyParser

server.use(jsonServer.bodyParser);

// Simulate delay on all requests

server.use(function(req, res, next) {

setTimeout(next, 0);

});

// Declaring custom routes below. Add custom routes before JSON Server router

// Add createdAt to all POSTS

server.use((req, res, next) => {

if (req.method === "POST") {

req.body.createdAt = Date.now();

}

// Continue to JSON Server router

next();

});

server.post("/courses/", function(req, res, next) {

const error = validateCourse(req.body);

if (error) {

res.status(400).send(error);

} else {

req.body.slug = createSlug(req.body.title); // Generate a slug for new courses.

next();

}

});

// Use default router

server.use(router);

// Start server

const port = 3001;

server.listen(port, () => {

console.log(`JSON Server is running on port ${port}`);

});

// Centralized logic

// Returns a URL friendly slug

function createSlug(value) {

return value

.replace(/[^a-z0-9\_]+/gi, "-")

.replace(/^-|-$/g, "")

.toLowerCase();

}

function validateCourse(course) {

if (!course.title) return "Title is required.";

if (!course.authorId) return "Author is required.";

if (!course.category) return "Category is required.";

return "";

}

createMockDb.js

/\* eslint-disable no-console \*/

const fs = require("fs");

const path = require("path");

const mockData = require("./mockData");

const { courses, authors, employee } = mockData;

const data = JSON.stringify({ courses, authors, employee });

const filepath = path.join(\_\_dirname, "db.json");

fs.writeFile(filepath, data, function(err) {

err ? console.log(err) : console.log("Mock DB created.");

});

apiUtils.js

export async function handleResponse(response) {

if (response.ok) return response.json();

if (response.status === 400) {

// So, a server-side validation error occurred.

// Server side validation returns a string error message, so parse as text instead of json.

const error = await response.text();

throw new Error(error);

}

throw new Error("Network response was not ok.");

}

// In a real app, would likely call an error logging service.

export function handleError(error) {

// eslint-disable-next-line no-console

console.error("API call failed. " + error);

throw error;

}

**courseApi.js**

import { handleResponse, handleError } from "./apiUtils";

const baseUrl = process.env.API\_URL + "/courses/";

export function getCourses() {

return fetch(baseUrl)

.then(handleResponse)

.catch(handleError);

}

export function saveCourse(course) {

return fetch(baseUrl + (course.id || ""), {

method: course.id ? "PUT" : "POST", // POST for create, PUT to update when id already exists.

headers: { "content-type": "application/json" },

body: JSON.stringify(course)

})

.then(handleResponse)

.catch(handleError);

}

export function deleteCourse(courseId) {

return fetch(baseUrl + courseId, { method: "DELETE" })

.then(handleResponse)

.catch(handleError);

}

**configureStore.js**

import { createStore, applyMiddleware, compose } from "redux";

import rootReducer from "./reducers";

import reduxImmutableStateInvariant from "redux-immutable-state-invariant";

import thunk from "redux-thunk";

export default function configureStore(initialState) {

const composeEnhancers =

window.\_\_REDUX\_DEVTOOLS\_EXTENSION\_COMPOSE\_\_ || compose;

return createStore(

rootReducer,

initialState,

composeEnhancers(applyMiddleware(thunk, reduxImmutableStateInvariant()))

);

}

**courseAction.js**

import \* as types from "./actionTypes";

import \* as courseApi from "../../api/courseApi";

export const loadCourseSuccess = courses => {

return { type: types.LOAD\_COURSE\_SUCCESS, courses };

};

export const createCourseSuccess = newCourse => {

return { type: types.CREATE\_COURSE\_SUCCESS, course: newCourse };

};

export const updateCourseSuccess = course => {

return { type: types.UPDATE\_COURSE\_SUCCESS, course };

};

export const loadCourses = () => {

return dispatch => {

return courseApi

.getCourses()

.then(courses => {

dispatch(loadCourseSuccess(courses));

})

.catch(error => {

throw error;

});

};

};

export const saveCourse = course => {

return dispatch => {

return courseApi

.saveCourse(course)

.then(data => {

Object.keys(data).map(key => {

console.log(key + " : " + data[key]);

});

data.id

? dispatch(updateCourseSuccess(data))

: dispatch(createCourseSuccess(data));

})

.catch(error => {

console.log("error: " + error);

throw error;

});

};

};

**Index.js**

import React from "react";

import { render } from "react-dom";

import { BrowserRouter as Router } from "react-router-dom";

import "bootstrap/dist/css/bootstrap.min.css";

import App from "./components/App";

import "./index.css";

import configureStore from "./redux/configureStore";

import { Provider as ReduxProvider } from "react-redux";

const store = configureStore();

render(

<ReduxProvider store={store}>

<Router>

<App />

</Router>

</ReduxProvider>,

document.getElementById("app")

);

**Webpack.config.dev.js**

const webpack = require("webpack");

const path = require("path");

const HtmlWebpackPlugin = require("html-webpack-plugin");

process.env.NODE\_ENV = "development";

module.exports = {

mode: "development",

target: "web",

devtool: "cheap-module-source-map",

entry: "./src/index",

output: {

path: path.resolve(\_\_dirname, "bundle"),

publicPath: "/",

filename: "bundle.js"

},

devServer: {

stats: "minimal",

overlay: true,

historyApiFallback: true,

disableHostCheck: false,

https: false,

headers: { "Access-Control-Allow-Origin": "\*" }

},

plugins: [

new webpack.DefinePlugin({

"process.env.API\_URL": JSON.stringify("http://localhost:3001")

}),

new HtmlWebpackPlugin({

template: "./src/index.html",

favicon: "./src/favicon.ico"

})

],

module: {

rules: [

{

test: /\.(js|jsx)$/,

exclude: /node\_modules/,

use: ["babel-loader", "eslint-loader"]

},

{

test: /(\.css)$/,

use: ["style-loader", "css-loader"]

}

]

}

};

Package.json

{

"name": "course\_dashboard",

"version": "1.0.0",

"description": "npm init",

"main": "index.js",

"scripts": {

"start": "run-p start:dev start:api",

"start:dev": "webpack-dev-server --config webpack.config.dev.js --port 3000",

"prestart:api": "node tools/createMockDb.js",

"start:api": "node tools/apiServer.js"

},

"repository": {

"type": "git",

"url": "git+https://github.com/RagaviVinod/course\_dashboard.git"

},

"author": "",

"license": "ISC",

"bugs": {

"url": "https://github.com/RagaviVinod/course\_dashboard/issues"

},

"homepage": "https://github.com/RagaviVinod/course\_dashboard#readme",

"dependencies": {

"bootstrap": "4.3.1",

"prop-types": "15.7.2",

"react": "16.8.4",

"react-datepicker": "^2.8.0",

"react-dom": "16.8.4",

"react-redux": "6.0.1",

"react-router-dom": "5.0.0",

"redux": "4.0.1",

"redux-thunk": "2.3.0"

},

"devDependencies": {

"@babel/core": "7.3.4",

"babel-loader": "8.0.5",

"babel-preset-react-app": "7.0.2",

"css-loader": "2.1.1",

"eslint": "5.15.2",

"eslint-loader": "2.1.2",

"eslint-plugin-import": "2.16.0",

"eslint-plugin-react": "7.12.4",

"babel-eslint": "10.0.1",

"style-loader": "0.23.1",

"webpack": "4.29.6",

"webpack-dev-server": "3.2.1",

"webpack-cli": "3.3.0",

"redux-immutable-state-invariant": "2.1.0",

"redux-mock-store": "1.5.3",

"fetch-mock": "7.3.1",

"json-server": "0.14.2",

"http-server": "0.11.1",

"npm-run-all": "4.1.5",

"node-fetch": "^2.3.0",

"rimraf": "2.6.3",

"html-webpack-plugin": "3.2.0",

"postcss-loader": "3.0.0",

"mini-css-extract-plugin": "0.5.0"

},

"babel": {

"presets": [

"babel-preset-react-app"

]

},

"eslintConfig": {

"extends": [

"eslint:recommended",

"plugin:react/recommended",

"plugin:import/errors",

"plugin:import/warnings"

],

"parser": "babel-eslint",

"parserOptions": {

"ecmaVersion": 2018,

"sourceType": "module",

"ecmaFeatures": {

"jsx": true

}

},

"env": {

"browser": true,

"node": true,

"es6": true,

"jest": true

},

"rules": {

"no-debugger": "off",

"no-console": "off",

"no-unused-vars": "warn",

"react/prop-types": "warn"

},

"settings": {

"react": {

"version": "detect"

}

},

"root": true

}

}

App.js

import React from "react";

import { Route, Switch } from "react-router-dom";

import HomePage from "./home/HomePage";

import AboutPage from "./about/AboutPage";

import Header from "./common/Header";

import PageNotFound from "./PageNotFound";

import CoursesPageContainer from "./courses/CoursesPageContainer";

import ManageCoursePageContainer from "./courses/ManageCoursePageContainer";

import EmployeeFormPageContainer from "./employee/EmployeeFormPageContainer";

function App() {

return (

<div className="container-fluid">

<Header />

<Switch>

<Route exact path="/" component={HomePage} />

<Route path="/about" component={AboutPage} />

<Route path="/courses" component={CoursesPageContainer} />

<Route path="/course/:slug" component={ManageCoursePageContainer} />

<Route path="/course" component={ManageCoursePageContainer} />

<Route path="/employee" component={EmployeeFormPageContainer} />

<Route component={PageNotFound} />

</Switch>

</div>

);

}

export default App;

EmployeeFormPage.js

import React from "react";

import { CheckBox } from "./items/CheckBox";

import { DateSelector } from "./items/DateSelector";

import { TextField } from "./items/TextField";

import { RadioButton } from "./items/RadioButton";

export class EmployeeFormPage extends React.Component {

state = {

employee: {

name: "",

dateOfBirth: new Date(),

gender: "",

personalDetails: []

}

};

handleChange = event => {

const name = event.target.name;

const value = event.target.value;

this.setState({ employee: { ...this.state.employee, name: value } });

};

handleSave = event => {

event.preventDefault();

this.props

.saveEmployee(this.state.employee)

.then(() => {

history.push("/employee");

})

.catch(() => {

history.push("/abc");

});

};

handleDateChange = date => {

this.setState({

employee: {

...this.state.employee,

dateOfBirth: date.toLocaleDateString()

}

});

};

handleRadioButtonClick = event => {

this.setState({

employee: { ...this.state.employee, gender: event.target.value }

});

};

handleCheckBoxClick = event => {

this.setState({

...this.state.employee,

personalDetails: [

this.state.employee.personalDetails.push(event.target.value)

]

});

};

render() {

const genders = ["Male", "Female", "Other"];

const info = [

"married",

"divorced",

"student",

"certified developer",

"beginner",

"architect"

];

return (

<form onSubmit={this.handleSave}>

<TextField

name="name"

value={this.state.employee.name}

onChange={this.handleChange}

label="Employee's Name "

/>

<DateSelector

name="dateOfBirth"

selectedDate={this.state.employee.dateOfBirth}

onChange={this.handleDateChange}

label="Date of birth "

/>

<RadioButton

name="gender"

label={genders}

onClick={this.handleRadioButtonClick}

/>

<CheckBox label={info} onClick={this.handleCheckBoxClick} />

<button type="submit" className="btn btn-primary">

Save

</button>

</form>

);

}

}

**ManageCoursePage.js**

import React, { useEffect, useState } from "react";

import { CourseForm } from "./CourseForm";

//import \* as courseActions from "../../redux/actions/courseActions";

//import \* as authorActions from "../../redux/actions/authorActions";

import PropTypes from "prop-types";

export const ManageCoursePage = ({

courses,

authors,

loadAuthors,

loadCourses,

saveCourse,

history,

ownProps,

...props

}) => {

const newCourse = {

id: null,

title: "",

authorId: null,

category: ""

};

const slug = ownProps.match.params.slug;

const getCourseSlug = (courses, slug) => {

return courses.find(course => {

return course.slug === slug;

});

};

const selCourse =

slug && courses.length > 0 ? getCourseSlug(courses, slug) : newCourse;

const [course, setCourse] = useState(selCourse);

useEffect(

() => {

if (courses.length === 0) {

loadCourses().catch(error => {

alert("Loading courses failed" + error);

});

} else {

setCourse(selCourse);

}

if (authors.length === 0) {

loadAuthors().catch(error => {

alert("Loading authors failed" + error);

});

}

},

[selCourse]

);

const handleChange = event => {

const { name, value } = event.target;

setCourse(prevState => ({

...prevState,

[name]: name === "authorId" ? parseInt(value, 10) : value

}));

};

const handleSave = event => {

event.preventDefault();

saveCourse(course)

.then(() => {

history.push("/courses");

})

.catch(() => {

history.push("/abc");

});

};

return (

<div>

<h2>Manage Course Page</h2>

<CourseForm

onChange={handleChange}

course={course}

authors={authors}

onSave={handleSave}

/>

</div>

);

};

ManageCoursePage.propTypes = {

authors: PropTypes.array.isRequired,

courses: PropTypes.array.isRequired,

loadCourses: PropTypes.func.isRequired,

loadAuthors: PropTypes.func.isRequired,

history: PropTypes.object.isRequired,

saveCourse: PropTypes.func.isRequired

};