Chapter 12 - Redux

Contents

What is Redux?

Redux is a State management tool for UI applications most commonly used in library such as React JS and frameworks such as angular and view etc.

What are the Advantages and Disadvantages of Redux?

Pros -

* Redux provides centralised state management system. In React state/data is stored locally within a component. To share this state with other components props are passed down from parent to child components and call backs are used from child to parent component. On the other hand, Redux state is stored globally in a central place called Redux Store which centralises all the data and makes them available across the components which need these data.

* Redux prevents Props drilling just like React Context API does.

* Redux is preferable over context API / React Context in large scale application

* Redux provides Performance Optimisations. In react whenever the state or prop gets changed the component tree using the state or prop gets rerendered. But in Redux when the data is changed inside store, a shallow copy of that data is created as a result rerender is less likely.

Cons -

* A huge learning curve for new comers

* Not suitable for small applications.

* Configuring Redux store is too much complex.

* Redux requires lots of boiler plate code making the code messy and unreadable.

* Debugging is difficult.

What is Redux Tool Kit (RTK)?

RTK enables us to write redux code in a concise way.

RTK abstracts the basic redux code preventing unnecessary boiler plate code which enables developer like us to write clean concise redux code.

What is Redux Store?

Redux store is a big JS object.

Redux store is basically a central place to store data needed by any components in react app.

We can have multiple react context to store multiple data. But we have a single redux store for global data management.

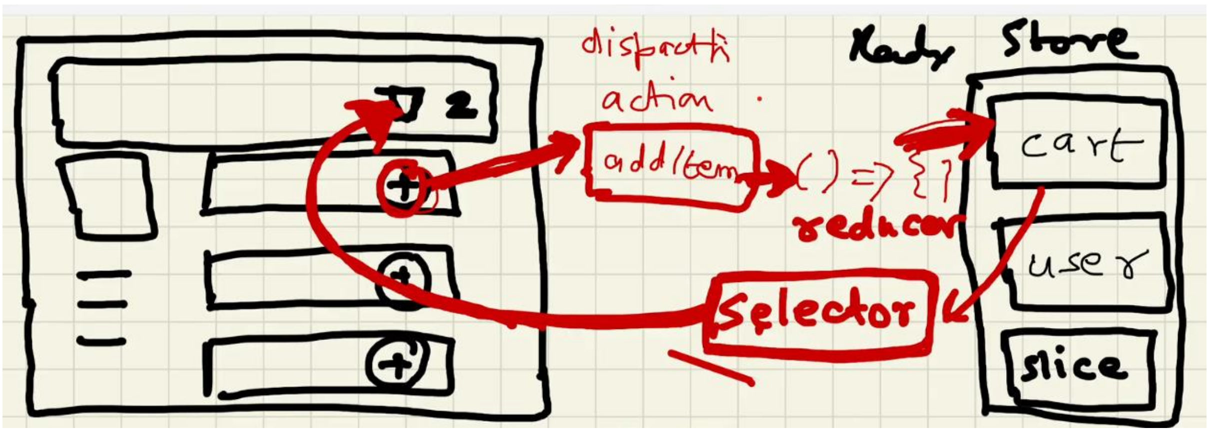
What are Slices?

Slices are logical separations or portions within the Redux Store.

* Each slice maintains a mapping between actions and reducer functions so that when the action is dispatched reducer of that action gets called. In short slices not only stores data portion wise but also it manages/updates these data.

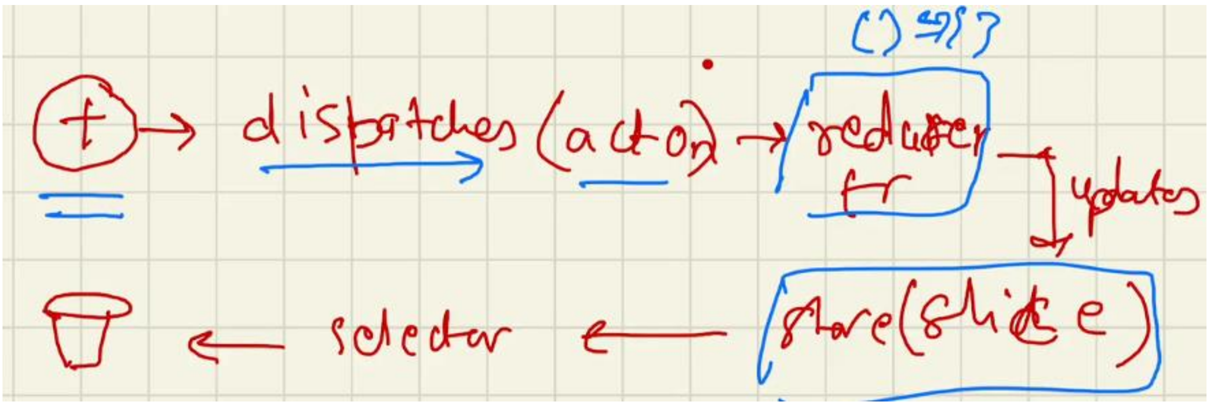
* In an ecommerce application we can have various slices inside the store. User slice to handle data associated with the logged in user, theme slices to handle light mode dark mode data related to the application UI, cart slice to handle data associated with cart.

Explain Redux Toolkit Architecture?



The way E commerce application works is, in E-cart system we can add items to the cart and then proceed for a check out.

If we consider this scenario in redux, when we click on plus (+) button in the UI we dispatch an action addItem and the action in turn calls a reducer function which updates the cart slice of the store. And If we want to read our cart slice data and updates the cart items count in UI, we use a selector. Technically we subscribe our store with our application with the help of a selector, meaning our cart items count in UI will get notified and updated when cart slice data is changed. This selector is a hook known as useSelector.The selector is named as Selector because we are selecting the slice out of the store, we are selecting a portion of the store.



Note: When we click on plus button we can't directly update the store why because we don't want random components randomly modify our store. Redux toolkit helps managing the state / data of react application in a predictable and organised way. To make sure everything stays organised and consistent we can't just change the state whenever we want. To change the state, we have to use specific functions and actions. This way redux can keep track of the changes.

How to setup Redux in a React App?

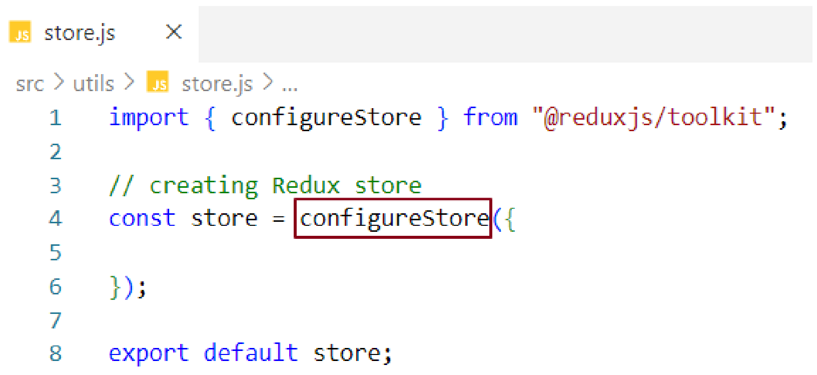
We install two libraries.

npm i @reduxjs/toolkit - This library provides functions using which we can create Redux store, Slices. This library provides the core functionality of Redux.

npm i react-redux - This library acts as a bridge between React app and Redux.

How to Create a Redux-Store?

RTK (Redux Tool Kit) provides an API configureStore which creates Redux store.

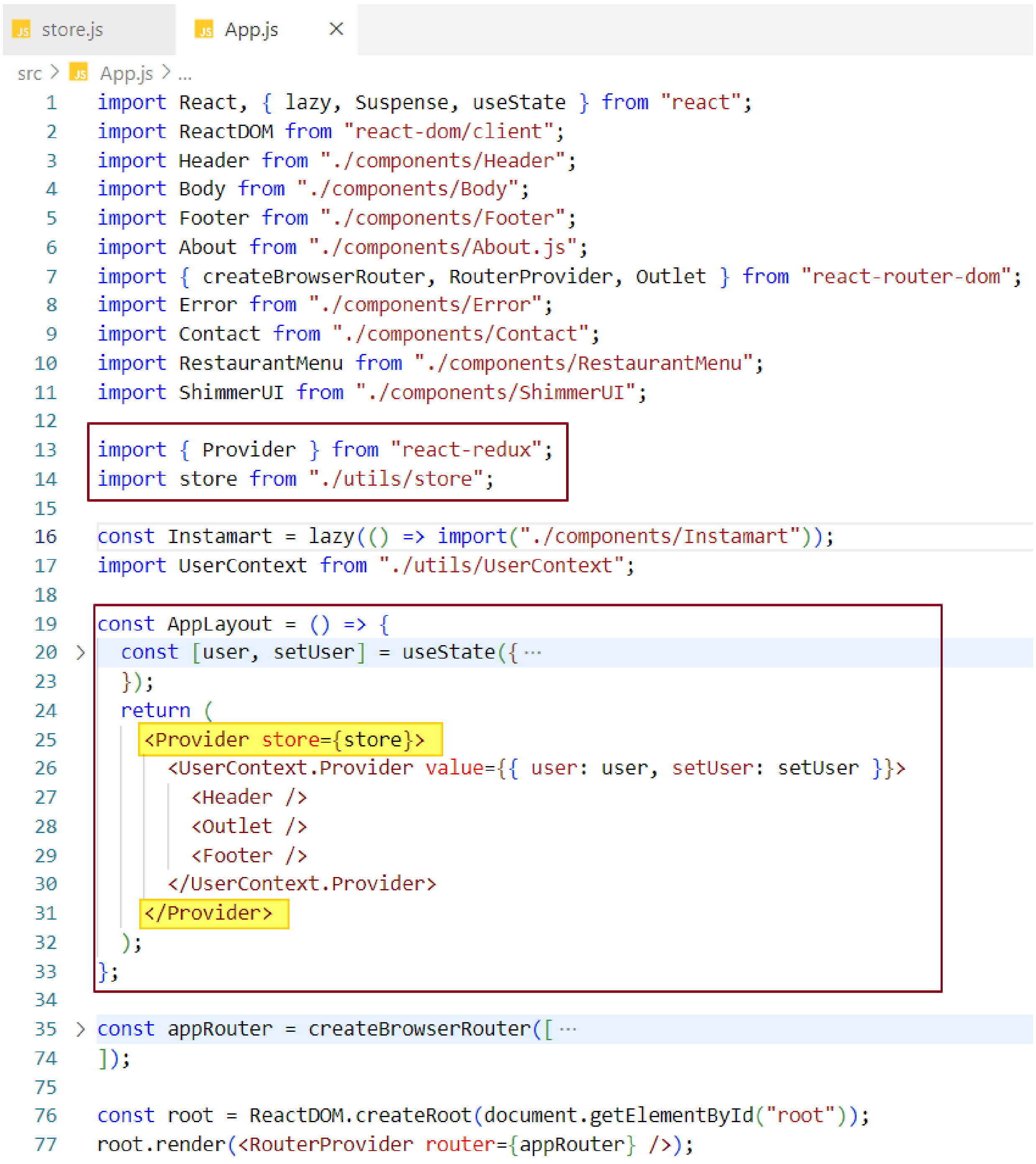


```javascript
import { configureStore } from "@reduxjs/toolkit";

// creating Redux store
const store = configureStore({

});

export default store;
```

Our store and application are separated entities.

How do I provide Redux store to our application?

React-redux library provides a component Provider which acts as a bridge between redux store and react application. To establish this bridge, we wrap the application root components within the Provider component and to make the store available to our application we pass store as a prop inside Provider Component.

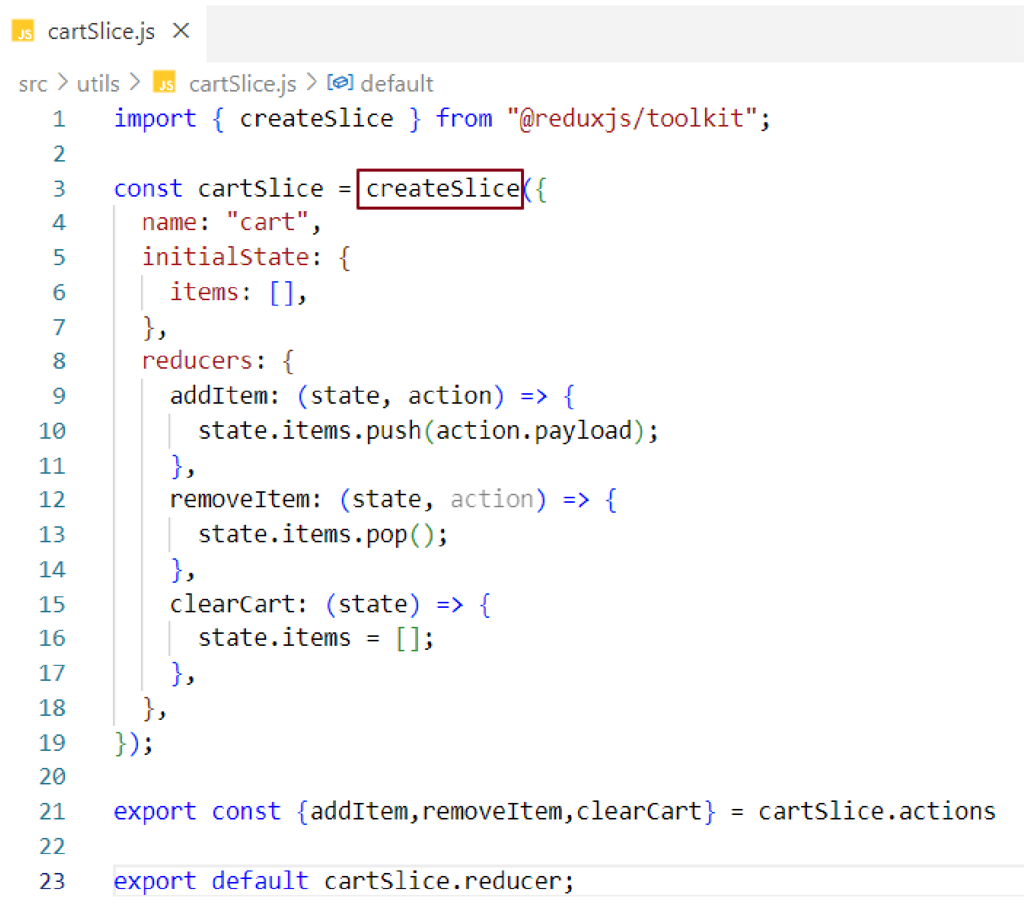

store.js   App.js

src > App.js > ...

```
import React, { lazy, Suspense, useState } from "react";
import ReactDOM from "react-dom/client";
import Header from "./components/Header";
import Body from "./components/Body";
import Footer from "./components/Footer";
import About from "./components/About.js";
import { createBrowserRouter, RouterProvider, Outlet } from "react-router-dom";
import Error from "./components/Error";
import Contact from "./components/Contact";
import RestaurantMenu from "./components/RestaurantMenu";
import ShimmerUI from "./components/ShimmerUI";

import { Provider } from "react-redux";
import store from "./utils/store";

const Instamart = lazy(() => import("./components/Instamart"));
import UserContext from "./utils/UserContext";

const AppLayout = () => {
  const [user, setUser] = useState({ ···
  });
  return (
    <Provider store={store}>
      <UserContext.Provider value={{ user: user, setUser: setUser }}>
        <Header />
        <Outlet />
        <Footer />
      </UserContext.Provider>
    </Provider>
  );
};

const appRouter = createBrowserRouter([ ···
]);

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(<RouterProvider router={appRouter} />);
```

How to create Slices inside Redux Store?

RTK provides an API createSlice which creates Slices for us. We are creating a cart slice in below example.



cartSlice.js ×

src > utils > cartSlice.js > default

```
1   import { createSlice } from "@reduxjs/toolkit";
2
3   const cartSlice = createSlice({
4     name: "cart",
5     initialState: {
6       items: [],
7     },
8     reducers: {
9       addItem: (state, action) => {
10        state.items.push(action.payload);
11      },
12      removeItem: (state, action) => {
13        state.items.pop();
14      },
15      clearCart: (state) => {
16        state.items = [];
17      },
18    },
19  });
20
21  export const {addItem,removeItem,clearCart} = cartSlice.actions
22
23  export default cartSlice.reducer;
```

name represents name of the slice which is cart.

initialState represents an object where we set initial state of the slice. In our case we have set items to an empty array because initially our cart will have zero items in it.

reducers represent an object where we map actions with reducer functions

cartSlice.reducer from line number 23 represents an object which wraps up all the reducer functions and export them as in one big reducer object.

In line number 21 we are destructuring all the actions from cartSlice.actions and exporting them at once.

Internal representation of slice object by redux developers **=>**

```
sliceObj = {
  actions: {
    action1,
    action2,
    action3,
    // ..
    // ..
  },
  reducer: reducerFunctionList,
};
```

what are reducer functions?

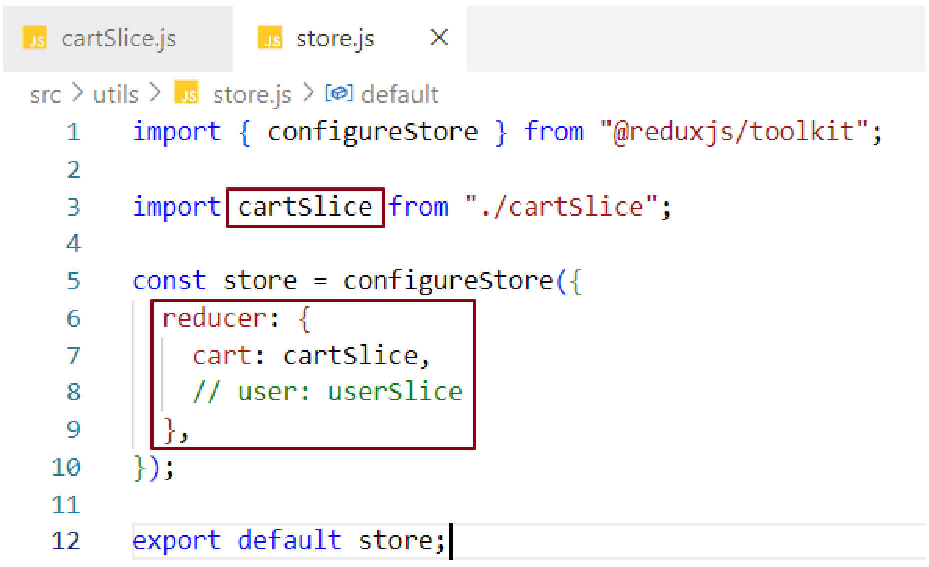
Reducer functions are normal JavaScript function which get called when associated action is dispatched. In reducer function we provide state management logic which updates the state based on the triggered action.

Reducer function takes in two parameters state and action where state is the current state and action is a plain JavaScript object having payload received from UI. When no action is dispatched state variable will have initial state of the cart slice (items = []) and when action is triggered state (items) will be updated

How do we add Slices to Redux Store?

configureStore receives an object where we add and configure all our slices. How?

The object has a key named reducer with a value a new object which contains mapping between slice name and the actual imported slice. For different slices, we add their configuration as a key value pair inside reducer property.



```
import { configureStore } from "@reduxjs/toolkit";

import cartSlice from "./cartSlice";

const store = configureStore({
  reducer: {
    cart: cartSlice,
    // user: userSlice
  },
});

export default store;
```

What is Redux Dev Tool?

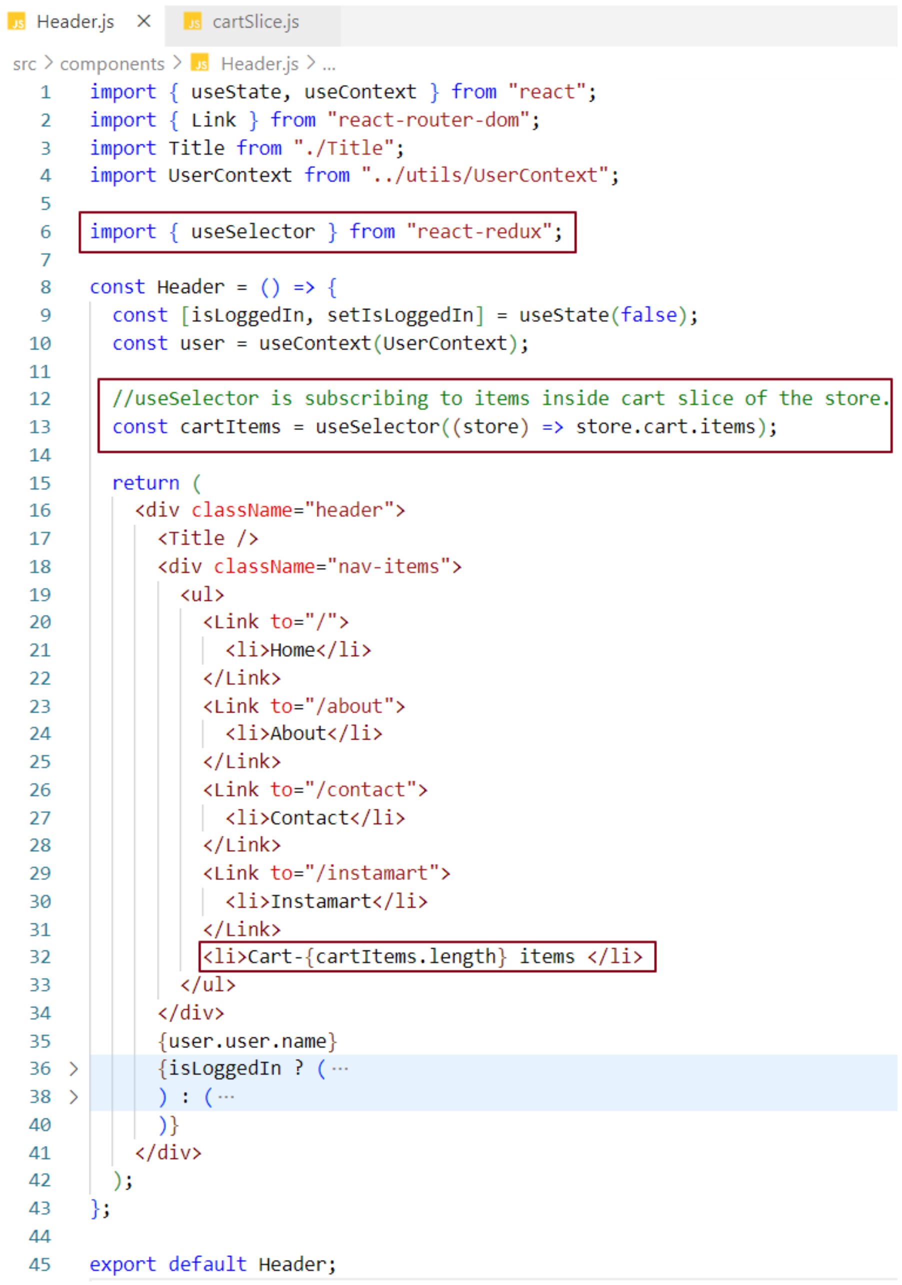
Redux Dev Tool lets us inspect every state and action payload.

Redux Dev Tools video guide:

https://www.youtube.com/watch?v=BYpuigD01Ew

How do React application subscribe to Redux Store?

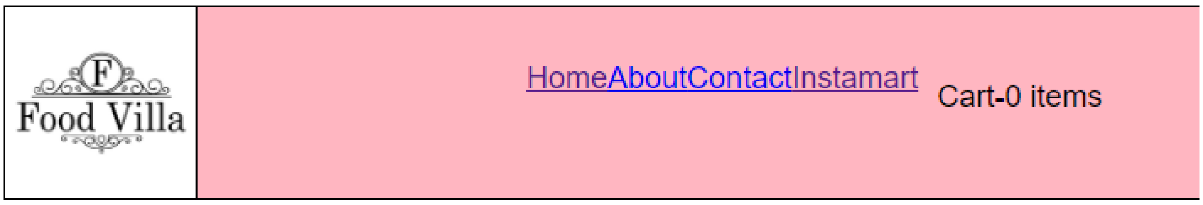
React application subscribe to the redux store using useSelector hook. useSelector comes from react-redux library because useSelector acts as a bridge between React and Redux store.



```
1   import { useState, useContext } from "react";
2   import { Link } from "react-router-dom";
3   import Title from "./Title";
4   import UserContext from "../utils/UserContext";
5
6   import { useSelector } from "react-redux";
7
8   const Header = () => {
9     const [isLoggedIn, setIsLoggedIn] = useState(false);
10    const user = useContext(UserContext);
11
12    //useSelector is subscribing to items inside cart slice of the store.
13    const cartItems = useSelector((store) => store.cart.items);
14
15    return (
16      <div className="header">
17        <Title />
18        <div className="nav-items">
19          <ul>
20            <Link to="/">
21              <li>Home</li>
22            </Link>
23            <Link to="/about">
24              <li>About</li>
25            </Link>
26            <Link to="/contact">
27              <li>Contact</li>
28            </Link>
29            <Link to="/instamart">
30              <li>Instamart</li>
31            </Link>
32            <li>Cart-{cartItems.length} items </li>
33          </ul>
34        </div>
35        {user.user.name}
36        {isLoggedIn ? ( ···
38        ) : ( ···
40        )}
41      </div>
42    );
43  };
44
45  export default Header;
```

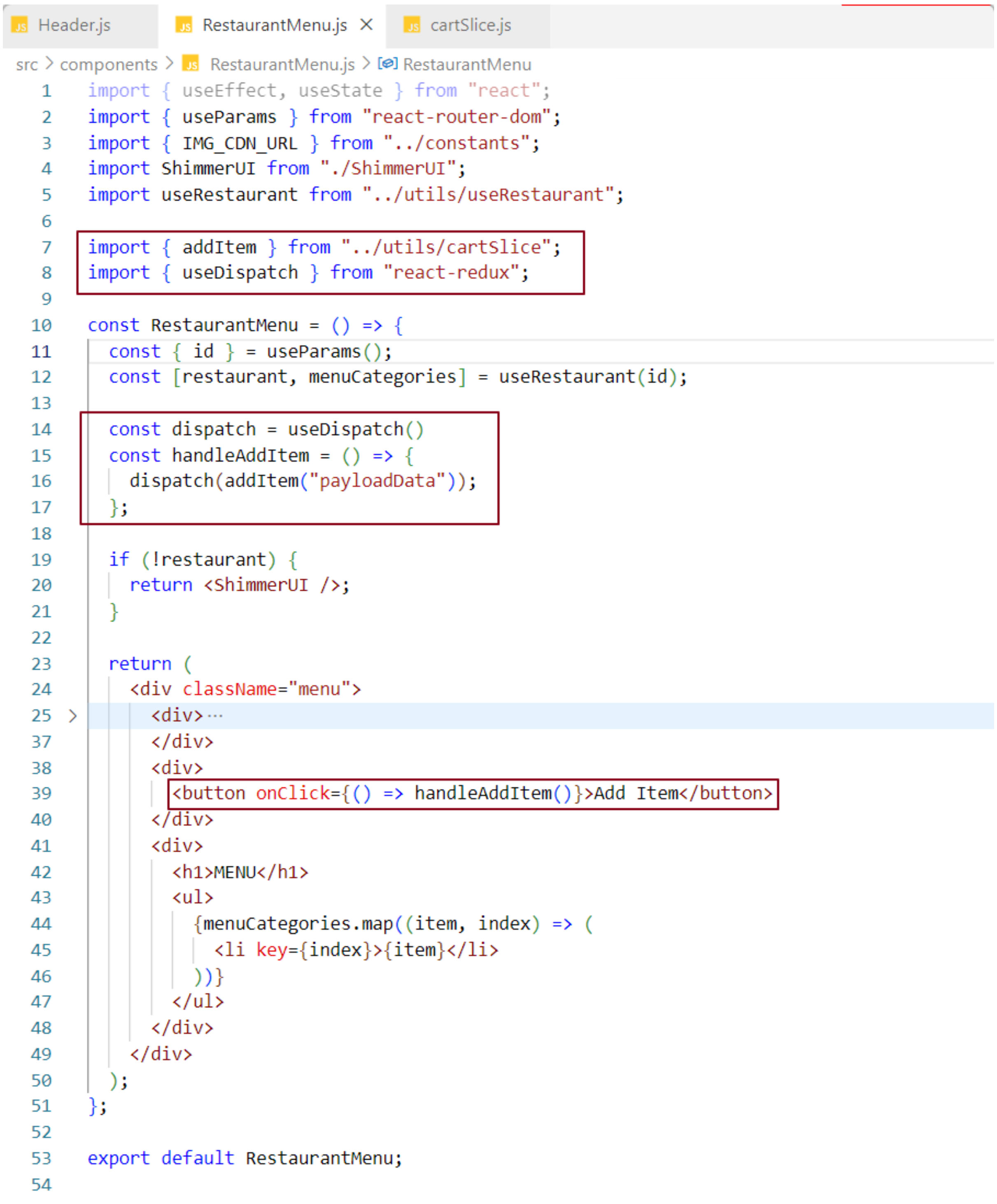
At line number 13 we are subscribing our react application to cartSlice items of the store.

At line number 32 we are using the subscription data and displaying on the UI. Currently in redux store cart slice has zero items Thus we expected 0 to be shown in the UI.



How do we dispatch an action?

React-redux library provides a hook called useDispatch that returns a function and the function takes an action with an action payload. This function is responsible for dispatching the action.



```javascript
import { useEffect, useState } from "react";
import { useParams } from "react-router-dom";
import { IMG_CDN_URL } from "../constants";
import ShimmerUI from "./ShimmerUI";
import useRestaurant from "../utils/useRestaurant";

import { addItem } from "../utils/cartSlice";
import { useDispatch } from "react-redux";

const RestaurantMenu = () => {
  const { id } = useParams();
  const [restaurant, menuCategories] = useRestaurant(id);

  const dispatch = useDispatch()
  const handleAddItem = () => {
    dispatch(addItem("payloadData"));
  };

  if (!restaurant) {
    return <ShimmerUI />;
  }

  return (
    <div className="menu">
      <div>···
      </div>
      <div>
        <button onClick={() => handleAddItem()}>Add Item</button>
      </div>
      <div>
        <h1>MENU</h1>
        <ul>
          {menuCategories.map((item, index) => (
            <li key={index}>{item}</li>
          ))}
        </ul>
      </div>
    </div>
  );
};

export default RestaurantMenu;
```

Code Explanation -

In line 7 we have imported addItem action which is exported by cartSlice Slice

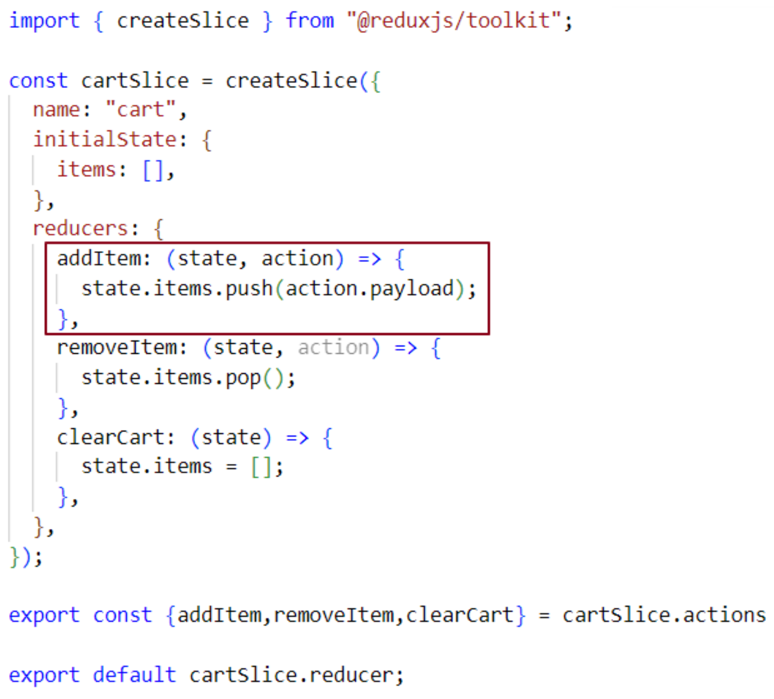
In line 14, we have used useDispatch hook which returns a function named dispatch.

In line 15, handleAddItem function is invoked when we click on Add Item button at line 39

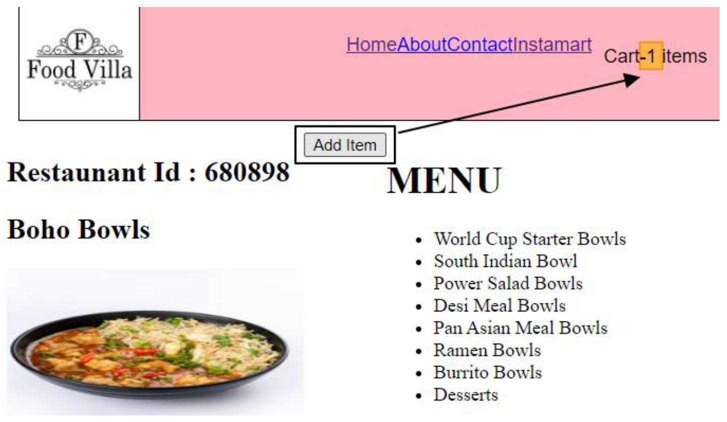
handleAddItem function calls dispatch function which dispatched addItem action with the PayloadData. This PayloadData will be received by addItem action which passes it into the mapped reducer function. This reducer function updates the items in the store

As the application is subscribed to card slice, as soon as the items gets updated in the store, our items count section in the UI will be updated.

In our case when we click on Add button, cart slice items will be filled with PayLoad data received from UI and items count will be updated. Meanwhile useSelector hook reads this updated value and displays it in the items count section in UI.



```
import { createSlice } from "@reduxjs/toolkit";

const cartSlice = createSlice({
  name: "cart",
  initialState: {
    items: [],
  },
  reducers: {
    addItem: (state, action) => {
      state.items.push(action.payload);
    },
    removeItem: (state, action) => {
      state.items.pop();
    },
    clearCart: (state) => {
      state.items = [];
    },
  },
});

export const {addItem,removeItem,clearCart} = cartSlice.actions

export default cartSlice.reducer;
```



Restaurant Id : 680898

MENU

Boho Bowls

- World Cup Starter Bowls
- South Indian Bowl
- Power Salad Bowls
- Desi Meal Bowls
- Pan Asian Meal Bowls
- Ramen Bowls
- Burrito Bowls
- Desserts

When we click on Add Item button the count in header section got updated to 1 from 0. And it keeps on incrementing as many time as we click on Add item button.