Date.....

Chapter-14 Let's Build our Store

Today we will see a better way of handling data for our application. If we are building a big app, handling data is most crucial/important thing and for that many company uses a library called **Redux**.

why do we use Redux?
→ we use redux to manage our data layer of application.

alternative of redux
→ MobX
→ RxJS

cons of redux
→ It is complex to setup
→ It has a huge learning curve (you need to learn a lot of complexe thing)
→ there's a lot of code to copy-paste.

Use redux only for large scale application where you need a lot of data handling., for small application you don't need redux.

→ Redux toolkit
→ we will be using redux toolkit instead of redux because :-
→ The Redux Toolkit package is intended to be the standard way to write Redux logic. It was originally created to help address three common concerns about Redux:
- "Configuring a Redux store is too complicated".
- "I have to add a lot of packages to get Redux to do anything useful"
- "Redux requires too much boilerplate code".

Teacher's Sign.....

Date.....

→ Today we will build our cart page (to add items/remove item)

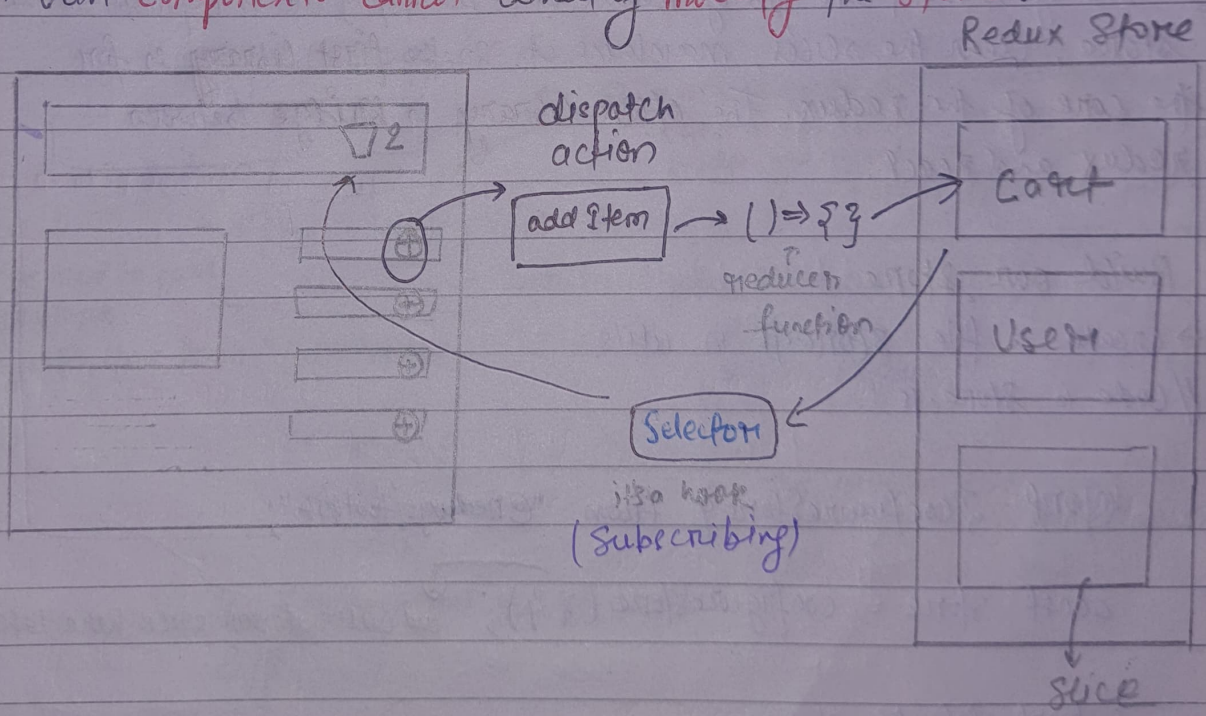
→ At the end of the day redux store is like big all object, which have different sections. and all components can access it.

→ Redux store is separate entity out of the application, my web app is different and my store is different.

→ In redux we have single thing to hold everything (only 1 Redux Store)

→ we have different sections of the store where we logically separate different data that is called slices of our store.
eg→ user slice, theme slice, cart slice. etc.

→ Our components cannot directly modify the store.



If we click on the (+) button it will dispatch an action that will calls the reducer function which updates the slice of our redux store.
we use selector to read data, and selector is a hook at the end of the day

Teacher's Sign .....

→ When I say our component has subscribed to the store it means it is reading from the store. basically it means it is synced to the store., it means whenever my store will modify my cart will automatically modify. (my UI)

→ Let's see everything in code.

① Install redux-toolkit library

→ npm i @reduxjs/toolkit

→ npm i react-redux

Q why two library?

ans:- Job of redux, core job of redux is to manage the store/maintain the store, clear the slices, maintain it. So the first library is for the core of the redux. The other library is bridge between redux and react.

② Build our store

→ create a file store.js in utils.

// code in Store.js

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

const store = configureStore({}); // we will put our slice here later

export default store;
```

Date.....

→ Now this store is different and our app is different, now I have to provide my store to my app. I need to tell that this is our store.

(3) providing our store to my app

→ Now I could provide this store to some part of my app or the whole application.

→ So I here I will provide my store to the whole app, for that I will go to my root component.

// in App.js

```
const AppLayout = () => {
  const [user, setUser] = useState({
    name: "Akshay Saini",
    email: "support@namastedev.com"
  });

  return (
    <Provider store={store}>
      <UserContext.Provider value={{ user: user, setUser: setUser }}>
        <Header />
        <Outlet />
        <Footer />
      </UserContext.Provider>
    </Provider>
  );
};
```

① import 'Provider' from 'react-redux' as this is our second library, used to bridge b/w react-redux

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

② we need to pass our store

```
import store from "./utils/store"
```

③ this name is very important react will be using this name to refer our store.

MacBook Air

Date.....

Now before to starting our action (to write action/dispatch, reducers) let's first ~~writing~~ fill our store with something

④ fill our store with slice

4.a → Creating slice

// create a file in utils cartSlice.js

// code in cartSlice.js

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

const cartSlice = createSlice({

  name: 'cart',                    → name of our slice
  initialState: {                  → initial value of our slice (cart)
    items: []
  },
  reducers: {                      → ① reducers are called on dispatch of an action
                                   This is a place where I will tell what action will call what reducer fun^n (mapping b/w action & reducer fun^n)
    addItem: (state, action) => {  ② This is the function that will be called when this action dispatches
                                   ③ This state is previous state. This is data which is coming in
      state.items.push(action.payload);   ④ logic to update the slice/state   ⑤ data is coming when button is clicked.
    },

    clearCart: (state) => {
      // state = [];
      state.items = [];
    },

    removeItem: (state, action) => {
      state.items.pop();           → write proper logic: find that item and remove it
    },                             Here I am just removing the last item.
  }
});
```

reducer

NOTE: this function doesn't return anything. It ~~just update~~ takes a state and directly modifies it.

Spiral

Teacher's Sign.....

Date.....

→ my component need to access the slice so. I need to export actions and reducers from the slice.

write this first maybe !!!

export default cartSlice.reducer; → this is reducer not reducers (combines all reducers and exports)

export const {addItems, removeItem, clearCart} = cartSlice.actions;

back of the scene kya chal raha h?

ye pura cartSlice ye return kar raha hoga.

```
cartSlice = {
    actions : {
        addItem,
        removeItem,
        clearCart.
    }
    reducer : reducers
}
```

4.6 → putting slice into our store

```
// code in store.js
import {configureStore} from "@reduxjs/toolkit";
import cartSlice from "./cartSlice";

const store = configureStore({
    reducer: {
        cart: cartSlice,
    },
});
export default store;
```

we are passing all reducers
① name of the slice → slice

Spiral

Teacher's Sign .....

Date.....

Conclusion till now

Create Store
- configureStore() imported from RTK

provide my store to app
- <Provider store={store} imported from react-redux

Slice -
- createSlice({ imported from RTK
  name: "",
  initialState:
  reducers: {
    addItem: (state, action) => {state = action.payload}
  }
})

export const {addItem, removeItem} = cartSlice.actions;
export default cartSlice.reducer;

put that slice into store
- {
  reducer: {
    cart: cartSlice,
    user: userSlice,
  }
}

Date.....

⑤ Now let's subscribe to our store.

// ~~code~~ Now let us print my total cart ~~value~~ length/items in header.
// write in my header.

```
import { useSelector } from "react-redux";
.
.
.
const cartItems = useSelector (store => store.cart.items);
.
.
.
<h2> Cart - {cartItems.length} items </h2>
```

this store could have millions of slice but I am concern about my cart slice and in that as well only items.

⑥ Now how will I dispatch my action
I will add a button to my menu list to add dish to my cart
Now how will I dispatch an action
// code to dispatch an action

```
const handleAddItem = () => {
    dispatch (addItem ("Grapes"));
};
```

action name

dispatch an action with the payload

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

Now in our component

```
const RestaurantMenu = () => {
    const dispatch = useDispatch ();
    const handleAddItem = () => {
        .
        .
        .
    };
```

Spiral

Teacher's Sign .....

Date.....

// code to activate my button (so that when I click there, it add my dish to the cart)

// in RestaurantMenu.js

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

Const RestaurantMenu = () => {

  const dispatch = useDispatch();
  const addFoodItem = (item) => {
      dispatch(addItem(item));
  };

  return (

    <button
      onClick = {() => addFoodItem(item)}>
      Add
    </button>

  );
};
```

⑦ Lets build my cart page

// create a new component cart.js (add routes accordingly)

Tell this to your interviewer (even if he doesn't as you

Date.....

// code in card.js

import { useSelector } from "react-redux";

const Cart = () => {

const cartItems = useSelector(store => store.cart.items)

return(

<h1> Cart Items - {cartItems.length} </h1>

);

};

THIS is A MOTOR PERFORMANCE IMPROVEMENT

(1) this is is where most people make performace mistake. what they do is the subscribe to store

(store => store)

OR

(store => store.cart)

(2) so every time my store changes it will re-render the component.

Install Redux DevTools (extension)