Redux or zustand

Ep. 12th

→ A predictable state container for JS Apps.

- ↳ Redux is not Madatory.
- ↳ Redux is not apart of React
- ↳ It makes our app easy to debug.

Vanila Redux → old way
vs. Redux Toolkit → Standard way of writting Redux.
→ new way

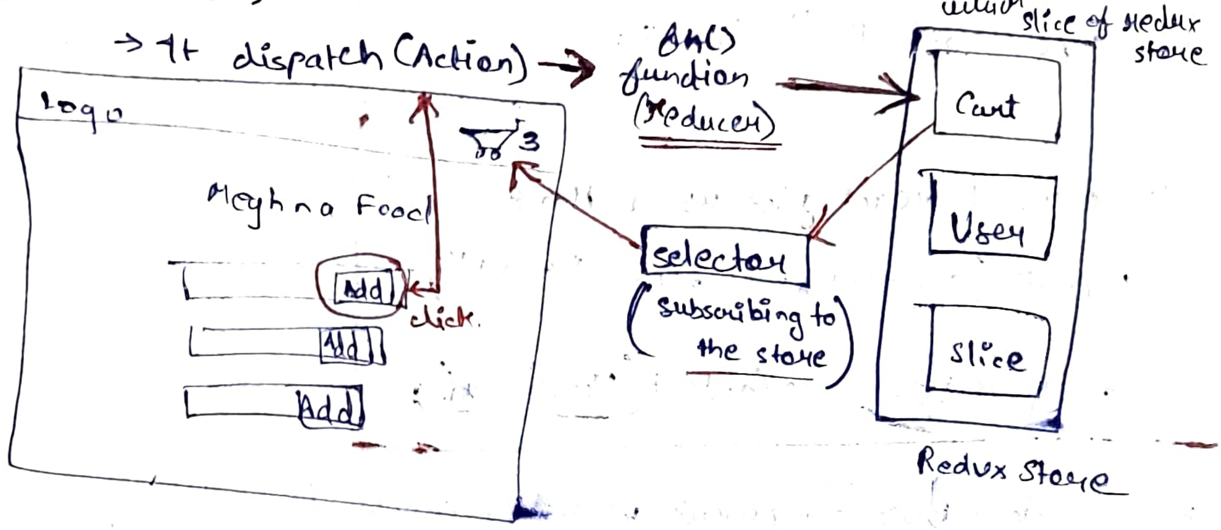
Adding cart feature using Redux

Redux is a big JS object kept in a centraal space.

→ to make partition we have slices → ex cart
ex user

To Add items ↓

→ It dispatch (Action) →



- Cart Component is subscribed to the store.

1. Install a) @redux js/toolkit
   and b) react-redux
2. Built our store
3. connect our store to our app
4. slice (cartSlice)
5. dispatch (action)
6. selector

② creating a store in utils/appStore. js

```
import {configureStore} from "@reduxjs/toolkit";
import cartReducer from "./cartSlice"
const appStore = configuredStore({
    reducer: {
        cart : cartReducer,
    },
});
export default appStore;
```

reducer of main store →

o connecti
① in Ap
import
return
< [
< /P
)
Creatin
→
Con
exporting actions }) →
exporting

- connecting Redux to react store

(1) in App.js

```
import {Provider} from "react-redux"
```

store name created in redux file

```
return (
    <Provider store ={appStore}>
        <div className = "app">



    </Provider>
)
```

Creating a Slice *

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

const cartSlice = createSlice({
        name : 'cart',
        initialState : {
            items : []          ← initially cart empty
        },

        reducers : {
            addItem : (state, action) => {
                state.items.push(action.payload);
            },
            removeItem: (state, action) => {
                state.items.pop();
            },
            clearCart : (state, action) => {
                state.items.length = 0;     ← making cart empty
            },
        },
})
```

we are mutating state here

actions

exporting actions →

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

exporting reducer →

```
export default cartSlice.reducer;
```

Subscribing to our Cart [Using Selector]

↳ It is a hook'

in Header.js

```
import {useSelector} from "read-redux"

const cart = useSelector((store) => store.cart.items);
```

↳ Now the cart will get access to cart items

Using dispatch action

~~In~~ItemList.js

```
import {addItem} from ..utils/cartSlice;
① import {useDispatch} from "react-redux";
② const dispatch = useDispatch();
```

~~<button add Items~~

~~on~~

```
const handleAddItems = (item)=>{
    dispatch(addItem(item));
};

<button
    onClick={()=> handleAddItem(item)}
..>
```

Read More

① onClick= {handle AddItem}
↳ just calling the function

② on Click ={()=> handle AddItem(Item)}
↳ call back function and passing ~~a~~ item

③ onclick= {handleAddItem (Item)}
↳ calling the function right away

Creating a Cart Page

in Component.js

```
const Cart = () ={
    return(

    );
export default Cart;
```

① ~~add~~ in app.js

import Cart from "/component/Cart"
add a path

```
{ path: "/card",
  element: <Cart/>,
}
```

② Create a Link on Cart

```
<Link to = "/cart"> . . . </Link>
```

~~#cart component~~

~~<div className="text-center m-4, p-4">~~

~~<h1> Cart </h1>~~

Reading to cart by Subscribing it

```
import ItemList from "./ItemList"
import {useSlector} from "react-redux"

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

return (
   <div>
      <h1>Cart</h1>
      <ItemList items = {cartItems}/>
   </div>
```

x");

tem)}

item

#Creating a Clear list button

in Cart.js

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

const dispatch = UseDispatch();
const handleClearCart = () => {
        dispatch (clearCart());
};

<button onClick = {handleClearCart}

{cartItems.length === 0 && (
      <h1> Cart is Empty </h1>
)}
```

① # While subscribing to store subscribe to only a particular part of the store. **Optimizing Redux**

Because if whole store is selected then it will render every time the store get updated.

(2) # When we create the store we use keyword " reducer " (a)

But when we create a slice and create actions in it we use word " reducers " (b) because we are creating many actions inside it

(c) While exporting it is a single reducer
export default cartSlice.reducer;

In Vanila redux (OLD)

① No Mutation

② return was compulsory

③ Create a whole new state and then replace it with old state for every small change

④ So when delete whole cart ↓ this will not work
state = []

Redux toolkit (New)

① we need to mutate

② Not ~~co~~ mandotary

③ Redux uses immer to do all old things automatically

④ Instead we do
state.items.length = 0;
or
return {item : []}
(Either mutate the state or return the whole new state)

Redux Dev Tools chrome extension, Makes easy to debug

→ Redux is used for large scale applications

H.W Read About RTK Query Overview.