12. Let's build our Store

**** Redux is not Mandatory in your Application ****

- If you are building small App using ~~Redux~~ React, you don't need Redux
- Also you can build big Apps without using Redux
- The Apps which are build using Redux can be build without using Redux also

* Redux and React are different libraries *

- Use Redux wisely, only when Required.

- Redux is not the only library for managing state, there exist one more library called (zustand)

Adv :- Redux offers great Solⁿ when creating large Scale Applⁿ
i.e, handling data, Managing store etc
handling state

- Redux offers easy debugging
(Redux devtools) => chrome extension.
- Predictable
- Centralized
- Flexible.

Redux has 2 libraries offered by Redux Team

1) React - Redux :- kind of bridge betⁿ React & Redux

2) Redux - Toolkit (RTK) :- (Newer Way of Writing Redux)
(latest Version) / Standard Version

⇓ standard way of Writing Redux logic

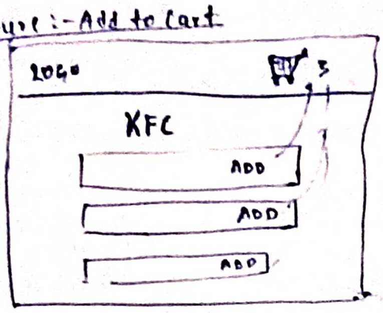
It Solve's 3 major Concerns of Redux

- Redux was very Compliacted to learn
- We have to add lot of package to get Redux to do anything useful
- "Redux requires too much boilerplate code"

eg:- Add to Cart Feature

Redux store: A big whole object & kept in Central global place. And any component can access it in our Application

Feature :- Add to Cart



- It is absolutely fine that we can keep object huge large data in a Central Object.
- Also, Our Redux does not become huge or very big, becoz we have slices in Redux

A small portion of Redux store.

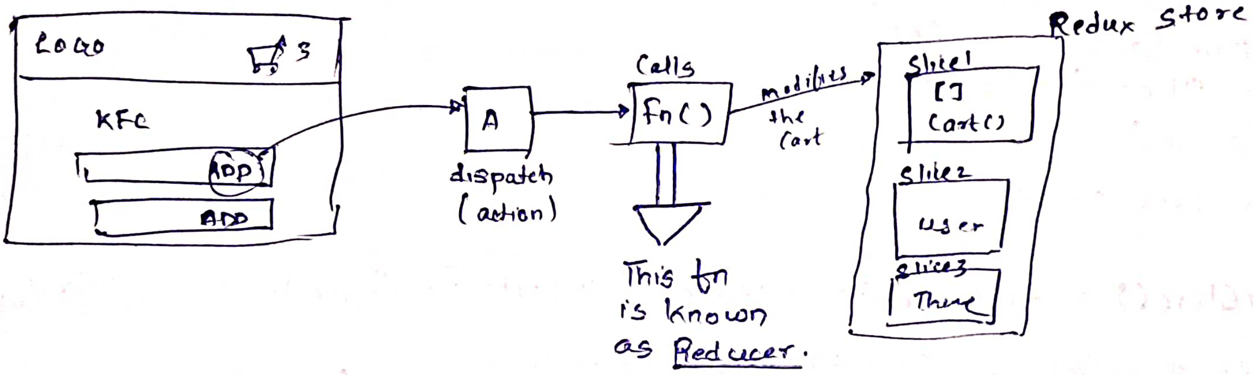
- we create Multiple slices in Redux Store.

Eg:- there can be a Cart Slice / User Slice / Theme Slice etc

- when you click on add button, you cannot directly add data to your Cart Slice. It's not possible to directly add Data to your cart Slice

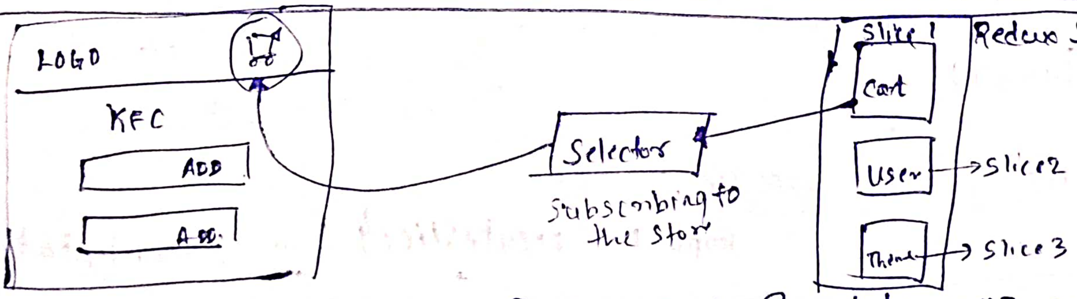
- when you click on add button, it dispatches an action, after dispatching an action, it calls an fn, and this fn modify the Cart

Write data



Hence,
- when you click on add btn, it first dispatches an action, calls a Reducer fn, And these Reducer function modifies our Cart Slice in our Redux Store
updates

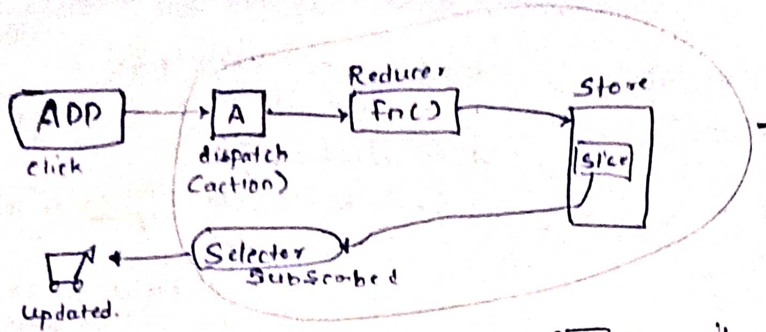
Read data



To Read Data from Cart Slice & update on UI, we use Something called as Selector

- Suppose, you want to Read Data from Cart Slice & display the data on UI Component, you will use a Selector. Selector will give data to you for the Cart.
- This phenomenon is known as Subscribing to Store

Working of Redux & Add to Cart feature implemention



→ happens behind the Scenes.

- When you click on (add) btn it will dispatch an action.
- It will call on Reducer fn(), which updates the Slice of our Redux store.
- Our Now, our Cart Component is Subscribed to the store using a Selector, it will [Cart] will automatically get upbated.

When you click on (ADD to cart) btn, the 🛒 cart will be update.

Redux Toolkit

1. Install@reduxjs/toolkit & Install react-redux [install using "npm i @reduxjs/toolkit" "npm i react-redux"]
2. Build our Store
3. Connect Our Store to our App
4. Create a Slice [cartSlice)
5. dispatch (Action)
6. Read data using Selector.

- **ConfigureStore()** ⇒ fn to Create a Store (Import {configureStore} from "@reduxjs/toolkit")
- We need to provide our Store to our Application.
  ↳ How? import {provider} from "react-redux"; } It act as a bridge between React & Redux

```
<Provider store={reduxStore}>
    <Whole Applayout/>
</Provider>
```

→ Provider will wrap your Whole App inside it
→ Pass your Redux store to Provider as a props

Creating a Slice : using:→ createSlice() fn
import {createSlice} from "@reduxjs/toolkit"

reducer is a combination of reducers of slice

Store

reducer : {
reducers of slices
}

Slice

reducers : { – reducers contains a combination of reducer fn
reduer fn 1
reducer fn 2
}

while exporting only one reducer is exported

CartSlice

Eg:-

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

const CartSlice = createSlice({
    name: "cart",
    initialState: {
        items: [],
    },
    reducers: {
        addItems: (state, action) => {
            state.items.push(action.payload);
        },
        removeItems: (State) => {
            State.items.pop();
        }
        clearCart: (state) => {
            State.items.length = 0;
        },
    }
});
export const {addItem, RemoveItem, clearCard} : CartSlice.actions;
export default ~~CartSlice~~ ~~CartSlice~~ cartslice.reducer;
```

object

object

fn

→ name of the cartSlice.

→ Initial state / Initial value of the Slice.

→ - reducers fn to modify the State

- It contains some action and state.

- Reducers contain multiple fn's to modify the State using actions

export 2 things

— Actions

— Reducers.

* createSlice() fn will return an object in the cartSlice.

* Cartslice object will have actions, reducer.

* each Slice will have a Reducer.

useSelector() hook For Reading

useDispatch() hook For Write / dispatching

↳ dispatch an Actions.

give by "react-redux"

Imp Tip

- Whenever you are using Selector, make sure you are Selecting write portion of the store, to increase performance.

- Our store will have only one reducer, but Slice will have multiple ~~[illegible]~~ reducers

* Redux devTools for debugging

↳ chrome Extensions

Context API → Using Context API, it increase the Performance and Code Complexity reduces.

Snapshot of data at current present time [data/info/state]

used to Complete certain task/Work

To use Context API, there are few Rules to use (Always follow):

① first you need to Create Context. [i.e. Preparing data at the initial Case, present time data]

② Providing Context. (transfer/Send data to the Components]

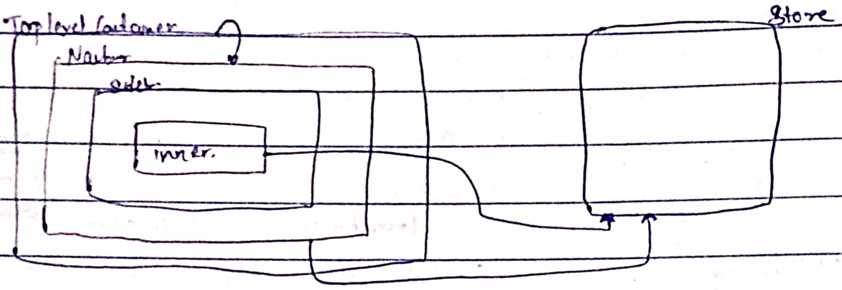
③ Consuming Context. (Use the data received).

Chai aur code

Redux → Independent State Management library.

↳ react-redux) library in Redux

changes are made using pure functions → Reducers



Store: Global variable type & which has all data & can be accessed easily

Reducers: functionality. (Slice → bigger Version of Reducer) type

useSelector: Select value from store

useDispatch: dispatch i.e. Send value to store

To install Redux in React use these command.

1) npm install @reduxjs/toolkit

2) npm install react-redux.

Step 1:

Process to make store.

1) Configure store import {configureStore} from '@reduxjs/toolkit';

- configureStore({}) (bydefault it takes object)
- Import the exported reducer

Step 2:

Process to make Reducers: Slice

1) import {createSlice, } from '@reduxjs/toolkit';
nanoid (unique id)

- initialState can be array or object
- CreateSlice({}) → It has name, initialState, reducers : {}

Reducers have Property & functionality

You need to export createSlice Reducer.
1) functionality inside Reducers. (Individual functionality)
2) Slice you created also need to be exported.

Step 3:- useSelector

import {useSelector} from 'react-redux'

useSelector(state => ())

useSelector gives you callback fn inside the parenthesis, help to access state using callback fn.

Step 4:- use Dispatch

import {useDispatch} from 'react-redux'

const dispatch = useDispatch()

dispatch uses reducer and changes value in store i.e. add value to store