**NAMASTE REACT – AKSHAY SAINI**

**CHAPTER-12 – LET’S BUILD OUR STORE**

1. Redux vs Context api

Ans:-

| Context API | Redux |
| --- | --- |
| Built-in tool that ships with React | Additional installation Required, driving up the final bundle size |
| Requires minimal Setup | Requires extensive setup to integrate it with a React Application |
| Specifically designed for static data, that is not often refreshed or updated | Works like a charm with both static and dynamic data |
| Adding new contexts requires creation from scratch | Easily extendible due to the ease of adding new data/actions after the initial setup |
| Debugging can be hard in highly nested React Component Structure even with Dev Tool | Incredibly powerful Redux Dev Tools to ease debugging |
| UI logic and State Management Logic are in the same component | Better code organization with separate UI logic and State Management Logic |

From the table, you must be able to comprehend where the popular opinion Redux is for large projects & Context API for small ones come from.

Both are excellent tools for their own specific niche, Redux is overkill just to pass data from parent to child & Context API truly shines in this case. When you have a lot of dynamic data Redux got your back!

Refer:- <https://dev.to/ruppysuppy/redux-vs-context-api-when-to-use-them-4k3p>

1. Advantages of using Redux-toolkit over Redux?

Ans:-

The **Redux Toolkit** package is intended to be the standard way to write [Redux](https://redux.js.org/" \t "_blank) 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"

1. Explain Dispatcher.

Ans:-

dispatch is a function of the Redux store. You call store.dispatch to dispatch an action. This is the only way to trigger a state change.

Refer:- <https://react-redux.js.org/using-react-redux/connect-mapdispatch>

import { useDispatch, useSelector } from "react-redux";

import { clearCart } from "../utils/cartSlice";

import FoodItem from "./FoodItem";

const Cart = () => {

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

  const dispatch = useDispatch();

  const handleClearCart = () => {

    dispatch(clearCart());

  };

1. Explain Reducer.

Ans:-

Reducers, as the name suggests, take in two things: previous state and an action. Then they reduce it (read it return) to one entity: the new updated instance of state. So reducers are basically pure JS functions which take in the previous state and an action and return the newly updated state.

1. Explain slice

Ans:-

A slice is the portion of Redux code that relates to a specific set of data and actions within the store 's state. A slice reducer is the reducer responsible for handling actions and updating the data for a given slice. This allows for smaller reducer functions that focus on a slice of state.

Refer createSlice.js in store

1. Explain Selector

Ans:-

Selector allows you to extract data from the Redux store state, using a selector function in your required component.

import { useDispatch, useSelector } from "react-redux";

import { clearCart } from "../utils/cartSlice";

import FoodItem from "./FoodItem";

const Cart = () => {

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

1. Explain createSlice and the configuration it takes.

Ans:-

Creating a slice requires a string name to identify the slice, an initial state value, and one or more reducer functions to define how the state can be updated. Once a slice is created, we can export the generated Redux action creators and the reducer function for the whole slice.

Refer:- <https://react-redux.js.org/tutorials/quick-start#create-a-redux-state-slice>

const cartSlice = createSlice({

  name: "cart",

  initialState: {

    items: [],

  },

  reducers: {

    addItem: (state, action) => {

      state.items.push(action.payload);

      //mapping between action from component(left side) and reducer function(right side)which takes 2 paraters state and action

    },

    clearCart: (state) => {

      state.items = [];

    },

    removeItem: (state, action) => {

      state.items.pop();

    },

  },

});

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

export default cartSlice.reducer;