**Assignment 13: time for the test**

***Theory:***

Q) useContext v/s Redux?

|  |  |
| --- | --- |
| **useContext** | **Redux** |
| useContext is a hook in React that allows you to access the value of a context directly in a functional component.  NOTE: In React, a context is a way to share data between components without having to explicitly pass props through each level of the component tree | Redux is an open-source JS library used for managing / handling the states of our application.  NOTE: In React, "handling the state of our application" refers to the management of data within a component or across components to keep track of changes, control the rendering of components, and maintain the application's interactivity.  State represents data that can change over time and affects how a component behaves or renders. |
| useContext is an ideal choice for the React Apps which does not have complex data management / handling.  It’s not that, complex data cannot be handled by context in React, it’s just that its bit harder way to do it. | The apps which have large data manipulations must use Redux as it offerseasy ways to manage the state / data of our application |

Q) Advantages of using Redux Toolkit(RTK) over Redux?

The main advantages of using Redux Toolkit over Redux are

1. Configuring the redux was too complex which is a major change in Redux Toolkit. RTK configuration is too easy.

1. Redux Toolkit is newer way of writing redux. It needs just 2 libraries to do all its work. One is @redux/toolkit and the other one is “react-redux”. BUT the older versions of redux needed many of the libraries to function

**NOTE:** Redux deals with the data-layer of the application where as react-redux is a bridge between data layer and the UI layer of the application

1. Earlier there was a different way of writing redux code. The syntaxes were complex, that’s the reason why the redux team itself revised their code so much that they come up with Redux Toolkit
2. One of the other important advantage of using RTK is that it gives us a very easy way to debug the code using redux dev tools

Q) Explain Dispatcher

We cannot directly modify the Slice. For this we need something called Dispatcher.

When an action takes place to modify the Slices

1. First, it Dispatches an ACTION
2. Then, this action, calls a FUNCTION (Known as a Reducer)
3. Finally, the reducer function will modify the Slice of a Redux store

So, dispatcher can be explained as an Action That calls a function called “Reducer” which modifies the Slice.

For example, if we consider the cart functionality, adding data to cart, deleting data from cart or updating the data in the cart are the different actions. These actions will dispatch an action which calls the function known as reducer function to modify the cart

Q) Explain Reducer

The dispatched action calla a function which is known as a reducer function which is used to modify the Slice in the redux store.

Q) Explain Slice

A slice is a Logical Portion of a Redux Store. As we know, redux store is a very big object that stores lot of data. We do slices to manage this data in a simple way. For example

1. All the data related to a cart functionality can be stored in a slice say cartSlice.
2. All the data related to a logged in user can be stored in a slice called loggedinUserSlice
3. Data related to a Theme can be kept in a slice say themeSlice

And this goes pn.

Slices help us to manage the data in the redux store very comfortably with out any confusion.

As explained above, if any action related to the cart would be done in cartSlice, any action related to the logged in user will be done in loggedinUserSlice and so on.

Q) Explain Selector

Selector is a named hook provided by “react-redux”, used to read the data from a slice. We use useSelector to read the data from a slice.

The phenomenon of using the Selector to read the data from the slice is know as “Subscring to the store”.

Let’s see the steps to read the data from a slice say cartSlice.

1. import { useSelector} from “react-redux”;
2. //Subscribing the header to our appStore using selector

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

where, store is an object, cart is the name given in the cartSlice and the items are the initialState values given

So, store.cart.items will get me all the cart values.

1. THEN, cartItems.length will give me the number of items present in the cart.

Q) Explain the createSlice and the configuration it takes

createSlice is a function provided by redux-toolkit which takes few configurations to create a slice in the redux store.

To explain it in better way lets take an example of a cart slice, where we try to create a slice and explain its configuration in detail

Eg:

Import {createSlice} from “@redux/toolkit”;

const cartSlice = createSlice({

name: cart,

initialState: {

items: [],

},

reducers: {

addToCart: (state, action) => {

state.items.push(action.payload);

},

removeFromCart: (state, action) => {

state.items.pop();

},

emptyTheCart: (state, action) => {

state.items.length = 0;

}

}

});

export { addToCart, removeFromCart , emptyTheCart } = cartSlice.actions;

export default cartSlice.reducer;

NOW lets see all the configurations in detail

1. name : is the name of the slice
2. initialState : it takes the initial state of the cart, which is given as empty array initially
3. reducer: will have all the reducer functions corresponding to the actions. In this case adding item to cart, deleting item from cart and emptying the cart are the actions from which ae will be writing the functions over here.

addToCart, removeFromCart and emptyTheCart are basically the dispatched action which calls the function to modify the cart.

These functions will basically take 2 arguments. State & action

state : to get the access to the state of the slice

action: to get the access to the actions of a slice

SO, the reducer functions will change the state / data of the cart depending on the action

1. Then export the reducer of cartSlice

export default cartSlice.reducer

1. //exporting all the actions inside the cartSlice’s “reducer-actions” which can be later imported to do the preferred actions

export { addToCart, removeFromCart , emptyTheCart } = cartSlice.actions;

***Coding:***

https://github.com/nandinihulsurkar/me-learning-react.git