Redux -> is an state management

Redux is an separate/independent state management library

Redux – react-redux like react react dom or react native

To connect with react react – redux

History

Statemanagement context API – included within react

Before context API – Flux proper data flow – how data flows / one way of data flow

Data structure

Redux – single source of truth/state should be readOnly/never modify state – changes should be made through a function(reducers)

A predictable state container for JS App

Redux – middleware – thunk/ saga helps in debugging

Redux-toolkit- avoids bulky setup/batteries included/built in middleware

1. Store- like global variable can contain many mini stores- single store - object
2. Reducers – controls functionality which helps to control state
3. useSelector- when any value is to be selected within store
4. useDispatch- when any value is to be sent.

Installation

npm install @reduxjs/toolkit

npm install react-redux

[configureStore()](https://redux-toolkit.js.org/api/configureStore)

[createReducer()](https://redux-toolkit.js.org/api/createReducer)

[createAction()](https://redux-toolkit.js.org/api/createAction)

[createSlice()](https://redux-toolkit.js.org/api/createSlice)

[combineSlices()](https://redux-toolkit.js.org/api/combineSlices)

[createAsyncThunk](https://redux-toolkit.js.org/api/createAsyncThunk)

[createEntityAdapter](https://redux-toolkit.js.org/api/createEntityAdapter)

[createSelector](https://redux-toolkit.js.org/api/createSelector)

creating store

new file

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

export const store = configureStore({})

create reduces/slices

creating reducer/slice

new file-> todoSlice.js

import { createSlice, nanoid } from "@reduxjs/toolkit";

nanoid -> creates unique id

declare initial val – can be object array

const initialState = {

    todos: [{id:1, text:"Hello World"}]

}

Creating slice / reducer

export const todoSlice = createSlice({

    name : 'todo', //cumpusion key

    initialState,

    reducers: {

        addTodo: (state, action)=>{ alaways access the state

            const todo= {

                id: nanoid(),

                text: action.payload

            }

            state.todos.push(todo)

        },

        removeTodo: (state,action) => {

            state.todos = state.todos.filter((todo)=> todo.id !== action.payload)

        },

    }

})

//exporting actions

export const {addTodo, removeTodo} = todoSlice.actions

//exporting reducers to sync with store

export default todoSlice.reducer

// importing reducer to store

import todoSlice from "../features/todo/todoSlice";

export const store = configureStore({

    reducer: todoReucer

})

//component

Dispatch sends to store

Dispatch and useSelect is from react-redux

import { UseDispatch, useDispatch } from 'react-redux'

        const dispatch = useDispatch()

dispatch uses reducer to change the value in store

first import the actions/ methods that you want to use from reducer/slice

import {addTodo} from '../features/todo/todoSlice'

  const addTodoHandler = (e)=>{

            e.preventDefault();

            dispatch(addTodo(input) input is action.payload

        }

// using the val dispatched

Import use selector

useSelctor(()=>{}) a method with call back function

import provider and store in main.jsx

  <Provider store={store}>

    <App />

  </Provider>,