REDUX TOOLKIT

Presented by hema vardhan

IMMER

till now we have copied the original state and manipilated in reducer and returned it and updated the original state
we spread the state and then change necessary key value in it
this makes complex for nested states
eg:const initialState = {

```
    name: 'hema vardhan',
    address: {
        city: 'hyd',
        street: '3rd street',
        nearby: 'lord shiv temple'
    }
}
```

to update street here we need to do in reducer return{

```
        ...state,
        address: {
            ...state.address,
            street: action.payload
        }
```

SOLUTION: IMMER

as a developer its difficult to keep track of all nested states so we use library immer

1. install: npm i immer
2. import : const produce = require('immer').produce // function
3. instead of spreading each object in nestead state we use

```
produce(originalObj, (OriginalObj_But_Can_be_mutable)=>{
      OriginalObj_But_Can_be_mutable.address.street = action.payload
 })
// eg from before problem
```

MIDDLEWARE

middleware is the suggested way to extend redux with custom functionalityprovides a third-party extension point between dispatching an action, and moment it reaches the reducer

- we use middleware for logging,crash reporting,Performing asynchronous tasks etc...

REDUX-LOGGER

it is a middleware used in redux

1. install: npm i redux-logger
2. import createLogger : const reduxLogger = require('redux-logger')
3. initialize function : const logger = reduxLogger.createLogger
4. redux provides a function to apply middleware : redux.applyMiddleware
   const applyMiddleware = redux.applyMiddleware
5. in createStore we pass 2nd argument as
   applyMiddleware(middleware1,middleware2,....)

here : const store = createStore(rootReducer,applyMiddleware(logger))

6. this logger middleware is useful to log out all logs every action happening in redux store and state

ASYNC ACTIONS

in synchronous actions:
as soon as an action was dispatched, the state was immidiately updated
as soon as action was dispatched, state was immi=ediately updated
if you dispatch the CAKE_ORDERED action the numOfCakes was decremented right away
for asynchronous actions:
let us say we want to fetch users from an api end point and stores it in the redux store
how to build state,actions,reducers

STATE IN ASYNC

state structure:

```
state = {
   loading : true,
  data : [ ],
  error: ''
}
```

loading: used to display loading spinner in your component while fetching data
data: to store list of users
error: this error property is used to store if theres any error occured in fetching data

ACTION IN ASYNC

1. FETCH_USERS_REQUESTED
2. FETCH_USERS_SUCCEEDED
3. FETCH_USERS_FAILED

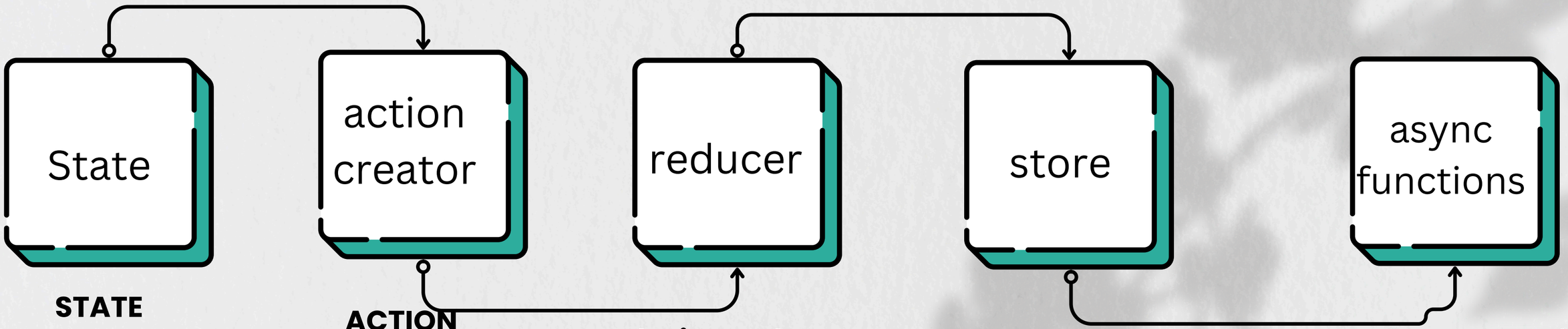
we are going to have 3 actions in our application
the 2nd and 3rd actions are dependent on 1st one
if 1st one succeeded then 2nd one is used
else 3rd one is used

REDUCER IN ASYNC

case: FETCH_USER_REQUESTED
loading: true
case: FETCH_USERS_SUCCEEDED
loading:false
users: data(from api)
case: fetch_users_failed
loading: false
error: error (from API)

if case 1 matches action then loading is set to true and then data is fetched
if case 1 fails it generates failed action and uses case 3
else uses case 2

FLOW



STATE
design a state

ACTION CREATOR
create an action creator using redux-thunk module npm

design reducer
Lorem ipsum dolor sit amet, consectetur adipiscing elit.

STORE
we create store and include middlewares like thunk to create async action creators

create async action creators

ASYNC ACTION CREATORS

to fetch api we use axios

redux-thunk : used to define async action creators (its a middleware)

1. install: npm i redux-thunk
2. apply redux-thunk middleware to createStore using applyMiddleware

const thinkMiddleware = require('redux-thunk').default

pass this middleware into applyMiddleware in createStore 2nd arg

- action creator usually returns action but redux-thunk helps in returning function in action creator
- speciality of this return function is it dosent have to be pure it can have sideeffects like async api calls
- these returned async fxns take dispatch as argument

```
const fetchUsers = ()=>{
    return async function(dispatch)=>{
     dispatch(fetchUserREq())
        try{
              const users = await axios(api)
              dispatch(fetchUsersSuccess(users))
        }catch(error){
            dispatch(fetchUsersFail(error.msg))
        }
    }
}
```

THANK YOU