const redux = require("redux");

const createStore = redux.createStore;

const BUY\_CAKE = "BUY\_CAKE";

function buyCake() {

  return {

    type: BUY\_CAKE,

    info: "First redux action",

  };

}

//(previousState, action) => newState

const initialState = {

  numOfCakes: 10,

};

const reducer = (state = initialState, action) => {

  switch (action.type) {

    case BUY\_CAKE:

      return {

        ...state,

        numOfCakes: state.numOfCakes - 1,

      };

    default:

      return state;

  }

};

const store = createStore(reducer);

console.log("initial state", store.getState());

const unsubscribe = store.subscribe(() =>

  console.log("updated state", store.getState())

);

store.dispatch(buyCake());

store.dispatch(buyCake());

store.dispatch(buyCake());

unsubscribe();

Redux is a predictable state container for Js apps.

Redux Is a library for javascript

Redux store sate of the app.

#npm install redux

**Three core concepts redux**:

Store---holds the state of the application

Action---that describes the changes in the state of the application.

Reducer—A reducer which carries out the state transition depending on the action.

Principles:

1.The state of the whole app is stored in an object tree within a single store;

Maintain our app state in a single object which would be managed bty the redux store.

2.The noly wat to change the state to emit an action, an object describing what happens.

3. To specify how the state is transformed by actions, you write pure reducers

Reducers-{previousState,action}=> newState

Javascript app

Subscribed dispatch

Reducer

Action

Redux store(state)

**Redux store**

**Resposibilities**

* **Holds** application state
* **Allows access to state via getState()**
* Allows state to be updated vi dispatch(action)
* Registers liseners via subscribes(listener)
* Handles registers via subscribe(lisener)
* Handles unregistering of listeners via the function returned by subscribe(listner)

Middleware:

Redux logger

Npm I redux-logger

Documentation

**Asyncronous actions-12 video**

**Appl**ication structures : fetches a list of users from an ApI end point and stores it in the redux store.

State?

State={

Loading: true,

Data:[],

Error:’’

}

Loading: display a loading spinner in your component

**Data:** List of users

**Error:** display eeror to the users

Actions?3

**FETCH\_USERS\_REQUEST :** Fetch list of users

**FETCH\_USERS\_REQUEST:**  fetched successfully.

**FETCH\_USERS\_REQUEST:** error fetching the data.

Reducers?

**Case:FETCH\_USERS\_REQUEST**

Loading:true

**Case:FETCH\_USERS\_REQUEST:**

Loading:false

Users:data(from API)

**Case:FETCH\_USERS\_REQUEST:**

Loading:false

error:data(from API)

#install packages for aPi call

1.axios –request to an api end point

2.redux-thunk---define async action creators