What is Redux?

“Redux is a predictable state container for Javascript apps”

* It is JavaScript apps
* It is a state container
* It is predictable

Redux is for JavaScript applications

Redux is not tied to React

Can be used with React, Angular, Vue or even vanila JavaScript

Redux is a library for JavaScript applications

Redux is state container

Redux stores the state of your application

State of an application is the state shared by al the individual components of that app

**\*\*\*\* (STORE, STATE, ACTIONS & REDUCERS)**

Summary

1. React is a library used to build user interfaces
2. Redux is a library for managing state in a predictable way in JavaScript applications
3. Redux toolkit is a library for efficient redux 𝑑𝑒𝑣𝑒𝑙𝑜𝑝𝑚𝑒𝑛𝑡
4. React-redux is a library that provides bindings to use React and redux (Toolkit) together in an application.

Cource structure:

* Learn Redux first
* Learn Redux Toolkit second
* Learn React Redux third

Prerequisites:

React Fundamentals

React Hooks

Make a project folder and inside – open terminal --

>npm init - -yes -- this will create a basic package.json file in the project

then install redux

>npm install redux

Three Core Concepts:

A **store** that holds the state of your application

An **action** that describes what happened in the application

A **reducer** which handles the action & decides how to update the state

Three Principles:

1. The global state of your application is stored as an object inside a single store

* Maintain our application state in a single object which would be managed by the redux store

Example: Cake Shop—

{

numberOfCakes:10

}

1. The only way to change the state is to dispatch an action, an object that describes what happened

* To update the state of your app, you need to let redux know about that with an action
* Not allowed to directly update the state object

Example:

Cake shop:

Not allow directly take the cake from the shelf in the shop

Scan the QR code and place an order – CAKE\_ORDERED

{

Type: ‘Cake\_Ordered’

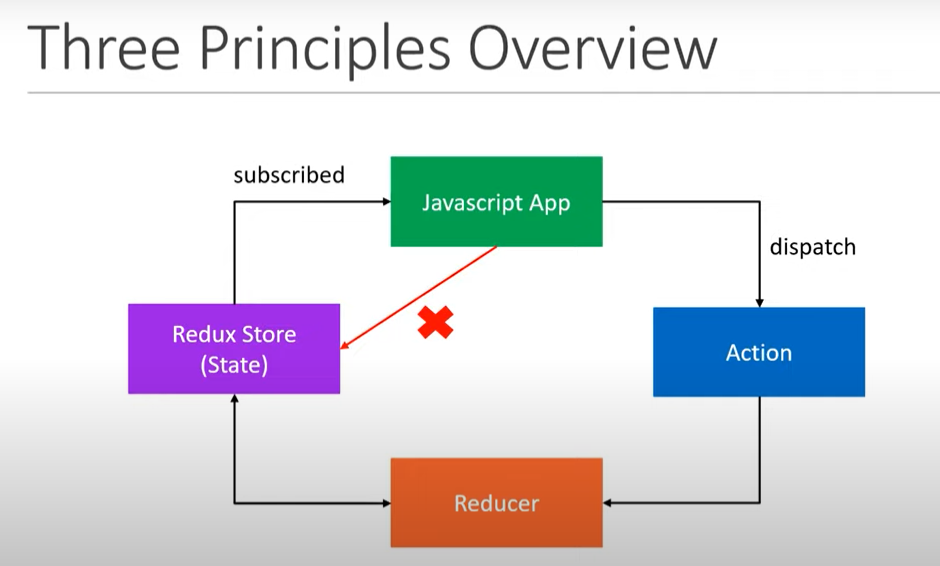
}

1. To specify how the state tree is updated based on actions, you write pure reducers

Reducer – (previousState, action) => newState

Example cake shop:

Reducer is the shopkeeper



Actions:

The only way your application can interact with the store

Carry some information from your app to the redux store

Actions are plain JavaScript objects

Have a ‘type’ property that describes something happened in the application

The ‘type’ property is typically defiened as string contants

Reducers:

Specify how the app’s state changes in response to actions sent to the store

Function that accepts state and action as arguments, and returns the next state

Of the application

(previousState, action) => newState

Redux Store:

One store for the entire application

Responsibilities –

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

**Restocking Cakes:**

Everyday, a vendor comes to the shop to restock the shelves.

The vendor can stock up one or more number of cakes depending on the previous day sales.

Async action creators:

2 packages needed

**Axios**: Requests to an API end point

**Redux-thunk**: Define async creators

Middleware

> npm I axios redux-thunk

Redux concerns:

Redux requires too much boilerplate code

* Action
* Action object
* Action creator
* Switch statement in a reducer

A lot of other packages have to installed to work with redux

* Redux-thunk
* Immer
* Redux-devtools

So, there was a need to improve the developer experience for redux

For this reason we need Redux toolkit library

Redux Toolkit

Redux toolkit is the offitial, opinionated, batteries-included toolset for efficient Redux development

* Abstract over the setup process
* Handle the most common use cases
* Include some useful utilities

npm install @reduxjs/toolkit

<https://redux-toolkit.js.org/introduction/getting-started>

* Redux toolkit provides create asunc thunk function to impelment the creation and dispatching of async functions
* We need to make redux store available to the react component tree
* To provide the redux store to our react application react-redux library exports a component called provider

\*\* Provider:: (Connecting react app with redux) We need to make redux store available to the react component tree and this is where the react redux library makes it’s first apperance . To provide the redux store to our react application react-redux library exports a component called provider. In main layout file:

 <html lang="en">

      <body className={inter.className}>

        <Provider store={store}>

          {children}

        </Provider>

      </body>

  </html>

**useSelector – hook ::** which is use to get hold any state that is maintained in the redux-store. To read data from the redux store in the react component we use the use selector hook. It is sort of a wrapper around store.getstate()

const role = useSelector((state: RootState) => state.userReducer.userInfo?.role);

const userName = useSelector((state: RootState)=> state.userReducer.userInfo?.name);

**useDispatch – hook ::** which is used to dispatch an action with react-redux.

const dispatch = useDispatch<AppDispatch>();

 onSubmit={(formData) => {

       dispatch(userLogin({

            email: formData.email,

            password: formData.password

          }))

    }}

Create async thunk dispatches the life cycle methods of a promise as actions

\*\*\* Handling the async task redux toolkit provide extra reducers.

<https://www.youtube.com/watch?v=0awA5Uw6SJE&list=PLC3y8-rFHvwiaOAuTtVXittwybYIorRB3&index=1&ab_channel=Codevolution>