**React Concept**

1->Don’t touch the DOM, (declarative)->one big object that act as component, you change it and I will change the dom. React-> this is the state make the change

2-> small component=> plain java script function()=> except prop and has html like structure and then render. It will have state

Component can be function or class component

3 -> one way data flow (state+ jsx)=> give component render in virtual dom

**React developer tools**

Node -v

npm -v

yarn -v

Install dependencies from package.json: npm install == yarn

Install a package and add to package.json: npm install package --save == yarn add package

Install a devDependency to package.json: npm install package --save-dev == yarn add package --dev

Remove a dependency from package.json: npm uninstall package --save == yarn remove package

Upgrade a package to its latest version: npm update --save == yarn upgrade

Install a package globally: npm install package -g == yarn global add package

npx create-react-app

cd

code .

npm start

json file=>

* react
* react dom
* script
  + start
  + build-> production build (build folder) => javascript
  + test
  + eject-> manual and granular control

babel-> code to basic javascript

web pack=> chunk (break it into chunck)

public folder ->index.html(root)->src

public folder has other static stuff

src folder->index.js->(react;react-dom)->App-> component

index.js=> reactdom.render()

classes are java script

hooks are react specific. hooks are use with function

Set state=> we are doing object. Assign () so it is creating new object in memory

Set state do=> Shallow merge, will I will update the state value of the key,

In case of shallow merge, it only merges the key that is asked for even though it is a complex.

State

Set State

Map

Map with id: Map optimised by using id

**Classes Life cycle hooks**

Lifecycle (unique to class component)-> functional component we don’t interact with the same

* Mounting -> componentDidMount
  + constructor-> render->
* Updating-> componentDidUpdate
  + New Props -> render
  + SetState() ->render
  + forceUpdate->render
* Unmounting -> componentWillUnmount

UseState

UseEffect

Virtual DOM is the JavaScript version of real DOM

React Router doms

BrowserRouter

Routes-> Route (Home)> Route+ route

<Link > + Outlet

FormControl

**Context** => Stored Data so that it can be access by anyone instead of props tree

Context provider

Use context

CreateContext

**Provider**

UseParam

UseNavigation

UseReducer

npm install styled-components

Promise

Observer

**Redux**

npm install redux react-redux redux-logger

npm install reselect

store->root reducer🡪 reducers ->(intital state, reducer, action and selector)

useSelect

useDispatch

reselect -> memorized

npm install redux-persist

npm install redux-thunk

npm install redux-saga

npm install @reduxjs/toolkit

.env file

Import element from stipe

Create instance and add to it

Redux vs context data access

Context

Redux

Redux vs Context Data Flow

value

action

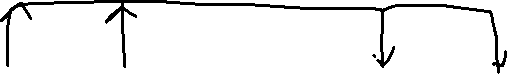
action

Reducer

Component 2

Component 1

Dispatch Action



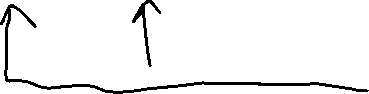
Component 1

Reducer

Reducer

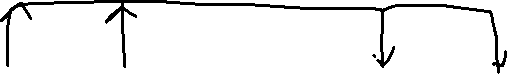
Component 2

Pass state



Root

Redux middleware



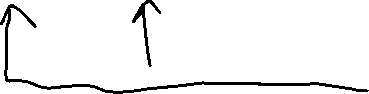
Component 1

Reducer

Reducer

Component 2

Pass state



Root

**Thunk** and **Saga** is for Async middle ware

Thunk act before reducer

Saga act after state is update, post reducer

Type scripts: tsconfig -> jsx== tsx, js== ts

Variable

Function

**Performance**

UseCallback(()=>{},[]) - cache the function

UseMemo(()=>{},[]) – cache the output

Lazy/ Suspend

import { useEffect, lazy, Suspense } from 'react';

import { useDispatch } from 'react-redux';

import { Routes, Route } from 'react-router-dom';

import Spinner from './components/spinner/spinner.component';

import { checkUserSession } from './store/user/user.action';

const Navigation = lazy(() =>

import('./routes/navigation/navigation.component')

);

const Shop = lazy(() => import('./routes/shop/shop.component'));

const Checkout = lazy(() => import('./routes/checkout/checkout.component'));

const Home = lazy(() => import('./routes/home/home.component'));

const Authentication = lazy(() =>

import('./routes/authentication/authentication.component')

);

const App = () => {

const dispatch = useDispatch();

useEffect(() => {

dispatch(checkUserSession());

}, []);

return (

<Suspense fallback={<Spinner />}>

<Routes>

<Route path='/' element={<Navigation />}>

<Route index element={<Home />} />

<Route path='shop/\*' element={<Shop />} />

<Route path='auth' element={<Authentication />} />

<Route path='checkout' element={<Checkout />} />

</Route>

</Routes>

</Suspense>

);

};

export default App;