5/July/23

React JS

- Installation →

① Install Node JS, Vs code

Create new folder React Folio
Change directory to React Folio
npx create-react-app demoshopapp
Change directory to demoshopapp
npm start

Option 2 →

Starter pack + Node JS

↓

GitHub.

OR

Create vite@latest

Package.JSON. → Contains Dependencies of React App. (imp data)

Index.JS → Entry point (FIRST to be executed)

~~Index.HTML~~ → INDEX.JS

```
import React from 'React'
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';

const root = ReactDOM.createRoot(document.
             getElementById('root'));

root.render(<App />);
```

APP.JS & APP.CSS.

```
import './App.css';

function APP() {

return (
<div className="APP">
Hello JEE
<div>
);
}

export default APP;
```

classmate
Date
Page

To make new Component →

Suppose navbar →

nav.JS
nav.css } Create these two Files

nav.JS
import './Nav.css';(;) → not necessary

function navbar () {
return(// Code)
}
export default navbar;

Naming Convention for Component

Itemname X

ItemName ✓

To use the Component →

go to App.JS

Write →

import Navbar from './Components/ITEM

and write <Item> </Item>
where we want to include Code.

React (Basics → Akshay Saini)

NPM : (npm init) → will add Package.json

Package Manager.

We want to Manage all the Packages which are required for React to run we need alot of superpowers And those superpowers comes from different Package which are installed using npm.

All those helper packages comes inside npm

~~Package.json~~

Parcell : npm install Parcel

npm install -D Parcel

↳ Dev dependency

-- save dev -

Dependencies

Now we will get Package-lock.json & Node Modules

We need parcel as in our Development Environment Parcell created a server for our app.

Package lock, locks the version keeps track of versions.

Node Module → Database for application

It takes minimum effort to inject a library into our code.

10/07/23

to Import React to App.js.

Import React from "React"
Import ReactDOM from "react-dom"

Also do,

< Script type = 'module' src = app.js />
in index.html.

Inception

LEC - 1

To Make Changes to HTML File

- Using HTML →

< h1 > Hello world < /h1 >

- Using JS

In HTML,

< div id = "root" > < /div >

In JS.

Const root = document.getElementById("root");

Const heading = document.createElement("h1");

heading.innerHtml = "Hello Duniya";

root.appendChild(heading);

React.createElement("tag", {}, "content")

Using React :

Inject React CDN Link &
React-dom CDN Link

Now ,

```
Const heading = React.createElement("h1",
```

```
Const root = ReactDOM.createRoot(document.getElement
                                  .id("root"));
```

```
root.render(heading);
```

We can add React to existing project also

like , →

```
<div id = "header"></div>
<div id = "root"></div>
<div id = "Footer"></div>
```

We can use react for any part just Make it id root & Modify that Section using React.

```
<div id = "root">
    <h1> Hello</h1>
    <h2> Hi </h2>
</div>
```

Create this structure using React →

```
<Script>

Const heading1 = React.CreateElement("h1",
{
    id = "title",
},
"Heading 2"
);

Const heading2 = React.CreateElement(
"h2",
{
    id : "title",
},
"Heading 2"
);
```

React Element OR Object

```
Const Container = React.CreateElement(
  "div",
  {
    id: "Container",
  },
  ['heading1', 'heading2']
);
```

↑ passed array of Objects as Content

```
Const root = ReactDOM.CreateRoot(document.getElementById("root"));
root.render(Container);
```

~~react~~

Const root = react.createroot() ↑ doc.getelementbyid

root.render

All react code will run inside Root.

react.createElement("tag", {}, "Content")

↑ return object (React element)

↑ attribute to tag

(HW)

Chapter 01: Inception

EMMET : Short hand notation to generate Full Code of HTML & CSS

Set of Shortcut & abbreviations to generate Complex Code Structure

Library Vs Framework : (Scala Academy)

it takes minimum Effort for a library to put into Code

By using library, you can Control flow of program. Library can be Invoked whenever & wherever in Code

we can think of a framework as set of libraries

Library	Framework
↓	
Provide functions, & Classes	Complete Foundation or environment
we can Control flow of application	Framework takes Control over flow.

Code written on someone else pc or server

Libraries are hosted on CDN

CDN → Content Delivery Network.

- ✓ Improve Performance
- ✓ Speed
- ✓ reduced server load
- ✓ Manage high traffic

React :→

- ✓ JS library
- ✓ Build user interface
- ✓ Famous for SPA
- ✓ Allows use of reusability of code using Concept of Component.

React DOM →

- ✓ package that Connects React with DOM
- ✓ Renders React Component into dom.
- ✓ Helps Manipulating DOM using React.
- ✓ Consists of DOM Manipulation functions

React development.JS Vs React production.JS

Igniting our APP

we need bundlers, (parcel, VEET, web

it gives access to alot of superpowers

to get Parcell →

we need a package Manager

↳ npm, yarn

npm → Pakkage Manager

npm init -y → skip options

npm init

↳ will create a file Pakkage.json

To install parcell → --save dev

npm install -D parcell

↓

we want parcell into our development environment & Not for production

we got Pakkage lock.json

ReactDom : For dom manipulation

To install React, React-DoM

npm install React // or : npm i react

npm install React-DoM

to start APP (Started Server)

npx parcell index.HTML
Entry Point

Since now my react & react DOM are in node Module

Import React into APP.js →

Import react from "react";

Import react-DoM from "react-DoM/Client"

do 1 more thing →

< Script type = "Module" Scr = "App.js" >
in index.html

we need to minify, bundle things up, optimise our app so we need some packages to get the pakkages we use npm

Pakkge lock.json → locks the version of pakkages
if Pakkage.json updated to current version we will loose the track of the exact version we used earlier

Node module → Database for pakkges.

if any topic is not clear Re-watch the Video.

Benifits of using Parcell →

1. HMR → Hot Module ~~Rendering~~ Replacement
   ↳ file watcher Algorithm → C++/C which re render page when we edit something (live server)

Parcell with help of node Modules

2. Bundling
3. Minifying
4. Cleaning our Code → Removing console.logs
5. Superfast build
6. Image Compress/ optimization
7. Caching while Development
8. Compression
9. Compatible with older version of browsers
10. HTTPS on dev
    ↳ --https

Required for React (Parts)

React → Modi
Parcell → Amit Shah
other node Module → other BJP Ministers

React Cannot do everything Alone it need helps of parcel & parcell needs Helps of whole Ministry

Laying the Foundation

IMP Points →

Package.JSON

"browsersList": [

"last two version of Chrome"

]

it will definately Run on this browser version + All other browsers.

PolyFill →

A Code which is replacement of newer version of code

We can create fun function that seems same as that newer function

this is done by babel.

ex →

old browser: es 6 was not there

Const, new X is not known

we don't need to write Poly Fill

it is already done by babel.

Concept of keys →

whenever a parent has multiple childerns we should always give keys to the childern

example →

```
<div>
    <h1> Hello </h1>
    <h2> world </h2>
</div>
```

In React, (Giving key)

```
Const heading1 = React.createElement(
"h1",
{
    key : "head1",
    ID : "title",
},
"Heading 1"
);
```

First Child

```
const heading2 = React.CreateElement(
"h2",
{
    key: "Head 2"
    ID: "title"
},
"World"
);

Const Container = React.CreateElement(
"Div",
{
    id: Container,
},
[Heading1, Heading2]
);
```

Why keys?

Suppose we are using the same code twice, if keys are given react will just update that part if keys are not given React had to render the whole document again which will Make app Slow.

```
<ul>
  <li> first </li>
  <li> second </li>
</ul>

<ul>
  <li> first </li>
  <li> second </li>
  <li> third </li>
</ul>
```

⇒ it will Afford in end because of same Sequence

But,

render: Updating in DOM

re-render: Changing entire dom-tree

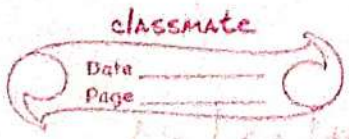
→
```
<ul>
    <li> Duke </li>
    <li> Villanova </li>
</ul>
```

→
```
<ul>
    <li> Connecticut </li>
    <li> Duke </li>
    <li> Villanova </li>
</ul>
```

Now, in this case if keys are not given React will have to re render the page (a lot of efforts)

if keys are given, it will just Render <li> Connecticut only otherwise it needs to re render all Content

Example with keys →



<li key = "2015" > Rohan

<li key>

More optimised, only "125" will be rendered

<li key = "123" > Rohan

<li key = "124" > Mithlesh

<li key = "125" > Akhil

<li key = "124" > Mithlesh

<li key = "123" > Rohan

Note: We pass keys as Props

17/07/23

HTML Like Syntax

Why JSX ?

Using React.createElement to build a complex structured code is hard

JSX allows us to write same code in less complex structure and less lines of code

Example →

Using React	Using JSX
Const heading =	Const heading = (
React.createElement(	<h1 id="title" key="heading">
"h1",	Namaste React
{ id : title,	</h1>
key : heading,	);
},	
"Namaste React"	
);	

Note : JSX is Not HTML inside JS
It is HTML Like Syntax.

classmate
Date
Page

Single line JSX Code → (Syntax)

```
Const heading = <h1> Hello </h1>
```

Multiline (Syntax)

```
Const heading = (
        // Code
);
```

Note : This Piece of Code is Called React Element

React Element Can be Created USing

→ React.CreateElement( )

→ JSX

IMP

React.createElement
↳ generates an object
↳ object is then Rendered on HTML DOM

Ugly, Less Readable
↳ JSX was built by Facebook developers

JS File. Execute this Code How?

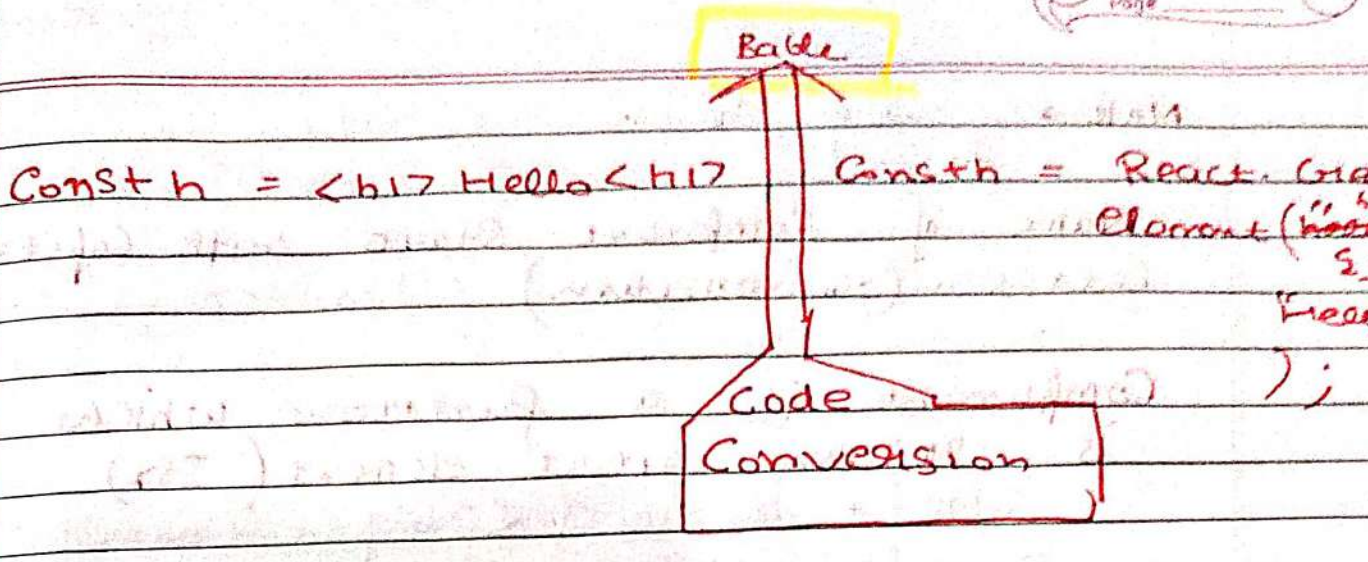
if we write JSX in browser Console it won't understand this Code

This Code is understood by Babel

Babel is Modern JS Compiler
↳ Code Converter

JSX → Babel → Normal Code

JSX inside Uses React.createElement
⇓
Gives object which is Rendered



JSX → Readability
No repetition
Maintained easily
Syntatical Sugar

JSX needs not to be imported because its just a Syntax

COMPONENTS In React

returns a react element → Functional Component (new)
→ Class Component (old)

* Functional Component → A JS Function

```
const HeaderComponent = () => {
    return <h1> Functional Component </h1>
};
```

Note →

Name of Component Starts with Capital letter (Convension)

Component is a function which is return react element (JSX)

When the Code is in Multiple lines we need to wrap it inside

(

);

in single line we don't need to put them.

two Ways of Creating Components. →

```
Const HeaderComponent = () => {
    return (
        <div>
            <h1>Hello</h1>
            <h2>Hi</h2>
        </div>
    );
};
```

OR (Just An Arrow Function thing)

~~Const HeaderComponent = () => {~~

```
Const HeaderComponent = () => (
<div>
    <h1>Hello</h1>
    <h2>Hi</h2>
</div>
);
```

9/2/24

Harkirat Week-6

☆ React Deep Dive 6.1

✱ React Component Return

A Component cannot return More than one element

because → it is easy to do reconsile for react when a component return One element.

✱ Step to create project →

npm create Vite@latest

Framework : React

Variant : JavaScript

open folder. run npm install
Now npm run dev to open in browser.

Re rendering → any time the dom is updated it is called re rendering

React developer tools → install this extension

TASK - 1

My name is Rohan

"title 2"

Click to update title

will update first line

```
function App() {
  const title = [title, setTitle] = useState("My name is Rohan")

  function SetTitleHandler() {
    setTitle("New Value")
  }

  return (
    <>
      <Header title={title} />
      <Header title="title 2" />
      <Button onClick={SetTitleHandler}>
        Update title
      </Button>
    </>
  );
```

Re-render

React did some work to calculate what all should be updated in the component

it happens when

1. A state variable got changed which is being used inside a component.
2. A parent component triggers re-rendering ^ re-rendering of all children components

Whenever parent rerender all children gets rendered again

Method 1 — this can be avoided by pushing state variable from parent component to individual or that children component

Method 2 — use React.memo

To use React Memo pass an existing component into React.memo(p) ;

example →

Before React.memo →

```
function Header({Title}){
  return(<div>
    {title}
  </div>
  )}
```

with React.memo()

```
Const Header = React.memo(function Header({Title}){
  return(
    <>{Title}</>
  );
});
```

Now Header Component will only re render if the props will be changed else it will not trigger a rerender when the parent component rerenders.

The parent will be rerendered but children using Memo will only be rerendered if the props gets changed.

Keys in React

Markdown SS

Javapoint

→ Iterating over a component
Useful when we dinamically created components or in case of lists
while rendering an array or MAP we need to mension key attribute

A key is a unique identifier. In React it is used to identify which items have changed, updated or deleted

It helps to determine which components in a collection needs to be rerendered instead of re-rendering entire set of components every time

example →

```
const list = ['Rohan', 'Akhil', 'Rohit']

list.

{list.map((index, listone) => {
  <li key={index}>{listone}</li>
})}
```

Harkirat example →

```
let counter = 4

function App() {
  const [todos, setTodos] = useState([{
    id: 1,
    title: "go to gym",
    Des: "Gym"
  },
  {
    id: 2,
    title: "Study",
    Des: "Study today",
  }
```

```
    {
      id : 3,
      title : "eat",
      Des : "r Eat food"
    }
  ])

function addtodo(){
  setTodos([...todos, {
    id : Counter++
    title : math.random()
    Des : math.random()
  }])
}
return(
  <>
  <button onclick={addtodo}>add todo</bu
  {todos.map(function(todo){
    <todo key={todo.id} title={todo.ti
      des={todo:des}/>
    }
  }
  </>
);
```

function Todo ({ title, Des }) {

return (

<>

<h1> {title} </h1>

<h5> {Des} </h5>

</>

)};

You need to pass in the key even it is not being used in that Component. I think keys helps in easy reconciliation

Wrapper Component

One Component which take other Component as input & Render it

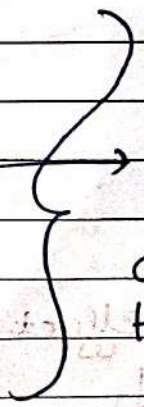
function App() {

return (<Card wrapper>

hi there

</Card wrapper>);

}



11/02/24

Week 7.1

[Routing, Prop drilling, Context API]

✗ Routing →

npm install react-router-dom

import {BrowserRouter} from 'react-router-dom'
{ routes } from "
{ route } from "

function App () {

return (

<BrowserRouter>
<Routes>
<Route Path = "/dashboard" element
={<Dashboard/>
/>

<Route Path = "/" element = {<Landing/>}

);

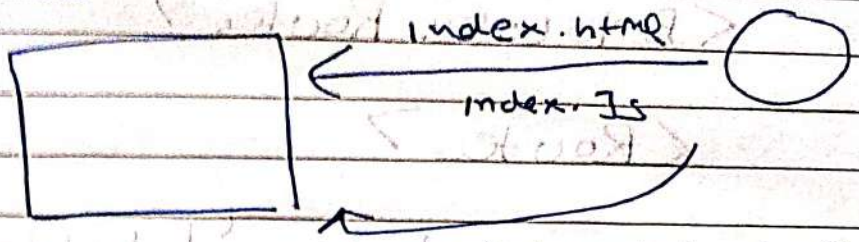
}

export default APP;

```
import { useNavigate } from 'react-router...
function App(){
  <BrowserRouter>
  Const navigate = useNavigate();

  return ( <Browser

    <div>
      <BrowserRouter>
      <button onClick={() => {
          navigate("/")
        }}> landing Page </button>

      <button onClick={() => {
          navigate("/dashboard")
        }}> landing Page </button>

    </div>

    <BrowserRouter>
      <Routes>
        <Route path="/dashboard" element={<Dash
        <Route path="/" element={<Landing/>}/>
      </Routes>
    </BrowserRouter>

  </div>
)}
```

Lazy loading →

Idealy



giving whole Files at once

lazy loding

User will get Code for landing Page at once

and whenever he will go to any other route files will be again fetched all Code will not brought together.

things to DO →

Const Dashboard = React.lazy(() => import("./-

↑

import like this

use default export

then you can use it in Route

→

```
function App(){
  return (
    <BrowserRouter>
      <Routes>
        <route path='/dashboard' element
          ={<Dashboard/>}
        }/>
      <Routes>
    </BrowserRouter>
```

How TO Implimant Navigation

good example

```
function App(){
  return (
    <div>
    <BrowserRouter>
      <Appbar/>
    <Routes>
      <Route path='/dash' element={ }/>
      <Route Path='/' element={<Login/>}/>
    </Routes>
    </BrowserRouter>
  </div>
)
}
```

Pass 6mb

```
function Appbar(){
  const navigate = useNevigate();
  return <div>
    <button onClick={()=>{
      navigate("/")
    }}> landing Page </button>

    <button onClick={()=>{
      navigate("/dashboard")
    }}> dashboard </button>
  </div>
}
```

Use Nevigate Hook Can only be used inside browser Router,

Prop Drilling

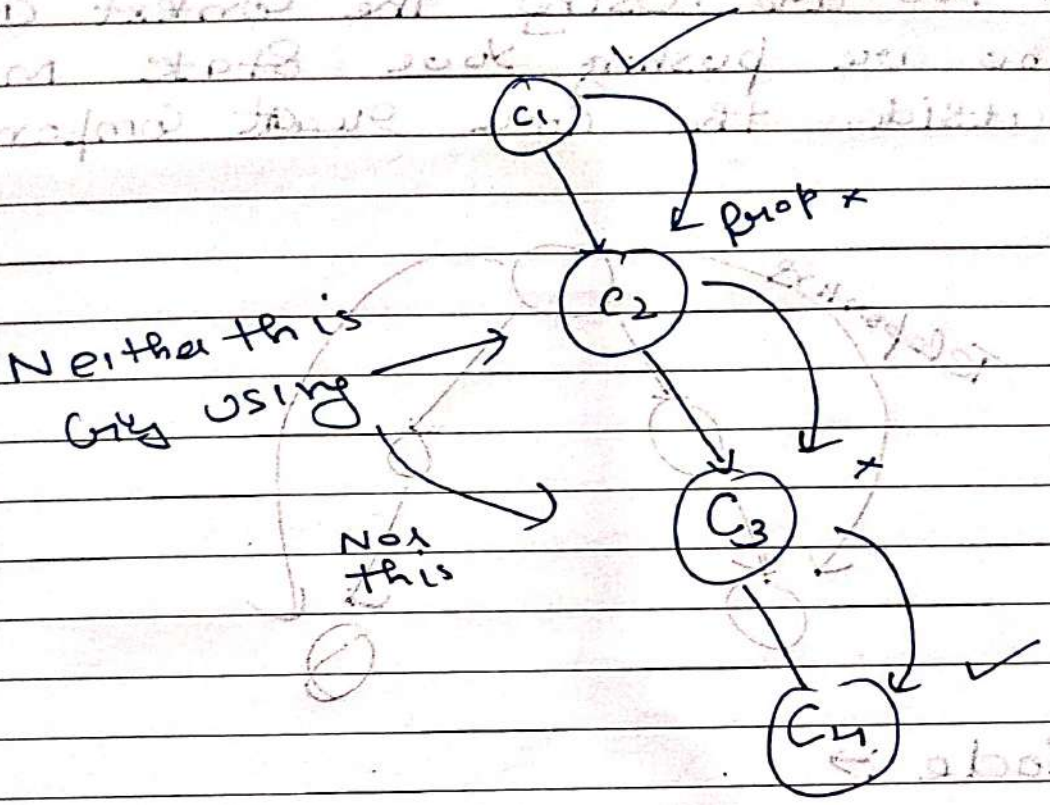
Example to understand →

```
function App() {
  Const [Count, SetCount] = use State(0)
  Return <div>
    <Count Count={Count} SetCount={SetCount} />
  </div>
}
```

drilling or passing a State Variable to children

```
function Button({Count, SetCount}){
  Return <div>
    <button onClick={() => {
      Setcount(Count + 1)
    }}> Increase </button>
  }

function Count({Count, setCount}){
  return <div>
    {Count}
    <Button Count={Count} SetCount={SetCount} />
  </div>
```

Count & SetCount are passed from App to Count and SetCount is not be used there it is being passed to another children of Count . this passing of a prop is called prop drilling



Prop passed from Component 1 to Component 4 through C_2 & C_3 which were not using it

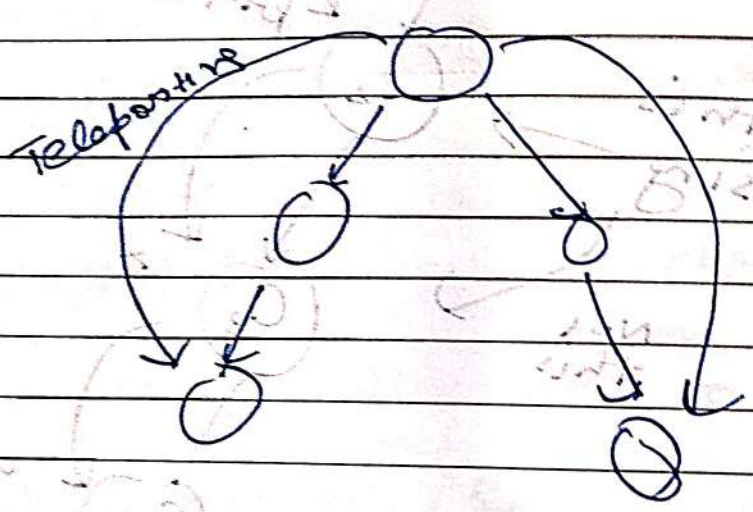
This a bad thing As it make the Code visually look unpealing to look.

Makes Code Highly unreadable

Context Api

To teleporte the data to components in a tree we use without drilling it down we use Context Api

If you are using the context api, you are pushing your State Management outside the Core react Component



Code :→

Context.jsx →

```
import { CreateContext } from 'react';

export Const CountContext = CreateContext(0)
```

17.2

State Management using Recoil

What is State Management ?

↳ Store the State (a Cleaner Manner

Recoil is a State Management library.

atom ⟶ use State

↑ ↑

get, update setCount

↑

less Rerenders

Cleaner Approach

Create → Store folder → atom folder → jsx File

inside src folder

export import { atom } from "recoil";

Const Countatom = atom({

Key : "Count Atom";

default : 0

});

to use it in a particular Component →

```
Const Count = UseRecoilValue(CountAtom)
```

~~Eor~~

```
Const [Count, SetCount] = useRecoilState(CountATom)
```

↑ Similar to useState

Proper Code →

```
function App() {
return
  <div>
    <RecoilRoot>
    <Count/>
    <RecoilRoot>
```

→ Anything that uses Recoil logic should be wrapped inside <RecoilRoot>

- Use State is good if it has to do with a Single component
- Recoil lets U use a Single State variable in Multiple Components.
- If You know a state variable needs to be defined and used inside a Same Component You Should use Use State.

npm install recoil

Things in Recoil

- ✓ Recoil Root
- ✓ atom
- ✓ useRecoilState
- ✓ useRecoilValue
- ✓ useSetRecoilState
- Selector

React Router - Dom !!

npm i react-router-dom

We have to create Routing Configration

import { createBrowserRouter } from "react-router-dom"

```
Const appRouter = createBrowserRouter([
{
 path: "/"
 element <App/>
}
```

```
{
  Path : "/about"
  Element : <About/> ,
},
{
  Path: "/Contact"
  Element : <Contact/>
}
])
```

// I need to provide it to render it
we need one More thing

```
import {RouterProvider} from "react-router-dom"

root.render(<RouterProvider router={appRouter} />)
```

35 Min

ERROR Page if wrong URL is entered

add errorElement : <Component />

↑ to Object

```
{ path : ——
  Element: ——
  ~~errorPage~~
  errorElement —— }
```

Hook for error component →

import { useRouteError } from 'react-router-dom'

Const err = useRouteError()

Returns object

More info about error

43.51

I want Header should be intact on all pages

↓

You have to create Childen Routes

```
Const App = () => {

  return (
    <div className='div'>
      <Header />
      <Outlet />
    </div>
  );
};
```

import {Outlet}

this will be filled with path which is opened in browser

```
const appRouter = createBrowserRouter([
{
  path: "/",
  element: <Applp/>
  children: [
  {
    path: "/"
    element: <body>
  },
  { path: '/about'
    element: <About>
  },
  {
    path: /Contact
    element: </Contact
  }
  ],
errorElement: <error/> ;
  },
]);
```

Same as href <a>

Link →

import {Link} from 'react-router-dom';

→
```
<li>
  <Link to="/about"> about </Link>
</li>
```

This will not reload the page

↳ Single Page Application

LETS Get Classy

Class Based Components

Creation →

```
class UserClass extends React.Component {
  render() {
    // Return JSX
  }
}
```

```
class UserClass extends React.Component {
  render() {
    return (
      <div>
        =
      </div>
    );
  }
}
export default UserClass;
```

↳ Same as Functional Component

Class which is rendering a piece of JSX

Usage is same as functional component

To Receive props in Class Based Component

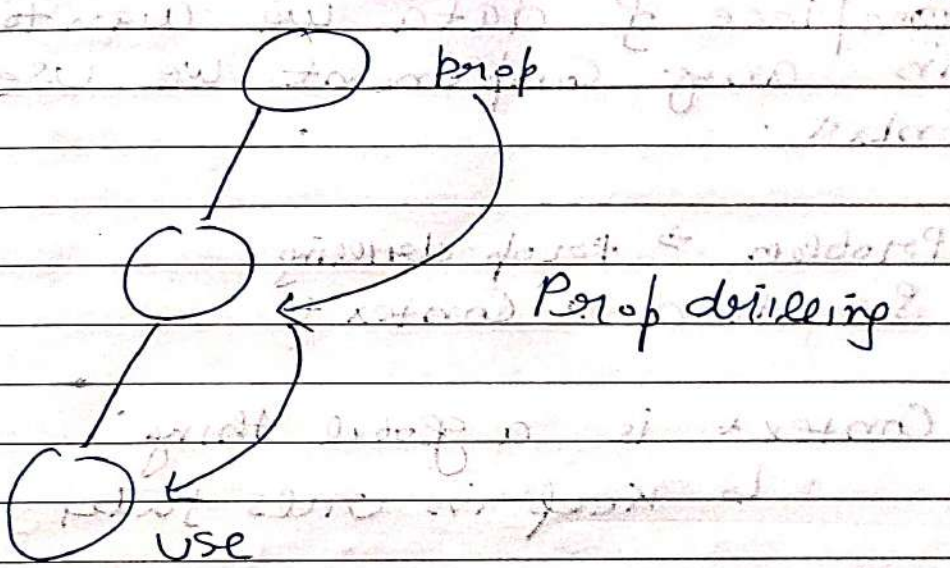
```
class userClass extends React.Component {
    constructor(Props) {
        super(props);
    }
    render() {
        return (
            <h2> {this.props.name} </h2>
        )
    }
}
```

Redux : State Management

Library :

Why State Mangement → (Managing data is crusial)

Suppose we want to use a data used in Parent Component to their Chidren. So we need to drill it to the Chidren



We want to keep it to a Central global place and use it any where we want without drilling it.

Now Suppose our data is present Some where else and we want it to Be used at Some where else.

React gives access to React Context

Its foolish to pass Props from a parent to ten level deep

We use Context
↳ global place where data is kept
↳ Anyone can access (any component)

Some piece of data we want to access in any component we use Context.

Problem → prop drilling
Solution → Context

Context is a global thing
↳ keep in utils folder

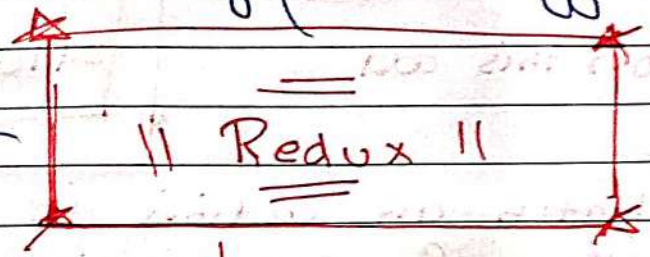
utils/UserContext.js

```
import { createContext } from 'react';
const UserContext = createContext({
    loggedInUser : "Default User",
});
export default UserContext;
```

Now, we can access it anywhere in app.

To Access →

```
import { useContext } from 'react';
import useContext from "../utils/useContext";
const data = useContext(useContext);
console.log(data.loggedInUser);
```



Redux vs react Redux

State Management Libraries
- React redux
- redux toolkit

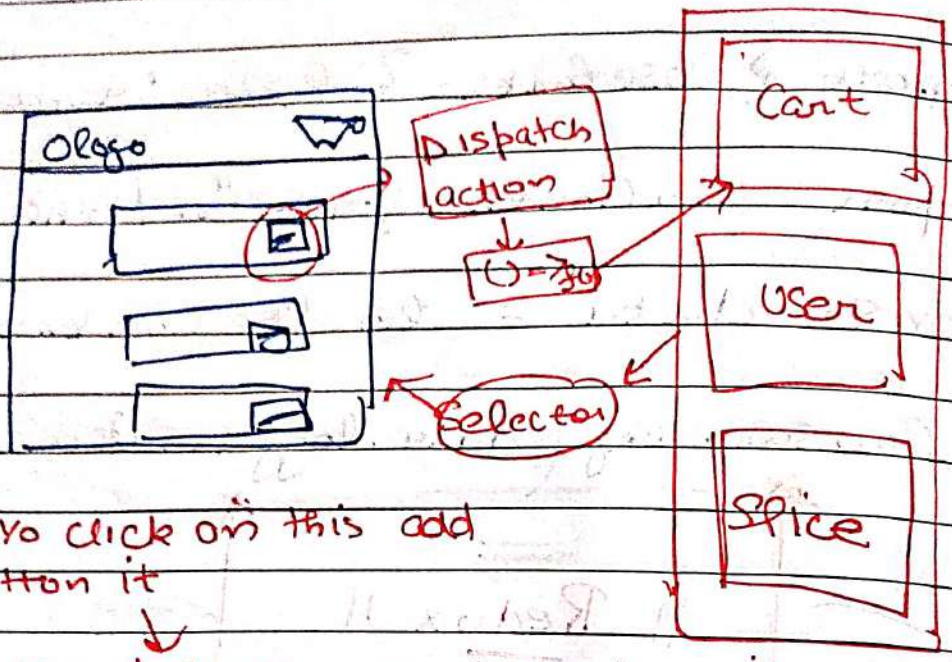
Redux Toolkit

↳ Set of utilities built on top of Redux aimed at simplifying & improving Redux development experience

It includes
- create slice
- configure store

less complicated than Redux

Architecture for Add to Cart



When Yo click on this add button it

When you dispatch an action it calls a function & that function will Modify the cart slice (the called function is known as reducer function.

Redux Store is like a big JS object kept at a central place

any Component can access it

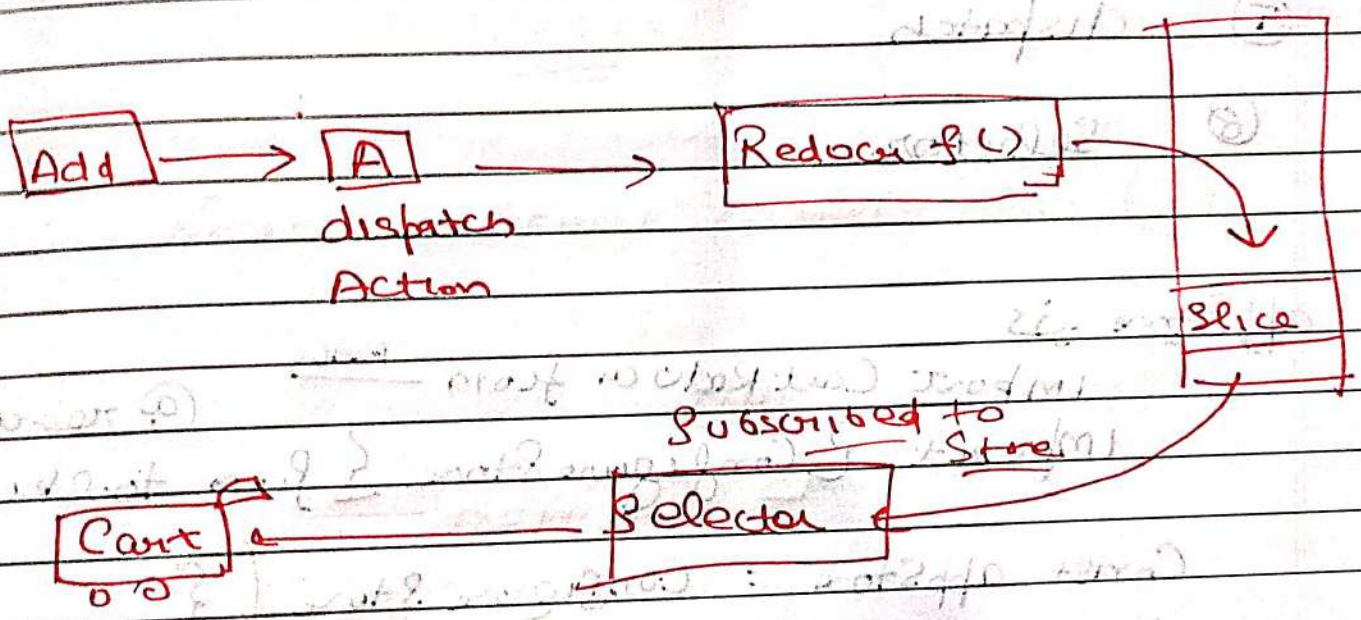
When You press Add button it dispatches an action which calls the reducer function which Updates the slice of redux Store

To read data from Cart

we use Selector

this Phenomeam is known as Subscribing to the Store.

↳ ie it is in Sync with Store whenever the data in Store changes it will trigger change in UI of component.



USAGE

① Install

↳ install @ reduxjs/toolkit & react-redux

② Build Store

③ Connecting Store to App

④ Slice

⑤ dispatch

⑥ Selector

appStore.js

```
Import CartReducer from Paths
Import { ConfigureStore } from @reduxjs/toolkit
Const appStore = ConfigureStore ({
  reducer : {
    Cart: CartReducer,
  },
});
Export default appStore
```

react-redux → Bridge between React & Redux

Now we need to provide Store to App.

```
<Provider store={appStore}>
    {where to use}
</Provider>
```

Slice →

```
Import { createSlice } from @reduxjs/toolkit

const cartSlice = createSlice({
    name: 'cart',
    initialState: {
        items: []
    },
    reducer: {
        addItems: (state, action) => {
            // Mutating the state
            state.items.push(action.payload);
        }
    }
```

```
removeItem: (state, action) => {
        state.items.pop();
    },
    clearCart: (state, action) =>
        state.items.length = 0;
    },
  },
});
export const {addItem, removeItem, clearCart} = cartSli
export default cartSlice.reducer;
```

export two things:

↳ Actions

↳ Reducer

→ Gives us Access to Store.

Subscribing To Store

We use a Selector (hook)

Reading Data →

```
Const Cartitems UseSelector(
    (Store) => Store.cart.items)
```

Updating Data ⇒

```
Const dispatch = UseDispatch()
~~dispatch~~
Onclick = { handleadditem }
```

```
Const handleAdditem = () => {
    dispatch( addItem("Pizza") )
```

↑ action.payload