# React JS

1. Find a suitable folder for land
2. npx create-react-app
3. follow (we suggest that you begin with)

* cd learning
* npm start
* code .

# Very important React update don’t allow render so

import React from 'react';

import ReactDOM from 'react-dom/client';

import App from './App';

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

root.render(

  <>

    <App />

  </>

);

# JSX (Look like HTML)

//if you want to use html in React

//by this method babel will compile the Code

import React from "react";

//if you want to use DOM in React

//ReactDOM.rendom(kiya likhna hai, kahan likhna hai) use 2 parameters

import ReactDOM from "react";

# Object and Array in JSX

<Fire data={["adil"]} text="Hi! i am Array" />

<Fire data={{name:"adil"}} text="Hi! i am Object" />

# React.Fragment

//will add extra div as a child of root

ReactDOM.render

(

    <div>

    <h1>Hello world</h1>

    </div>,

    document.getElementById("root")

)

//or

//not add extra div but still not feel good to read

//called react fragment

ReactDOM.render

(

    <React.Fragment>

    <h1>Hello world</h1>

    </React.Fragment>,

    document.getElementById("root")

)

//or

//it is also called react fragment

ReactDOM.render

(

    <>

    <h1>Hello world</h1>

    </>,

    document.getElementById("root")

)

# Challenge no 1

import React from 'react';

import ReactDOM from 'react-dom';

//it is also called react fragment

ReactDOM.render

(

    <>

    <h1>Hello world</h1>

    <p>paragraph</p>

    <ol>

      <li>Games of Throne</li>

      <li>Suits</li>

      <li>Flash</li>

      <li>Vikings</li>

      <li>Under-thrill</li>

    </ol>

    </>,

    document.getElementById("root")

)

# JavaScript inside JSX

//simple Array

const a = ["adil","younas"]

ReactDOM.render

(

  <>

  <h1>Hello world {a[0]} </h1>

  </>,

  document.getElementById("root")

)

//simple object

const obj = {

name:"adil",

father:"younas"

}

ReactDOM.render

(

  <>

  <h1>Hello world {obj.father}</h1>

  </>,

  document.getElementById("root")

)

//Number function

ReactDOM.render

(

  <>

  <h1>Hello world {Math.random()}</h1>

  </>,

  document.getElementById("root")

)

# JavaScript Template literals inside JSX

const name = "Adil"

const profession = "Web Development"

ReactDOM.render

(

  <>

  <h1>{`Hello world my name is ${name}

  and my profession is ${profession}`}</h1>

  </>,

  document.getElementById("root")

)

# Challenge 2

const name = "adil"

const profession = "Web Development"

const date = new Date()

ReactDOM.render

(

<>

<h1>{`Adil`}</h1>

<p>{`Currunt Date is : ${date.toLocaleDateString()}`}</p>

<p>{`Currunt Time: ${date.toLocaleTimeString()} `}</p>

</>,

document.getElementById("root")

)

# Self-closed Tag, attribute in JSX

* <img /> is called Self-closed Tag
* Attribute must be in CamelCase (like contentEditable)
* Attributes except Style can be written by src = “something”

const image = <https://i.pinimg.com/200x/96/14/41/961441ac981ada78f04b0ec5e1503e21.jpg>

ReactDOM.render

(

<>

<h1 contentEditable="true">{`Adil`}</h1>

<a href='https://www.google.com' target="\_blank">

  <img src= {image} alt= "image"/>

</a>

</>,

document.getElementById("root")

)

# CSS Styling in React

External Styling

* It’s important to import CSS file for the sake of use
* className should be in camelCase convention

//it is a relative path

import "./index.css"

ReactDOM.render

(

<>

<div className='container'>

<h1 className='heading' id>{`Adil`}</h1>

</div>

</>,

document.getElementById("root")

)

Inline Styling (not best)

* Inline Styling Required Object { { } } as style property as in following

const heading = {

    color:"red",

    text-transform: "Capitalize"

}

<div style="color:red;">

//VS

<div style= { {color:"red",textTransform: "uppercase"} }  >

Best Inline Styling

* As we know style attribute require object as a property then

const heading = {

    color:"red",

    textTransform: "uppercase",

    textAlign:"center"

}

ReactDOM.render

(

<>

<h1 style= { heading }>{`Adil`}</h1>

</>,

document.getElementById("root")

)

# Challenge 3

* Create a react app from Scratch
* Show a heading that says Hello Sir, Good Morning, if time is between 1 to 11am
* Good afternoon, if 12pm to 7pm
* Good night, if 7pm till Midnight
* Apply CSS in it, then dynamically change the color of greeting parts only using inline CSS style. Ex. Green, orange, black etc…..

import React from 'react';

import ReactDOM from 'react-dom';

//it is a relative path

import "./index.css"

let greetingStyle = { }

//to check code is working or not use

// new Date(2022,9,27,14)

let date = new Date()

date = date.getHours()

let greeting = ""

if (date >= 0 && date <= 12) {

    greeting = "Good Morning"

    greetingStyle.color = "green"

} else if(date > 12 && date <= 19){

    greeting = "Good Afternoon"

    greetingStyle.color = "orange"

}else{

    greeting = "Good Night"

    greetingStyle.color = "black"

}

ReactDOM.render

(

<>

<div className='container'>

    <h1 className='h1'>Hello Sir, <span style={greetingStyle} >  {greeting} </span> </h1>

</div>

</>,

document.getElementById("root")

)

With Switch Statement

import React from 'react';

import ReactDOM from 'react-dom';

//it is a relative path

import "./index.css"

const greetingStyle = {}

//to check code is working or not use

// new Date(2022,9,27,14)

let date = new Date()

date = date.getHours()

let greeting = ""

switch (true) {

    case (date >= 0 && date <= 12):

        greeting = "Good Morning"

        greetingStyle.color = "green"

        break;

    case (date > 12 && date <= 19):

        greeting = "Good Afternoon"

        greetingStyle.color = "orange"

        break;

    default:

        greeting = "Good Night"

        greetingStyle.color = "black"

        break;

}

ReactDOM.render

    (

        <>

            <div className='container'>

                <h1 className='h1'>Hello Sir, <span style={greetingStyle} >  {greeting} </span> </h1>

            </div>

        </>,

        document.getElementById("root")

    )

Output



# Export & Import

Export

let youtuber = "adil younas"

let youtuber2 = "Thapa Technical"

let youtuber3 = "Code with harry"

//for default value if there is no default then it is OK

export default youtuber;

//for multiple values

export {youtuber2,youtuber3}

Import

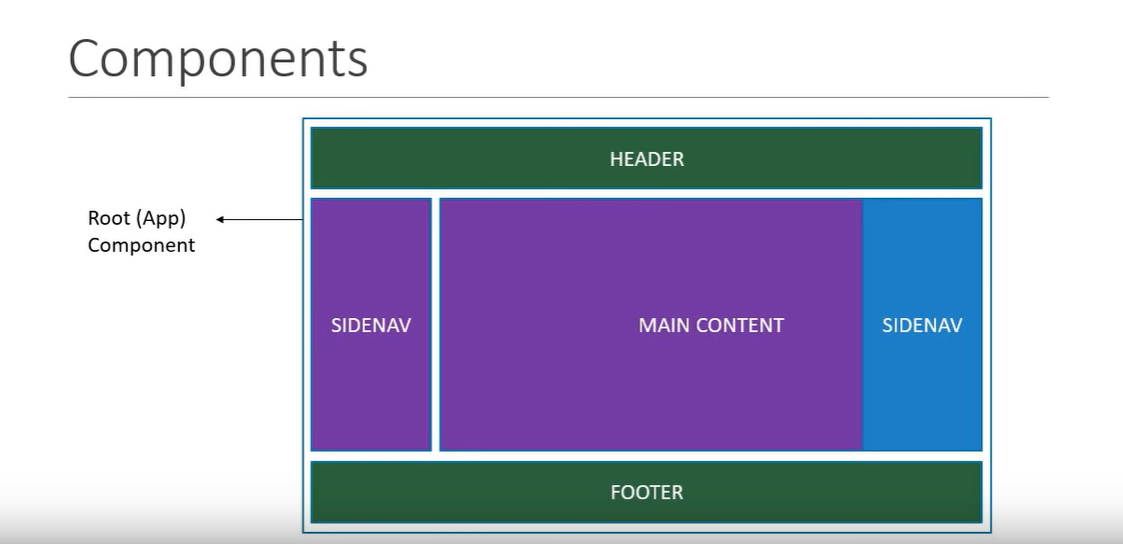
//it is a relative path

//youtube for default export

//{youtube2,youtuber3} for all

import youtuber,{youtuber2,youtuber3} from "./App"

# Now what Is component??



# Types of components

1. class components
2. functional components

class component

class Fire extends React.Component{

    render(){

        return(

            <>

                <h1>Fire breathing</h1>

            </>

        )

    }

}

Functional component

function home(){

    return(

        <>

            <h1>Home of Adil</h1>

         </>

    )

}

# Props

* In ReactJS, the props are **a type of object where the value of attributes of a tag is stored**. The word “props” implies “properties”
* Props is like a object of Attributes simply

Props in functional components

Note: Here you have to pass parameter of any name to receive data

Note2: Props call every element

Note 3 : Props are unchangeable

<Home data={["adil"]} text="Hi! i am Array" />

<Home data={{name:"adil"}} text="Hi! i am Object" />

export default function home(prop){

    return(

        <>

            <h1>{prop.data[0]}</h1>

            <h1>{prop.data.name}</h1>

            <h1>{prop.text}</h1>

         </>

    )

}

Another Example of Props

import React from "react";

const Obj = [{

    image:"https://wallpapercave.com/wp/wp4589844.jpg",

    Title:"Demon Slayers",

    available:"NetFlix",

    link:"https://www.w3schools.com/howto/howto\_css\_blurred\_background.asp"

}]

export default Obj

import React from "react";

import ReactDOM  from "react-dom";

import Card from "./Cards"

import Obj from "./data"

ReactDOM.render(

    <>

        <Card imageSrc = {Obj[0].image}

            title = {Obj[0].Title}

            availAbleOn = {Obj[0].available}

            Links = {Obj[0].link}

        />

    </>,document.getElementById("root")

)

import React from "react";

import "./index.css"

function Card(props){

    return(

        <>

    <div className="main">

        <div className="container">

            <div className="cardimage">

                <img src= {props.imageSrc} alt="picture"/>

            </div>

            <div className="cardinfo">

                <h1>{props.title}</h1>

                <p>{props.availAbleOn}</p>

                <button><a href={props.Links}>Watch Now</a></button>

            </div>

        </div>

</div>

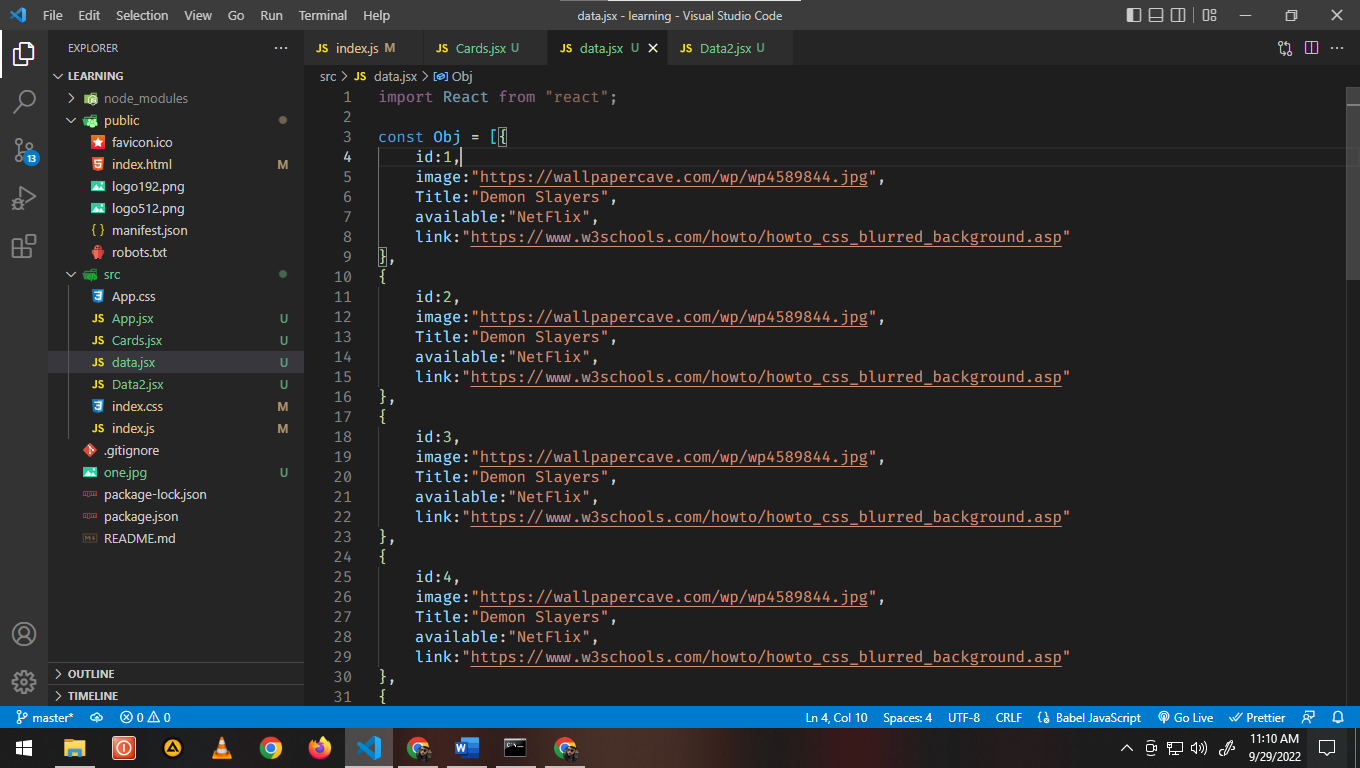
        </>

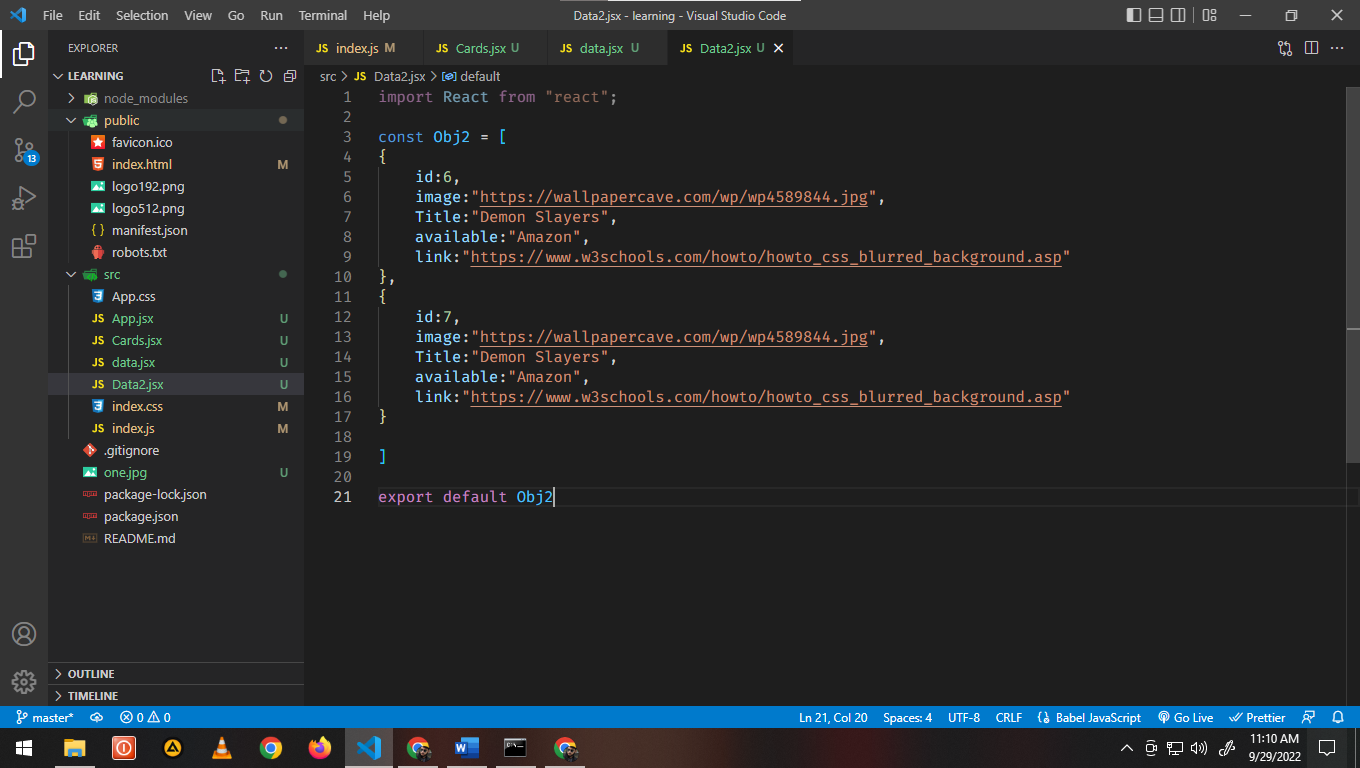
)}

export default Card;

# Sorting / Search At last using props

* It is ok to have different files, consist data in the form of Array of an Object





import React from "react";

import ReactDOM  from "react-dom";

import Card from "./Cards"

import Obj from "./data"

import Obj2 from "./Data2"

function myFunction(val,index){

    return(

        <Card

            key={val.id}

            imageSrc = {val.image}

            title = {val.Title}

            availAbleOn = {val.available}

            Links = {val.link}

        />

    )

}

let select = "Amazon"

// let select = "Netflix"

if(select == "Amazon"){

    ReactDOM.render(

        <>

            {Obj.map(myFunction)}

        </>,document.getElementById("root")

    )

}else if(select == "Netflix"){

    ReactDOM.render(

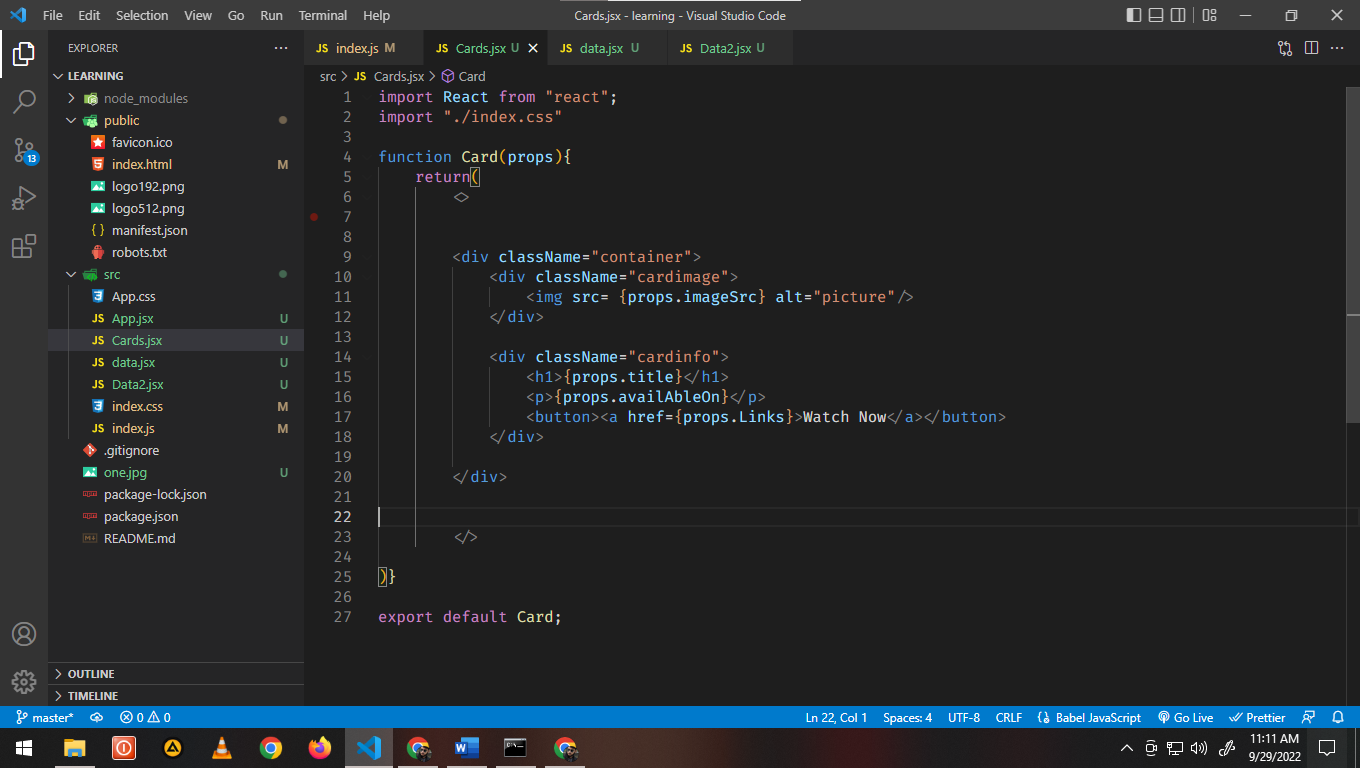
        <>

            {Obj2.map(myFunction)}

        </>,document.getElementById("root")

    )

}



Practice the Above as Complete for better understanding next

# Hooks & Rules

* Hooks is used in a functional component
* Functional component must be in PasCal convention
* Don’t call hooks inside loops, conditions, or nested functions
* React Hooks are **simple JavaScript functions that we can use to isolate the reusable part from a functional component**
* Hooks are the new feature introduced in the React 16.8 version. It **allows you to use state and other React features without writing a class**. Hooks are the functions which "hook into" React state and lifecycle features from function components. It does not work inside classes.

useState Hook (return same present in it)

Syntx:

import React,{useState} from "react";

const [name,setName] = useState("Adil")

Example 1 Toggle of useState Hook:

import React,{useState} from "react";

import ReactDOM from "react-dom";

import App from "./App"

ReactDOM.render(

  <>

    <App />

  </>,document.getElementById("root")

)

import React,{useState} from "react";

const [name,setName] = useState("Adil")

function App() {

  function myFunction(){

  let val = name

  if (val === "Adil") {

    setName("Aqil")

  } else if(val === "Aqil") {

    setName("Abeeha")

  } else if(val === "Abeeha"){

    setName("Younas")

  }else if(val ==="Younas"){

    setName("Adil")

  }

}

  return(

    <>

      <h1> {name} </h1>

      <button onClick={myFunction}>Click me</button>

    </>

  )

}

export default App;

# Use Array in useState Hook

* According to my information array can’t be update through useState() dynamically

Example 2 **Erase data** of useState Hook:

import React,{useState} from "react";

import ReactDOM from "react-dom";

import App from "./App"

ReactDOM.render(

  <>

    <App />

  </>,document.getElementById("root")

)

import React, { useState } from "react";

function App() {

  const Obj = [

    {

      id: 1,

      name: "Adil",

      father: "Younas",

      age: 25,

    },

    {

      id: 2,

      name: "Adil",

      father: "Younas",

      age: 25,

    },

  ];

  function myFunction() {

    setName([]);

  }

  const [name, setName] = useState(Obj);

  return (

    <>

      {name.map((val) => {

        return (

          <h1 key={val.id}>{`name is ${val.name}

                              father name is ${val.father}

                                and age is ${val.age}`}</h1>

        );

      })}

      <button onClick={myFunction}>Clear</button>

    </>

  );

}

export default App;

Example 3 **Simple Todo list using** useState Hook:

import ReactDOM from "react-dom";

import React from "react";

import App from "./App"

ReactDOM.render(

  <>

    <App />

  </>,document.getElementById("root")

)

import React, { useState } from "react";

function App() {

  const Array = [

    {

      id: 0,

      name: "Adil",

      father: "Younas",

      age: 25

    },

    {

      id: 1,

      name: "Aqil",

      father: "Younas",

      age: 22

    },

    {

      id: 2,

      name: "Rizwan",

      father: "Saleem",

      age: 24

    }

  ]

  const [data, setData] = useState(Array)

  function clearFunction(id) {

    const myNewArray = data.filter((Element) => {

      return Element.id !== id;

    })

    setData(myNewArray)

  }

  return (

    data.map((value) => {

      return (

        <h1 key={value.id}> Name: {value.name} Father: {value.father} age: {value.age}

          <button onClick={() => clearFunction(value.id)}>Click </button>

        </h1>

      )

    })

  )

}

export default App;

Example 4 Short circuit useState Hook:

* Or operator || show the left side if condition is true
* And operator && show the right side if condition is true

import React, { useState } from "react";

function App() {

const [data, setData] = useState("s")

return(

  <>

    <h1>name: {data || "Rizwan"}</h1>

    <h1>name: {data && "Rizwan" }</h1>

  </>

)

}

export default App;

# Use Object in useState Hook

import ReactDOM from "react-dom";

import React from "react";

import App from "./App"

ReactDOM.render(

  <>

    <App />

  </>,document.getElementById("root")

)

import React, { useState } from "react";

function App() {

const [data, setData] = useState(

{

name: "adil",

gender: "male",

age: 25,

id: 1

}

);

function myFunction() {

setData({ ...data, name: "kaboo" })

}

return (

<>

  <h1>{`name is ${data.name} and age ${data.age}`}</h1>

  <button onClick={myFunction}>click me</button>

</>

)

}

export default App;

# Controlled Form using useState hook + validation through if else

import React, { useState } from "react";

function App() {

const [email, setEmail] = useState("")

const [password, setPassword] = useState("")

const [newEntry, setNewEntry] = useState([])

const submitFunction = (e)=>{

  if (email && password) {

    e.preventDefault()

    const allEntries = {email:email,password:password }

    setNewEntry([...newEntry,allEntries])

    setEmail("")

    setPassword("")

    // console.log(allEntries);

  } else {

    alert("fill the data in field")

  }

}

return(

  <>

<form action="" onSubmit={submitFunction}>

  <label htmlFor="Email">Email</label>

  <input type="email" value={email}

  onChange={(e)=> setEmail(e.target.value) } />

  <label htmlFor="password">Password</label>

  <input type="password" value={password}

  onChange={(e)=> setPassword(e.target.value) } />

  <button type="submit" >submit</button>

</form>

  </>

)

}

export default App;

Practice the Above as Complete for better understanding next

# useEffect Hook in React

* useEffect() can’t be use in condition, but condition can be use in it
* can’t use async await directly with useEffect()
* Use in a component, when the page if load and component do all things after that effect hook show its face which is called side effect of effect hook.
* useEffect() render when all work is done this behavior is called side effect of useEffect() as mention above.
* As we all know that useSate() re-render the component, so when useState() re-render the component useEffect() will also re-render.

useEffect dependency list Array

const [count,setCount] = useSate(0)

* [ ] means on page load useEffect() run for one time
* [count] means on count change useEffect() will run again and again

Clean Up function for EventListner

<https://www.youtube.com/watch?v=5gCtW7RCtQA&list=PLwGdqUZWnOp0ningKFp1aslpZkCDAzML8&index=13&ab_channel=ThapaTechnical>

Simple increment and decrement in title by useEffect hook

import React, { useState, useEffect } from "react";

function App() {

  const [count, setCount] = useState(0)

  useEffect(() => {

    if (count === 0) {

      document.title = ``

    } else {

      document.title = `Check(${count})`

    }

  }, [count])

  return (

    <>

      <h1>{count}</h1>

      <button

      onClick={() => setCount(count + 1)}>Increment</button>

    </>

  )

}

export default App;

Important Project to DO

Manipulate user data through fetch() API using useEffect() + async await + Icons + Loading…….

* Open react icon website
* Install npm react package
* Import it and use it

import React, { useState, useEffect } from "react";

import Create from "./Create";

import { AiFillAlipaySquare } from "react-icons/ai";

function App() {

  const [data, setData] = useState([]);

  const [loading,setLoading] = useState(true)

  const myFunction = async () => {

    try {

      const response =

      await fetch("https://jsonplaceholder.typicode.com/users");

      const jsonData = await response.json();

      setData(await jsonData);

      setLoading(false)

    } catch (error) {

      console.log(error);

    }

  };

  useEffect(() => {

    myFunction();

  }, []);

if(loading === true){

  return <Create  />

}else if(loading === false){

  return (

    <>

      {data.map((val) => {

        return (

          <>

            <h1>{val.name} <AiFillAlipaySquare /> </h1>

            <p>{val.username}</p>

            <p>{val.email}</p>

            <p>{val.address.street}</p>

          </>

        );

      })}

    </>

  );

}

}

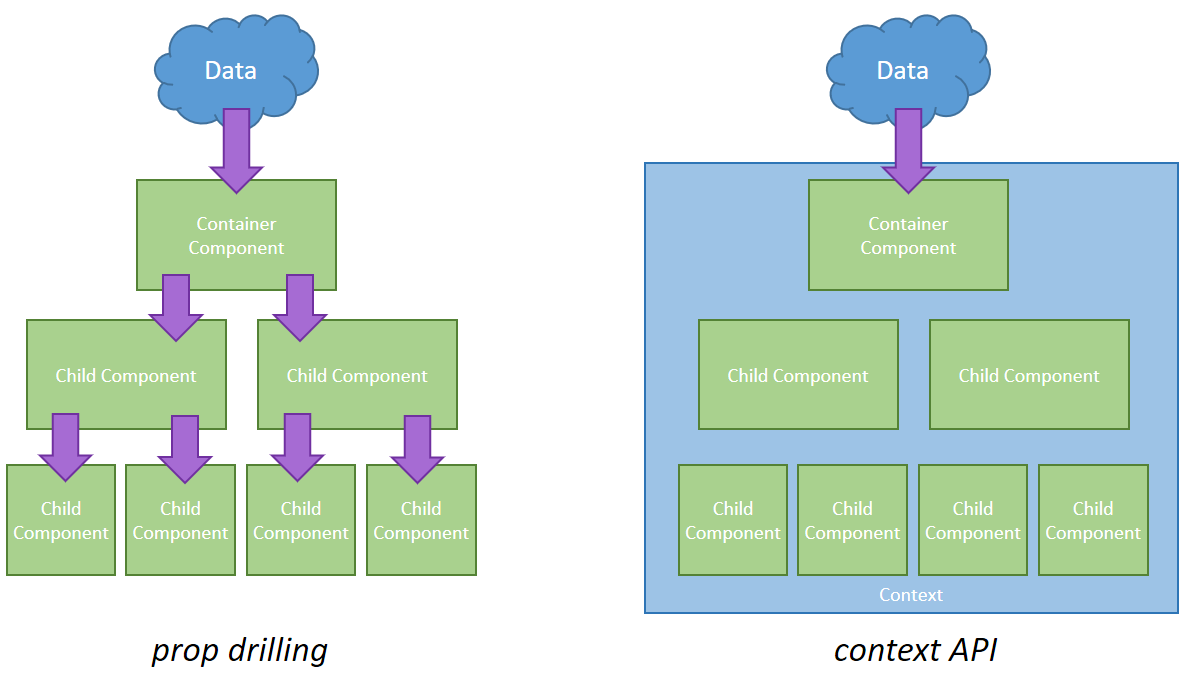
export default App;

# useReducerHook()

<https://www.youtube.com/watch?v=VdXGIEYZuCw&list=PLwGdqUZWnOp0ningKFp1aslpZkCDAzML8&index=18&ab_channel=ThapaTechnical>

# useContext() Hook

props drilling



* Props drilling is a way through which we pass data from component present up level in hierarchy to the last component present in hierarchy or somewhere between both, by passing data each element present in that hierarchy
* Props drilling is a problem for developers so they use context API through which they can pass data without passing to all middle ones

Context API

1. Create context

//instead of biodata you can write anything.

import React, {createContext} from "react";

const BioData = createContext()

1. Provider (Provider pass data to component B and all below it)

//after providing value to provider export BioData component

import CompB from "./CompB"

import React, {createContext} from "react";

const BioData = createContext()

const Obj = {

    name:"adil",

    father:"younas"

}

function CompA(){

    return (

        <BioData.Provider value = {Obj}>

             <CompB />

        </BioData.Provider>

    )

}

export default CompA;

export {BioData}

1. Consumer (consumer code is though so we use useContext Hook to replace)

import React, { useContext } from "react";

import { BioData } from "./CompA";

function ComC(){

    const Data = useContext(BioData)

    return <h1>{Data.father}</h1>

}

export default ComC;

# React Router

<https://www.youtube.com/watch?v=2x2zp7xnq7s&list=PL8p2I9GklV46L0l9WB6wBYV87j8AcraJX&index=4&ab_channel=CodeStepByStep>

* Router is use to switch pages, if you use react router then it is called client-side routing means the files server give you are switching VS In normal websites we use router which render data from server again and again.

1. Import {BrowserRouter, Routes, Route} from react-router-dom in App.js page
2. In self close tage which is Route in our case, element must be equal to component

Formula

//1

import {

  BrowserRouter as Router,

  Routes,

  Route

} from "react-router-dom";

Always use Router > Routes > Route and navbar inside the Router like this

 <Router>

        <Navbar />

        <Routes>

          <Route exact="true" path="/" element={<Home />} />

          <Route exact="true" path="/service" element={<Service />} />

          <Route exact="true" path="/contact" element={<Contact />} />

          <Route exact="true" path="/about" element={<About />} />

          <Route exact="true" path="/\*" element={<ErrorPage />} />

        </Routes>

      </Router>

* Allways wraped NavLink into Router

React 6.0 router update

import React from 'react';

import ReactDOM from 'react-dom/client';

import App from './App';

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

root.render(

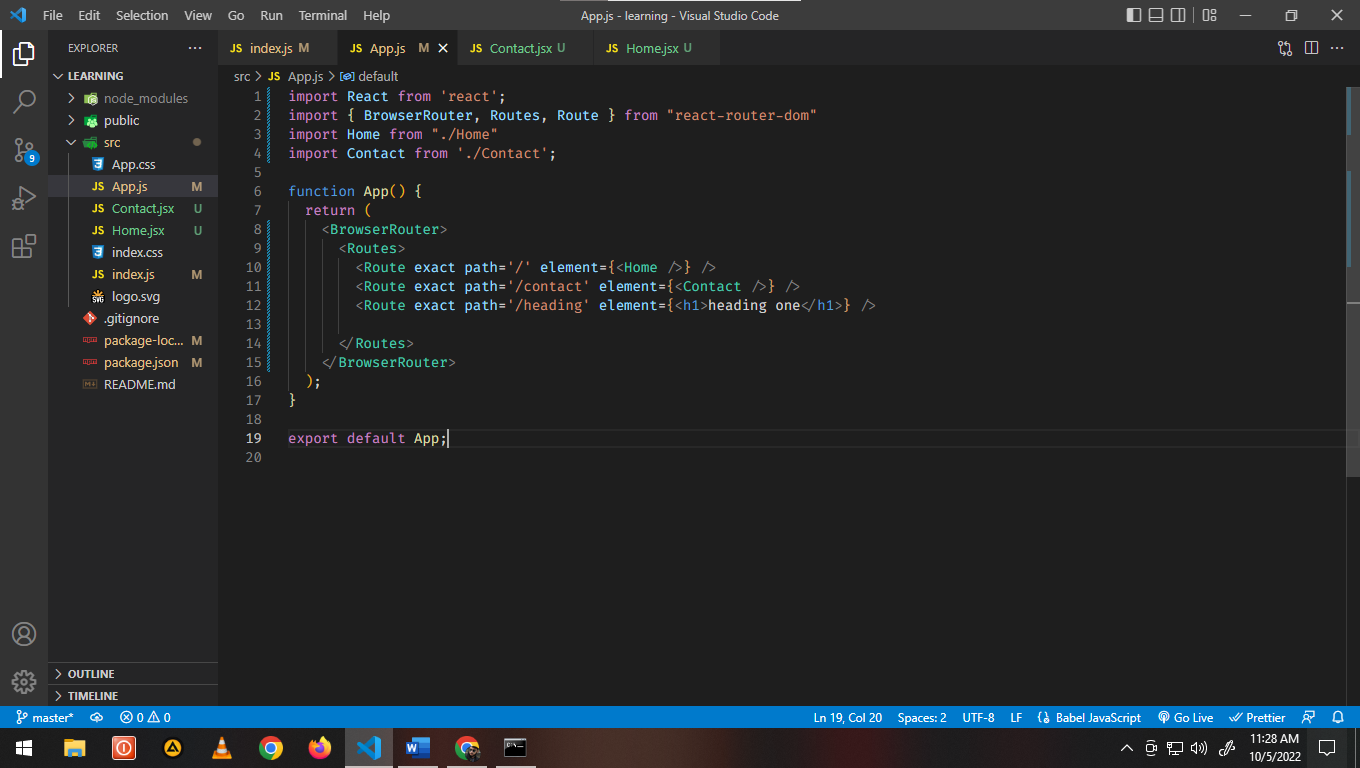
  <>

    <App />

  </>

);

Example of Base link



import React from 'react';

import { BrowserRouter, Routes, Route, Link } from "react-router-dom"

import Home from "./Home"

import Contact from './Contact';

function App() {

  return (

    <BrowserRouter>

      <Routes>

        <Route exact path='/' element={<Home />} />

        <Route exact path='/contact' element={<Contact />} />

        <Route exact path='/heading' element={<h1>heading one</h1>} />

      </Routes>

    </BrowserRouter>

  );

}

export default App;

Example for Navbar

import React from "react";

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

function Navbar(){

return(

    <>

        <ul>

            <li><Link to= "/" >Home</Link>  </li>

            <li><Link to= "/contact" >contact</Link>  </li>

        </ul>

    </>

)

}

export default Navbar;

Example for 404 Error page or Home default

import React from 'react';

import { BrowserRouter, Routes, Route, Link } from "react-router-dom"

import Home from "./Home"

import Contact from './Contact';

import Navbar from './Navbar';

import Page from './Page404';

function App() {

  return (

    <BrowserRouter>

    <Navbar />

      <Routes>

        <Route exact path='/' element={<Home />} />

        <Route exact path='/contact' element={<Contact />} />

        <Route exact path='/\*' element={<Home />} />

        <Route exact path='/heading' element={<h1>heading one</h1>} />

      </Routes>

    </BrowserRouter>

  );

}

export default App;

Dynamic routing with params

NavLink Active

* When ever you click on any link line home, contact in navbar you see a active class automatically added in it. So, we just put style in parent class of home, contact etc. in active class and active class will reflect the style
* Home by default does not leave active class
* So style active class in css and the problem will be used

useSearchParams hook

* To understand useSearchParams hook first you understand the parameters in URL after ?age=10&city=gujranwala
* Just like useState hook in syntax wise but import from react-router-Dom instead of react
* It has many methods but here we use only two get and set
* setSearchParam and setState are not function they are side kicks means they are callback function
* setSearchParam has parameters in form of object like setSearchParam({age:10}) to set age in parameter

const [searchparam,setSearchParams] = useSearchParams()

    console.log(searchparam.get("10"));

useNavigation Hook

* it is use for buttons etc
* it is used as a call back first import it from react-router-dom
* store it in variable and use it

import React from "react";

import { useNavigate } from "react-router-dom";

import Navbar from "./Navbar";

function Contact(){

const navigate = useNavigate()

    return(

        <>

            <Navbar />

        <h1>contact page</h1>

        <button onClick= {()=> navigate("/") }>click to navigate</button>

        </>

    )

}

export default Contact;

# Redux + Redux Again

State reducer k andar he hoti hai or reducer se ho gi or action reducer ko triger kry ga

Note: payload is not mention in this document so that to use payload you have to use useDispatch you can pass the payload and receive it in reducer if type: “increment” you will receive it in increment

Note 2: In reducer file you will notice an object after initial state object look like

{

increment: ()=>{ }

}

So you can write javascript in it but under dispatch you notice object with object elements

1. Create store by toolkit

import {configureStore} from "@reduxjs/toolkit"

const store = configureStore({ reducer:{}})

2.Wrap App by Provider by react-redux

import {Provider} from "react-redux"

import store from "./redux/store"

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

root.render(

  <React.StrictMode>

  <Provider store={store}>

    <App  />

  </Provider>

  </React.StrictMode>

);

3.Create slice from toolkit

* initialState spelling is fixed
* use reducers
* The last two exports are important

import { createSlice } from "@reduxjs/toolkit";

let initialState = {

  address:"",

  pinCode:"",

  phoneNo:"",

  state:"",

  country:"",

  city:""

};

const shippingInfoSlice = createSlice({

  name: "ShippingIno",

  initialState,

  reducers: {

    shippingInfoAction: (state, action) => {

      state.ShippingInfo = action.payload;

    },

  },

});

export default shippingInfoSlice.reducer;

export const { shippingInfoAction } = shippingInfoSlice.actions;

4.Add reducer in store

Here first is a state

import shippingInfoSlice from "./Redux/Reducers/ShippingInfoReducer";

import { configureStore } from "@reduxjs/toolkit";

const store = configureStore({

  reducer: {

    productSlice,

    userSlice,

    cartSlice,

    shippingInfoSlice

  },

});

export default store;

5. import useDispatch from “react-redux”

* Actually useDipatch() trigger the action in state and change state
* data is sending the action the payload

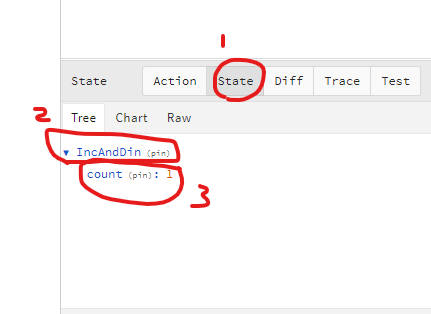
import { shippingInfoAction } from "../Reducers/ShippingInfoReducer";

const dispatch = useDispatch()

dispatch(shippingInfoAction(data));

6. import useSelector form “react-redux”

* Through useSelector I am calling a function who has state (state)=>{ }



* Actually it the the way through you draw anything from the store
* And state show here that you want data from state

import React from "react";

import { useDispatch } from "react-redux";

import {useSelector} from "react-redux"

const Home = () => {

  const dispatch = useDispatch();

  const {count} = useSelector(state=>state.IncAndDin)

