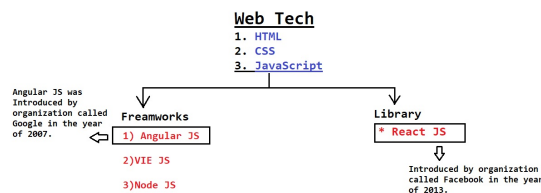




React JS - Prajwal Sir

▼ Note

- HTML is used to create a structure for a Web Page.
- CSS is used to design the Web Page.
- JS is used to add functionality and make Web Page dynamic.
- The only way to see the JavaScript Output is Browser. and There is Another platform to run on JavaScript code which is Node JS. And for Node JS Express JS is Freamework. (Advance to Advance)
- There is good salary for MARN and MEAN developers (MARN →) and (MEAN →)



1. Installation ?

▼ Ans

1. Download Node.js from Google.
2. Install Node.js
3. Create a Separate folder for Node.js.
4. Close all the Tabs and Open CMD in that particular React folder.
5. Pass the command (npx create-react-app ProjectName) in CMD.
6. It will download some backfiles and It will give message called 'Happy Hacking'.
7. Pass the command (cd ProjectName)
8. Pass the command (npm start) —>(We will getting default User Interface with the React Logo).
9. Type (code .) → To get Visual Code

2. What is React JS ?

▼ Ans

- React JS is a Library of JavaScript, where Library consist of collection of Pre-defined codes. by using these codes we are going to create Web pages. Extension is .jsx → Javascript and XML → Xtensible Markup Language
- Using HTML, CSS, JS. We can create Proper, Structural, Beautiful, Functionality Web Page. But, In order to make Web Page Perform better and Faster, We are going to use React JS and also by using React JS we can develop Single Page Application.
- Collection of Libraries is called Framework.
- React JS was Introduced by organization called Facebook in the year of 2013.
- Angular JS was Introduced by organization called Google in the year of 2007.
- Node JS is the Server side language for JavaScript. It is not a Framework or Library.

3. What is Single-Page Application and Multi-Page Application ?

▼ Ans

- Single-page application is a web application that loads only one single web page, and then updates the body content of that single web page without loading is called Single-page application.
- Multi-Page Application is a web application that has more than one web pages with static information and for every change request a new web page will be rendered from the server in the browser is called Multi-Page Application.

4. What are the Features of React JS ?

▼ Ans

1. We can develop Single Page Application

- Whenever we click on hypertext, Generally in Multipage Application, It will takes time to reload the whole web-page and it makes the web-page slower. but, In SPA, If we move to the next web-page. It will not reload the entire web-page else it will be load only the respected component which will be faster then MPA.
- We cannot create SPA using only WEB TECH in order to create SPA we need React-JS.
- (Ex: YouTube, Instagram.. but not Amazon)

2. It is a Declarative

- Declarative means where the programmer specifying what should be achieved, rather than specifying how should be achieved. React JS is a library of JavaScript, library consist of collection of Pre-defined codes. All the pre-defined codes are already declared inside the react library.
- There is no need to write logics everywhere, we can use inbuilt methods which is already present in the library.

3. It follows Component-Based-Architecture.

- Component Based Architecture is a framework for developing an application based on reusable components. Each component has different functionalities and that can be stored in a library and dropped into an application without modifying other components. React JS follows Component Based Architecture, Because it is easy to maintain the codes from the perspective of devops.
- Components are nothing but separate block of one single webpage.
- In one react project their can be any num of components
- Facebook which is built by React is maintaining more then 30,000+ components.

5. What is Component Based Architecture ?

▼ Ans

- Component Based Architecture is a framework for developing an application based on reusable components. Each component has different functionalities and that can be stored in a library and dropped into an application without modifying other components.

6. Main folders of React JS ?

▼ Ans

▼ node modules

- Since React is a library of JavaScript, It consist of Pre-Defined codes. All those pre-Defined codes are declared inside "node modules" folder ("do not touch this folder")

▼ public

- "index.html" is present in this folder. It consists of 1 div element with no content but with Id value called "root". We have to maintain this "index.html" file as it is.

▼ src

- After installing we will be having default **src** which is no use for us. Delete that default **src** and create new **"src"** in the same place. Inside **src**, We have to maintain two main files : -

1. **index.js**

- It is known as root file of React Project, Because we are going to create root between **index.js** and **index.html**

2. **App.jsx**

- It is a component where the content should be written to display on the UI

3. **package.json** (Directories of React project)

4. **package-lock.json** (Directories of React project)

7. How to create a root for a component ?

▼ Ans

- Call **createRoot()** function to create root to displaying React components inside Browser DOM node. We have to create a root between **index.html** and **index.js**

8. What is **createRoot()** and **render()** ?

▼ Ans

- **createRoot()** and **render()** are functions declared inside **"react-dom"** package present in **"node-modules"** folder.
- **createRoot()** function is used to create the root to display the React Components inside a Browser Dom node.
- **render()** function is used to push the React component inside Browser DOM node. It will push the element into particular file.

9. What is **"root"** ?

▼ Ans

- **"root"** is the id value given to the div Tag in the **index.html**.

10. What is Components ? uses of Components ?

▼ Ans

- Components are nothing but separate block of one single web page. They are independent and reusable. We should merge all components in a parent component, which will be the final UI of our application.
- We can declare components in 2 ways
 1. Self Closing Tags. EX: `<ComponentName/>`
 2. Paired Tags. EX: `<ComponentName></ComponentName>`
- Uses
 1. Reusability - We can reuse the components.
 2. Extensibility - We can combined the component with others to create new behaviors.
 3. Replaceability - Similar functionality components can be swapped.

11. What are the types of Components ?

▼ Ans

1. **Function Based Components**

- Function Based Components are simply a JavaScript Functions. We can create a Functional Based Component in React JS by writing a JavaScript Function.
- Syntax

```
const ComponentName = () =>{
    return(
        <div>
            //Content
        </div>
    )
}
```

2. Class Based Components

- **Class Based Components** are simply a JavaScript Class. We can create a Class Based Component in React JS by writing a JavaScript Class.
- **Syntax**

```
import {Component} from "react"

class ComponentName extends Component
{
    render()
    {
        return(
            //content
        )
    }
}
```

12. How to Print "Hello World" in React JS ?

▼ Ans

1. Open particular folder of React JS.
2. Open CMD in that folder.
3. Type code . to open Visual code.
4. Delete Existing src folder and create your own src folder.
5. Create two file inside src folder → 1. index.js 2. App.jsx

▼ index.js

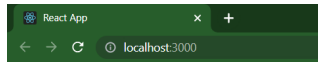
```
import {createRoot} from "react-dom/client"
import App from "./App"
createRoot(document.getElementById("root")).render(<App/>)
```

▼ App.jsx

```
const App=()=>{
    return(
        <h1>Hello World</h1>
    )
}
export default App
```

▼ Output

1. To Run click on Terminal and click New Terminal.
2. Type (npm start)



Hello World

- render means it will push the element into particular file.

13. Rules to follow while Creating a Components ?

▼ Ans

- First Letter should be capital. (Ex: Nav.jsx, Content.jsx, Footer.jsx)
- The Files whichever we created and given first letter as capital will be always considered as components.

14. What is XML and HTML ?

▼ Ans

XML	HTML
1). className (Ex: <div className='a'> HI </div>)	1). class (<div class='a'> HI </div> (Selector))
2). Events (Ex: <div onClick='b()'> HI </div>)	2). Events (Ex: <div onclick='b()'> HI </div>)
3). Everything should be wrapped inside one single tag. Otherwise we will get Error	4). We will get output in any case.

15. Import Every component into Single component ?

▼ Ans

▼ index.js

```
import {createRoot} from "react-dom/client"
import App from "./App"
createRoot(document.getElementById('root')).render(<App/>)
```

▼ App.jsx

```
import Nav from "./Components/Nav"
import Content from "./Components/Content"
import Footer from "./Components/Footer"

const App={()=>{
  return(
    <div>
      <Nav/>
      <Content/></Content>
      <Footer/></Footer>
    </div>
  )
}
export default App
```

▼ Nav.jsx (Child → App.jsx)

```
const Nav={()=>{
  return(
    <div>
      <h1>NavBar</h1>
    </div>
  )
}
```

```
}
export default Nav
```

▼ Content.jsx (Child → App.jsx)

```
const Content={()=>{return(
  <div>
    <h1>Content</h1>
  </div>
)}}
export default Content
```

▼ Footer.jsx (Child → App.jsx)

```
const Footer={()=>{return(
  <div>
    <h1>Footer</h1>
  </div>
)}}
export default Footer
```

16. Code for Class Based Components ?

▼ Ans

▼ index.js

```
import {createRoot} from 'react-dom/client'
import App from './App'
createRoot(document.getElementById('root')).render(<App></App>)
```

▼ App.jsx

```
import Cbc from "../Components/Cbc"
const App={()=>{
  return (<Cbc/>)
}}
export default App
```

▼ Cbc.jsx

```
import {Component} from "react"
class Cbc extends Component
{
  render()
  {
    return(<h1>!!! hi Ho Are You !!!</h1>)
  }
}
export default Cbc
```

```
!!! hi Ho Are You !!!
```

17. Difference between Function Based Component and Class Based Component ?

▼ Ans

<u>FBC</u>	<u>CBC</u>
1). FBC is a Stateless	1). CBC is Stateful.
2). We use JS function in FBC	2). We use JS class in CBC

3). FBC supports Hooks	3). CBC does not supports Hooks
4). FBC does not supports Life-cycle methods.	4). CBC supports Life-cycle methods.
5). "this" keyword cannot be used in FBC.	5). "this" keyword can be used in CBC.

18. What is Props ? Why we need Props ?

▼ Ans

- Props are Inbuilt objects in React. Props are used to pass the data from the parent component to the child component . Props are immutable.

▼ Syntax App.jsx

```
const App={()=>{
  return(
    <div>
      <hello data="Galaxy">
    </div>
  )
}
```

```
export default App
```

▼ Code 1 (Passing String)

▼ App.jsx

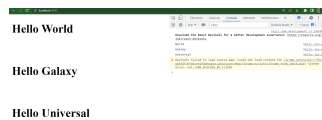
```
import Hello from "../Components/Hello"
const App={()=>{
  return (<div> <Hello data="World"/><br /> //here data is prop name
    <Hello data="Galaxy"/><br />
    <Hello data="Universal"/>
  </div>)
}
```

```
export default App
```

▼ Hello.jsx

```
const Hello=(a)=>{
  console.log(a.data)
  return(
    <div>
      <h1>Hello {a.data}</h1> <br/>
    </div>
  )
}
```

```
export default Hello
```



▼ Code 2 (Passing Array)

▼ App.jsx

```
import Hello from "../Components/Hello"
const App={()=>{
  return (<div> <Hello data={['Hi','Hello','Bye']}/>
  </div>)
}
```

```
export default App
```

▼ Hello.jsx

```
const Hello=(a)=>{
  console.log(a.data[2])
  return(
    <div >
      <h1> Hello {a.data[0]}</h1>
      <h1> Hello {a.data[1]}</h1>
      <h1> Hello {a.data[2]}</h1>
    </div>
  )
}
export default Hello
```



Hello Hi
Hello Hello
Hello Bye

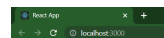
▼ Code 3 (Passing Object)

▼ App.jsx

```
import Hello from "../Components/Hello"
const App=()=>{
  let obj={
    Name:"Abc",
    id:123
  }
  return (<div> <Hello data={obj}/>
    </div>)
}
export default App
```

▼ Hello.jsx

```
const Hello=(a)=>{
  console.log(a.data.Name)
  return(
    <div >
      <h1> Hello {a.data.Name}</h1>
    </div>
  )
}
export default Hello
```



Hello Abc

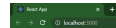
▼ Code 4 (Passing Object Array)

▼ App.jsx

```
import Hello1 from "../Components/Hello1"
const App=()=>{
  let obj=[ {Name:"HTML",id:123},
    {Name:"CSS",id:456},
    {Name:"JS",id:789} ]
  return (<div> <Hello1 data={obj}/>
    </div>)
}
export default App
```


▼ Hello1.jsx

```
const Hello1=(a)=>{
  console.log(a.data[0].Name) //for Console output
  return(
    <div >
      <h1> Hello {a.data[0].Name}</h1><br />
      <h1> Hello {a.data[1].Name}</h1><br />
      <h1> Hello {a.data[2].Name}</h1>
    </div>
  )
}
export default Hello1
```



Hello HTML

Hello CSS

Hello JS

19. What is Props Drilling ?

▼ Ans

- Props drilling is the process of passing the data from the top component tree through all the child components, Where some of the child components does not need the data. It is a dis-advantage in React. To avoid this disadvantage, We are going to **useContext()** hook.

▼ App.jsx (Parent Component)

```
import Child from "../Components/Child"

const App=()=>{
  return (<div>
    <Child abc="Hi"/>
  </div>)
}
export default App
```

▼ Child.jsx (Child Component)

```
import GrandChild from "../Grandchild"
const Child=(a)=>{
  return(
    <div>
      <GrandChild cba={a.abc}/>
    </div>
  )
}
export default Child
```

▼ Grandchild.jsx (Grand child Component)

```
const GrandChild=(b)=>{
  return(
    <div>
      {b.cba} Good-Morning
    </div>
  )
}
export default GrandChild
```



Hi Good-Morning

▼ Using **useContext()** for same example (don't worry you will get it in further topics)

▼ App.jsx (Parent Component)

```
import Child from "../Components/Child"
import { createContext } from "react"

export let userData = createContext()

const App=()=>{
  return (
    <div>
      <userData.Provider value="Hi">
        <Child/>
      </userData.Provider>
    </div>
  )
}
export default App
```

▼ Child.jsx (Child Component)

```
import GrandChild from "../Grandchild"

const Child=(a)=>{
  return(
    <div>
      <GrandChild/>
    </div>
  )
}
export default Child
```

▼ Grandchild.jsx (Grand child Component)

```
import { useContext } from "react"
import { userData } from "../App"

const GrandChild=(b)=>{
  let user = useContext(userData)
  return(
    <div>
      {user} Good-Morning
    </div>
  )
}
export default GrandChild
```

20. Props Drilling Practice Code ?

▼ Ans

- Props drilling tree

▼ index.js

```
import {createRoot} from 'react-dom/client'
import App from './App'
createRoot(document.getElementById('root')).render(<App></App>)
```

▼ App.jsx (Parent Component)

```
import A from "../Components/A"
import D from "../Components/D"
import E from "../Components/E"

const App=()=>{
```

```

    return (
      <div>
        <A abc="manu(App to C)"/>
        <D bac="manu(App to D)"/>
        <E opp="manu(App to F)"/>
      </div>
    )
  }
  export default App

```

▼ A.jsx

```

import B from "./B"
const A=(amanu)=>{
  return(
    <div>
      <B abd={amanu.abc}/>
    </div>
  )
}
export default A

```

▼ B.jsx

```

import C from "./C"
const B=(bmanu)=>{
  return(
    <div>
      <C abe={bmanu.abd}/>
    </div>
  )
}
export default B

```

▼ C.jsx

```

const C=(cmanu)=>{
  return(
    <div>
      {cmanu.abe}
    </div>
  )
}
export default C

```

▼ D.jsx

```

const D=(zyx)=>{
  return(
    <div>
      {zyx.bac}
    </div>
  )
}
export default D

```

▼ E.jsx

```

import F from "./F";
const E=(emanu)=>{
  return(
    <div>
      <F zmanu={emanu.opp}/>
    </div>
  )
}
export default E

```

▼ F.jsx

```
const F=(fmanu)=>{
  return(
    <div>
      {fmanu.zmanu}
    </div>
  )
}
export default F
```

21. What is props.children ?

▼ Ans

- **props.children** is a special prop which is used to access the content which are written inside the opening and closing tag.
- **“children”** is inbuilt keyword
 - Ex: App.jsx , PropChild.jsx

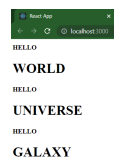
▼ App.jsx

```
import Propchild from "../Components/Propchild"

const App=()=>{
  return (
    <div>
      <Propchild><h1>WORLD</h1></Propchild>
      <Propchild><h1>UNIVERSE</h1></Propchild>
      <Propchild><h1>GALAXY</h1></Propchild>
    </div>
  )
}
export default App
```

▼ Propchild.jsx

```
const Propchild=(x)=>{
  console.log(x.children)
  return(
    <div><h1>HELLO {x.children}</h1></div>
  )
}
export default Propchild
```



```
HELLO
WORLD
HELLO
UNIVERSE
HELLO
GALAXY
```

22. prop class ?

▼ Ans

▼ App.jsx

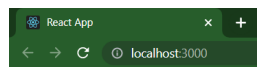
```
import Propclass from "../Components/Propclass"

const App=()=>{
  return (
    <div>
      <Propclass data="Bye"/>
    </div>
  )
}
```

```
}  
export default App
```

▼ Propclass

```
import { Component } from "react";  
class Propclass extends Component{  
  render(){  
    console.log(this.props)  
    return(  
      <div>  
        Hii {this.props.data}  
      </div>  
    )  
  }  
}  
export default Propclass
```



Hii Bye

18. What is States ?

▼ Ans

- States are used to create **dynamic** (the data which will be keep on changing) **data on the web page**. States are **Mutable**.
- Function Based Components are “Stateless” that means, We cannot create States inside FBC. So that, We can make FBC as Stateful by using a feature called Hooks. “**useState()**” is a Hook which makes FBC as Stateful.

19. Difference between props and states ?

▼ Ans

20. What is Hook ?

▼ Ans

- React Hooks are simply a JavaScript functions that allow us to Reuse Stateful Logics without modifying our components. This makes easy to share Hooks and write codes.
- Hooks are like inbuilt methods in react
- Hooks always starts from a prefix word “use”
- Whenever we wanted to use hooks we have to import it from react library, import statements are mandatory
- Hooks are only used in Function Based Component.
- We have many Hooks in react these are the few below “hooks”.

21. What is **useState()** ?

▼ Ans

- **useState()** is a hook which is used to store state value in the Function Based Component. It accepts initial value such as number, boolean ,Array, object and string. This makes Function Based Component as Stateful.
- Syntax

```
let [state, setState] = useState(0)
```

```
let      = variable
state    = variable name
setState = It is a method which is used to update the value of the variable.
```

▼ We can pass all data-type as a value for the states

```
1)Passing a String
  Ex: let[state,setState]=useState("Hello")

2)Passing a Number
  Ex: let[num,setName]=useState(0)

3)Passing an Object
  Ex: let[obj,setObj]=useState({name:"abc",id:123})

1)Passing an Array
  Ex: let[arr,setArr]=useState("Hello","Hi","Namaskara")
```

▼ Example 1

▼ App.jsx

```
import States from "../Components/States"

const App=()=>>{
  return (
    <div>
      <States/>
    </div>
  )
}
export default App
```

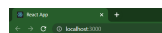
▼ States.jsx

```
import { useState } from "react";

const States=()=>>{
  let [abc,setAbc]=useState("Hungry")

  let btn=()=>>{
    setAbc("I am Full")
  }

  return(
    <div>
      <h1>{abc}</h1>
      <button onClick={btn}>Food</button>
    </div>
  )
}
export default States
```



Hungry

Food



I am Full

Food

▼ Example 2 (Passing Array)

▼ App.jsx

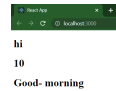
```
import StateArray from "../Components/StateArray"

const App=()=>{
  return (
    <div>
      <StateArray/>
    </div>)
}
export default App
```

▼ StateArray.jsx

```
import { useState } from "react"

const StateArray=()=>{
  let [arr, setArr]=useState(["hi",10,"Good- morning"])
  return(
    <div>
      <h1>{arr[0]}</h1>
      <h1>{arr[1]}</h1>
      <h1>{arr[2]}</h1>
    </div>
  )
}
export default StateArray
```



▼ Example 3 (Using map)

▼ App.jsx

```
import Statearr from "../Components/Statearr"

const App=()=>{
  return (
    <div>
      <Statearr></Statearr>
    </div>)
}
export default App
```

▼ StateArray.jsx

```
import { useState } from "react"
const Statearr = () =>{

  let[arr, setArr] = useState(["Hi",10,"GoodMorning","Manu"])

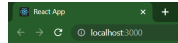
  return(
    <div>
      {arr.map((x)=>{
        return(
          <h1>{x}</h1>
        )})
      }
    </div>
  )
}
export default Statearr
```

▼ StateArray.jsx (We can write like this also)

```
import { useState } from "react"
const Statearr = () =>{

  let[arr, setArr] = useState(["Hi",10,"GoodMorning","Manu"])

  return(
    <div>
      {arr.map(x=><h1>{x}</h1>)}
    </div>
  )
}
export default Statearr
```



Hi
10
GoodMorning
Manu

▼ Example 4 (Fetching Data using Json)

- COPY AND PASTE THE CODE FORM ==> <https://api.github.com/users>

▼ App.jsx

```
import FetchData1 from "../Components/Fetchdata1"

const App=()=>{
  return (
    <div>
      <FetchData1/>
    </div>
  )
}
export default App
```

▼ FetchData1.jsx

```
import { useState } from "react"
import dummyData from "../Userdata.json"

const FetchData1=()=>{

  let[content,setContent]=useState(dummyData)

  console.log(content);

  return(
    <div>
      {content.map((x)=>{
        return(
          <div>
            <h1>{x.login}</h1>
            <img src={x.avatar_url} alt="" />
          </div>
        )
      })}
    </div>
  )
}
export default FetchData1
```

▼ Userdata.json

```
[
  {
    "login": "mojombo",
```



```

    "id": 1,
    "node_id": "MDQ6VXNlcjE=",
    "avatar_url": "https://avatars.githubusercontent.com/u/1?v=4",
    "gravatar_id": "",
    "url": "https://api.github.com/users/mojombo",
    "html_url": "https://github.com/mojombo",
    "followers_url": "https://api.github.com/users/mojombo/followers",
    "following_url": "https://api.github.com/users/mojombo/following{/other_user}",
    "gists_url": "https://api.github.com/users/mojombo/gists{/gist_id}",
    "starred_url": "https://api.github.com/users/mojombo/starred{/owner}/{/repo}",
    "subscriptions_url": "https://api.github.com/users/mojombo/subscriptions",
    "organizations_url": "https://api.github.com/users/mojombo/orgs",
    "repos_url": "https://api.github.com/users/mojombo/repos",
    "events_url": "https://api.github.com/users/mojombo/events{/privacy}",
    "received_events_url": "https://api.github.com/users/mojombo/received_events",
    "type": "User",
    "site_admin": false
  },
  {
    "login": "defunkt",
    "id": 2,
    "node_id": "MDQ6VXNlcjI=",
    "avatar_url": "https://avatars.githubusercontent.com/u/2?v=4",
    "gravatar_id": "",
    "url": "https://api.github.com/users/defunkt",
    "html_url": "https://github.com/defunkt",
    "followers_url": "https://api.github.com/users/defunkt/followers",
    "following_url": "https://api.github.com/users/defunkt/following{/other_user}",
    "gists_url": "https://api.github.com/users/defunkt/gists{/gist_id}",
    "starred_url": "https://api.github.com/users/defunkt/starred{/owner}/{/repo}",
    "subscriptions_url": "https://api.github.com/users/defunkt/subscriptions",
    "organizations_url": "https://api.github.com/users/defunkt/orgs",
    "repos_url": "https://api.github.com/users/defunkt/repos",
    "events_url": "https://api.github.com/users/defunkt/events{/privacy}",
    "received_events_url": "https://api.github.com/users/defunkt/received_events",
    "type": "User",
    "site_admin": false
  },
  {
    "login": "pjhyett",
    "id": 3,
    "node_id": "MDQ6VXNlcjM=",
    "avatar_url": "https://avatars.githubusercontent.com/u/3?v=4",
    "gravatar_id": "",
    "url": "https://api.github.com/users/pjhyett",
    "html_url": "https://github.com/pjhyett",
    "followers_url": "https://api.github.com/users/pjhyett/followers",
    "following_url": "https://api.github.com/users/pjhyett/following{/other_user}",
    "gists_url": "https://api.github.com/users/pjhyett/gists{/gist_id}",
    "starred_url": "https://api.github.com/users/pjhyett/starred{/owner}/{/repo}",
    "subscriptions_url": "https://api.github.com/users/pjhyett/subscriptions",
    "organizations_url": "https://api.github.com/users/pjhyett/orgs",
    "repos_url": "https://api.github.com/users/pjhyett/repos",
    "events_url": "https://api.github.com/users/pjhyett/events{/privacy}",
    "received_events_url": "https://api.github.com/users/pjhyett/received_events",
    "type": "User",
    "site_admin": false
  }
]

```



22. Increment and Decrement Project ? (Interview Project)

▼ Ans

▼ App.jsx

```
import States from "../Components/States"
```

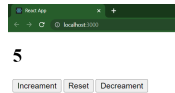
```
const App=()=>{
  return (
    <div>
      <States/>
    </div>
  )
}
export default App
```

▼ States.jsx

```
import { useState } from "react";

const States=()=>{
  let [count,setCount]=useState(0)

  let incre=()=>{
    setCount(count+1)
  }
  let decre=()=>{
    setCount(count-1)
  }
  let reset=()=>{
    setCount(0)
  }
  return(
    <div>
      <h1>{count}</h1>
      <button onClick={incre}>Increment</button> <span></span>
      <button onClick={reset}>Reset</button> <span></span>
      <button onClick={decre}>Decrement</button>
    </div>
  )
}
export default States
```



23. Fetching the Data from Database(JSON) using **useState()** Hook ?

▼ Ans

▼ Fetchdata2.jsx

```
import { useState } from "react"
import content from "../userdata.json"
const Fetchdata2=()=>{
  let [users,setuser]=useState(content)
  return(
    <div>
      <h1>{users.map((x)=>{
        return(
          <div>
            <img src={x.avatar_url}/>
            <h1>{x.login}</h1>
          </div>
        )
      })}</h1>

      </div>
    )
  }
}
export default Fetchdata2
```

24. Facebook Like button or Instagram Heart button and it should be increment ? (Interview Project)

▼ Ans

- font awesome cdn (Saerch in Google) —> past this Link in index.html
- font awesome —> past this link in our Component

▼ App.jsx

```
import Heart from "../Components/Heart"

const App={()=>{
  return(
    <div>
      <Heart/>
    </div>
  )
}
export default App
```

▼ Heart.jsx

```
import { useState } from "react"

const Heart={()=>{
  let [count,setCount]=useState(0)

  let incre={()=>{
    setCount(count+1)
  }
  return(
    <div>
      <i class="fa-solid fa-heart" onClick={incre}></i><sup>{count}</sup>
    </div>
  )
}
export default Heart
```

25. What is List and Keys ?

▼ Ans

- Lists will be always created using JavaScript “map”. Lists is used to display the data in an ordered format. map() function is used for transversion the lists.
- Keys are unique identification for the elements which are present inside Lists.

26. How to fetch only selected contents ?

▼ Ans

▼ App.jsx

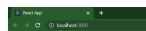
```
import FetchData1 from "../Components/Fetchdata1"

const App={()=>{
  return (
    <div>
      <FetchData1/>
    </div>)
}
export default App
```

▼ Fetchdata1.jsx

```
import { useState } from "react"
import dummyData from "../Userdata.json"
```

```
const FetchData1={()=>{
  let [content, setContent]=useState(dummyData)
  console.log(content);
  return(
    <div>
      {content.slice(0,5).map((x)=>{
        return(
          <div>
            <h1>{x.login}</h1>
          </div>
        )
      })}
    </div>
  )
})
export default FetchData1
```



mojombo
defunkt
pjhyett

27. What is Fragment ?

▼ Ans

- Fragment helps to prevent of object while creating a list.
- It will act like a container or element but it is not a container or element.
- It is also known as Invisible container.
- Whenever you are using Fragment, import statement is mandatory.
- Fragments can be declared like "<Fragment></Fragment>" or "<></>"
- Fragment Tag acts as a container. It reduces or hides the unnecessary tags in an console or UI output.

▼ App.jsx

```
import FetchData1 from "../Components/Fetchdata1"

const App={()=>{
  return (
    <div>
      <FetchData1/>
    </div>
  )
}
export default App
```

▼ Fetchdata1.jsx

- To avoid the repeated value in content we use key attribute. key attribute → Data add on Information.

```
import { Fragment, useState } from "react"
import dummyData from "../Userdata.json"

const FetchData1={()=>{
  let [content, setContent]=useState(dummyData)
  console.log(content);
  return(
    <div>
      {content.map((x)=>{
        return(
          <Fragment key={x.id}>
            <h1>{x.login}</h1>
            <img src={x.avatar_url} alt="" />
          </Fragment>
        )
      })}
    </div>
  )
})
```

```

    )
  }
  export default FetchData1

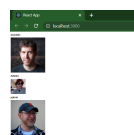
```

▼ Userdata.json

```

[
  {
    "login": "mojombo",
    "id": 1,
    "node_id": "MDQ6VXNlcjE=",
    "avatar_url": "https://avatars.githubusercontent.com/u/1?v=4",
    "gravatar_id": "",
    "url": "https://api.github.com/users/mojombo",
    "html_url": "https://github.com/mojombo",
    "followers_url": "https://api.github.com/users/mojombo/followers",
    "following_url": "https://api.github.com/users/mojombo/following{/other_user}",
    "gists_url": "https://api.github.com/users/mojombo/gists{/gist_id}",
    "starred_url": "https://api.github.com/users/mojombo/starred{/owner}/{repo}",
    "subscriptions_url": "https://api.github.com/users/mojombo/subscriptions",
    "organizations_url": "https://api.github.com/users/mojombo/orgs",
    "repos_url": "https://api.github.com/users/mojombo/repos",
    "events_url": "https://api.github.com/users/mojombo/events{/privacy}",
    "received_events_url": "https://api.github.com/users/mojombo/received_events",
    "type": "User",
    "site_admin": false
  },
  {
    "login": "defunkt",
    "id": 2,
    "node_id": "MDQ6VXNlcjI=",
    "avatar_url": "https://avatars.githubusercontent.com/u/2?v=4",
    "gravatar_id": "",
    "url": "https://api.github.com/users/defunkt",
    "html_url": "https://github.com/defunkt",
    "followers_url": "https://api.github.com/users/defunkt/followers",
    "following_url": "https://api.github.com/users/defunkt/following{/other_user}",
    "gists_url": "https://api.github.com/users/defunkt/gists{/gist_id}",
    "starred_url": "https://api.github.com/users/defunkt/starred{/owner}/{repo}",
    "subscriptions_url": "https://api.github.com/users/defunkt/subscriptions",
    "organizations_url": "https://api.github.com/users/defunkt/orgs",
    "repos_url": "https://api.github.com/users/defunkt/repos",
    "events_url": "https://api.github.com/users/defunkt/events{/privacy}",
    "received_events_url": "https://api.github.com/users/defunkt/received_events",
    "type": "User",
    "site_admin": false
  },
  {
    "login": "pjhyett",
    "id": 3,
    "node_id": "MDQ6VXNlcjM=",
    "avatar_url": "https://avatars.githubusercontent.com/u/3?v=4",
    "gravatar_id": "",
    "url": "https://api.github.com/users/pjhyett",
    "html_url": "https://github.com/pjhyett",
    "followers_url": "https://api.github.com/users/pjhyett/followers",
    "following_url": "https://api.github.com/users/pjhyett/following{/other_user}",
    "gists_url": "https://api.github.com/users/pjhyett/gists{/gist_id}",
    "starred_url": "https://api.github.com/users/pjhyett/starred{/owner}/{repo}",
    "subscriptions_url": "https://api.github.com/users/pjhyett/subscriptions",
    "organizations_url": "https://api.github.com/users/pjhyett/orgs",
    "repos_url": "https://api.github.com/users/pjhyett/repos",
    "events_url": "https://api.github.com/users/pjhyett/events{/privacy}",
    "received_events_url": "https://api.github.com/users/pjhyett/received_events",
    "type": "User",
    "site_admin": false
  }
]

```



28. React CSS ?

▼ Ans

▼ Inline

- In Inline CSS, CSS will be applied only for particular element by using an attribute called "**style**". We should mention all the CSS properties in the form of Objects.

▼ App.jsx

```
import Inline from "../Components/Inline"

const App=()=>{
  return(
    <div>
      <Inline/>
    </div>
  )
}
export default App
```

▼ Inline.jsx

```
const Inline=()=>{
  return(
    <div>
      <h1 style={{background:"red",fontSize:"50px"}}>INLINE CSS</h1>
    </div>
  )
}
export default Inline
```



▼ global.css

- In global.css, We will be maintaining one separate CSS file for whole React project. for every components, style will be applying in that file.
- While creating a CSS file
 - a. extension should be .css
 - b. first letter should be small case and also we have to import the global.css file in the parent component which is App.

▼ App.jsx

```
import Style2 from "../Components/Style2"
import "../Components/global.css"
import Style3 from "../Components/Style3"
const App=()=>{
  return(
    <div>
      <Style2/>
      <Style3/>
    </div>
  )
}
export default App
```

▼ Style2.jsx

```
const Style2={()=>{
  return(
    <div>
      <h1>GLOBAL CSS</h1>
    </div>
  )
}
export default Style2
```

▼ Style3.jsx

```
const Style3={()=>{
  return(
    <div>
      <h2>GLOBAL CSS 11111111111</h2>
    </div>
  )
}
export default Style3
```

▼ global.css

```
h1{
  background-color: aqua;
}
h2{
  background-color: bisque;
}
```

GLOBAL CSS
GLOBAL CSS mmmmm

▼ module.css

- In module.css, We will be maintaining individual CSS file for each component. For every component, we have to import the individual CSS files in their respective components.
- Extension to follow while creating a CSS file
 - filename.module.css
 - Everything should be in lower case

▼ App.jsx

```
import Style from "../Components/Style"

const App={()=>{
  return (
    <div>
      <Style/>
    </div>
  )
}
export default App
```

▼ Style.jsx

```
import design from "../style.module.css"

const Style={()=>{
  return(
    <div id={design.nav}>
      <h1>STYLE</h1>
    </div>
  )
}
```

```
}  
export default Style
```

▼ style.module.css

```
#nav{  
  background-color: black;  
  color: white;  
}
```



29. What are the types of Selectors ?

▼ Ans

There 5 types of Selector are there:-

▼ Simple Selectors

▼ Universal

- The `*` selector selects all elements.
- `* selector` can also select all elements inside another element.

▼ Example 1

```
<!DOCTYPE html>  
<html>  
<head>  
<style>  
  * {  
    background-color: yellow;  
  }  
</style>  
</head>  
<body>  
  
<h1>Demo of the * selector</h1>  
  
<div class="intro">  
  <p id="firstname">My name is Manu.</p>  
  <p id="hometown">I live in Harapanalli.</p>  
</div>  
<p>My best friend is No-One.</p>  
  
</body>  
</html>
```

Demo of the * selector

My name is Manu.
I live in Harapanalli.
My best friend is No-One.

▼ Example 2

```
<!DOCTYPE html>  
<html>  
<head>  
<style>  
  div * {  
    background-color: cyan;  
  }  
</style>  
</head>  
<body>  
  
<h1>Demo of the * selector</h1>
```



```

<div class="intro">
  <p id="firstname">My name is Manu.</p>
  <p id="hometown">I live in Harapanalli.</p>
</div>

<p>My best friend is No-One.</p>

</body>
</html>

```

Demo of the * selector

My best friend is No-One

▼ Type

- Type selector selects the element based on TagName.

▼ Example

```

<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    div {
      background-color: red;
    }
  </style>
</head>
<body>

  <div> This is a div </div>
  <span> This is a span </span>
  <ul>
    <li>Item 1</li>
    <li>Item 2</li>
    <li>Item 3</li>
    <li>Item 4</li>
  </ul>

</body>
</html>

```

▼ Class

- Class selector selects the element based on class attribute.
- class attribute value can be duplicate.

▼ Example

```

<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    .A{
      background-color: blue;
    }
  </style>
</head>
<body>

  <div class="A"> This is a div </div>
  <span> This is a span </span>
  <ul>
    <li class="A">Item 1</li>
    <li>Item 2</li>
    <li>Item 3</li>
    <li>Item 4</li>
  </ul>

</body>
</html>

```

```
</body>
</html>
```

▼ Id

- Id selector selects the element based on Id attribute.
- Id attribute value cannot be duplicate.

▼ Example

```
<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    #A{
      background-color: cyan;
    }
  </style>
</head>
<body>

  <div id="A"> This is a div </div>
  <span> This is a span </span>
  <ul>
    <li>Item 1</li>
    <li>Item 2</li>
    <li>Item 3</li>
    <li>Item 4</li>
  </ul>

</body>
</html>
```

▼ Combination Selectors

Combination Selectors selects more than one Simple Selectors.

▼ And (".")

- It selects the elements that match all the Selectors in the list.

▼ Example

```
<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    div.A
    {
      background-color: cyan;
    }
  </style>
</head>
<body>

  <div class="A"> This is a div </div>
  <div> Manu </div>
  <span> This is a span </span>
  <ul>
    <li class="A">Item 1</li>
    <li>Item 2</li>
    <li>Item 3</li>
    <li>Item 4</li>
  </ul>

</body>
</html>
```

▼ Or (",")

- It selects elements that match any Selectors in the list.

▼ Example

```
<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    div, li.A{
      background-color: cyan;
    }
  </style>
</head>
<body>

  <div class="A"> This is a div </div>
  <div> Manu </div>
  <span> This is a span </span>
  <ul>
    <li class="A"> Item 1 </li>
    <li>Item 2</li>
    <li>Item 3</li>
    <li>Item 4</li>
  </ul>

</body>
</html>
```

▼ Descendant (" ")

- It selects the elements inside of other element. whether it is a direct child or child of a child or child of a child of a child.

▼ Example 1

```
<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    ul li{
      background-color: red;
    }
  </style>
</head>
<body>

  <div class="A"> This is a div </div>
  <div> Manu </div>
  <span> This is a span </span>
  <ul>
    <li class="A"> Item 1 </li>
    <li>Item 2</li>
    <li>Item 3</li>
    <li>Item 4</li>
  </ul>

</body>
</html>
```

▼ Example 2

```
<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    div b{
      background-color: red;
    }
  </style>
</head>
<body>

  <div class="A"> This is a div </div>
  <span> This ia a Span </span>
  <div>
```

```

        <span>
          <b> Nested Text </b>
        </span>
      </div>
      <b> Nested Text 1 </b>
      <ul>
        <li class="A"> Item 1 </li>
        <li>Item 2</li>
        <li>Item 3</li>
        <li>Item 4</li>
      </ul>
    </body>
  </html>

```

▼ Direct child (">")

- It selects the elements inside of other element. but it should be a direct child element.

▼ Example

```

<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    span > b{
      background-color: red;
    }
  </style>
</head>
<body>

  <div class="A"> This is a div </div>
  <span> This ia a Span </span>

  <div>
    <span>
      <b> Nested Text </b>
    </span>
  </div>

  <b> Nested Text 1 </b>
  <ul>
    <li class="A"> Item 1 </li>
    <li>Item 2</li>
    <li>Item 3</li>
    <li>Item 4</li>
  </ul>

</body>
</html>

```

▼ General-sibling ("~")

- It selects all the elements that are siblings of the first element and come after the first element.

▼ Example 1

```

<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    li.A ~ li{
      background-color: red;
    }
  </style>
</head>
<body>

  <div class="A"> This is a div </div>
  <span> This ia a Span </span>

  <div>

```

```

    <span>
      <b> Nested Text </b>
    </span>
  </div>

  <ul>
    <li class="A"> Item 1 </li>
    <li>Item 2</li>
    <li>Item 3</li>
    <li>Item 4</li>
  </ul>

</body>
</html>

```

▼ Example 2

```

<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    li.A ~ li{
      background-color: red;
    }
  </style>
</head>
<body>

  <div class="A"> This is a div </div>
  <span> This ia a Span </span>

  <div>
    <span>
      <b> Nested Text </b>
    </span>
  </div>

  <ul>
    <li> Item 1 </li>
    <li class="A">Item 2</li>
    <li>Item 3</li>
    <li>Item 4</li>
  </ul>

</body>
</html>

```

▼ Adjacent-sibling ("+")

- It selects only one elements that are siblings of the first element and come after the first element.

▼ Example

```

<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    li.A + li{
      background-color: red;
    }
  </style>
</head>
<body>

  <div class="A"> This is a div </div>
  <span> This ia a Span </span>

  <div>
    <span>
      <b> Nested Text </b>
    </span>
  </div>

```

```

<ul>
  <li> Item 1 </li>
  <li class="A">Item 2</li>
  <li>Item 3</li>
  <li>Item 4</li>
</ul>

</body>
</html>

```

▼ Pseudo-Class Selectors

A pseudo-class is used to define a special state of an element. For example, Style an element when a user mouses over it, Style an element when it gets focus.

▼ **Hover (" : hover")**

- It selects the elements that are hovered by the mouse.

▼ Example

```

<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    li:hover{

        background-color: red;
    }
  </style>
</head>
<body>

  <div class="A"> This is a div </div>
  <span> This ia a Span </span>

  <div>
    <span>
      <b> Nested Text </b>
    </span>
  </div>

  <ul>
    <li> Item 1 </li>
    <li class="A">Item 2</li>
    <li>Item 3</li>
    <li>Item 4</li>
  </ul>

</body>
</html>

```

▼ **Focus (" : focus")**

- It selects the elements that are Focused.

▼ Example

```

<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    input:focus{

        background-color: red;
    }
  </style>
</head>
<body>

  <label for="">Name</label>
  <input type="text">

</body>
</html>

```

▼ Required (" : required")

- It selects the input that are required.

▼ Example

```
<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    input:required{
      background-color: red;
    }
  </style>
</head>
<body>
  <label for="">Name</label>
  <input required type="text">

</body>
</html>
```

▼ Checked (" : checked")

- It selects the checkboxes / radio buttons that are checked

▼ Example

```
<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    input:checked{
      margin: 50px;
    }
  </style>
</head>
<body>
  <label for="">Name</label>
  <input type="checkbox">

</body>
</html>
```

▼ Disabled (" : disabled")

- It selects the input that are disabled.

▼ Example

```
<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    input:disabled{
      margin: 50px;
    }
  </style>
</head>
<body>
  <label for="">Name</label>
  <input disabled type="checkbox">

</body>
</html>
```

▼ First Child (" : first-child")

- It selects the elements that are the first child inside a container.

▼ Example

```
<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    li:first-child
    {
      background-color: red;
    }
  </style>
</head>
<body>
  <ul>
    <li> Item 1 </li>
    <li> Item 2 </li>
    <li> Item 3 </li>
    <li> Item 2 </li>
  </ul>
</body>
</html>
```

▼ Last Child (" : last-child")

- It selects the elements that are the last child inside a container.

▼ Example

```
<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    li:last-child
    {
      background-color: red;
    }
  </style>
</head>
<body>
  <ul>
    <li> Item 1 </li>
    <li> Item 2 </li>
    <li> Item 3 </li>
    <li> Item 2 </li>
  </ul>
</body>
</html>
```

▼ Nth Child (" : nth-child(2n)")

- It selects the elements that are the nth child inside a container based on the formula.

▼ Example

```
<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    li:nth-child(2)
    {
      background-color: red;
    }
  </style>
</head>
<body>
  <ul>
    <li> Item 1 </li>
    <li> Item 2 </li>
    <li> Item 3 </li>
    <li> Item 2 </li>
  </ul>
```



```
</body>
</html>
```

- **nth-child(2n)** —> It will select every alternate 2nd elements.
- **nth-child(3n)** —> It will select every alternate 3rd elements.
- **nth-child(2n-1)** —> It will select every other elements but it's will offset negative one.
- **nthlast-child(2n)**

▼ Only Child (" : only-child")

- It selects the elements that are the only child inside a container.

▼ Example

```
<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    span:only-child
    {
      background-color: red;
    }
  </style>
</head>
<body>
  <span> Manu </span>
  <div>
    <span> Good Vibes Only </span>
  </div>
</body>
</html>
```

▼ First Of Type (" : first-of-type")

- It selects the elements that are the first of a type inside a container.

▼ Example 1

```
<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    span:first-of-type
    {
      background-color: red;
    }
  </style>
</head>
<body>
  <span> Manu </span>
  <div>
    <span> Good Vibes Only </span>
  </div>
  <ul>
    <li>1</li>
    <li>2</li>
    <li>3</li>
    <li>4</li>
    <li>5</li>
    <li>6</li>
  </ul>
</body>
</html>
```

▼ Example 2

```
<!DOCTYPE html>
<html lang="en">
```

```

<head>
  <style>
    li:first-of-type
    {
      background-color: red;
    }
  </style>
</head>
<body>
  <span> Manu </span>
  <div>
    <span> Good Vibes Only </span>
  </div>
  <ul>
    <li>1</li>
    <li>2</li>
    <li>3</li>
    <li>4</li>
    <li>5</li>
    <li>6</li>
  </ul>
</body>
</html>

```

▼ Last Of Type (" : last-of-type")

- It selects the elements that are the last of a type inside a container.

▼ Example

```

<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    li:last-of-type
    {
      background-color: red;
    }
  </style>
</head>
<body>
  <span> Manu </span>
  <div>
    <span> Good Vibes Only </span>
  </div>
  <ul>
    <li>1</li>
    <li>2</li>
    <li>3</li>
    <li>4</li>
    <li>5</li>
    <li>6</li>
    <span>dnn</span>
  </ul>
</body>
</html>

```

▼ Nth Of Type (" : nth-of-type(2n)")

- It selects the elements that are the nth of a type inside a container based on the formula.

▼ Example

```

<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    li:nth-of-type(2n)
    {
      background-color: red;
    }
  </style>
</head>
<body>
  <span> Manu </span>

```

```

<div>
  <span> Good Vibes Only </span>
</div>
<ul>
  <li>1</li>
  <li>2</li>
  <li>3</li>
  <li>4</li>
  <li>5</li>
  <li>6</li>
</ul>
</body>
</html>

```

▼ Nth Last Of Type (" nth-last-of-type(2n)")

- It selects the elements that are the nth of a type inside a container based on the formula counting from the end.

▼ Example

```

<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    li:nth-last-of-type(2n)
    {
      background-color: red;
    }
  </style>
</head>
<body>
  <span> Manu </span>
  <div>
    <span> Good Vibes Only </span>
  </div>
  <ul>
    <li>1</li>
    <li>2</li>
    <li>3</li>
    <li>4</li>
    <li>5</li>
    <li>6</li>
  </ul>
</body>
</html>

```

▼ Only Of Type (" : only-of-type")

- It selects the elements that are the only of a type inside a container.

▼ Example

```

<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    span:only-of-type
    {
      background-color: red;
    }
  </style>
</head>
<body>
  <span> Manu </span>
  <div>
    <span> Good Vibes Only </span>
  </div>
  <ul>
    <li>1</li>
    <li>2</li>
    <li>3</li>
    <li>4</li>
    <li>5</li>
  </ul>

```

```

        <li>6</li>
      <span>Good Vibes</span>
    </ul>
  </body>
</html>

```

▼ Not (" : not(.green)")

- It selects the elements that do not match the selector inside the not selector.

▼ Example

```

<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    li:not(.green)
    {
      background-color: red;
    }
  </style>
</head>
<body>
  <span> Manu </span>
  <div>
    <span> Good Vibes Only </span>
  </div>
  <ul>
    <li>1</li>
    <li>2</li>
    <li>3</li>
    <li class="green">4</li>
    <li>5</li>
    <li>6</li>
    <span>Good Vibes</span>
  </ul>
</body>
</html>

```

▼ Pseudo-Elements Selectors

Pseudo-element is used to style specified parts of an element. For example, • Insert content before, or after, the content of an element.

▼ Before (" : : before")

- It creates an empty element directly before the children of selected element.

▼ Example

```

<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    span::before
    {
      content: 'Before ';
      background-color: red;
    }
  </style>
</head>
<body>
  <span> Manu </span>

  <ul>
    <li>1</li>
    <li>2</li>
    <li>3</li>
    <li>4</li>
    <li>5</li>
    <li>6</li>
    <span>Good Vibes</span>
  </ul>

```

```
</body>
</html>
```

▼ After (" : : after")

- It creates an empty element directly after the children of selected element.

▼ Example

```
<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    span::after
    {
      content: 'After ';
      background-color: red;
    }
  </style>
</head>
<body>
  <span> Manu </span>

  <ul>
    <li>1</li>
    <li>2</li>
    <li>3</li>
    <li>4</li>
    <li>5</li>
    <li>6</li>
  </ul>
  <span>Good Vibes</span>
</body>
</html>
```

▼ Attribute Selectors

Attribute selector is used to select elements with a specified attribute and value.

▼ Has Attribute (" [manu] ")

- It selects the elements that have that attribute.

▼ Example

```
<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    [manu]
    {
      background-color: red;
    }
  </style>
</head>
<body>
  <span manu> Good Vibes Only </span>
  <div>
    <b> Ok Prends Bye</b>
  </div>
</body>
</html>
```

▼ Exact Attribute (" [manu="1"] ")

- It selects the elements that have that attribute with exactly that value.

▼ Example

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```

<style>
  [manu="1"]
  {
    background-color: red;
  }
</style>
</head>
<body>
  <span manu="1"> Good Vibes Only </span>
  <div>
    <b> Ok Prends Bye</b>
  </div>
</body>
</html>

```

▼ Begins With Attribute (" [manu^="1"] ")

- It selects the elements that have that attribute which start with that value.

▼ Example

```

<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    [manu^="12"]
    {
      background-color: red;
    }
  </style>
</head>
<body>
  <span manu="123"> Good Vibes Only </span>
  <div>
    <b> Ok Prends Bye</b>
  </div>
</body>
</html>

```

▼ Ends with Attribute (" [manu\$="1"] ")

- It selects the elements that have that attribute which end with that value.

▼ Example

```

<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    [manu$="23"]
    {
      background-color: red;
    }
  </style>
</head>
<body>
  <span manu="123"> Good Vibes Only </span>
  <div>
    <b> Ok Prends Bye</b>
  </div>
</body>
</html>

```

▼ Substring Attribute (" [manu*="1"] ")

- It selects the elements that have that attribute which contain that value anywhere.

▼ Example

```

<!DOCTYPE html>
<html lang="en">
<head>

```

```

<style>
  [manu*="23"]
  {
    background-color: red;
  }
</style>
</head>
<body>
  <span manu="1233"> Good Vibes Only </span>
  <div>
    <b> Ok Prends Bye</b>
  </div>
</body>
</html>

```

30. Create a Spotify Project ?

▼ Ans

▼ App.jsx

```

import Spotify from "../Components/Spotify"
const App=()=>{
  return(
    <div>
      <Spotify/>
    </div>
  )
}
export default App

```

▼ Spotify.jsx

```

import style from "../spotify.module.css"
import pic from "../bubbles-mobile.svg"

const Spotify=()=>{
  return(
    <div>
      <section id={style.nav}>
        <article>
          <div className={style.Logo}><svg viewBox="0 0 63 26" xmlns="http://www.w3.org/2000/svg" preserveAspectRatio=
          <div className={style.Menu}>
            <ol>
              <li><a href="">Premium</a></li>
              <li><a href="">Sipport</a></li>
              <li><a href="">Download</a></li>
              <li><a href=""></a>|</li>
              <li><a href="">Sign up</a></li>
              <li><a href="">Login</a></li>
            </ol>
          </div>
        </article>
      </section>
      <img src={pic} alt="" />
    </div>
  )
}
export default Spotify

```

▼ spotify.module.css

```

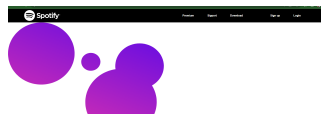
*{
  padding: 0;
  margin: 0;
  box-sizing: border-box;
}
#nav{
  height: 80px;
  width: 100%;
  background-color: black;
  display: flex;

```

```

        justify-content: center;
    }
    #nav>article{
        height: 80px;
        width: 90%;
        /* border: 2px solid red; */
        display: flex;
    }
    #nav>article>.Logo{
        height: 80px;
        width: 50%;
        /* background-color: antiquewhite; */
        display: flex;
        align-items: center;
    }
    #nav>article>.Menu{
        height: 80px;
        width: 50%;
        /* background-color:aqua; */
    }
    #nav>article>.Logo>svg{
        width: 25%;
        fill: white;
    }
    #nav>article>.Menu>ol{
        height:100%;
        display: flex;
        list-style-type: none;
        justify-content: space-evenly;
        align-items: center;
    }
    #nav>article>.Menu>ol>li>a{
        text-decoration: none;
        color: white;
        font-family: Arial, Helvetica, sans-serif;
        font-weight: bold
    }
}

```



31. What is **useRef()** ?

▼ Ans

- **useRef()** is a hook which is used to access a DOM element directly by using **ref** attribute.
- **useRef()** returns an Object called “**current**”. We initialize `const count = useRef(0);` It's like doing this `const count = {current: 0}` We can access the count by using `count.current`
- For more —> "<https://youtu.be/t2ypzz6gJm0>"

32. Reference Project ?

▼ Ans

- **ref** is an attribute and inbuilt object which is used to target the elements. default key and value pair is “**current**” and “**undefined**”. In order to use **ref** attribute, We must use **useRef()** hook.

▼ App.jsx

```

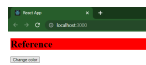
import Reference from "../Components/Reference"
const App=()=>{
    return(
        <div>
            <Reference/>
        </div>
    )
}
export default App

```


▼ Reference.jsx

```
import { useRef } from "react"

const Reference=()=>{
  let demoRef=useRef()
  console.log(demoRef)
  let btn={()=>{
    demoRef.current.style.background="red"
  }}
  return(
    <div>
      <h1 ref={demoRef}>Reference</h1>
      <button onClick={btn}>Change color</button>
    </div>
  )
}
export default Reference
```



33. Background THEME project ?

▼ Ans

▼ App.jsx

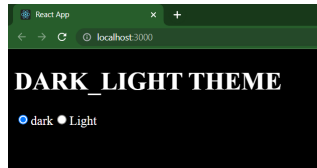
```
import Theme from "../Components/Theme"

const App=()=>{
  return(
    <div>
      <Theme/>
    </div>
  )
}
export default App
```

▼ Theme.jsx

```
const Theme=()=>{
  let dark={()=>{
    {
      document.body.style.background="black"
      document.body.style.color="white"
    }
  }}
  let light={()=>{
    {
      document.body.style.background="pink"
      document.body.style.color="black"
    }
  }}
  return(
    <div>
      <h1>DARK_LIGHT THEME</h1>
      <input type="radio" name="a" onClick={dark}/>
      <label >dark</label>
      <input type="radio" name="a" onClick={light}/>
      <label >Light</label>

    </div>
  )
}
export default Theme
```



34. What are the Form Handling in React JS ?

▼ Ans

1. Uncontrolled Forms

- It is a form, Where it is created using Reference concept.
- In FBC, We use `useRef()` hook to create Uncontrolled forms.
- These Forms are completely handled by Dom itself.
- Suppose if we wanted to take any data from user first the data will be taken by DOM and then we will be taken the data from Dom.

2. Controlled Forms

- These forms are created using states in React.
- In FBC, We use `useState()` hook to create controlled forms.
- These forms are completely handled by developers where we will be taking the data directly from the users by using `onChange` event.

35. Controlled and Uncontrolled Forms ?

▼ Ans

- Controlled —> states
- Uncontrolled —> Ref

▼ Uncontrolled (Example 1)

▼ App.jsx

```
import Uncontrolled from "../Components/Uncontrolled"

const App=()=>>{
  return(
    <div>
      <Uncontrolled/>
    </div>
  )
}
export default App
```

▼ Uncontrolled.jsx

```
import { useRef } from "react"

const Uncontrolled=()=>>{
  let name=useRef()
  let email=useRef()

  let formHandle=(e)=>{
    let nameData=name.current.value
    let emailData=email.current.value

    e.preventDefault()
    console.log(nameData)
    console.log(emailData)
  }
  return(
```

```

    <div>
      <form action="">
        <label htmlFor="">Name: </label>
        <input type="text" ref={name}/><br/><br/>
        <label htmlFor="">Email: </label>
        <input type="text" ref={email}/> <br /><br />
        <button onClick={formHandle}>Submit</button>
      </form>
    </div>
  )
}
export default Uncontrolled

```

▼ Uncontrolled (Example 2)

▼ App.jsx

```

import Uncontrolled from "../Components/Uncontrolled"

const App=()=>{
  return(
    <div>
      <Uncontrolled/>
    </div>
  )
}
export default App

```

▼ Uncontrolled.jsx

```

import { useRef } from "react"

const Uncontrolled=()=>{
  let name=useRef()
  console.log(name)

  let formHandle=(e)=>{
    e.preventDefault()
    console.log(e)
    console.log(name.current.value)
  }
  return(
    <div>
      <form onSubmit={formHandle}>
        <label htmlFor="">Name: </label>
        <input type="text" ref={name}/><br/>
        <input type="submit" value="submit"/>
      </form>
    </div>
  )
}
export default Uncontrolled

```

▼ Uncontrolled (Example 3)

▼ App.jsx

```

import Uncontrolled from "../Components/Uncontrolled"

const App=()=>{
  return(
    <div>
      <Uncontrolled/>
    </div>
  )
}
export default App

```

▼ Uncontrolled.jsx

```
import { useRef,useState } from "react"

const Uncontrolled=()=>{
  let num1=useRef()
  let num2=useRef()
  let [res,setRes]=useState("")

  let add=()=>{
    let a=Number(num1.current.value)
    let b=Number(num2.current.value)
    console.log(a+b)
    setRes(a+b)
  }
  let sub=()=>{
    let a=Number(num1.current.value)
    let b=Number(num2.current.value)
    console.log(a-b)
    setRes(a-b)
  }
  let multi=()=>{
    let a=Number(num1.current.value)
    let b=Number(num2.current.value)
    console.log(a*b)
    setRes(a*b)
  }
  return(
    <div>
      <label htmlFor="">Enter First Number</label>
      <input type="number" ref={num1}/><br /><br />
      <label htmlFor="">Enter Second Number</label>
      <input type="number" ref={num2}/><br />
      <button onClick={add}>+</button><br />
      <button onClick={sub}>-</button><br />
      <button onClick={multi}>*</button><br />
      <h1>Answer = {res}</h1>
    </div>
  )
}
export default Uncontrolled
```

▼ Controlled (Example 1)

▼ App.jsx

```
import Controlled from "../Components/Controlled"

const App=()=>{
  return(
    <div>
      <Controlled/>
    </div>
  )
}
export default App
```

▼ Controlled.jsx

```
import { useState } from "react"

const Controlled=()=>{
  let [name,setName]=useState("")
  let [email,setEmail]=useState("")

  let nameData=(e)=>{
    console.log(e.target.value);
    setName(e.target.value)
  }
  let emailData=(e)=>{
    console.log(e.target.value);
    setEmail(e.target.value)
  }
  let formHandle=(e)=>{
    e.preventDefault()
    console.log(name,email)
  }
}
```

```

    }
    return(
      <div>
        <form action="">
          <label>NAME: </label>
          <input type="text" placeholder="Enter Your Name" value={name} onChange={nameData}/> <br /><br />
          <label>Email: </label>
          <input type="email" placeholder="Enter your Email" value={email} onChange={emailData}/><br /><br />
          <button onClick={formHandle}>Submit</button>
        </form>
        {name}
      </div>
    )
  }
  export default Controlled

```

▼ Controlled (Example 2)

▼ App.jsx

```

import Controlled from "../Components/Controlled"

const App=()=>{
  return(
    <div>
      <Controlled/>
    </div>
  )
}
export default App

```

▼ Controlled.jsx

```

import { useState } from "react"

const Controlled=()=>{
  let [n1,setN1]=useState("")
  let [n2,setN2]=useState("")
  let [ans,setAns]=useState(0)

  let n1data=(e)=>{
    console.log(e.target.value);
    setN1(e.target.value)
  }
  let n2data=(e)=>{
    console.log(e.target.value);
    setN2(e.target.value)
  }
  let add=(e)=>{
    e.preventDefault()
    let ans=Number(n1)+Number(n2)
    console.log(ans)
    setAns(ans)
  }
  let sub=(e)=>{
    e.preventDefault()
    let ans=Number(n1)-Number(n2)
    console.log(ans)
    setAns(ans)
  }
  let multi=(e)=>{
    e.preventDefault()
    let ans=Number(n1)*Number(n2)
    console.log(ans)
    setAns(ans)
  }
  return(
    <div>
      <form action="">
        <label>Number 1</label>
        <input type="text" value={n1} onChange={n1data}/><br/><br />
        <label>Number 2</label>

```

```

        <input type="text" value={n2} onChange={n2data}/><br/><br />
        <button onClick={add}>+</button><br /><br />
        <button onClick={sub}>-</button><br /><br />
        <button onClick={multi}>*</button><br /><br />
        <h1>{ans}</h1>
      </form>
    </div>
  )
}
export default Controlled

```

▼ Controlled (Example 3)

▼ Controlled.jsx

```

import { useState } from "react"

const Controlled={()=>{
  let [n1,setN1]=useState("")
  let [n2,setN2]=useState("")
  let [ans,setAns]=useState(0)

  let n1data=(e)=>{
    console.log(e.target.value);
    setN1(e.target.value)
  }
  let n2data=(e)=>{
    console.log(e.target.value);
    setN2(e.target.value)
  }
  let add=(e)=>{
    e.preventDefault()
    let ans=Number(n1)+Number(n2)
    console.log(ans)
    setAns(ans)
  }
  let sub=(e)=>{
    e.preventDefault()
    let ans=Number(n1)-Number(n2)
    console.log(ans)
    setAns(ans)
  }
  let multi=(e)=>{
    e.preventDefault()
    let ans=Number(n1)*Number(n2)
    console.log(ans)
    setAns(ans)
  }
}
return(
  <div>
    <form action="">
      <label>Number 1</label>
      <input type="text" value={n1} onChange={n1data}/><br/><br />
      <label>Number 2</label>
      <input type="text" value={n2} onChange={n2data}/><br/><br />
      <button onClick={add}>+</button><br /><br />
      <button onClick={sub}>-</button><br /><br />
      <button onClick={multi}>*</button><br /><br />
      <h1>{ans}</h1>
    </form>
  </div>
)
}
export default Controlled

```

36. What is High Order Component ?

▼ Ans

- A components which accepts another components as an arguments is called High Order Components. The main use of Higher Order Component is reusability. In case ,If we want to write the same logic for multiple different Component then We can write our logics in a HOC, instated of writing in every single Components then we can use it in another components.

- map is higher order function.

▼ App.jsx

```
import Amanu from "../Components/Amanu"
import Bmanu from "../Components/Bmanu"

const App=()=>{
  return(
    <div>
      <Amanu/>
      <Bmanu/>
    </div>
  )
}
export default App
```

▼ Hoc.jsx

```
import { useState } from "react";

const Hoc=(Wrappedcomp)=>{
  function NesteHoc(){
    let [count,setCounnt] = useState(0)

    let incre=()=>{
      setCounnt(count+1)
    }
    return(
      <div>
        <Wrappedcomp data={count} Func={incre}/>
      </div>
    )
  }
  return NesteHoc
}
export default Hoc
```

▼ Amanu.jsx

```
import Hoc from "../Hoc"

const Amanu=(x)=>{
  return(
    <div>
      {x.data}
      <button onMouseOver={x.Func}>INCREAMENT</button>
    </div>
  )
}
export default Hoc(Amanu)
```

▼ Bmanu.jsx

```
import Hoc from "../Hoc"

const Bmanu=(x)=>{
  return(
    <div>
      {x.data}
      <button onClick={x.Func}>INCREAMENT</button>
    </div>
  )
}
export default Hoc(Bmanu)
```

37. How many types of Hooks are there ?

▼ Ans

1. Basic Hooks

2. Advanced Hooks

Basic Hooks	Advanced Hooks
1). useState()	1). useRef()
2). useEffect()	2). useCallback()
3). useContext()	3). useMemo()
	4). useNavigate()
	5). useParams()

38. What is **useEffect()** ?

▼ Ans

- **useEffect()** is a Hook which is used to apply the Side Effects and also it is mainly used for fetching the data from the database and it takes 2 parameter.

1. Call Back Function

- If we pass only call back function, then the **useEffect()** will execute for every change which we make on the component.

2. Dependencies

- If we pass call back function along with empty dependency, then **useEffect()** will execute only one time when components gets rendered.

▼ App.jsx

```
import UseEffec from "../Components/UseEffec"

const App={()=>{
  return(
    <div>
      <UseEffec/>
    </div>
  )
}
export default App
```

▼ UseEffec.jsx

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

const UseEffec={()=>{
  let [state, setState] = useState(0)
  let [state1, setState1] = useState(0)
  let [state2, setState2] = useState(0)

  useEffect(()=>{
    console.log("Component is Loaded");
  },[state,state1])
  return(
    <div>
      {state}
      <button onClick={()=>{setState(state+1)}}>INCREAMENT</button> <br />
      {state1}
      <button onClick={()=>{setState1(state1+5)}}>INCREAMENT1111</button><br />
      {state2}
      <button onClick={()=>{setState2(state2+100)}}>INCREAMENT22222</button>
    </div>
  )
}
export default UseEffec
```

39. Fetching the Data from Database using **useEffect()** Hook and Printing in Console Window ?

▼ Ans

1. Types **"npm install axios"** and It will download some files → We can see that in package.json → "axios": "^1.2.3"
2. Go to google Search —Type→ **"jsonplaceholder"** or **"<https://jsonplaceholder.typicode.com/posts>"** (For dummy Data)

▼ App.jsx

```
import Fetchdata from "../Components/Fetchdata"

const App=()=>{
  return(
    <div>
      <Fetchdata/>
    </div>
  )
}
export default App
```

▼ Fetchdata.jsx

```
import axios from "axios"
import { useEffect } from "react"

const Fetchdata=()=>{
  useEffect(()=>{
    axios.get("https://jsonplaceholder.typicode.com/posts")
      .then((response)=>{
        console.log("Got The Data")
        console.log(response.data)
      })
      .catch(()=>{
        console.log("I did not get the Data")
      })
  },[])
  return(
    <div>

    </div>
  )
}
export default Fetchdata
```

- **get()** → is used to fetch the data from the particular URL.

40. Fetching the Data from Database using **useEffect()** and **useState()** Hooks and Printing in UI ?

▼ Ans

▼ Fetchdata.jsx

```
import axios from "axios"
import { useState ,useEffect } from "react"

const Fetchdata=()=>{

  let [content , setContent]=useState([])

  useEffect(()=>{
    axios.get("https://jsonplaceholder.typicode.com/posts")
      .then((response)=>{
        console.log("Got The Data")
        setContent(response.data)
        console.log(response.data)
      })
  },[])
  console.log(content)

  return(
```

```

        <div>
          {content.map((x)=>{
            return(
              <div>
                <h1>{x.id}</h1>
                <h3>{x.title}</h3>
              </div>
            )
          })}
        </div>
      )
    }
  }
  export default Fetchdata

```

41. Fetching the Particular Data from Database using onChange event ?

▼ Ans

▼ FetchData4.jsx

```

import axios from "axios"
import { useState ,useEffect } from "react"

const FetchData4={()=>{

  let [content , setContent]=useState([])
  let [id,setId] = useState("")

  useEffect(()=>{
    axios.get(`https://jsonplaceholder.typicode.com/posts/${id}`)
    .then((response)=>{
      setContent(response.data)
      console.log(response.data)
      console.log("Got the Data")
    })
  },[id])

  let idData=(e)=>{
    setId(e.target.value)
    console.log(e.target.value)
  }

  return(
    <div>
      <input type="text" value={id} onChange={idData}/><br />
      {content.title}
    </div>
  )
}
export default FetchData4

```

42. Fetching the Particular Data from Database using Submit button ?

▼ Ans

▼ FetchData5.jsx

```

import axios from "axios"
import { useState ,useEffect } from "react"

const FetchData5={()=>{

  let [content , setContent]=useState([])
  let [id,setId] = useState("")
  let [btn,setBtn] = useState("")

  useEffect(()=>{
    axios.get(`https://jsonplaceholder.typicode.com/posts/${btn}`)
    .then((response)=>{
      setContent(response.data)
      console.log(response.data)
      console.log("Got the Data")
    })
  },[btn])
}

```

```

    let idData=(e)=>{
      setId(e.target.value)
      console.log(e.target.value)
    }
    let FormHandling={()=>{
      setBtn(id)
    }}
  }
  return(
    <div>
      <label htmlFor="">Enter ID</label>
      <input type="text" value={id} onChange={idData}/>
      <button onClick={FormHandling}>SUBMIT</button>
      {content.title}
    </div>
  )
}
export default FetchData5

```

43. What is `useContext()` ?

▼ Ans

- To avoid props drilling, we use `useContext()`.

▼ App.jsx

```

import { createContext } from "react"
import X from "../Components/X"

export let userData = createContext()
const App={()=>{
  return(
    <div>
      <userData.Provider value="Hi">
        <X/>
      </userData.Provider>
    </div>
  )
}
export default App

```

▼ X.jsx

```

import Y from "../Y"

const X={()=>{
  return(
    <div>
      <Y/>
    </div>
  )
}
export default X

```

▼ Y.jsx

```

import { useContext } from "react"
import { userData } from "../App"

const Y={()=>{
  let user = useContext(userData)
  return(
    <div>
      {user} Good Morning
    </div>
  )
}
export default Y

```

44. `useContext()` ? (Passing an Array)

▼ Ans

▼ App.jsx

```
import { createContext } from "react"
import X from "../Components/X"

export let userData = createContext()
const App={()=>{
  return(
    <div>
      <userData.Provider value={['Hi', 'Hello']}>
        <X/>
      </userData.Provider>
    </div>
  )
}
export default App
```

▼ X.jsx

```
import Y from "../Y"

const X={()=>{
  return(
    <div>
      <Y/>
    </div>
  )
}
export default X
```

▼ Y.jsx

```
import { useContext } from "react"
import { userData } from "../App"

const Y={()=>{
  let user = useContext(userData)
  return(
    <div>
      {user[0]} Good Morning <br />
      {user[1]} Good Morning
    </div>
  )
}
export default Y
```

45. Age and Salary Increment Project ? (Components Reusing Concept)

▼ Ans

▼ App.jsx

```
import Main from "../Components/Main"

const App={()=>{
  return(
    <div>
      <Main/>
    </div>
  )
}
export default App
```

▼ Main.jsx

```
import Count from "../Count"
import Button from "../Button"
import { useState } from "react"

const Main = () => {
  let [age, setAge] = useState(25)
  let [salary, setSalary] = useState(25000)

  let increAge = () => { setAge(age + 1) }
  let increSalary = () => { setSalary(salary + 5000) }

  return (
    <div>
      <Count data={age}> Age </Count>
      <Button Func={increAge}>
      <Count data={salary}> salary </Count>
      <Button Func={increSalary}>
    </div>
  )
}
export default Main
```

▼ Count.jsx

```
const Count = (x) => {
  return (
    <div>
      {x.children}={x.data}
    </div>
  )
}
export default Count
```

▼ Button.jsx

```
const Button = (x) => {
  return (
    <div>
      <button onClick={x.Func}> Incre </button>
    </div>
  )
}
export default Button
```

46. What is **useCallback()** ? (You have doubt remember and ask)

▼ Ans

▼ App.jsx

```
import Main from "../Components/Main"

const App = () => {
  return (
    <div>
      <Main/>
    </div>
  )
}
export default App
```

▼ Main.jsx

```
import Count from "../Count"
import Button from "../Button"
import { useState } from "react"
import { useCallback } from "react"
```

```
const Main=()=>{
  let [age,setAge]= useState(25)
  let [salary,setSalary]=useState(25000)

  let increAge=useCallback(()=>{
    setAge(age+1)
  },[age])

  let increSalary=useCallback(()=>{
    setSalary(salary+5000)
  },[salary])
  return(
    <div>
      <Count data={age}>Age</Count>
      <Button Func={increAge}>Age</Button>
      <Count data={salary}>Salary</Count>
      <Button Func={increSalary}>Salary</Button>
    </div>
  )
}
export default Main
```

▼ Count.jsx

```
import React from "react";
const Count=(x)=>{
  console.log("(Count) Rendering 1",x.children);
  return(
    <div>
      <h1>{x.children}={x.data}</h1>
    </div>
  )
}
export default React.memo(Count)
```

▼ Button.jsx

```
import React from "react"
const Button=(x)=>{
  console.log("(Button) Rendering 2",x.children)
  return(
    <div>
      <button onClick={x.Func}>Incre {x.children}</button>
    </div>
  )
}
export default React.memo(Button)
```

47. What is **useMemo()** ?

▼ Ans

- The point of using **useMemo()** is If we are writing complex functionality happen, that lagging can effect will happen, that lagging can effect for every. we are using **useMemo()** hook, If we use **useMemo()** hook that lagging will happen only for that particular elements

▼ App.jsx

```
import Perform from "../Components/Perform"

const App=()=>{
  return(
    <div>
      <Perform/>
    </div>
  )
}
export default App
```

▼ Perform.jsx

```
import { useMemeo } from "react"
import { useState } from "react"

const Perform=()=>{
  let [count1 , setCount1] = useState(0)
  let [count2 , setCount2] = useState(0)

  let Incre1 = ()=>{setCount1(count1+1)}
  let Incre2 = ()=>{setCount2(count2+1)}

  let Even = useMemeo(()=>{
    let i=0
    while(i<100000) i++
    return count1%2==0
  },[count1])

  return(
    <div>
      <center>
        {count1} is {Even?"Even":"ODD"} Man <br />
        <button onClick={Incre1}>Click Me 1 </button> <br />

        {count2} <br />
        <button onClick={Incre2}>Click Me 2 </button>
      </center>
    </div>
  )
}
export default Perform
```

48. useMemeo() and useCallback() ?

▼ Ans

- The **useMemeo()** and **useCallback()** Hooks are advance hook and both are similar. The main difference is that **useMemeo()** returns a memoized value and **useCallback()** returns a memoized function. These Hook only runs when one of its dependencies updated. This can improve performance of an React Application.

40. Single Page Application Project ? (FindCoder)

▼ Ans

- For Single Page Application, We should download --> **"npm install react-router-dom"**
- Open FindCoder Website —> ["https://www.findcoder.io/"](https://www.findcoder.io/)

▼ App.jsx

```
import Navbar from "../Components/Navbar"
import { BrowserRouter,Routes,Route } from "react-router-dom"
import Expolore from "../Components/Explore"
import Hire from "../Components/Hire"
import Dev from "../Components/Dev"
import Challenge from "../Components/Challenge"

const App=()=>{
  return(
    <div>
      <BrowserRouter>
        <Navbar/>
        <Routes>
          <Route path="/explore" element={}<Expolore/>}/>
          <Route path="/hire" element={}<Hire/>}/>
          <Route path="/dev" element={}<Dev/>}/>
          <Route path="/challenge" element={}<Challenge/>}/>
        </Routes>
      </BrowserRouter>
    </div>
  )
}
export default App
```

▼ Navbar.jsx

```
import style from "./nav.module.css"
import Logo from "./Logo"
import Menu from "./Menu"
import Button from "./Button"
const Navbar={() => {
  return (
    <div>
      <section id={style.nav} >
        <article>
          <div className={style.Logo}><Logo/> </div>
          <div className={style.Menu} ><Menu/> </div>
          <div className={style.Button} ><Button/> </div>
        </article>
      </section>
    </div>
  )
}
export default Navbar
```

▼ Logo.jsx

```
import style from "./nav.module.css"
import { Link } from "react-router-dom"

const Logo={() => {
  return (
    <div id={style.logoBlock}>
      <Link to="/Home">
        <svg width="38" height="22" viewBox="0 0 38 22" fill="none" xmlns="http://www.w3.org/2000/svg"><path d="M0.356678"
        <p>FindCoder</p>
      </Link>
    </div>
  )
}
export default Logo
```

▼ Menu.jsx

```
import style from "./nav.module.css"
import { Link } from "react-router-dom"
const Menu={() => {
  return (
    <div id={style.menuBlock}>
      <ol>
        <li><Link to="/explore">Explore work</Link> </li>
        <li><Link to="/hire"> Hire</Link> </li>
        <li><Link to="/dev">Dev</Link> </li>
        <li><Link to="/challenge">Challenge</Link> </li>
      </ol>
    </div>
  )
}
export default Menu
```

▼ Explore.jsx

```
import style from "./menu.module.css"
import pic from "./pic1.jpg"
const Expolore={() => {
  return (
    <div>
      <section id={style.exploreBlock} >
        <article>
          <div className={style.summary}>
            <div className={style.summaryText}>
              <h1>The Only Platform To</h1>
              <h2>Show Off Your Skill</h2>
              <p>collab or hires coders with no middleman. An open platform
```



```

        to hire coders based on skills and projects.</p><br /><br />
        <button className={style.button1}>EXPLORE WORK</button>
        <button className={style.button2}>HIRE TALENTS</button>
      </div>
    </div>
    <div className={style.image}>
      <img src={pic} alt="" />
    </div>
  </article>
</section>
</div>
)
}
export default Expolore

```

▼ Button.jsx

```

import style from "../nav.module.css"
const Button={()=>{
  return(
    <div id={style.buttonBlock}>
      <button>
        <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke-width="2" stroke="currentColor">
          <p>Get Started</p>
        </button>
      </div>
    )
  }
}
export default Button

```

▼ Challenge.jsx

```

const Challenge={()=>{
  return(
    <div>
      <h1> I am Challange</h1>
    </div>
  )
}
export default Challenge

```

▼ Dev.jsx

```

const Dev={()=>{
  return(
    <div>
      <h1>I am Dev</h1>
    </div>
  )
}
export default Dev

```

▼ Hire.jsx

```

const Hire={()=>{
  return(
    <div>
      <h1>Hello</h1>
      <h1>I am Hire</h1>
    </div>
  )
}
export default Hire

```

▼ nav.module.css

```

*{
  padding: 0;
  margin: 0;
  box-sizing: border-box;
}
#nav{
  height: 70px;
  width: 100%;
  box-shadow: 0px 0px 5px rgb(45, 146, 226);
  display: flex;
  justify-content: center;
}
#nav>article{
  height: 70px;
  width: 95%;
  /* border: 1px solid black; */
  display: flex;

}
#nav>article>.Logo{
  height: 70px;
  width: 30%;
  /* background-color: aqua; */
}
#nav>article>.Menu{
  height: 70px;
  width: 40%;
  /* background-color: red; */
}
#nav>article>.Button{
  height: 70px;
  width: 30%;
  /* background-color: blue; */
}
#logoBlock{
  height: 100%;
  width: 100%;
  display: flex;
  align-items: center;
}
#logoBlock>p{
  margin-left: 12px;
  font-size: 26px;
  font-family: Arial, Helvetica, sans-serif;
  color: rgb(45, 146, 226);
}
#menuBlock{
  height: 100%;
  width: 100%;
}
#menuBlock>ol{
  height: 100%;
  display: flex;
  justify-content: space-around;
  list-style-type: none;
  align-items: center;
}
#menuBlock>ol>li>a{
  text-decoration: none;
  font-family: Arial, Helvetica, sans-serif;
  font-size: 14px;
  font-weight: bold;
  color: rgb(86, 85, 85);
}
#buttonBlock{
  height: 100%;
  width: 100%;
  display: flex;
  align-items: center;
  justify-content: flex-end;
}
#buttonBlock>button{
  border: 0px solid rgb(23,124,226);
  height: 30px;
  width: 130px;
  /* border: none; */
  border-radius: 4px;
  display: flex;
  color: white;
}

```

```

        align-items: center;
        margin-left: auto;
        margin-right: 5px;
        background-color: rgb(23,124,226);
    }
    #buttonBlock>button>svg{
        height: 90%;
        margin-left: 8px;
        color: white;
    }
    #buttonBlock>button>p{
        margin-left: 10px;
        font-size: 14px;
        font-family:sans-serif ;
        color: white;
    }
}

```

▼ menu.module.css

```

#exploreBlock{
    height: 70vh;
    width: 100%;
    display: flex;
    justify-content: center;
}
#exploreBlock>article{
    height: 70%;
    width: 90%;
    /* border: 1px solid black; */
    display: flex;
}
#exploreBlock>article>.summary{
    height: 100%;
    width: 50%;
    /* border: 1px solid black; */
    /* display: flex; */
}
#exploreBlock>article>.summary>.summaryText{
    margin-top: 12%;
    margin-left: 5%;
    height: 50%;
}
#exploreBlock>article>.summary>.summaryText>h1{
    font-size: 36px;
    font-weight: bold;
    color: rgb(69,72,77);
    font-family: Arial, Helvetica, sans-serif;
}
#exploreBlock>article>.summary>.summaryText>h2{
    font-size: 46px;
    font-weight: bold;
    font-family: Arial, Helvetica, sans-serif;
    color: rgb(23,124,226);
}
#exploreBlock>article>.summary>.summaryText>p{
    font-size: 20px;
    margin-top: 12px;
    color: rgb(97, 97, 97);
    font-family: Verdana, Geneva, Tahoma, sans-serif;
}

#exploreBlock>article>.summary>.summaryText>.button1{
    background-color: rgb(23,124,226);
    border: 1px solid rgb(23,124,226);
    /* border-color: rgb(23,124,226); */
    color: white;
    font-family: Arial, Helvetica, sans-serif;
    padding: 7px;
    border-radius: 4px;
}
#exploreBlock>article>.summary>.summaryText>.button2{
    /* border-color: rgb(23,124,226); */
    color: rgb(23,124,226);
    border: 1px solid rgb(23,124,226);
    background-color: white;
    font-family: Arial, Helvetica, sans-serif;
    padding: 7px;
}

```

```

border-radius: 4px;
margin-left: 10px
}

#exploreBlock>article>.image{
  height: 100%;
  width: 50%;

  display: flex;
  justify-content: space-around;
}
#exploreBlock>article>.image>img{
  height:100% ;
  margin-top: 8px;
}

```

41. CRUD Operation Project ?

▼ Ans

- If we want to do CRUD operation Project then We should create one Database by name DatabaseName.json in one separate folder and We should always run 2 commands :-

1. **"npm start"**
2. **"npx json-server Backend/DatabaseName.json --watch port 5000"**

- And Install before starting the project

- **For Single Page Application** ---> **"npm install react-router-dom"**
- **For Axios** —> **"npm install axios"**
- **For BackEnd purpose** —> **"npm install json-server"**
- **Port opening** —> **"npx json-server Backend/db.json - -watch port 5000"**

▼ index.js

```

import {createRoot} from 'react-dom/client'
import App from './App'
createRoot(document.getElementById('root')).render(<App></App>)

```

▼ App.jsx

```

import Homepage from './Crud/Homepage'
import { BrowserRouter, Routes, Route } from "react-router-dom"
import Createusers from './Crud/Createusers'
import Users from './Crud/Users'
import Editusers from './Crud/Editusers'

const App=()=>{
  return(
    <div>
      <BrowserRouter>
        <Homepage/>
        <Routes>
          <Route path="/" element={<Createusers/>}></Route>
          <Route path="/user" element={<Users/>}></Route>
          <Route path="/Edit/:index" element={<Editusers/>}></Route>
        </Routes>
      </BrowserRouter>
    </div>
  )
}
export default App

```

▼ Homepage.jsx

```

import style from './home.module.css'
import { Link } from "react-router-dom"

```

```

const Homepage={()=>{
  return(
    <section id={style.nav}>
      <Link to="/">Create-User</Link>
      <Link to="/user">User</Link>
    </section>
  )
}

export default Homepage

```

▼ home.module.css

```

*{
  margin: 0%;
  padding: 0%;
  box-sizing: border-box;
}
#nav{
  height: 70px;
  width: 100%;
  background-color: black;
  display: flex;
  justify-content: space-around;
  align-items: center;
}
#nav>a{
  color: white;
  background-color: black;
  text-decoration: none;
  font-family: Arial, Helvetica, sans-serif;
  font-size: 20px;
  border: 2px solid white;
  border-radius: 30px;
  padding: 10px;
}
#nav>a:hover{
  color: black;
  transform: scale(1.1);
  background-color: white;
  font-weight: bold;
}
#myForm{
  height: 90vh;
  width: 100%;
  background: linear-gradient(white,black);
  display: flex;
  justify-content: center;
  align-items: center;
}
#myForm>table{
  height: 250px;
  width: 280px;
  border: 1px solid white;
  border-radius: 5px;
  box-shadow: 0px 0px 5px white;
}
#myForm>table label{
  color: white;
  font-style: italic;
  font-family: Arial, Helvetica, sans-serif;
  margin-left: 10px;
}
#myForm>table input{
  background-color: transparent;
  border: none;
  border-bottom: 1px solid white;
  color: white;
}
#myForm>table input:focus{
  outline: none;
}
button{
  height: 30px;
  width: 130px;
  background-color: gainsboro;
  border: none;
}

```

```

        border-radius: 100px;
    }
    button:hover{
        transform: scale(1.1);
        transition: 0.5s;
        cursor: pointer;
    }
    h4{
        font-style: italic;
        font-weight: bold;
        font-family: Arial, Helvetica, sans-serif;
        color: black;
        text-shadow: 0px 0px 5px white;
    }
    #myUsers{
        height: 90vh;
        width: 100%;
        background: linear-gradient(white,black);
        display: flex;
        justify-content: space-around;
        align-items: center;
        flex-wrap: wrap;
        padding: 20px;
    }
    #cards{
        height: 300px;
        width: 300px;
        border: 1px solid white;
        border-radius: 5px;
        box-shadow: 0px 0px 5px white;
    }
    #cards:hover{
        transform: scale(1.1);
        transition: 1s;
    }
    #cards>table{
        padding: 40px;
    }
    #cards td{
        color: white;
        padding: 10px;
        font-family: Arial, Helvetica, sans-serif;
    }
    #cards td>button{
        height: 20px;
        width: 80px;
        background-color: white;
        display: inline-block;
    }
    #cards td>button>a{
        text-decoration: none;
        color: black;
    }
}

```

▼ Createusers.jsx

```

import axios from "axios"
import { useState } from "react"
import { useNavigate } from "react-router-dom"
import style from "./home.module.css"

const Createusers={()=>{
    let [name, setName] = useState("")
    let [salary, setSalary] = useState("")
    let [company, setCompany] = useState("")

    let navigate = useNavigate()

    let nameData = (e) =>{setName(e.target.value)}
    let salaryData = (e) =>{setSalary(e.target.value)}
    let companyData = (e) =>{setCompany(e.target.value)}

    let formhandle = (e) =>{
        e.preventDefault()
        let payload ={name,salary,company}
        axios.post("http://localhost:3000/content",payload)
        .then(()=>{console.log("Data Has been added")})
        .catch(()=>{"Something is wrong"})
    }
}

```

```

    navigate("/user")
  }

  return(
    <div id={style.myForm}>

      <table>
        <tr>
          <th colspan="2"><h4>User Details</h4></th>
        </tr>
        <tr>
          <td><label>Name</label></td>
          <td>:<input type="text" value={name} onChange={nameData}/></td>
        </tr>
        <tr>
          <td><label>Salary</label></td>
          <td>:<input type="text" value={salary} onChange={salaryData}/></td>
        </tr>
        <tr>
          <td><label>Company</label></td>
          <td>:<input type="text" value={company} onChange={companyData}/></td>
        </tr>
        <tr>
          <th colspan="2"><button onClick={formhandle}>Submit</button></th>
        </tr>
      </table>

    </div>
  )
}
export default Createusers

```

▼ Users.jsx

```

import axios from "axios"
import { useEffect, useState } from "react"
import style from "../home.module.css"
import { Link } from "react-router-dom"

const Users={()=>{

  let [data, setData] = useState([])

  useEffect(()=>{
    axios.get("http://localhost:3000/content")
    .then((response)=>{
      console.log("Got The Datas")
      setData(response.data)
    })
    .catch(()=>{console.log("Didn't get the data")})
  },[])

  let deleteData=(id)=>{
    axios.delete(`http://localhost:3000/content/${id}`)
    window.location.assign("/user")
  }

  return(
    <div id={style.myUsers}>
      {data.map((x)=>{
        return(
          <div id={style.cards}>
            <table>
              <tr>
                <th colspan="2"><h4>Employee {x.id}</h4></th>
              </tr>
              <tr>
                <td>Name:</td>
                <td>:{x.name}</td>
              </tr>
              <tr>
                <td>Salary:</td>
                <td>:{x.salary}</td>
              </tr>
              <tr>
                <td>Company:</td>
                <td>:{x.company}</td>
              </tr>
            </table>
          </div>
        )
      })}
    </div>
  )
}

```

```

                <td><button><Link to={`/Edit/${x.id}`}>Edit</Link></button></td>
                <td><button onClick={()=>{deleteData(x.id)}}>Delete</button></td>
            </tr>
        </table>
    </div>
    )
    }
    </div>
    )
}
export default Users

```

▼ Editusers.jsx

```

import axios from "axios"
import { useEffect , useState } from "react"
import { useParams, useNavigate } from "react-router-dom"
import style from "../home.module.css"

const Editusers=(()=>{

    let [name, setName] = useState("")
    let [salary, setSalary] = useState("")
    let [company, setCompany] = useState("")

    let navigate = useNavigate()

    let {index} = useParams()
    console.log(index)

    useEffect(()=>{
        axios.get(`http://localhost:3000/content/${index}`)
        .then((resp)=>{
            console.log(resp.data)
            setName(resp.data.name)
            setSalary(resp.data.salary)
            setCompany(resp.data.company)
        })
    },[])

    let nameData=(e)=>{
        e.preventDefault()
        setName(e.target.value)
    }
    let salaryData=(e)=>{
        e.preventDefault()
        setSalary(e.target.value)
    }
    let companyData=(e)=>{
        e.preventDefault()
        setCompany(e.target.value)
    }

    let formhandle=(e)=>{
        e.preventDefault()

        let payload={name,salary,company}
        axios.put(`http://localhost:3000/content/${index}`,payload)
        .then(()=>{
            console.log("Data has been updated")
        })
        .catch(()=>{
            console.log("Not updated")
        })
        navigate("/user")
    }

    return(
        <div id={style.myForm}>
            <table>
                <tr>
                    <th colspan="2"><h4>Update User Details</h4></th>
                </tr>
                <tr>
                    <td><label>Name</label></td>
                    <td><input type="text" value={name} onChange={nameData}/></td>
                </tr>
                <tr>

```



```

        <td><label>Salary</label></td>
        <td><input type="text" value={salary} onChange={salaryData}/></td>
      </tr>
      <tr>
        <td><label>Company</label></td>
        <td><input type="text" value={company} onChange={companyData}/></td>
      </tr>
      <tr>
        <th colspan="2"><button onClick={formhandle}>Submit</button></th>
      </tr>
    </table>
  </div>
)
}

export default Editusers

```

▼ db.json

```

{
  "content": [
    {
      "name": "asfn,",
      "salary": "kdna",
      "company": "kwbd",
      "id": 1
    },
    {
      "name": "db",
      "salary": "jff",
      "company": "sdcn",
      "id": 2
    },
    {
      "name": "db",
      "salary": "jff",
      "company": "sdcn",
      "id": 3
    },
    {
      "name": "db",
      "salary": "jff",
      "company": "sdcn",
      "id": 4
    },
    {
      "name": "db",
      "salary": "jff",
      "company": "sdcn",
      "id": 5
    }
  ]
}

```

42. What are the Life Cycle methods ?

▼ Ans

3 Life Cycle Stage are there :-

1. Mounting Stage

- In this stage Component is getting born, It is in Mounting Stage.
- Whenever we refresh Component gets reborn, Constructor method, render method and componentDidMount() method is getting loaded in this stage

2. Updating Stage

- In this stage, componentDidMount() method will get load. Whenever make changes in UI, componentDidMount() method and render method is getting loaded.

3. UnMounting Stage

- In this stage, `componentWillUnmount()` method will load.

▼ App.jsx

```
import { useState } from "react"
import Lifecycle from "../Components/Lifecycle"

const App=()=>{
  let [state, setState] = useState(true)
  return(
    <div>
      <button onClick={()=>{setState(true)}}>Mount Component</button>
      <button onClick={()=>{setState(false)}}>Mount Component</button>
      {state?<Lifecycle/>:null}
    </div>
  )
}
export default App
```

▼ Lifecycle.jsx

```
import React, {Component} from "react"

class Lifecycle extends Component
{
  constructor()
  {
    console.log("Constructor Method")
    super()

    this.state = {
      count:0
    }
  }

  incre=()=>{
    this.setState({
      count:this.state.count+1
    })
  }
  componentDidMount(){
    console.log("ComponentDidMount (it got born)")
  }
  componentDidUpdate(){
    console.log("Component got update")
  }
  render()
  {
    console.log("Render Method")
    return(
      <div>
        {this.state.count}
        <button onClick={this.incre}>Increment</button>
      </div>
    )
  }
  componentWillUnmount(){
    alert("Component Killed")
    console.log("Component Killed")
  }
}

export default Lifecycle
```

▼ Note : - Mention this in Interview

- Mention Function Based Component in Introduction before they ask. or otherwise
- “I have much more concentrated on Function Based Components”
- For creating new project —> “`npx create-react-app projectname`” —> “`cd projectname`” —> “`npm start`”
- For Single Page Application ---> “`npm install react-router-dom`”

- **For Axios** —> **“npm install axios”**
- **For BackEnd purpose** —> **“npm install json-server”**
- **Port opening** —> **“npx json-server Backend/db.json - -watch port 5000”**
- **For installing node_modules** —> **“npm i”**
- “Color picker eye dropper” —> add this Extension to your Browser for Picking the Right Color.
- Standard height of Navigation bar is 70 to 80 px.