REACT INTERVIEW QUESTIONS

1) npm alternative ⟹ "yarn" package manager made by facebook

2) React library or framework
⇒ React is JS library made by facebook on 29 may 2013

3) Can you send fun as prop
⇒ Yes, you can send fun() as prop using concept of lifting up state

4) Can you change Props in receiving Component
⇒ no, you can't change Props data in receiving component, you can only change it in parent component with help of State

5) Can you pass html as props data ⇒ yes, you can send HTML using this.props.children & if func() component then use this.children

6) Can you use Components state outside state & state public or private
⇒ • no, you can't use it outside
• State is public

7) Can you use "useState" in class Component? ⟹ no you can't use

8) What you use to hide & show in React & JQuery
• React : when you use React you use "state"

9) Can you send Super child as prop in Component ⇒ yes, you can send child from any Component to other using props even Super child (माता child के माता child)

10) when you hid & show your Component does a new life cycle start or not
⇒ new life cycle will start & new component will be made

11) Can you get or use props inside class constructor
⇒ yes you can use props in class constructor using this.props

12) Order of execution of constructor DidMount & render
⇒ 1) Constructor call
2) render call
3) ComponentDidMount & we call API here generally
• use of ComponentDidMount is API call.
• ComponentDidMount जाता पूरा HTML mount (execute) होगा उसके बाद it get call

13) Can we use setState in render()

→ no, don't use setState in render bcoz it gets call on each update :. setState will run in loop

∴ don't update state in render

14) Can you block or stop the execution of ComponentDidUpdate

→ yes, you can use a method called shouldComponentUpdate() & if this method returns false then DidUpdate will not execute

15) Can you call API in ComponentDidUpdate()

⇒ yes, when you want to do conditional call then you can do in DidUpdate

16) shouldComponentUpdate & ComponentDidUpdate, which executes first b/w these two

⇒ shouldComponentUpdate will get execute first & then on the value of this DidUpdate will run

17) will componentWillUnmount() work if the component in which it has been used a functional Component

⇒ no, you need a class Component in which this willUnmount() is called then only it will get execute, no matter what is the parent component but the component in which willUnmount is being used should be a class component

18) Can you use hooks in class component

⇒ No, hooks are only for functional component

19) where you will make fun in parent or child component if parent is calling or using child component in loop

⇒ in parent component make fun

20) which hook is used as substitute for pure Component in functional Component

⇒ useMemo()

21) can you use more then one Ref in component & can you use more then one Ref on a element

⇒ yes you can have more then one Ref in component

22) ComponentWillUnMount
⇒ component के DOM से हटने से just पहले call होता है

23) Can we use useState inside useEffect
⇒ yes, But use it conditionally

24) Can we use Controlled Component in Class Component
⇒ yes you can use in both functional & Class, both Controlled and UnControlled anywhere you can use.

25) if some Component is Controlled by both state also & JS also then what is that component called
⇒ it is called UnControlled Component, even if Ref is used then also it will be UnControlled Component

26) if we use <a> tag inplace of <Link> from React-router then will it work
⇒ yes it will work but it will Reload the Page
∴ don't use <a> tag, use <Link> imported from router

27) Why we use <Switch> from React-router-dom
⇒ Switch पहला match होता Route path ढूंढेगा जो मिलेगा उस पर direct कर देगा & नहीं मिला तो "*" वाले पर ले जाएगा।

28) diff b/w PUT & PATCH API call
⇒ Put = method is method using we send data inside req & used to ~~modify~~ replace existing data
PATCH = is only used to modify the existing data acc to condition

29) Can you send body inside get API request
⇒ no, you can only send a body (data) inside POST req

30) Can you send POST req & delete the data
⇒ it totally depends on the Backend developer how he has define the POST, DELETE request or API call

CH : 0 INTRODUCTION TO REACT

#1 INTRODUCTION

* REACT JS

- React is a Java Script Library (front End Library)
- many people compare React directly with frameworks like Vue JS or Angular JS & say it as framework but Remember React is a Library
- React is used to make single Page application & React is maintained & made by facebook
- Single Page Application : web application which only loads for one time & all the navigations happen without Reloading the web Page
- React is based on Components & using them again & again
- React doesn't have Router also
- Extension : React Developer tools Chrome extension install

VS Code Extensions :

1) Thunder Client → alternative of postman
2) EST React/Redux/GreaphQL
3) Auto Rename tag
4) Prettier

- Angular ; Vue JS also are used to make single page Application , but React is fast
- React is fast becoz it uses Virtual DOM . ∴ the item which you want to change, impact happens only their ∴ React is fast
- you can also do Mobile App Development with React - Native
- React was Released on 29 May 2013
- whenever you make a component in React, you can use it as HTML tag
- CRA : Create React App you need to set up CRA & you will get set up of React

* REACT WITHOUT INSTALING

Steps:
1) open folder in Vs Code
2) write HTML (!) and pase the 3 <script> tag links
   <script src " " > </script> React Link
   <script src " " > </script> React Dom Link
   <script src " " > </script> babel Link
3) write React Code in body
4) yhis is using CDN

* INSTALING REACT JS

- As React is a library you can install it using npm/npx which is installed when you install Node

Steps:

- npm install -g create-react-app // yhis insteall react globally
- you can also use react for a specif project by installing it in a specific folder using npx

Steps:
1) open terminal & open the folder where you want to do work
2) cd/React > npx create-react-app name_for_project
   npm start → // it will start your first React app (project)

* create-react-app Package

- yhis "create-react-app" Package set up the basic app of react
- eg:
  npx install create-react-app my-app
  cd my-app
  npm start

NOTE

- Components are independent and reusable bits of Code (like navbar) yhey serve the same purpose as JS fun, but work in isolation & return HTML. 2 types of Components Class & functions
- you can use these components any where in your website as HTML tags

#2 REACT FOLDER STRUCTURE

* FILE STRUCTURE

- when you do "create-react-app" some files get create & install the two most imp files are 1) public folder
  2) src folder

- **structure :**

file_name
- > node_modules
- > public → imp
- > src → imp
- > gitignore
- > package-lock.json
- > package.json

IMP

public
- > index.html

src
- > app.js
- > app.css
- > index.js

- package.json : it includes all packages you install. you can do "npm install" to install all packages if lost
- package-lock.json : it keeps the details of package.json file & it has meta data of all packages, like version of packages etc
- gitignore : it tells what should you (commit) push while uploading project on git/github
- node_modules : this are the packages & libraries we have installed along with their dependencies. if you delete this file you can get it back by "npm install" & all node_modules will install again from package.json

* Public folder

- index.html : this "public" folder contains index.html file which has :

```
<div id = "root" > </div>  → v. imp
```

- don't do anything in this file, all the react at the end comes in this "root" div
- at the end, सारी Rendering index.html में ही होगी
- with help of JS में id = "root" वाला div populate होगा

* src folder

- We will put our components in "src" folder & इन components में हम "props" को भेजेंगे & components की "state" होगी
- props : components के props मतलब details , eg HTML form में "details" भरना is :. details are props
- State : form की जो heading होगी , तो state बोल सकते है
- eg : HTML form ⟶ is one component
  details filled in form ⟶ are props
  heading of form ⟶ is one state
- App.js : this App.js is one component , & it is seen when you do "npm start"
- App.test.js : it contains unit test (contains) cases . & these are run by developers for testing
- index.js : Entry point of react .
  - <App/> yhis says that App.js वाले component को render कर दो & उनका/ उसके सबके content को doc.getElementById ('root') पर लेके जाओ
  - import 'App from 'App.js'; index.js में app component को import किया है
  - you can make other components & put it in app.js component also
  - you can use components inside components
  - all the components should be imported in index.js of src
  - put all components in src folder
  - Components are of 2 types
    1) class Based
    2) function Based
  - create-react-app uses function based component

* Creating first Component

- when you do "npm start", the website which is visible on your 3000 port is of "app.js" component which is imported in index.js of src folder
- first file which loads is index.js & app.js file is imported in index.js & is used in index.js

Open src → create a file : name : user.js

- this "user.js" is our new component, write the following code in "user.js"
- eg:

```
function useres ()          //this name should be equal to the file (component) name
{
    return (
        <div className = "App">
            <h1> hello World </h1>
        </div>
    );
}
export default users;
```

- now import this file "user.js" in "index.js".

Code:

```
import Users from './user';   //import the component in index.js
ReactDom.render(
    <React.StrictMode>
        <users />
    </React.StrictMode>
//other code
```

- now when you do "npm start" you can see Hello world

NOTE: npm का alternative है "yarn" package manager made by Facebook

CH:1 COMPONENTS & JSX

#1 COMPONENTS

* Components

- **Components:** is A peice of code that can be reused, such as functions, but components are more powerful than functions.
- eg: Header, footer, sidebar, navbar
- eg: for a home bedroom, hall, kitchen all are components
- Components in React can be Reused on same page as well as diff page
- **React Components Types**

  1) functional Components
  2) Class based Components

① functional Component

- you have to make a function of same name as of the file and export the functional component so that it can be imported into diff componenters
- syntax:

```
function file_name ()
{
    return (
        <>
            // JSX body
        </>
    );
}
export default User ——→ // export the component
```

- React functional Component has to be returned & you can only return one tag :. use <> </>

- eg:

src
> Usex.js

```
function User()
{
  return (
      <h1> User Component </h1>
  );
}
export default User
```

src
> App.js

```
import User from './User'
function App()
{
  return (
    <div className="App">
      <h1>          </h1>
      <User/> =>//use the component
      <User> </User>
    </div>
  );
};
export App ; or
export default App ;
```

- you have created "User" component & used the User component in "App" component
- you can use the component as HTML tag like "<User/>" or "<User> </User>"
- props: you can use props to send data b/w components or b/w diff components
- you can create 2 components in same file & use them without Exporting

NOTE

bootstrap!

- you can add bootstrap components (CSS & JS) in index.html of public folder
- put <script> tag of bootstrap after </body> & <link> tag of CSS in <head> tag

(2) Class Component

- you can make class component in new file as well as some other component file also like function component
- Syntax:

```
import React, {Component} from 'react' // import react
export default class file_name extends Component // you can also write export at the end after class
{
    render ()
    {
        return (
            <>
                // JSX
            </>
        );
    };
}
```

- eg: src > User.js

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

class Users extends Component
{
    render()
    {
        return (
            <>
                <h1> Class Component </h1>
            </>
        );
    };
};
Export default class Users
```

- now you can use this component like functional component
- you can make a component inside a component & use this component into only the component which is within that ~~file~~ file

#2 JSX (JavaScript XML / JS Extention)

* JSX

- JSX एक HTML & JS का mixture है with .js extention
- HTML ही है JSX, but you can write JS b/w " { } " curly Brackets
- JSX file का extention ".js" ही होता है
- className : in place of HTML "class" you have to use "className" becoz class is a reserve key word in JS
- htmlFor : A in place of HTML "for" which is used in <lable> tag as an attribute is replaced by "htmlFor"
- tabIndex : in place of HTML tab index you have to replace it with "tabIndex"
- in React when you are returning JSX, you have to only return one element tag
- you can import css in JSX or .js file by just importing it : import './file-name.css';
- the HTML tag which are having only opening tag have to be written as <img />
  eg: < />
- "Babel" compiles JSX down to react.createElement() calls
- JSX means JS के ऊपर HTML
- you can also use React without JSX, with pure JS also but it will become complex

NOTE

Working: index.html में "div id=root". index.js में लिखा है जहा पर id=root है वहा पर ये उपर वाले components डाला दो (app component)

- :: app component (app.js) index.html में जायेगी & दिखेगी port 3000 पर (localhost:3000)
- you can use React any where (with Angular etc) becoz React is a library

#3 Click Event & function

- when you want to call a function inside JSX (functional component), then you have to call the function without the brackets "()"
- eg: Src
  > App.js

```
function App()
{
    function apple()
    {
        alert("function called");
    }
    return (
        <div className="App">
            <h1> hello </h1>
            //← <button onClick={apple()}> click me </button>
            <button onClick={apple}> click me </button>
        </div>
    );
};
export default App
```

don't give "()" bracket

wrong it won't work

- if you write <button onClick={alert("hello")}> click </button> then it will get automatically call, to avoid this write arrow function
- eg

```
<button onClick={()=>alert("hello")}> click </button>
```

OR

you can call fun using arrow fun

```
<button onClick={()=>apple()}> click </button>
```

CH:2 PROPS & STATE

#1 PROPS

- you can pass data from one Component to another using props
- **props** : props is short form of properties.
  - when you make component you can send data into the component using props
  - you can pass any type of data using "props" from one component to another & multiple data also
  - props is a obj which includes all data that has passed
- below we will pass data from App.js to Student.js using props

- Src
  > App.js

```
import Student from './student';
function App()
{
  return (
    <div className="app">
      <Student name="Ennu" />
      <Student name="Vishu"
               other={{
                 add: 'delhi',
                 mob: '000'
               }} />
    </div>
  );
}
export default App;
```

We have use Student component & passed parameters or data into it

data variables

data values

Src
> student.js

```
function Student(props)
{
  console.log(props)
  return (
    <div style={{backgrdCol:blue}}>
      <h1> hello : {props.name}</h1>
      <h2> Email : {props.mail} </h2>
      <h4> add : {props.other.add}
      </h4>
    </div>
  );
}
export default student
```

you can use any variable name

Note:
all the data like "name" other is inside "props" object

data from App to student

* updating "props" onClick button

- for this you need to use "state" also beoz for changing prop data you need to change the "state" of component

App.js

```
import React,{useState} from 'react'
import Student from './student'
function App()
{
  const [name, setName] = useState("Vishu")
  return (
    <div>
      <h1> Props in react </h1>
      <Student name={namey}/>
      <button onClick={()=>{setName("Euuu")}}> Update.
      </button>
    </div>
  );
};
export default App
```

Student.js

```
function Student (props)
{
    console.log(props)
    return (
        <div>
            <h1> Hello : {props.name} </h1>
        </div>
    );
};
export default Student
```

* Prop Type

- you can set the type (data type) of the variables of props, the one component is sending to other
- you have to write all these "prop types" & "default Props" in the component in which you are using these Props data or Receiving data
- in our Case it is Student.js
- student.js

```
import propTypes from 'prop-types'
function Student(props)
{
    return (
        <div>
            <h1> Hello : {prop.name} </h1>
        </div>
    );
}
export default Student

Student.propTypes = {
    name : propTypes.string,
    other : propTypes.object.isRequired
}
```

if you set some number value of name then you will get error

// isRequired करने से "other" कि value set होनी ही चाहिए, you can't keep it undefined otherwise you will get error.

- by default set को हो tho error नहीं आएगा

* default Props

- you can set some default values of the Props data & if the props data is not set then, these default values of the props will be consider
- eg: Student.js

```
import propTyper from 'prop-types
function student (props)
{

        // body

}

Student.defaultprops = {
        name : 'vishwajeet Emm',
        other : {
            add : 'wadi',
            mob : '8600'.
        }
}
```

- now if you don't pass the data in props then default values for the data will be used

#2 PROPS IN CLASS COMPONENT

*Props :

- Props are used to send data (of any data type) from one component to another
- you can send any no. of props & even "link" also
- all the data can be accessed in form of props properties because props is a object.
- we will be sending data from App to students
- you can send multiple "data" for same prop name

Src
>App.js

```
import student from './student'
function App()
{
  return
  {
    <div>
      <h1>          </h1>
      <Student name="Anil"/>
      <Student name="Euuu" email=
              "Euuu@gmail" />
    </div>
  };
}
export default App
```

Src
>student.js

```
import React from 'react'
class Student extends React.Component
{
  render()
  {
    return(
    <div >
      <h1> hello {this.props.name} </h1>
      <h3> {this.props.email} </h3>
    </div>
    );
  }
    //Class Component
```

NOTE:
you cant change props in receiving components (student)
you can only change props in parent or sending component like App
this applys for both class & functional Component

- you can update props data using state (on click), but this has to be done in app component or parent component only

App.js

```
import Student from './student'
import React from 'react'
class App extends React.Component
{
    constructor()
    {
        super();
        this.state = {
            name = "Ennn"
        }
    }
    render()
    {
        return (
            <div className="App" >
                <h1> Prope1 </h1>
                <Student onClick={()=> this.setState({ name: "Visha" })}>
                    Update name
                <Student name = {this.state.name} > </student>
                <button onClick={()=> this.setState({name: "vi"})} >
                    update name </button>
            </div>
        );
    }
}
export default App
```

* now on button clicke name will be change

#3 State

* need for state

- you can't change the "value" of variables inside components
- for updating variables you need to change the state of the component

* State

- using state you can update "values" of variables
- State belongs to an component
- State मतलब अभी किस state मे है, और मे अलग state मे चला न जाना।
- variables को component मे update करने के लिए component state change होनी चाहिए & this is called Rerendering
- Rerendering : मतलब component फिर से execute होगा
- you need to define state & import it for using it

Code:

```
import React, {useState} from 'react'

function App()
{
//destructuring const [text, setText] = useState("your text default");

return (
        // function component
);
}
```

useState is a hook

text एक state variable है
और default value "your text" है

setText
is a fun
& it is used
to update text variable

this is dy default value

- here in plate of "setText" you can also use "updateText"
- "setText" is used to update "text" variable

- Syntax

```
import React, {useState} from 'react'
function Component_name ()
{
    const [data, setdata ] = usetState ("initial value")
    return (
        // your function component Code
    )
}
```

- इस "data" या "text" जहा भी use कर रहे है & आगे उसे update करना है तो तो new value इस setdata/settext को देगे & सारी जगह "text/data" की value update होगी
- hooks जो है, इस बीना class बनाऐ class के fun() & features use करने मे (सहायता) सहाएक होता है
- you can't update text as

```
text = "some value"
```

- you can update it by

```
settext = "new value"
```

- eg:

```
import React, {useState} from 'react'
function App()
{
    const [data, setdata]= useState("Guru");
    function update()
    {
        setdata = "vishu";
    }
    return (
        <div>
            <h1>{data} </h1>
            <button onClick = {update} > Update click </button>
        </div>
    );
}
export default App
```

#14 State IN class Component

- you can use state in class component also
- eg:

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

class App extends Component
{
    constructor()  //you have to define constructor for using this
    {
        super();  // this is JS functionality, & you can use this
                  & parent constructor also get call
                  for that you need super()
        this.state = {
            data: "Sumit";
        }
    }
    apple()
    {
        this.setState({data: "vishu"})
    }
    render()
    {
        return (
            <div className="App">
                <h1> {this.state.data} </h1>
                <button onClick={()=> this.apple()}>
                    Update Data click </button>
            </div>
        );
    }
}
```

if you don't write arrow fun then without click the fun apple will get execute & pgm will have bugs

- if you don't use "super()" you will get error

CH: 9 onClick & onChange Event

#1) Get Input box Value

* onChange Event

- using this "onChange" you can get the data which is enter in a input feild by user
- Code :

```
import React, {useState} from 'react'
function App()
{
    const [data, setData] = useState(null)
    function getData(event)
    {
        console.warn(event.target.value)
        setData(event.target.value) // "event.target.value" will have the entered text
    }
    return(
        <div className="App">
            <h1> {data} </h1>
            <input type="text" onChange={getData} />
        </div>
    );
}
```

this is event obj

whenever you click button or enter text getData fun() will get call & you will get data Entered in input field

- you can also write one more state & print data on button click

Code:

```
const [print, setPrint] = useState(false)
fun getData(event)
{
    setPrint(false)
    setData(event.target.value)
}
return(
    <div>
        { print? <h1> {data} </h1> : null }
        <button onClick={()=>setPrint(true)}> Print </button>
    </div>
)
```

eg:

```
import React, {useState} from 'react'
export default function Textform ()
{
    const handleUpClick = () =>{
        console.log ("uperCase was click");
        setText ("you have clicked on handle Up click");
    }   setText(text.toUpperCase()); ---> // this will make text as upper case

    const handleonChange = (event) =>{
        console.log ("on change");
        setText(event.target.value);
    }                                  -> it has the value of <textarea> tag

    const [text, setText] = useState ('Enter text');

    return (
        <div>
            <h1> lower To Upper </h1>
            <textarea value={text} onChange={handleonChange}>
                                    </textarea>
            <button onClick={handleUpClick}> Convert </button>
        </div>
    )

};
```

when you listen for onChange event you will get event obj

- this is a page in which when you will enter text it will become upper case

#2 HIDE and show/toggle Element

* hide & show

- you can use two buttons to hide & show any element using "onClick event"

Code:

```
import React, {useState} from 'react'
function App()
{
    const [status, setStatus] = useState(true) // this is called defining state

    return (
        <div className = "App">
            {
                status? <h1> Hello </h1> : null
            }
            <button onClick = {()=> setStatus(false)}> Hide </button>
            <button onClick = {()=> setStatus(true)}> Show </button>
        </div>
    );
}
```

* toggle

- you can do it on sigle button also

Code:

जैसे ही click करोंगे जो present status है उसका opposite होगा

```
<button onClick = {()=> setStatus(!status)}> Toggle </button>
```

#3 FORM

*getting form data

- you can get the form data Entered by user when clicked Submit button

Code :

```
import React, {useState} from 'react'
function App ()
{
    const [name, setName] = useState("")
    const [tnc, setTnc] = useState(false);
    const [interest, setInterest] = useState("");
    function getData (e)
    {
        e.preventDefault() // by this when you click on Submit page will not reload & will not redirected

        console.log(name, tnc, interest); // you will get the entered data output in console
    }
    return (
        <div>
            <h1> form handle </h1>
            <form onSubmit={getData}>
                <input type=text onChange={(e) => setName(e.target.value)}
                <select onChange={(e)=>setInterest(e.target.value)}>
                    <option> Select option
                    <option> marvel
                </select>
                <input type="checkbox" onChange={(e)=>
                    setIntrest(e.target.checked)} />
                <button type="submit"> Submit </button>
            </form>
        </div>
    );
}
```

#4 IF - ELSE

* If Else

- if you want to use if-else condition you can use it.

Code:

```
import {useState} from 'react'
function profile()
{
    const [loggIn, states] = useState(false)
    if(loggIn)
    {
        return(
            <div>
                <h1> Welcome User </h1>
            </div>
        );
    }
    else
    {
        return(
            <div>
                <h1> Welcome </h1>
                <button onClick={()=>{states:true}>
                    logIn </button>
            </div>
        );
    }
}
export default profile;
```

- you can also use single line syntax as

```
return(
    <div>
        {
            loggIn ? <h1> Welcome User </h1> : <h1> Guest </h1>
        }
    </div>
)
```

#5 PASS FUNCTION AS PROPS

* passing function as props

- you can send fun as props parameter from one component to another
- we will send fun from App.js to User.js

App.js

```
import user from './user'
function App()
{
    function getData()
    {
        alert("hello from APP")
    }
    return(
      <div>
        <User data={getData} />
      </div>
    );
}
export default App
```

User.js

```
import React from 'React'
function User(props)
{
    return(
      <div>
        <h1> User Component </h1>
        <button onClick={
            ()=>props.data()}>
            click to call </button>
        <button onClick={
            props.data}>
            another way call </button>
      </div>
    );
}
```

- here 'getData()' fun from parent (App) is going inside child component (User)