1) What is React and Why do we use React? (Complete Interview Answer):

React ek JavaScript library hai jiska use User Interface build karne ke liye kiya jata hai, mainly Single Page Applications banane ke liye. React ko Facebook ne develop kiya tha.

React ka main benefit ye hai ki ye component-based architecture follow karta hai. Matlab UI ko small, reusable parts me divide kar diya jata hai jinko components kehte hain, jaise header, footer, cards, etc. Isse code reuse hota hai aur application ko maintain karna easy ho jata hai.

React Virtual DOM ka use karta hai. Virtual DOM, Real DOM ka lightweight copy hota hai, jaise ek blueprint ya naksha. Jab UI me koi change hota hai, React pehle Virtual DOM update karta hai, phir sirf jo necessary changes hote hain unhe Real DOM me apply karta hai. Isse performance fast ho jati hai kyunki poora page reload nahi hota.

2) What is Single Page Application (SPA)?

Single Page Application wo application hoti hai jisme sirf ek HTML page load hota hai. Uske baad jo bhi content change hota hai wo JavaScript ke through dynamically load hota hai, bina page reload ke. Isi ko Single Page Application kehte hain

3) Difference between Library and Framework

Library aur Framework dono code reuse ke liye use hote hain, lekin main difference control ka hota hai.

Library me control developer ke paas hota hai. Hum decide karte hain kab aur kaunsa code call karna hai. React ek library hai, isliye hum React ko apne code ke andar use karte hain jab hume zarurat hoti hai.

Framework me control framework ke paas hota hai. Framework application ka flow decide karta hai aur developer ko uske rules follow karne padte hain. Is concept ko Inversion of Control kehte hain.

Example ke liye, React ek library hai jabki Angular ek framework hai.

4)What is JSX?

JSX ka full form JavaScript XML hai. JSX ek syntax extension hai jo React me use hota hai, jisme hum JavaScript ke andar HTML jaisa code likh sakte hain.

JSX ka main benefit ye hai ki ye UI code ko readable aur easy to understand bana deta hai, kyunki UI aur logic ek hi jagah likhe ja sakte hain. Isse code maintain karna bhi easy ho jata hai.

Browser JSX ko directly samajh nahi sakta, isliye React JSX ko directly render nahi karta. Babel ka use karke JSX ko normal JavaScript me convert kiya jata hai.

JSX React me compulsory nahi hai, lekin zyada tar developers JSX use karte hain kyunki ye code ko simple aur clean banata hai. Agar hum JSX use nahi karte, to hum plain JavaScript me `React.createElement()` function ka use karke UI create kar sakte hain.

Example ke liye, JSX me `<h1>Hello</h1>` likhte hain, lekin internally ye `React.createElement("h1", null, "Hello")` me convert ho jata hai.

5) What is Babel?

Babel ek JavaScript compiler hai. Ye JSX aur modern JavaScript (ES6+) code ko normal JavaScript me convert karta hai, jise browser easily samajh aur execute kar sake. React me Babel JSX ko `React.createElement()` me convert karta hai

6) What are components in React?

React me component UI ka ek small, independent aur reusable part hota hai. Component ek JavaScript function ya class hota hai jo JSX return karta hai.

React component-based architecture follow karta hai, jisme poori UI ko small reusable components me divide kar diya jata hai, jaise header, footer, navbar, card, etc.

Component ka main benefit ye hai ki hum ek baar component bana kar usse multiple jagah use kar sakte hain, jaise Home page aur About page dono me same header component use karna. Isse code reuse hota hai aur application ko maintain karna easy ho jata hai.

React me mainly do types ke components hote hain: Functional Components aur Class Components, lekin aajkal mostly Functional Components use kiye jate hain.

7) What is the difference between functional and class components in React?

React me mainly do types ke components hote hain: Functional Components aur Class Components.

Functional Component ek normal JavaScript function hota hai jo JSX return karta hai. Ye zyada easy to understand hota hai aur aajkal React me yahi recommended hai. Functional components hooks ka use karke state aur lifecycle features handle karte hain, jaise `useState` aur `useEffect`.

Class Component JavaScript ES6 class hota hai jo `React.Component` ko extend karta hai. Isme JSX return karne ke liye `render()` method use karna padta hai. Class components me state aur props ko access karne ke liye `this` keyword ka use hota hai aur lifecycle methods jaise `componentDidMount`, `componentDidUpdate` hote hain.

Aajkal functional components zyada use hote hain kyunki unka syntax simple hota hai, code readable hota hai aur hooks ki wajah se state aur lifecycle handle karna easy ho gaya hai.

Syntax Examples
Functional Component

```
function Card() {

 return (

  <div>Card Component</div>

 );

}
```

Class Component

```
class Card extends React.Component {

 render() {

  return (

   <div>Card Component</div>

  );

 }}
```

8) What is a prop in React

Props ka full form properties hota hai. Props React me data pass karne ka mechanism hai, jiske through parent component apna data child component ko send karta hai.

Props basically ek object hota hai aur child component me props read-only hote hain, matlab child component parent se aaye data ko change nahi kar sakta.

Props ka main use ye hai ki hum ek component ka data dusre component me reuse kar sakte hain. Jaise agar parent component me API se data fetch kiya gaya hai, to us data ko props ke through multiple child components me pass kiya ja sakta hai. Isse code reuse hota hai aur application maintainable banti hai.

Ye components ko dynamic banate hain. Ek hi component ko hum alag-alag data (props) dekar alag-alag tarike se render kar sakte hai

Props functional components aur class components dono me use kiye ja sakte hain, lekin functional components me props use karna zyada easy hota hai.

Parent Component
```
function Parent() {

  return (

    <Child name="Aman" age={20} />

  );}
```

Child Component (Normal way)
```
function Child(props) {

  return (

    <h1>{props.name} - {props.age}</h1>

  );}
```

Child Component (Destructuring way)
```
function Child({ name, age }) {

  return (

    <h1>{name} - {age}</h1>

  );}
```

9) What is state in react

State React ka ek built-in object hota hai jo component ke data ko store karta hai. Jab state change hoti hai, React automatically component ko re-render karta hai jisse UI update ho jati hai.

State component ka private data hota hai, matlab sirf wahi component apni state ko access aur update kar sakta hai. State ko directly update nahi kar sakte, uske liye hamesha setter function ka use karna padta hai.

Functional components me state handle karne ke liye `useState` hook ka use hota hai. `useState` ek array return karta hai jisme pehla value current state hota hai aur dusra ek function hota hai jisse state update ki jati hai.

Example ke liye, counter app me count value state hoti hai. Jab button click hota hai aur state update hoti hai, to UI automatically change ho jati hai.

Syntax Example

const [count, setCount] = useState(0);

setCount(count + 1);

10) What is the difference between Props and State

Props aur State dono React me data handle karne ke liye use hote hain, lekin dono ka purpose alag hota hai.

Props parent component se child component me data pass karne ke liye use hote hain. Props read-only hote hain, matlab child component props ko change nahi kar sakta.

State component ke andar manage hoti hai aur component ka private data hoti hai. State change ho sakti hai aur jab state update hoti hai to React component ko re-render karta hai.

State directly dusre component me access nahi hoti, lekin parent component apni state ko props ke through child component me pass kar sakta hai

Props	State
Parent se milti hai	Component ke andar banti hai
Read-only hoti hai	Change ho sakti hai
Child change nahi kar sakta	Component khud change karta hai
Data pass karne ke liye	Data manage karne ke liye
Reusable components banane me help	Dynamic UI banane me help

11) What is UseEffect

useEffect React ka ek hook hai jo side effects handle karne ke liye use hota hai. Side effects ka matlab hota hai aise kaam jo component ke render hone ke baad perform hote hain, jaise API call, data fetch karna, timer set karna ya event listener add karna.

useEffect ka main use functional components me lifecycle behavior handle karna hota hai. Pehle class components me componentDidMount, componentDidUpdate aur componentWillUnmount use hote the, useEffect in sabka replacement hai.

useEffect do arguments leta hai: pehla ek function hota hai jisme side-effect code hota hai, aur dusra dependency array hota hai jo decide karta hai effect kab chalega.

Agar dependency array empty ho [], to effect sirf first render par chalta hai. Agar dependencies di ho, to unke change hone par effect chalta hai. Aur agar dependency array hi na ho, to effect har render par chalta hai.

useEffect ek cleanup function bhi return kar sakta hai jo component unmount hone par ya effect dubara run hone se pehle execute hota hai. Iska use timers, event listeners clean karne ke liye hota hai taaki memory leak na ho.

```
useEffect(() => {
  // side effect logic
  return () => {
    // cleanup (optional)
  }}, [dependencies]);
```

Example

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

function Counter() {

  const [count, setCount] = useState(0);


  useEffect(() => {

    console.log("Count changed:", count);

  }, [count]);


  return (

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

      Count: {count}

    </button>

  );

}
```

12) What Is life cycle behavior

Lifecycle behaviour ka matlab hota hai component ke life ke different phases: kab component create hota hai, kab update hota hai aur kab remove hota hai.

React me lifecycle ke 3 phases hote hain: Mounting (first render), Updating (state ya props change), aur Unmounting (component remove hona).

Class Component	`useEffect`

Class Component	`useEffect`
componentDidMount	`useEffect(() => {}, [])`
componentDidUpdate	`useEffect(() => {}, [value])`
componentWillUnmount	Cleanup function

13) What is Cleanup Function

Cleanup function useEffect ke andar return hota hai aur component unmount hone par ya effect re-run hone se pehle chalta hai. Iska use timers, subscriptions aur event listeners ko clean karne ke liye hota hai.

```
useEffect(() => {
  const timer = setInterval(() => {
    console.log("Running");
  }, 1000);

  return () => {
    clearInterval(timer);
  };
}, []);
```

14) What is componentDidMount componentDidUpdate componentWillUnmount

`componentDidMount` **class component ka lifecycle method hai jo component ke first time render hone ke baad run hota hai. Iska use API call, data fetch ya event listener add karne ke liye hota hai. Functional component me iska equivalent** `useEffect` **hota hai jab dependency array empty ho, jaise** `useEffect(() => {}, []).`

`componentDidUpdate` **class component ka lifecycle method hai jo tab run hota hai jab component update hota hai, yaani jab state ya props change hoti hai—jaise counter increment hone par. Functional component me ye** `useEffect` **ke saath dependency array dene par hota hai, jaise** `useEffect(() => {}, [count]).`

`componentWillUnmount` **class component ka lifecycle method hai jo tab run hota hai jab component screen se remove hone wala hota hai. Iska use cleanup ke liye hota hai—jaise timer clear karna ya event listener remove karna. Functional component me iska equivalent** `useEffect` **ka cleanup function hota hai.**

15)What are controlled and uncontrolled components in React?

React mein forms handle karne ke do tarike hote hain:
1) Controlled Components
2) Uncontrolled Components

Controlled component wo hota hai jisme form ka data React ke state ke through control hota hai. Input ki value state se aati hai aur har change par `onChange` ke through state update hoti hai. Isliye React hi data ka single source of truth hota hai.

Example ke liye, jab hum input ki value ko state se bind kar dete hain aur typing par state update hoti hai, to use controlled component kehte hain. Isme real-time validation aur better control possible hota hai.

Uncontrolled component wo hota hai jisme form ka data React state ke through control nahi hota, balki direct DOM handle karta hai. Yahan hum `useRef` ka use karke input ki value access karte hain. Isme koi state nahi hoti aur value submit ke time direct DOM se milti hai.

Controlled components zyada preferred hote hain kyunki unme validation, debugging aur form handling easy hoti hai.

Controlled:

```
function Form() {

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



  return (

    <input

      type="text"

      value={name}

      onChange={(e) => setName(e.target.value)}

    />

  );

}
```

Uncontrolled

```
function Form() {

  const inputRef = useRef();



  const handleSubmit = () => {

    console.log(inputRef.current.value);

  };



  return (

    <>

      <input type="text" ref={inputRef} />

      <button onClick={handleSubmit}>Submit</button>

    </>

  );}
```

16) What are event handlers in React, and how do they work?

Event handlers React me functions hote hain jo user ke actions par run hote hain, jaise button click, input change, form submit, mouse over, ya key press. Simple words me, user jo action karta hai usko handle karne ke liye event handlers use hote hain.

React me events camelCase me likhe jate hain, jaise `onClick`, `onChange`, `onSubmit`. React me event handler ko function reference diya jata hai, function call nahi kiya jata, kyunki agar function call kar denge to component render hote hi function execute ho jayega.

Event handlers ko event object bhi milta hai jisse hum input ki value ya event details access kar sakte hain.

Basic Example (Correct)

```
function App() {

  const handleClick = () => {

    console.log("Button clicked");

  };

  return (

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

  );

}
```

Event Handler pass karne ke 3 IMPORTANT ways

1)Inline Event Handler

-> Small logic ke liye OK, heavy logic ke liye avoid

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

2)Event Handler with Parameters (Interview Favourite)

```
<button onClick={() => handleDelete(id)}>Delete</button>

function handleDelete(id) {

  console.log(id);}
```

->Jab hume id, state, ya custom value **pass karni ho**

3)Event Object use karke (`e`)

```
<input
  type="text"
  onChange={(e) => console.log(e.target.value)}
/>
```

->Input value, checkbox, keyboard events ke liye

17) react key prop

`key` React me ek special prop hota hai jo list ke elements ko uniquely identify karne ke liye use hota hai. Jab hum lists render karte hain, React `key` ki help se ye samajhta hai kaunsa item add hua, remove hua ya update hua hai.

`key` React ke Virtual DOM comparison me help karta hai, jisse unnecessary re-rendering avoid hoti hai aur performance better hoti hai.

Best practice ye hai ki `key` ke liye hamesha ek unique aur stable value use karein, jaise item ki `id`. Index ko key banana avoid karna chahiye kyunki list change hone par bugs aa sakte hain.

```
const users = [
  { id: 1, name: "Aman" },
  { id: 2, name: "Ravi" }
];
function App() {
  return (
    <ul>
      {users.map(user => (
        <li key={user.id}>{user.name}</li>
      ))}
 </ul>
);}
```

18) Why should we avoid using index as key in React

Index ko key ke roop me use karna avoid karna chahiye kyunki jab list me items add, remove ya reorder hote hain, to index change ho jata hai.

Index change hone ki wajah se React items ko sahi tarah identify nahi kar pata, jisse wrong re-rendering hoti hai aur UI bugs aa sakte hain, jaise wrong item update ho jana.

Isliye hamesha ek unique aur stable value, jaise item ki `id`, ko key ke roop me use karna chahiye