### 1. **What is the virtual DOM?**

Answer:  
The Virtual DOM is a lightweight JavaScript object that is a copy of the real DOM. When state changes in a component, a new Virtual DOM tree is created. React compares it with the previous tree using a process called "diffing", and applies the minimal necessary changes to the real DOM.

### 2. **What are the differences between functional and class components?**

Answer:

| Feature | Functional Component | Class Component |
| --- | --- | --- |
| Syntax | Functions | Classes |
| State | useState hook | this.state |
| Lifecycle methods | useEffect | componentDidMount, etc. |
| Performance | Better (lighter) | Slightly heavier |

### 3. **What are hooks in React?**

Answer:  
Hooks are special functions that let you “hook into” React state and lifecycle features in functional components. Common hooks include:

* useState – to manage state
* useEffect – to manage side effects
* useContext – to access context
* useRef – to access DOM or store mutable values

### 4. **What is the useEffect hook?**

Answer:  
useEffect is used to perform side effects (like data fetching, subscriptions, or DOM updates) in functional components.

useEffect(() => {

fetchData();

}, [dependency]); // Runs when dependency changes

### 5. **What is the difference between props and state?**

Answer:

| Props | State |
| --- | --- |
| Read-only | Mutable (can be changed) |
| Passed from parent | Managed within the component |
| Cannot change itself | Can change using setState or useState |

### 6. **How does React handle component re-rendering?**

Answer:  
React re-renders a component whenever its state or props change. It uses the Virtual DOM to compute the minimal set of updates and applies them efficiently to the actual DOM.

### 7. **What is the key prop and why is it important?**

Answer:  
The key prop helps React identify which items have changed, are added, or are removed in a list. It improves rendering performance and avoids unwanted re-renders.

{items.map(item => <li key={item.id}>{item.name}</li>)}

### 8. **What is context in React?**

Answer:  
Context allows passing data through the component tree without passing props manually at every level.

const MyContext = React.createContext();

<MyContext.Provider value={someValue}>

<Child />

</MyContext.Provider>

Use useContext(MyContext) in a child component to access the value.

### 9. **What are controlled and uncontrolled components?**

Answer:

* Controlled Component: Input value is controlled by React state.
* Uncontrolled Component: Input value is controlled by the DOM (via refs).

// Controlled

<input value={inputValue} onChange={(e) => setInputValue(e.target.value)} />

// Uncontrolled

<input ref={inputRef} />

### 10. **How do you optimize performance in React?**

Answer:

* Use React.memo() to avoid re-rendering
* Use useCallback and useMemo to memoize functions/values
* Split code using lazy loading (React.lazy + Suspense)
* Avoid unnecessary state updates

### 11. **What is the difference between useEffect and useLayoutEffect?**

Answer:

* useEffect runs after the paint.
* useLayoutEffect runs before the paint — useful when you need DOM measurements or layout changes.

### 12. **What is React.memo and how does it work?**

Answer:  
React.memo is a higher-order component that memoizes a component — it prevents re-renders if props haven't changed.

const MyComponent = React.memo(function MyComponent(props) {

return <div>{props.name}</div>;

});

### 13. **How do you handle forms in React?**

Answer:  
Use controlled components with useState to manage form inputs and a handleSubmit function for form submission.

### 14. **What are Higher Order Components (HOC)?**

Answer:  
A HOC is a function that takes a component and returns a new component with additional functionality.

const withAuth = (Component) => (props) => {

return isLoggedIn ? <Component {...props} /> : <Redirect to="/login" />;

};

### 15. **What is Redux and when would you use it?**

Answer:  
Redux is a state management library used when:

* You need centralized state
* Many components need access to the same data
* You want predictable state updates using actions and reducers