

React JS Interview Questions (2 Years Experience)

1. What are the main features of React?

- Virtual DOM
- JSX (JavaScript XML)
- Component-based architecture
- One-way data binding
- Performance optimization
- Unidirectional data flow

2. Explain the Virtual DOM and how React uses it.

The Virtual DOM is a lightweight JavaScript object that React uses to improve performance. React keeps a virtual representation of the UI in memory and compares it with the real DOM using a process called 'diffing'. Only the changed elements are updated in the real DOM, minimizing performance hits.

3. What are React Hooks? Name a few commonly used ones.

Hooks are functions that let you use state and other React features without writing a class. Common hooks include:

- useState
- useEffect
- useContext
- useRef
- useMemo
- useCallback

4. Difference between useEffect and useLayoutEffect?

- useEffect runs asynchronously after paint.
- useLayoutEffect runs synchronously after all DOM mutations but before paint. It's useful when DOM measurement is required.

5. What is prop drilling and how do you avoid it?

Prop drilling is the process of passing data from a parent to a deeply nested child component through intermediate components. It can be avoided using:

- React Context API
- Redux or other state management libraries

6. How do you optimize performance in a React app?

- Use React.memo for functional components
- Use useMemo and useCallback hooks



- Lazy load components with `React.lazy`
- Code splitting with `dynamic import()`
- Avoid unnecessary re-renders

7. What is the difference between controlled and uncontrolled components?

- Controlled components are managed by React state.
- Uncontrolled components manage their own state using refs.

8. What is reconciliation in React?

Reconciliation is the process through which React updates the DOM. It compares the previous and current Virtual DOM and only applies necessary changes.

9. Explain the Context API in React.

The Context API is used to share values between components without having to pass props manually at every level. It consists of `React.createContext()`, `Provider`, and `useContext()`.

10. What are keys in React and why are they important?

Keys help React identify which items have changed, are added, or are removed in a list. They should be unique and stable for optimal performance and correct component behavior.

11. What is the difference between class components and functional components?

Class components are ES6 classes that extend from `React.Component` and use lifecycle methods.

Functional components are plain JavaScript functions and rely on React Hooks for managing state and side effects.

****Key Differences:****

- Functional components are shorter and easier to test.
- Functional components use Hooks for lifecycle and state management.
- Class components are slightly more verbose and were used before Hooks were introduced.

12. What are React lifecycle methods?

Lifecycle methods are special methods in class components that allow you to run code at specific points in a component's lifecycle.

- ****Mounting:**** `constructor()`, `componentDidMount()`
- ****Updating:**** `shouldComponentUpdate()`, `componentDidUpdate()`
- ****Unmounting:**** `componentWillUnmount()`

In functional components, similar effects are achieved using the `'useEffect'` hook.

13. What is Redux and why is it used in React applications?

Redux is a state management library that helps manage and centralize application state. It's especially useful for large applications where data needs to be shared among many components.

****Core Concepts:****

- ****Store:**** Holds the application state.
- ****Actions:**** Describe what happened.
- ****Reducers:**** Specify how the state changes in response to actions.

React-Redux provides bindings for integrating Redux with React components.

14. What is the difference between useMemo and useCallback?

- `useMemo` memoizes the result of a function computation.
- `useCallback` memoizes the function definition itself.

Use `useMemo` to avoid recalculating expensive computations, and `useCallback` to avoid re-creating functions on each render.

15. How do you handle side effects in React?

Side effects in React are handled using the `useEffect` hook in functional components.

****Common side effects include:****

- Fetching data
- Subscribing to services
- Manually manipulating the DOM

`useEffect(() => { ... }, [dependencies])` runs after render and when dependencies change.

16. What are fragments in React?

Fragments let you group multiple elements without adding extra nodes to the DOM.

Instead of returning multiple `<div>` elements, you can use:

`<></>` or `<React.Fragment></React.Fragment>`

17. How does React handle forms?

React uses controlled components to manage form data. The input values are tied to component state using `useState` and updated via `onChange` handlers.

****Example:****

```
```\njsx\nconst [name, setName] = useState('');
```

```
return <input value={name} onChange={(e) => setName(e.target.value)} />;
...
```

## 18. What are error boundaries in React?

Error boundaries are React components that catch JavaScript errors in child components during rendering.

They log errors and display a fallback UI instead of crashing the app.

You can create one using `componentDidCatch()` and `getDerivedStateFromError()` in a class component.

## 19. What is lazy loading in React?

Lazy loading is a technique to load components only when they are needed. It reduces the initial bundle size and improves performance.

**Example:**

```
```jsx  
const MyComponent = React.lazy(() => import('./MyComponent'));  
<Suspense fallback={<div>Loading...</div>}>  
  <MyComponent />  
</Suspense>  
```
```

## 20. How do you handle routing in React?

React Router is the standard library for routing in React.

**Common components include:**

- `<BrowserRouter>`
- `<Routes>` and `<Route>`
- `<Link>` and `<NavLink>`

**Example:**

```
```jsx  
<Routes>  
  <Route path="/" element={<Home />} />  
  <Route path="/about" element={<About />} />  
</Routes>  
```
```