Basic React Questions

1. **What is React and why is it used?**
   - **Answer:** React is a JavaScript library for building user interfaces. It's used to create interactive UI components that efficiently update and render data changes. React's virtual DOM allows for efficient updates to minimize re-rendering of the entire UI.
2. **What is a component in React?**
   - **Answer:** A component in React is a reusable building block for UI elements. It encapsulates the logic and presentation of a part of the UI. Components can be functional (using functions) or class-based (using ES6 classes).
3. **Explain the concept of JSX.**
   - **Answer:** JSX (JavaScript XML) is a syntax extension for JavaScript used in React to write HTML-like code directly within JavaScript. It allows developers to write UI components using a familiar HTML/XML syntax, which gets transpiled into JavaScript.

Intermediate React Questions

4. **How does state management work in React?**
   - **Answer:** State in React is an object that holds component's dynamic data and determines its behavior and rendering. It can be modified using `setState()` method, triggering re-rendering of components when state changes.
5. **What are hooks in React? Provide examples.**
   - **Answer:** A hook is a function that lets you add stateful logic and other features to functional components. It enables components to manage their own state, handle side effects, and interact with the React lifecycle. In essence, hooks are essential tools for building dynamic and interactive user interfaces in React applications. Examples include:
     - **useState**: Allows functional components to manage and update their own state. And triggers re-renders when state changes.
     - **useEffect**: Is a Hook in React that allows you to perform side effects in functional components. Handles side effects like data fetching, subscriptions, or DOM manipulation.
     - `const [count, setCount] = useState(0);`

       `useEffect(() => { // Side effect code here }, [dependency]);`
6. **Explain the lifecycle methods of a React component.**
   - **Answer:** React class components have lifecycle methods:
     - **Mounting:** `constructor`, `render`, `componentDidMount`
     - **Updating:** `shouldComponentUpdate`, `render`, `componentDidUpdate`
     - **Unmounting:** `componentWillUnmount` These methods allow developers to perform actions at different stages of a component's lifecycle.

Advanced React Questions

7. **How do you optimize performance in a React application?**

- **Answer:** Performance optimization techniques in React include:
  - Memoization with `React.memo` and `useMemo` to avoid unnecessary renders.
  - Virtualization for long lists using libraries like `react-virtualized`.
  - Code splitting and lazy loading components with `React.lazy` and `Suspense`.
  - Optimizing images and assets, using production builds, and profiling tools like React Developer Tools.

8. **What is server-side rendering in React and how does it compare to client-side rendering?**
   - **Answer:** Server-side rendering (SSR) renders React components on the server before sending HTML to the client, improving initial load times and SEO. Client-side rendering (CSR) renders components in the browser, often faster for subsequent interactions but slower for initial page load and SEO.

9. **How do you handle side effects in React?**
   - **Answer:** Side effects (like data fetching, subscriptions) in React are handled using `useEffect` hook:

     ```
     useEffect(() => { // Side effect code here (e.g., API calls) return () => { // Cleanup code (optional) }; }, [dependency]);
     ```

     Use dependencies to control when the effect runs or cleanup resources in the return function.