Basic Level

1. **What is React?**

   Answer: React is a JavaScript library developed by Facebook for building user interfaces, particularly single-page applications. It allows for the creation of reusable UI components and manages the view layer of web and mobile apps.

2. **What is JSX and why is it used in React?**

   Answer: JSX stands for JavaScript XML. It is a syntax extension for JavaScript that allows developers to write HTML-like code directly in JavaScript. It makes it easier to visualize the UI and integrates tightly with React's component-based architecture.

3. **What is the difference between state and props?**

   Answer:

   - **State:** Managed within a component, mutable, and used to handle dynamic data.
   - **Props:** Passed from parent to child components, immutable, and used to pass data and event handlers.

4. **How does React handle events?**

   Answer: React uses Synthetic Events, a cross-browser wrapper around native events, providing a consistent API. Event handlers are passed as props, and the event object in React is normalized for consistency.

5. **What are keys in React, and why are they important?**

   Answer: Keys help React identify which items in a list have changed, been added, or removed. They are essential for optimizing the rendering process by ensuring that React updates only the necessary parts of the DOM.

Intermediate Level

1. **Explain the purpose of `useEffect` and how it can be used.**

   Answer: `useEffect` is a Hook that allows you to perform side effects in functional components. It can be used to fetch data, subscribe to events, or manually manipulate the DOM. It takes two arguments: a function and a dependency array. The function runs after the render, and the dependency array specifies when the effect should re-run.

2. **What are keys in React, and why are they important?**

Answer: Keys are unique identifiers used to identify which items in a list have changed, been added, or removed. They help React optimize the rendering process by tracking changes efficiently. Without keys, React would re-render all the items, leading to performance issues.

3. **What is the difference between `useState` and `useReducer`?**

   Answer:

   - **useState:** Suitable for managing simple state in a component.
   - **useReducer:** Ideal for managing more complex state logic and state transitions. It provides a more scalable way to handle multiple state variables and complex updates.

4. **Explain the lifecycle methods of a React component.**

   Answer:

   - **Mounting:** `componentDidMount()` - Called after the component is rendered.
   - **Updating:** `componentDidUpdate()` - Called after the component is updated.
   - **Unmounting:** `componentWillUnmount()` - Called before the component is removed from the DOM.
   - **Error Handling:** `componentDidCatch()` - Used for error handling in the component tree.

5. **What are React Hooks? Can you name a few commonly used hooks?**

   Answer: Hooks allow functional components to use state and other React features without writing a class. Common hooks include:

   - `useState`: For managing local state.
   - `useEffect`: For side effects like data fetching.
   - `useContext`: For accessing context.
   - `useReducer`: For more complex state logic.

6. **What is the Context API, and how is it used?**

   Answer: The Context API allows for sharing values like theme or authentication across components without passing props down manually at every level. It's used with `React.createContext()` and `Context.Provider`.

7. **How does React's reconciliation process work?**

   Answer: React's reconciliation process involves diffing the new virtual DOM with the previous one to identify changes. React then updates only the changed elements in the real DOM, optimizing performance.

Advanced Level

1. **What is the purpose of Redux, and how does it work with React?**

   Answer: Redux is a state management library that centralizes the application's state in a single store. React components can connect to the store to read state and dispatch actions. It provides a predictable state container for JavaScript applications.

2. **Explain higher-order components (HOCs) in React.**

   Answer: HOCs are functions that take a component and return a new component, enhancing it with additional props or logic. They are used to reuse component logic, such as authentication or data fetching.

3. **What are React Portals, and when would you use them?**

   Answer: React Portals allow rendering of child components into a DOM node outside of the parent component's hierarchy. They are used for modals, tooltips, or overlays that need to escape the parent's styling constraints.

4. **Describe the concept of React's Concurrent Mode.**

   Answer: Concurrent Mode, now part of React's Concurrent Features, helps applications remain responsive by allowing React to interrupt rendering, prioritize tasks, and manage asynchronous data fetching. It enhances the user experience on slower devices and networks.

5. **How would you optimize the performance of a React application?**

   Answer: Performance optimization in React can be achieved by:

   - Using `React.memo` to prevent unnecessary re-renders.
   - Using `useCallback` and `useMemo` to memoize functions and values.
   - Implementing lazy loading and code splitting.
   - Reducing the number of re-renders by lifting state and minimizing props passed to child components.

6. **How does React's reconciliation process work?**

Answer: React's reconciliation process involves the comparison of the new virtual DOM with the previous one to determine the minimal set of changes needed to update the real DOM. This process, known as "diffing," ensures efficient updates by only re-rendering parts of the UI that have changed.

7. **What are React Portals, and when would you use them?**

   Answer: React Portals provide a way to render children into a DOM node that exists outside the parent component's DOM hierarchy. They are often used for modals, tooltips, and overlays, where you need to break out of the parent container's overflow and z-index constraints.

8. **Explain how you would implement lazy loading in a React application.**

   Answer: Lazy loading in React can be implemented using `React.lazy()` and `Suspense`. `React.lazy()` allows you to dynamically import a component, and `Suspense` lets you display a fallback UI (like a loading spinner) while the component is being loaded:

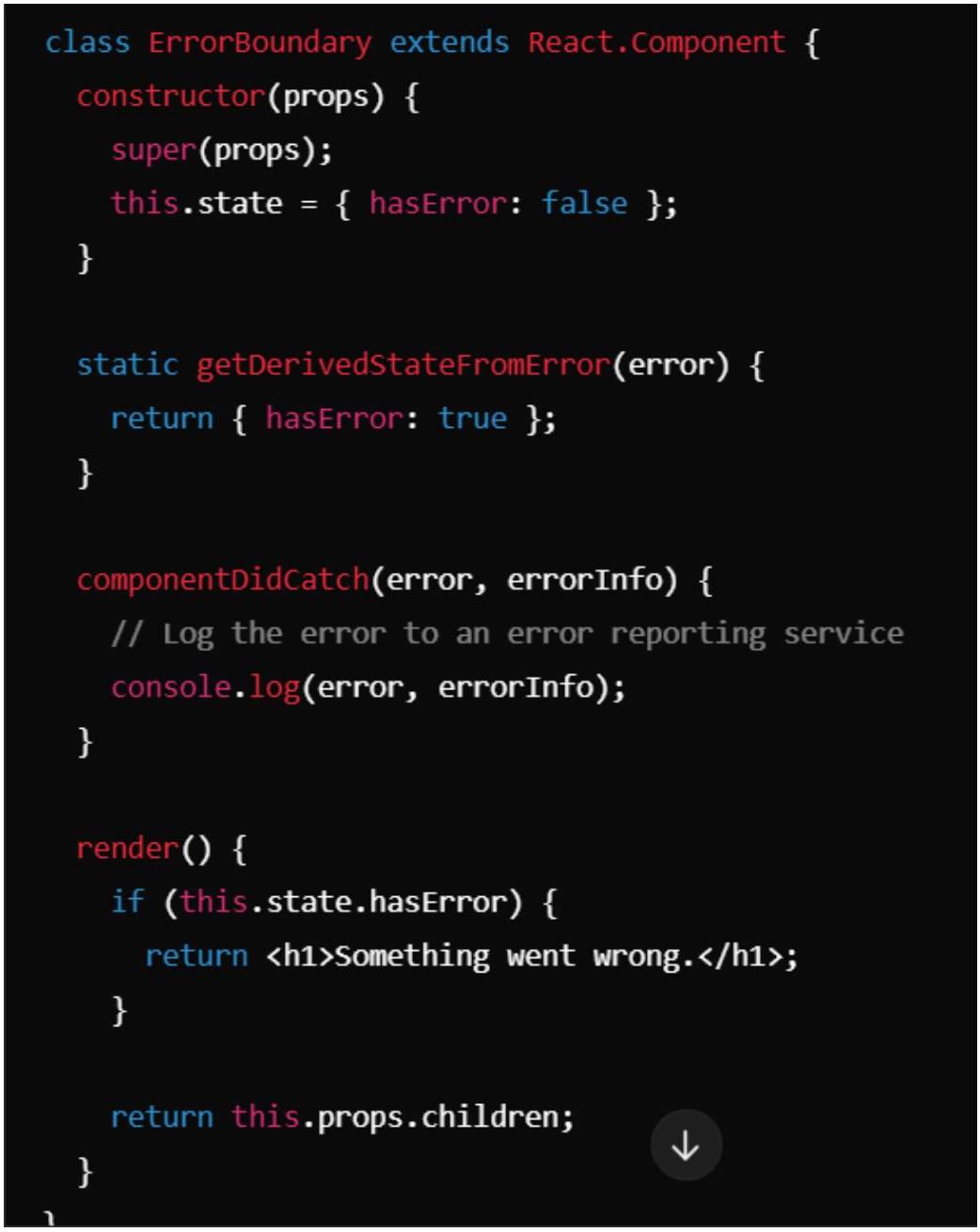
```javascript
const LazyComponent = React.lazy(() => import('./LazyComponent'));

function App() {
  return (
    <React.Suspense fallback={<div>Loading...</div>}>
      <LazyComponent />
    </React.Suspense>
  );
}
```

9. **Describe the concept of React's Concurrent Mode.**

   Answer: Concurrent Mode (now referred to as Concurrent Features) is an experimental set of features in React that helps applications stay responsive and gracefully adjust to the user's device capabilities and network speed. It allows React to interrupt rendering, prioritize tasks, and handle asynchronous data fetching.

10. . **How would you handle error boundaries in a React application?**

Answer: Error boundaries are React components that catch JavaScript errors anywhere in their child component tree, log those errors, and display a fallback UI. They are created by defining a class component with a `componentDidCatch` lifecycle method and a `getDerivedStateFromError` method:



```
class ErrorBoundary extends React.Component {
  constructor(props) {
    super(props);
    this.state = { hasError: false };
  }

  static getDerivedStateFromError(error) {
    return { hasError: true };
  }

  componentDidCatch(error, errorInfo) {
    // Log the error to an error reporting service
    console.log(error, errorInfo);
  }

  render() {
    if (this.state.hasError) {
      return <h1>Something went wrong.</h1>;
    }

    return this.props.children;
  }
}
```

State Management and Lifecycle

1. **What is the difference between controlled and uncontrolled components?**
    - **Answer:**
        - **Controlled Components:** These components have their form data controlled by React state. The value of the input field is determined by the state.
        - **Uncontrolled Components:** These components manage their own state internally. Refs are used to access the DOM values directly.
2. **What are React Fragments, and why are they useful?**
    - **Answer:** React Fragments (`<React.Fragment>` or `<>`) allow you to group multiple elements without adding extra nodes to the DOM. This helps in reducing unnecessary wrapper elements, thus keeping the DOM clean.
3. **How does the `useContext` Hook work?**
    - **Answer:** `useContext` allows components to access context values without needing to wrap them in a `Context.Consumer`. It simplifies context usage in functional components by directly retrieving the context value.

Advanced React Concepts

4. **What is Server-Side Rendering (SSR) in React?**
    - **Answer:** SSR is a technique used to render React components on the server and send fully rendered pages to the client. This improves performance and SEO. Frameworks like Next.js enable SSR in React applications.
5. **What is the difference between `React.Component` and `React.PureComponent`?**
    - **Answer:**
        - **`React.Component`:** Regular component that doesn't implement `shouldComponentUpdate` by default.
        - **`React.PureComponent`:** Implements a shallow comparison of props and state in `shouldComponentUpdate` to avoid unnecessary renders when the output is not affected.
6. **Explain the concept of code splitting in React.**
    - **Answer:** Code splitting is a feature that allows you to split your code into various bundles which can be loaded on demand. This can significantly improve the performance of your application by reducing the size of the initial load bundle. Tools like `React.lazy()` and `import()` help implement code splitting.

React Ecosystem

7. **What is React Router, and how does it work?**
    - **Answer:** React Router is a library for routing in React applications. It enables navigation between different views of components in a single-page application without reloading the page. It uses a component-based approach for defining routes.
8. **What is the difference between `Link` and `NavLink` in React Router?**

- **Answer:**
  - **`Link`:** Used to create links that navigate to different routes without refreshing the page.
  - **`NavLink`:** Similar to `Link`, but it can apply active styles to the link when it matches the current URL, which is useful for navigation bars.

9. **How does `React.memo` work?**
   - **Answer:** `React.memo` is a higher-order component that prevents a functional component from re-rendering if its props have not changed. It's similar to `PureComponent` but for functional components.

React Performance Optimization

10. **What are the differences between useMemo and useCallback?**
    - **Answer:**
      - **`useMemo`:** Memoizes the result of a computation and returns the cached value if the dependencies haven't changed.
      - **`useCallback`:** Memoizes a function and returns the cached function reference if the dependencies haven't changed.
11. **How would you implement a virtual list in React?**
    - **Answer:** A virtual list renders only the visible items to the user, improving performance when dealing with large lists. Libraries like `react-virtualized` or `react-window` can be used to implement this.