Basics Interview Questions:

1.What is React.js? Done

-JS library , use to make reusable components

2.Explain the key features of React.js. Done

-VDOM,Reusable,JSX,Community,Unidirectional,Hooks

3.What are the differences between React class components and functional components? Done

-syntax, lifecycle methods, easy to learn, keywords (class), render method

4.How do you create a React component? Done

-Class based and functional

5.What is JSX in React? How is it different from HTML? Done

-HTML like syntax with JS

6.What are React hooks? Provide some examples of built-in hooks. Done

-function allow developers to add state and other react feature in functional compo

7.How do you pass data from a parent component to a child component in React? Done

-props and redux also

8.What is state in React, and how is it different from props? Done

-State: mechanism for holding and managing data within a component

-Props: pass data from a parent component to a child component

9.Explain the component lifecycle methods in React class components. Done

10.What is the virtual DOM in React, and how does it improve performance? Done

-light weight representation of DOM, through diff algo and reconcilation

11.How do you handle events in React? Done

- **Preventing Default Behavior:, Event Handling in Functional Components: Binding Event Handlers:**

12.What is conditional rendering in React? Provide an example. Done

Conditional rendering allows you to create dynamic user interfaces that adapt to the application's state, making it easier to show different content or UI components based on various conditions and user interactions.

How do you handle forms in React?

Using Controlled and uncontrolled

13.Explain the concept of lifting state up in React.

-moving the state management from a child component to its parent component

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

15.How do you perform AJAX requests in React?

16.What are Higher-Order Components (HOCs) in React, and why would you use them?

17.What is React Router, and how do you use it for routing in React applications?

React Router is a popular library for handling routing in React applications. It provides a declarative way to define navigation and allows you to create single-page applications (SPAs) with multiple views or pages. React Router ensures that the URL of your application reflects the current state of the UI, enabling users to navigate between different views without triggering a full page reload.

BrowserRouter: ,Route: ,Link and NavLink:, Switch: ,Routes:

18.What are the benefits of using Redux with React?

19.Explain the differences between controlled and uncontrolled components in React.

-------------------------------

Advanced interview Questions:

2.What are React Fragments, and why are they useful?

3.How does React achieve code splitting, and why is it beneficial for performance?

4.Explain the concept of lazy loading in React and how it can improve the initial load time of an application.

5.What are React portals, and in what scenarios would you use them?

6.How do you optimize the performance of a React application? Discuss some best practices and tools for performance optimization.

7.What are the different methods for handling forms in React, and what are the pros and cons of each approach?

8.Explain the concept of "render props" and how it can be used to share code between components.

9.What are React error boundaries, and how do they help in preventing the entire application from crashing due to errors in a specific component?

10.Discuss the differences between server-side rendering (SSR) and client-side rendering (CSR) in React. When would you choose one over the other?

11.How do you handle authentication and authorization in a React application?

12.What is the significance of the Context API in React, and how can it be used to manage global state?

13.Explain the differences between React's reconciliation and virtual DOM algorithms, and how they contribute to React's performance.

14.What are Higher-Order Components (HOCs), and how can you refactor components using HOCs?

15.How does React work with third-party libraries or non-React code? Provide examples of integrating React with other JavaScript frameworks.

16.What are React hooks, and how do they change the way you write React components compared to class components?

17.How do you handle routing in a large-scale React application? Discuss different routing libraries and their benefits.

18.Explain the concept of memoization in React. How does it help in optimizing functional components?

19.What are the different testing approaches and libraries available for testing React components and applications?

20.How do you manage state in a React application without using Redux? Discuss alternative state management solutions.

21.What are the best practices for optimizing SEO in a React application?

1. **Explain the Virtual DOM in React and how it improves performance.**
2. **What are the differences between state and props in React? When would you use one over the other?**
3. **How do you handle forms in React? Explain controlled and uncontrolled components.**
4. **What are React hooks? How do they differ from class components, and what are the most commonly used hooks?**
5. **How does React Router work, and how do you implement routing in a React application?**
6. **What is Redux, and why would you use it in a React application? How does it differ from React's built-in state management?**
7. **Explain the concept of higher-order components (HOC) and their use cases.**
8. **What are the lifecycle methods in React class components, and how have they evolved with the introduction of hooks?**
9. **How do you optimize the performance of a React application?**
10. **What are React fragments, and how do they improve rendering efficiency?**
11. **How would you handle errors in a React application, and what are the best practices for error boundaries?**
12. **What are the key differences between React and other JavaScript frameworks/libraries like Angular and Vue.js?**
13. **How can you prevent unnecessary re-renders in React components?**
14. **Explain the concept of React context and when it's useful to use it.**
15. **What are the different ways to style React components? Compare CSS modules, styled-components, and inline styles.**
16. **How do you perform server-side rendering (SSR) with React? What are the benefits of SSR?**
17. **Explain the purpose of keys in React lists and the potential issues of using an index as a key.**
18. **What are React's PropTypes and how do they help in debugging and maintaining a React codebase?**
19. **How would you handle performance optimization for React applications that deal with a large number of items in a list?**
20. **Discuss the concept of code splitting in React and how it can be achieved.**

------------------------------------------------

Implementations based questions