React Native

Here is a list of 100 React Native interview questions and answers, categorized by basic, intermediate, advanced, and technical levels:

Basic Questions (1-25)

1. **What is React Native?**
    - React Native is a JavaScript framework used for building mobile applications for iOS and Android using the same codebase. It uses React and native components to build mobile apps.
2. **What is JSX in React Native?**
    - JSX is a syntax extension for JavaScript. It looks similar to HTML, but it is used to describe what the UI should look like. It is later transformed into JavaScript code.
3. **What is the difference between React and React Native?**
    - React is a library for building user interfaces for web applications, while React Native is a framework for building mobile applications using React.
4. **What are Props in React Native?**
    - Props are the short form for properties. They are used to pass data from one component to another, making the component reusable.
5. **What is the use of `State` in React Native?**
    - State is used to store and manage data within a component. It allows the component to re-render when the data changes.
6. **What is the purpose of `componentDidMount` in React Native?**
    - `componentDidMount` is a lifecycle method in React that is called after a component is mounted. It's used for tasks like API calls or setting up subscriptions.
7. **What is the Virtual DOM in React Native?**
    - The Virtual DOM is an in-memory representation of the real DOM elements. React Native uses it to optimize rendering by comparing the virtual DOM with the actual DOM and applying the minimal updates required.

8. **What are functional components in React Native?**
    - Functional components are stateless components that are defined as JavaScript functions, which receive `props` as arguments and return JSX.
9. **What are class components in React Native?**
    - Class components are components that extend `React.Component` and can hold state and lifecycle methods.
10. **What are the advantages of using React Native?**
    - It enables building cross-platform apps with a single codebase, offering faster development and a native-like performance experience.
11. **What is `useState` hook in React Native?**
    - `useState` is a Hook that allows you to add state to a functional component in React Native.
12. **How do you create a component in React Native?**
    - Components can be created using either JavaScript functions or ES6 classes. Functional components are more commonly used today.
13. **What is the role of the `render()` method in React Native?**
    - The `render()` method is used in class components to return the JSX that defines the structure of the component.
14. **What is a `key` prop in React Native?**
    - The `key` prop is a special string attribute used to help React identify which items have changed, are added, or are removed in a list.
15. **What is the difference between `state` and `props` in React Native?**
    - `State` is used to manage local data within a component, while `props` are used to pass data from a parent component to a child component.
16. **What is the `FlatList` component in React Native?**
    - `FlatList` is a performance-optimized component for rendering large lists of data. It only renders items that are currently visible on the screen.

17. **What is `StyleSheet` in React Native?**

    - `StyleSheet` is an abstraction for creating stylesheets in React Native. It uses a JavaScript object to define styles, which are then converted to native styles.

18. **How do you make HTTP requests in React Native?**

    - You can use `fetch` API or libraries like Axios to make HTTP requests in React Native.

19. **What is the `Text` component in React Native?**

    - The `Text` component is used to display text in React Native apps. It supports nesting and styling.

20. **What are controlled components in React Native?**

    - Controlled components are components where form data is handled by the React state, and the value of the input is controlled by state.

21. **What is the `TouchableOpacity` component in React Native?**

    - `TouchableOpacity` is a wrapper for making any element touchable and provides feedback with reduced opacity when touched.

22. **What is the purpose of `setState()` in React Native?**

    - `setState()` is used to update the state of a component, triggering a re-render of the component with the new state.

23. **What is `useEffect` hook in React Native?**

    - `useEffect` is a hook used to perform side effects in functional components, such as data fetching, subscriptions, or manually changing the DOM.

24. **What are React Native's key performance optimization techniques?**

    - Techniques include lazy loading, using `FlatList` for large data sets, optimizing images, and using native modules for performance-critical tasks.

25. **How do you debug React Native applications?**

    - Debugging can be done using tools like React Native Debugger, Chrome DevTools, and `console.log` statements for tracking issues.

Intermediate Questions (26-50)

26. **What is React Navigation in React Native?**
    - React Navigation is a popular library used for navigation and routing in React Native applications. It provides stack, tab, and drawer navigation options.

27. **What is the difference between `useEffect` and `componentDidMount`?**
    - `useEffect` is for functional components and runs after every render, whereas `componentDidMount` is used in class components and runs once after the initial render.

28. **How do you pass data between components in React Native?**
    - Data can be passed from parent to child components via `props` or between sibling components using state management solutions like `Context` or third-party libraries.

29. **What is Redux in React Native?**
    - Redux is a state management library that helps manage the application state in a predictable way. It uses actions, reducers, and a single store to maintain state.

30. **What is `React Context API`?**
    - The Context API provides a way to pass data through the component tree without passing props manually at every level.

31. **What are Higher Order Components (HOC) in React Native?**
    - HOCs are functions that take a component and return a new component with enhanced functionality or behavior.

32. **What is the purpose of the `useReducer` hook in React Native?**
    - `useReducer` is used for managing complex state logic in functional components, especially when the state transitions are based on actions.

33. **What is the difference between `state` and `useState` in React Native?**

- `state` is a class component concept, while `useState` is a hook used for managing state in functional components.

34. **How do you optimize an app for better performance in React Native?**

    - Techniques include reducing unnecessary renders, using `FlatList` for large lists, memoization with `React.memo()`, and using `shouldComponentUpdate()` in class components.

35. **What is the difference between `useCallback` and `useMemo` hooks?**

    - `useCallback` returns a memoized version of a function, while `useMemo` returns a memoized value based on dependencies.

36. **What is the purpose of `React Native Elements`?**

    - `React Native Elements` is a UI library that provides customizable and reusable UI components to simplify React Native development.

37. **How do you handle navigation parameters in React Navigation?**

    - Parameters can be passed via the `navigation.navigate()` function and accessed using `route.params` in the target screen component.

38. **What is the `FlatList` performance optimization in React Native?**

    - `FlatList` optimizes performance by rendering only the visible items and unloading items that are off-screen, which reduces memory consumption.

39. **What is `AsyncStorage` in React Native?**

    - `AsyncStorage` is a simple key-value store for storing data locally on the device, such as user preferences or session data.

40. **How do you integrate native modules in React Native?**

    - Native modules can be integrated by creating custom Java/Kotlin or Objective-C/Swift code that exposes functionality to JavaScript through a bridge.

41. **What is `react-native link` and how is it used?**

    - `react-native link` is a command used to link native dependencies to a React Native project. It automates the process of adding native code into a

React Native app.

42. **What is the role of `AppRegistry` in React Native?**
    - `AppRegistry` is responsible for registering the root component of the app and scheduling the component to be rendered on the screen.

43. **What are the differences between `ScrollView` and `FlatList`?**
    - `ScrollView` renders all the child components at once, while `FlatList` renders only the visible items to improve performance.

44. **What is a `TouchableWithoutFeedback` component in React Native?**
    - `TouchableWithoutFeedback` is used to detect touch gestures and trigger an action, without the visual feedback like opacity change.

45. **What is a `Modal` component in React Native?**
    - A `Modal` is a component that renders content above the current view and can be used for dialogs, popups, or full-screen overlays.

46. **How do you handle images in React Native?**
    - Images can be handled using the `Image` component. You can either use local images from the assets or fetch images from remote URLs.

47. **What is `SafeAreaView` in React Native?**
    - `SafeAreaView` is a component that renders content within the safe area boundaries, preventing UI elements from overlapping with system UI (like notches or bottom bars).

48. **How do you implement animations in React Native?**
    - Animations in React Native can be created using the `Animated` library, which provides various types of animations such as translation, scaling, and opacity.

49. **What is `react-navigation-stack` used for?**
    - `react-navigation-stack` is a package that enables stack-based navigation in React Native, managing navigation between different screens with push and pop methods.

50. **What is the role of `React Native Debugger`?**

    - React Native Debugger is a debugging tool that combines Redux DevTools and Chrome DevTools to provide an enhanced debugging experience for React Native apps.

Advanced Questions (51-75)

51. **What is the purpose of using `shouldComponentUpdate`?**

    - `shouldComponentUpdate` is used to optimize performance by preventing unnecessary re-renders when the component's state or props have not changed.

52. **How does the React Native bridge work?**

    - The React Native bridge enables communication between JavaScript and native modules. It allows JavaScript to call native code and vice versa using a bridge mechanism.

53. **What is `React Native Paper`?**

    - React Native Paper is a UI component library that implements Material Design for React Native apps, providing customizable components like buttons, cards, and dialogs.

54. **How do you implement deep linking in React Native?**

    - Deep linking in React Native can be implemented using the `react-navigation` library in combination with the `Linking` API to handle URLs and navigate to specific screens.

55. **What are `Hermes` and its benefits in React Native?**

    - Hermes is an open-source JavaScript engine optimized for React Native. It improves app startup time, reduces memory usage, and decreases APK size.

56. **What is the difference between React Native's `ActivityIndicator` and `Spinner`?**

    - `ActivityIndicator` is the standard React Native component for displaying loading indicators, while `Spinner` is a more customizable component often

used in other UI libraries.

57. **How do you configure Redux with React Native?**
    - Redux is configured in React Native by setting up the `createStore`, `Provider`, and `reducer` functions, and connecting them to the components using `connect` or `useSelector`.

58. **How does React Native handle background tasks?**
    - React Native can handle background tasks through third-party libraries like `react-native-background-fetch`, which allows the app to perform background jobs like syncing data or sending notifications.

59. **What is the role of `getDerivedStateFromProps`?**
    - `getDerivedStateFromProps` is a lifecycle method in class components that is used to update the state based on changes in the props.

60. **How do you handle state management in a large React Native application?**
    - State management in large React Native apps is usually handled using Redux, Context API, or third-party solutions like Recoil or MobX.

61. **How do you integrate Firebase with React Native?**
    - Firebase is integrated with React Native using the `react-native-firebase` library. You configure Firebase in both Android and iOS projects and use Firebase services like authentication, database, and storage.

62. **How do you manage form validation in React Native?**
    - Form validation can be handled using libraries like `Formik` or `React Hook Form` in combination with validation libraries like `Yup`.

63. **What is the `React Native Web`?**
    - `React Native Web` is a project that enables React Native components to be used in web applications, allowing for shared codebases across web and mobile.

64. **What is `react-native-reanimated`?**
    - `react-native-reanimated` is a library for building complex animations with React Native. It provides a more performant and flexible approach to

animations compared to the `Animated` API.

65. **What is the role of `Animated API` in React Native?**

    - The `Animated API` allows for smooth and declarative animations in React Native. It provides building blocks like `Animated.View`, `Animated.Text`, and more for creating animations.

66. **How do you handle API error handling in React Native?**

    - API error handling in React Native can be done using `try-catch` blocks, checking the response status, and displaying user-friendly error messages or alerts.

67. **What is `react-native-svg` used for?**

    - `react-native-svg` is a library used to render scalable vector graphics (SVG) in React Native, enabling the use of vector-based images for mobile apps.

68. **What is the `React Native CLI`?**

    - The React Native CLI is a command-line interface that is used to initialize and manage React Native projects, providing commands for running, building, and testing React Native apps.

69. **What are `React Native Navigation`'s major components?**

    - Major components of `React Native Navigation` include `Stack Navigator`, `Drawer Navigator`, and `Tab Navigator`, which allow for organizing navigation flow between screens.

70. **How do you handle performance issues in React Native?**

    - Performance can be optimized using profiling tools, reducing unnecessary renders, lazy loading components, and using `FlatList` for large datasets.

71. **What are `React Native Gesture Handler` and `React Native Reanimated`?**

    - `React Native Gesture Handler` provides an API for handling gestures in a performant and flexible way, while `React Native Reanimated` is used for creating complex animations and transitions.

72. **How do you implement a custom native module in React Native?**

    - A custom native module can be created by writing platform-specific code (Java/Swift/Objective-C) and exposing it to JavaScript using the React Native bridge.

73. **What is the purpose of `useLayoutEffect` in React Native?**

    - `useLayoutEffect` is similar to `useEffect`, but it is called synchronously after the DOM has been painted, making it suitable for measuring DOM elements and performing layout-related tasks.

74. **What is `Expo` in React Native?**

    - Expo is a set of tools and libraries for React Native that simplifies the development process. It provides an easy way to build and deploy apps without the need for Xcode or Android Studio.

75. **How do you use Firebase Cloud Messaging (FCM) with React Native?**

    - Firebase Cloud Messaging can be integrated with React Native using the `react-native-firebase` package, allowing push notifications to be sent to users.

Technical Questions (76-100)

76. **How do you implement custom animations in React Native?**

    - Custom animations can be created using the `Animated` API or `react-native-reanimated` by defining animation values and linking them to component styles.

77. **How do you handle device-specific code in React Native?**

    - Device-specific code can be handled using platform-specific file extensions (e.g., `.ios.js`, `.android.js`) or by using the `Platform` module to detect the device type.

78. **How do you test a React Native application?**

    - React Native applications can be tested using Jest for unit tests, `react-native-testing-library` for component tests, and `detox` for end-to-end testing.

79. **How do you manage background tasks in React Native?**

    - Background tasks can be managed using libraries like `react-native-background-fetch` or `react-native-background-task` to handle periodic tasks.

80. **What are React Native's major lifecycle methods?**

    - React Native's major lifecycle methods include `componentDidMount`, `componentDidUpdate`, `componentWillUnmount`, and `shouldComponentUpdate`.

81. **How does React Native manage memory and garbage collection?**

    - React Native uses JavaScript's automatic garbage collection system to manage memory. However, developers should be mindful of memory leaks caused by lingering references or events.

82. **How do you bundle a React Native application for production?**

    - A production bundle is created using `react-native bundle` command, which compiles JavaScript, images, and assets into a single file for production.

83. **How do you implement theming in React Native?**

    - Theming can be implemented using context or libraries like `styled-components` or `react-native-paper` to provide consistent styles across the app.

84. **How do you optimize a React Native app for a faster startup?**

    - Optimizations include reducing bundle size, lazy loading assets, using Hermes for improved JavaScript performance, and deferring non-essential tasks.

85. **What is the `Linking` API in React Native?**

    - `Linking` is used for deep linking and opening external URLs, such as opening a website or dialing a phone number.

86. **How do you handle app navigation state persistence in React Native?**

    - App navigation state persistence can be handled using libraries like `redux-persist` or by using `React Navigation`

's state persistence options.

87. **What is the difference between `react-navigation` and `react-native-navigation`?**
    - `react-navigation` is a JavaScript-based navigation library, while `react-native-navigation` is a native navigation solution that provides better performance for large applications.

88. **How do you access native modules in React Native?**
    - Native modules are accessed using the bridge by invoking specific methods exposed by the native code via JavaScript.

89. **What are some common performance bottlenecks in React Native?**
    - Common performance bottlenecks include large component trees, non-optimal list rendering, excessive re-renders, and large image assets.

90. **What is the role of `async` and `await` in React Native?**
    - `async` and `await` are used for handling asynchronous operations in a more readable and maintainable way, avoiding callback hell.

91. **How do you set up a CI/CD pipeline for React Native?**
    - CI/CD pipelines can be set up using tools like Jenkins, CircleCI, or GitHub Actions, with automated builds, tests, and deployments.

92. **How do you handle touch gestures in React Native?**
    - Touch gestures can be handled using the `react-native-gesture-handler` library, which provides gesture recognition for tap, swipe, and drag interactions.

93. **How do you use native modules to access device capabilities?**
    - Native modules are created in platform-specific languages (Java/Swift/Objective-C) to access device features, and then exposed to JavaScript via the React Native bridge.

94. **What is the difference between `require` and `import` in React Native?**
    - `require` is used for loading modules at runtime, while `import` is a static import used for importing modules at compile-time.

95. **How do you manage assets in React Native?**

    - Assets can be managed by placing them in the appropriate directory and using `require()` or `Image` components to reference and display them in the app.

96. **What are the best practices for optimizing React Native apps for performance?**

    - Best practices include lazy loading components, using `FlatList` for lists, avoiding unnecessary re-renders, and optimizing image loading and memory usage.

97. **How do you handle network requests and responses in React Native?**

    - Network requests can be handled using the `fetch` API or Axios, and responses can be processed with promise chaining or async/await.

98. **What is the purpose of using `react-native-paper`?**

    - `react-native-paper` is a UI component library that implements Material Design for React Native apps, making it easier to build consistent and accessible interfaces.

99. **How do you manage updates in React Native apps?**

    - Updates are managed by creating new versions of the app and submitting them to the app stores or using over-the-air (OTA) updates with tools like `CodePush`.

100. **What is `expo` and how does it relate to React Native?** - `Expo` is a framework that simplifies React Native development by providing a set of pre-configured libraries and tools for building and deploying apps quickly.