

React Native Questionnaire

- [React Native Questionnaire](#)

React Native Questionnaire

1. What are the key differences between React Native and ReactJS?

Sample answer:

React Native is used to develop mobile apps for iOS and Android, whereas ReactJS is used to build web apps in a web browser.

Both use reusable JavaScript XML components, but the syntax varies: while React Native uses app-view components like `<View>` and `<Text>`, ReactJS uses HTML tags such as `<div>` and `<h1>`.

2. What products and apps is React Native best used for?

Sample answer:

React Native is a great option for developing a hybrid app that does not require extremely high performance.

Cross-platform compatibility means development teams can save lots of time when using React Native compared to a native framework.

However, it might not be suitable when designing complex apps or if developers aren't already well-versed in React code.

3. What is component-driven development?

Sample answer:

Component-driven development (CDD) is a development methodology where the build process is anchored around components rather than objects. Components are loosely coupled and each one serves its own purpose.

When put together, components (buttons, navigation bars, images) form the program as a whole. React Native is a component-driven framework.

4. What is the role of AsyncStorage in React Native?

Sample answer:

AsyncStorage is React Native's key-value, unencrypted storage module that allows developers to store data for offline use. Typically, it's used to store data when an app is not linked to a cloud service, or when specific features require data storage.

5. What is the role of Flexbox in React Native?

Sample answer:

In React Native apps, Flexbox is used to provide a consistent layout across different screen types. The Flexbox algorithm helps to structure the positioning of different components and create a responsive UI for the end user.

6. What coding languages are compatible with React Native?

Sample answer:

While React Native is generally used with JavaScript, compatibility with other coding languages, including Python, C++, and C, is also possible through the framework's Java Native Interface (JNI).

7. What are the main disadvantages of using React Native?

Sample answer:

As with any software framework, React Native has its fair share of drawbacks. These include:

- **Non-nativity:** React Native isn't a native solution, which means its apps may be slightly slower than native ones
- **Debugging issues:** React Native is built using Javascript, Objective-C, Java, and C or C++, which can make debugging more difficult
- **Memory management:** Limitations on memory mean React Native is not suited to developing computation-intensive apps
- **Low security:** React Native's open-source design leaves apps more exposed to threats, which is especially dangerous for apps containing sensitive information, such as banking services
- **Learning curve:** React Native is one of the more challenging software frameworks to learn, especially for junior developers

8. What is the interaction manager in React Native?

Sample answer:

In React Native, the interaction manager is used to defer the execution of a function until a specified 'interaction' has been completed.

This is important as React Native is single-threaded in nature, meaning queued animations on the UI can become congested. The interaction manager helps with this issue and ensures that animations run smoothly in a scheduled manner.

9. How do you create basic text input in React Native?

Sample answer:

The insertion of basic text in React Native apps is handled by the Text and TextInput components. TextInput allows users to type on the app. We can implement it using the following syntax: `<import { Text, TextInput, View } from 'react-native'>`.

10. How can sensitive data be stored securely in React Native?

Sample answer:

Most React Native data is stored in Async Storage. As an unencrypted, local form of storage, it's not suitable for storing sensitive data such as tokens and passwords.

Alternatively, React Native Keychain offers a secure form of storage that also works similarly to Async Storage. For iOS, Keychain storage can be used to protect sensitive data, while Android developers can use Facebook Conceal and Android Keystore.

11. What are some differences between using React Native for iOS and for Android?

Sample answer:

Around 85% of React Native code is cross-platform, which means that most processes are the same on both iOS and Android. However, there are a few minor differences. These include:

- iOS development uses Mac and Xcode, whereas Android development draws on Android SDK and an emulator
- Third-party plugins that don't offer native functionality will need to be used differently
- The bridging process can be slightly different when developing complex apps

12. What are the differences between Flexbox in browser and in React Native?

Sample answer:

Flexbox generally works in the same way in React Native as it does in CSS on the web. However, there are a few minor differences in the values. These include:

- The default value in React Native is column, whereas the default value for CSS is row
- The default value in React Native is flex-start, whereas the default value for CSS is stretch
- The default value in React Native is 0, whereas the default value for CSS is 1

13. What steps can you take to resolve persistent memory leak issues?

Sample answer:

A memory leak occurs when memory that is no longer needed by an app remains in the app rather than being returned to the operating system. This is one of the most common causes of performance issues.

In theory, memory management is handled automatically by the garbage collector. However, this process is still prone to errors. Debugging tools can be used to identify memory leak issues. Some of the most common causes for issues are:

- Timers and listeners in `componentDidMount`
- Inline styles
- Closure scope leaks
- The use of `console.log`

Debugging can identify the root of the memory leak; once it has been removed, the issue should be resolved.

14. What are the main performance issues in React Native and what causes them?

Sample answer:

Some of the most common performance issues in React Native include:

- **High CPU usage:** Offloading complex functions to the JavaScript thread can cause performance issues
- **Memory leak:** Information can be lost in the Bridge during the transfer from the Primary to the React Native domains, especially in Android apps
- **Slow navigation:** Multiple thread bridging can also cause slower navigation times

15. How can app performance be optimized in React Native?

Sample answer:

There are several techniques that we can use to optimize an app's performance in React Native, such as:

- Remove all console statements
- Resize and scale down images
- Cache images internally
- Compress or convert raw JSON data
- Use code splitting for large lists
- Schedule animations
- Remove unnecessary libraries and features

16. How is React Native code processed to display the final output on the screen?

Sample answer:

The process for rendering code in React Native is the following:

1. When the app is opened, the main thread (or UI thread) starts execution by loading JavaScript bundles
2. Once the JavaScript code has been loaded successfully, the main thread sends it to the second JS thread where more calculations are performed
3. When React Native starts rendering, the reconciler algorithm generates a virtual DOM or layout, which is then sent to a third shadow thread
4. The shadow thread calculates a new DOM and sends the layout characteristics to the main UI thread
5. The UI thread then renders the received DOM for display on the mobile app

17. What is the role of the bridge in React Native?

Sample answer:

The bridge acts as a transport layer between JavaScript and Native modules. In the rendering process:

- The bridge first receives the user response to open the app from the Native module
- It then passes the serialized payload to the JavaScript module
- Once the event has been processed and a virtual DOM has been generated in the JavaScript module, the bridge receives the serialized batched response
- The bridge passes the serialized batched response to the Native module for final rendering

18. What is the interaction manager in React Native?

Sample answer:

In React Native, the interaction manager is used to defer the execution of a function until a specified 'interaction' has been completed.

This is important as React Native is single-threaded in nature, meaning queued animations on the UI can become congested. The interaction manager helps with this issue and ensures that animations run smoothly in a scheduled manner.

19. What is the role of fabric in React Native?

Sample answer:

Fabric is a modern type of architecture that was first created in 2018 and aims to address some of React Native's performance issues. Fabric modernizes the framework's rendering layer by allowing specified priority tasks to be executed synchronously and, therefore, quicker.

20. How do you create basic text input in React Native?

Sample answer:

The insertion of basic text in React Native apps is handled by the Text and TextInput components. TextInput allows users to type on the app. We can implement it using the following syntax: `<import { Text, TextInput, View } from 'react-native'>`.

21. How can you optimize the performance of images in React Native?

Sample answer:

There are several useful tricks for optimizing the performance of images in React Native. These include:

- Using image caching tools
- Using PNG or WEBP formats rather than JPEG
- Using smaller images
- Reducing the number of renders

22. What is the role of timers in a React Native app?

Sample answer:

In React Native, timers allow developers to manipulate the order in which events in a program occur. There are four different types of timers, each one serving a different purpose:

1. **Timeout** implements a delay
2. **Interval** allows repeat actions to occur at given intervals
3. **Immediate** allows actions to occur as soon as possible
4. **Animation** allows animations to display when the program is ready to render frames

23. How do you create a basic button in React Native?

Sample answer:

We can create basic buttons using the following syntax: `<import { View, Button, StyleSheet } from "react-native">`. Basic buttons support a minimal level of customization and can be modified using

TouchableOpacity or TouchableWithoutFeedback.

24. What is the role of fast refresh in React Native?

Sample answer:

Fast refresh allows developers to get near-instant feedback on recent changes in their app. Once 'Enable fast refresh' in the developer menu is toggled, any new edits in the program become visible within a few seconds for an easy evaluation.

25. How do you ensure animations run smoothly in React Native apps?

Sample answer:

Several steps can be taken to optimize animations in React Native. These include:

- **Use lazy loading** so components are only rendered when in use
- **Remove animated values from the state** to avoid unnecessary overhead
- **Use shouldComponentUpdate** to fast-track the rendering process
- **Use useNativeDriver for Android** to transfer all the animating work to the native layer

26. What steps would you take in React Native if you have an app that crashes continually?

Sample answer:

Use a third-party error reporting integration to pull up an error report and further diagnose the bug. These plugins help collect, organize, and analyze crash reports and also provide quick fixes so the app can get back up and running. Popular error reporting plugins include:

- Bugsnag
- Crashlytics
- Sentry
- TestFairy
- Rollbar

27. What is the role of TouchableOpacity in React Native?

Sample answer:

In React Native, TouchableOpacity is a wrapper used to change the transparency of a button. When used on a button, opacity reduces in response to touch, allowing users to see the background whenever they press it.

28. How can you optimize FlatList items in React Native?

Sample answer:

There are several techniques for optimizing the performance of FlatList items. For example, we can:

- Avoid using 1080P HD images
- Optimize the maxToRenderPerBatch prop
- Use the getItemLayout prop
- Use the keyExtractor prop
- Use fewer views
- Optimize the windowSize prop

28. How are hot reloading and live reloading in React Native different?

Sample answer:

Live reloading in React Native refreshes the entire app when a file changes, whereas hot reloading only refreshes the files that were changed.

When hot reloading is used on an app, the state remains the same and the developer is returned to the page they started on. The opposite is true for live reloading.

29. When would you use ScrollView over FlatList and vice versa?

Sample answer:

ScrollView loads all data items on one screen for scrolling purposes. All the data is stored on RAM, which can cause performance issues for large amounts of data.

FlatList only displays items that are currently shown on the screen (10 by default), thus sidestepping any performance issues.

Therefore, it is best to use FlatList for large datasets, whereas ScrollView can be used for smaller datasets.

30. How are vector icons used in React Native?

Sample answer:

Vectors are an integral part of app design in React Native. They can initially be installed by running the following command: `<npm install react-native-vector-icons>`. Selected vector icons can then be imported into the React Native app for use.

31. When should setNativeProps be used in React Native?

Sample answer:

In React Native, setNativeProps is used to change a component directly on a DOM node rather than in the app's state. While this can help to solve issues, it also makes code more difficult to work with in the future.

It's considered a backup solution to performance issues linked to animations, and should only be used if setState and shouldComponent fail to resolve the problem.

32. How do you handle element size in React Native?

React native follows the box-model concept of CSS. The size of the element is calculated based on the size of content, padding, border, margin. The simplest way to set the size of an element is to set width and height CSS property for an element. All dimensions in React Native is unitless and represent density-independent pixels. By setting fixed height and width, the element will look

exactly the same size on different screen sizes. But there is an instance where you want to give the width and height of an element in percentage.

Directly use of percentage is not supported in React native but React native does give a dimension module which can be used to give width in percentage. Dimension module gives the width and height of the mobile device. This information can be used to set the style of an element in runtime.

Below is the example of how to use Dimension module from React native:

Importing module from React Native:

```
import { Dimension } from 'react-native';
```

Figure out width and height of the device:

```
const deviceWidth = Dimension.get("window").width;
```

```
const deviceHeight = Dimension.get("window").height;
```

Calculate the style value:

```
Width: deviceWidth*<percentageValue>/100
```

But the simplest way is by setting width and height CSS for an element.

33. Give a quick overview of the life cycle of a React component?

Below image summarise the lifecycle methods exposed by react component and the order in which lifecycle methods are being called by React.

life cycle of a React component

React 16 was a major release for ReactJS. With this version react has marked some of the lifecycle methods like `componentWillReceiveProps`, `ComponentWillUpdate` as deprecated. In a later release of React, the framework will be removing these methods. In place of deprecated methods, they added a few new lifecycle methods. The above picture shows the lifecycle method of a component. We can divide the lifecycle methods of the component into three categories.

- **Mounting:** This includes the beginning phase of the component. The first constructor of the component gets invoked. Constructor receives props as an argument from the parent component. Constructor is the place where we define an initial state of a component. After constructor and before render, lifecycle method `getDerivedStateFromProps` is invoked, it returns the state object or null to avoid the render. Then Render method is invoked which renders the JSX. Then at the end, `componentDidMount` is invoked.
- **Updating:** A component gets an update via a change in state or change in props. In both of these cases, the `getDerivedStateFromProps` method is invoked first where we get props and state as an argument and returns updated state. Then `shouldComponentUpdate` methods are invoked. If this method returns true then the only component gets the update otherwise component update is aborted. Then render method is called and after this `getSnapshotBeforeUpdate` is called. In the end, the `componentDidUpdate` method is called

- **Unmounting:** `componentWillUnmount` lifecycle method is invoked before the unmounting of the component takes place

34. List the users of React Native?

Today, thousands of React Native built-in apps are available in the market. Here is the list of users who uses React Native apps:

- Facebook
- Facebook Ads Manager
- Instagram
- F8
- Airbnb
- Skype
- Tesla
- Bloomberg
- Gyroscope
- Myntra
- UberEats

35. For what XHR Module is used in the React Native?

In React Native, the XHR module is used to implement the **XMLHttpRequest**. It is an object for interacting with remote services. This object consists of two parts, front-end and back-end, where the front-end allows interacting within JavaScript. It sends the request to the XHR back-end, which is responsible for a processing network request. The back-end part is called Networking.

36. Is React Native a Native mobile app?

Yes, React Native is a native mobile app, which compiles a native mobile app using native app components. It is neither a Hybrid mobile app that uses WebView to run the HTML5 app nor a mobile web app. The React Native framework builds a real mobile app, which is indistinguishable from an app built using Objective-C/Swift or Java.

37. What is meant by **InteractionManager**, and why it is Important?

The **InteractionManager** is a native module in React Native, which is responsible for differing the execution of a function until an interaction has finished. To handle this deferral, we need to call **InteractionManager.runAfterInteractions(() => {...})**.

The **InteractionManager** is important because React Native has **two threads**. One is JavaScript UI thread, which handles drawing updates to the screen, and the second thread used for all task, not on the UI thread. Since React Native has only one thread for making UI updates, it can get overloaded and drop frames, especially in navigation screen animations. So, developers use the **InteractionManager** to ensure that the function is executed after these animations occur. As a result, we do not drop frames on the UI thread.

38. What is **ListView**?

ListView is a core component of React Native, which contains a list of items and displays in vertical scrollable lists.

39. What is the storage system in the React Native?

React Native storage is a simple, unencrypted, asynchronous, persistent system, which stores the data globally in the app. It stores data in the form of a **key-value** pair. React Native provides **AsyncStorage** class to store data globally. Using the **AsyncStorage** class, we need to have a data backup and synchronization class. It is because data saved on the device is not permanent and not encrypted.

40. Why do React Native use **Redux**?

Redux is a state container for JavaScript applications. It is a state management tool, which helps you to write applications that behave consistently, can run in a different environment, and are easy to test.

React Native uses **Redux** because it allows developers to use one application state as a global state and interact easily with the state from any React component. It can combine with any framework or library

41. What are the advantages of using React Native?

Sample answer:

Since its launch in 2015, React Native has built a reputation as a reliable and effective JavaScript framework. Some of its key strengths include:

- **Cross-platform compatibility:** Most of the code is cross-platform, meaning developers only have to create one app rather than two separate apps for both iOS and Android
- **Real-time feedback:** React Native offers a 'hot reloading' feature where developers can immediately view the changes they've made in a separate preview window
- **Flexible user interface:** React Native's interface is slick and makes it easy for multiple developers to work on a project together
- **Third-party plugins:** React Native is compatible with many third-party plugins that can be used to support and improve the app development process
- **Community:** As a popular open-source framework, React Native has a large community of developers that exchange knowledge

42. How do native apps differ from hybrid apps?

Hybrid apps are developed to be used across all platforms, whereas native apps are developed for a particular platform. React Native is used for the development of hybrid apps.

While hybrid apps are faster to develop and typically require less maintenance than native apps, they may perform slightly worse than their native counterparts.

43. What coding languages are compatible with React Native?

Sample answer:

While React Native is generally used with JavaScript, compatibility with other coding languages, including Python, C++, and C, is also possible through the framework's Java Native Interface (JNI).

44. What is the role of the bridge in React Native?

Sample answer:

The bridge acts as a transport layer between JavaScript and Native modules. In the rendering process:

- The bridge first receives the user response to open the app from the Native module
- It then passes the serialized payload to the JavaScript module
- Once the event has been processed and a virtual DOM has been generated in the JavaScript module, the bridge receives the serialized batched response
- The bridge passes the serialized batched response to the Native module for final rendering

45. What is the interaction manager in React Native?**Sample answer:**

In React Native, the interaction manager is used to defer the execution of a function until a specified 'interaction' has been completed.

This is important as React Native is single-threaded in nature, meaning queued animations on the UI can become congested. The interaction manager helps with this issue and ensures that animations run smoothly in a scheduled manner.

46. What is the role of fabric in React Native?**Sample answer:**

Fabric is a modern type of architecture that was first created in 2018 and aims to address some of React Native's performance issues. Fabric modernizes the framework's rendering layer by allowing specified priority tasks to be executed synchronously and, therefore, quicker.

47. How do you create basic text input in React Native?**Sample answer:**

The insertion of basic text in React Native apps is handled by the Text and TextInput components. TextInput allows users to type on the app. We can implement it using the following syntax: `<import { Text, TextInput, View } from 'react-native'>`.

48. What is the role of timers in a React Native app?

Sample answer:

In React Native, timers allow developers to manipulate the order in which events in a program occur. There are four different types of timers, each one serving a different purpose:

1. **Timeout** implements a delay
2. **Interval** allows repeat actions to occur at given intervals
3. **Immediate** allows actions to occur as soon as possible
4. **Animation** allows animations to display when the program is ready to render frames

49. How can sensitive data be stored securely in React Native?

Sample answer:

Most React Native data is stored in Async Storage. As an unencrypted, local form of storage, it's not suitable for storing sensitive data such as tokens and passwords.

Alternatively, React Native Keychain offers a secure form of storage that also works similarly to Async Storage. For iOS, Keychain storage can be used to protect sensitive data, while Android developers can use Facebook Conceal and Android Keystore.

50. What are some differences between using React Native for iOS and for Android?

Sample answer:

Around 85% of React Native code is cross-platform, which means that most processes are the same on both iOS and Android. However, there are a few minor differences. These include:

- iOS development uses Mac and Xcode, whereas Android development draws on Android SDK and an emulator
- Third-party plugins that don't offer native functionality will need to be used differently
- The bridging process can be slightly different when developing complex apps

51. How do you ensure animations run smoothly in React Native apps?

Sample answer:

Several steps can be taken to optimize animations in React Native. These include:

- **Use lazy loading** so components are only rendered when in use
- **Remove animated values from the state** to avoid unnecessary overhead
- **Use shouldComponentUpdate** to fast-track the rendering process
- **Use useNativeDriver for Android** to transfer all the animating work to the native layer

52. What steps would you take in React Native if you have an app that crashes continually?

Sample answer:

Use a third-party error reporting integration to pull up an error report and further diagnose the bug. These plugins help collect, organize, and analyze crash reports and also provide quick fixes so the app can get back up and running. Popular error reporting plugins include:

- Bugsnag
- Crashlytics
- Sentry
- TestFairy
- Rollbar

53. What is the role of TouchableOpacity in React Native?

Sample answer:

In React Native, TouchableOpacity is a wrapper used to change the transparency of a button. When used on a button, opacity reduces in response to touch, allowing users to see the background whenever they press it.

54. What are the differences between Flexbox in browser and in React Native?

Sample answer:

Flexbox generally works in the same way in React Native as it does in CSS on the web. However, there are a few minor differences in the values. These include:

- The default value in React Native is column, whereas the default value for CSS is row
- The default value in React Native is flex-start, whereas the default value for CSS is stretch
- The default value in React Native is 0, whereas the default value for CSS is 1

55. What steps can you take to resolve persistent memory leak issues?

Sample answer:

A memory leak occurs when memory that is no longer needed by an app remains in the app rather than being returned to the operating system. This is one of the most common causes of performance issues.

In theory, memory management is handled automatically by the garbage collector. However, this process is still prone to errors. Debugging tools can be used to identify memory leak issues. Some of the most common causes for issues are:

- Timers and listeners in componentDidMount
- Inline styles
- Closure scope leaks
- The use of console.log

Debugging can identify the root of the memory leak; once it has been removed, the issue should be resolved

56. How are hot reloading and live reloading in React Native different?

Sample answer:

Live reloading in React Native refreshes the entire app when a file changes, whereas hot reloading only refreshes the files that were changed.

When hot reloading is used on an app, the state remains the same and the developer is returned to the page they started on. The opposite is true for live reloading.

57. When would you use ScrollView over FlatList and vice versa?

Sample answer:

ScrollView loads all data items on one screen for scrolling purposes. All the data is stored on RAM, which can cause performance issues for large amounts of data.

FlatList only displays items that are currently shown on the screen (10 by default), thus sidestepping any performance issues.

58. How are vector icons used in React Native?

Sample answer:

Vectors are an integral part of app design in React Native. They can initially be installed by running the following command: `<npm install react-native-vector-icons>`. Selected vector icons can then be imported into the React Native app for use.

59. When should setNativeProps be used in React Native?

Sample answer:

In React Native, `setNativeProps` is used to change a component directly on a DOM node rather than in the app's state. While this can help to solve issues, it also makes code more difficult to work with in the future.

It's considered a backup solution to performance issues linked to animations, and should only be used if `setState` and `shouldComponent` fail to resolve the problem.

60. List the steps to avoid Props drilling.

The given steps can be used to avoid props drilling:

- React Context API
- Composition
- Render props
- HOC
- MobX or Redux

