React Native Interview Questions

interviewbit.com/react-native-interview-questions

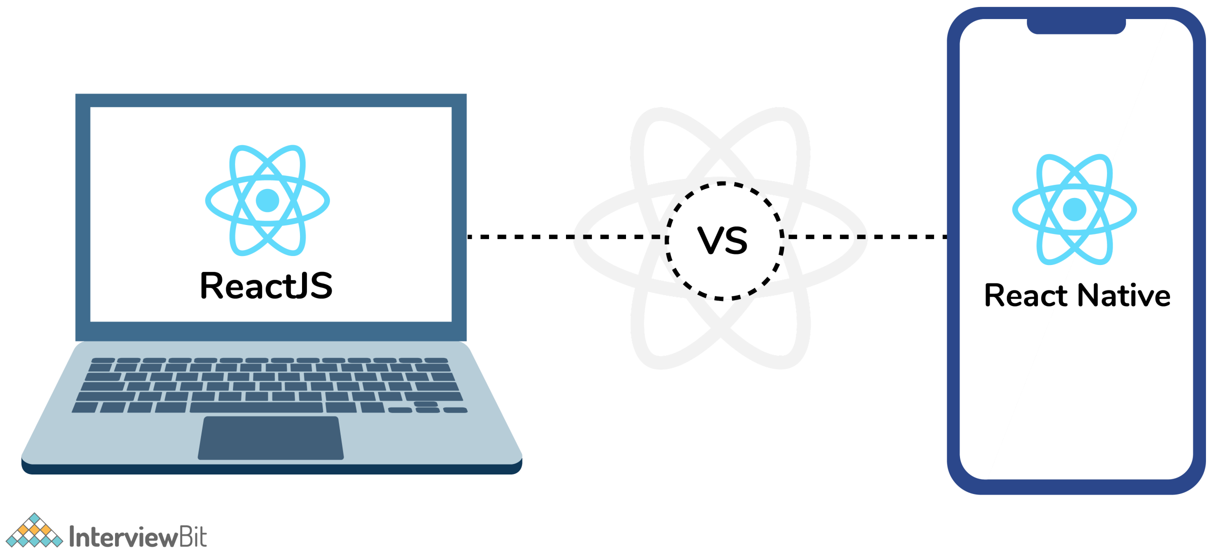
React Native is a JavaScript-based mobile application framework, designed to create mobile applications for iOS and Android by providing coders a tool to use React along with the native mobile platform. The major advantage of React Native is that code can be once written and shared between both IOS and Android. Mobile applications that feel truly "native" in terms of both look and feel can be built with Javascript itself. Learn More.

Crack your next tech interview with confidence!

React Native Basic Interview Questions

1. How Different is React-native from ReactJS ?

- **Usage Scope**
  ReactJs - React is a JavaScript library for building Responsive User Interfaces for Building Web Application.
  React Native - It is a framework for creating mobile applications with a native feel.
- **Syntax**
  Both React and React Native uses JSX (JavaScript XML) syntax but React uses html tags like <div> <h1> <p> etc while React Native uses <view> <text> etc.
- **Animation And Gestures**
  React uses CSS animations on a major scale to achieve animations for a web page while The recommended way to animate a component is to use the Animated API provided by React-Native.
- **Routing Mechanism**
  React uses a react-router for routing and does not have any inbuilt routing capabilities but React Native has a built-in Navigator library for navigating mobile applications.



REACT JS	REACT NATIVE
It is used for developing web applications.	It is used for developing mobile applications.
It uses React-router for navigating web pages.	It has a built-in navigator library for navigating mobile applications.
It uses HTML tags.	It does not use HTML tags.
It provides high security.	It provides low security in comparison to ReactJS.
In this, the virtual DOM renders the browser code.	In this, Native uses its API to render code for mobile applications.

2. What is Flexbox and describe any elaborate on its most used properties?

It is a layout model that allows elements to align and distribute space within a container. With Flexbox when Using flexible widths and heights, all the inside the main container can be aligned to fill a space or distribute space between elements, which makes it a great tool to use for responsive design systems.

Property	Values	Description
flexDirection	‘column’,'row'	Used to specify if elements will be aligned vertically or horizontally
justifyContent	‘center’,'flex-start','flex-end','space-around','space-between'	Used to determine how should elements be distributed inside the container
alignItems	‘center’,'flex-start','flex-end','stretched'	Used to determine how should elements be distributed inside the container along the secondary axis (opposite of flexDirection)

3. Describe advantages of using React Native?

There are multiple advantage of using React Native like,

- **Large Community**
  React Native is an Open Source Framework, it is completely community driven so any challenges can be resolved by getting online help from other developers.

- **Reusability**
  Code can be written once and can be used for both IOS and ANDROID, which helps in maintaining and as well debugging large complex applications as no separate teams are needed for supporting both the platforms, this also reduces the cost to a major extent.

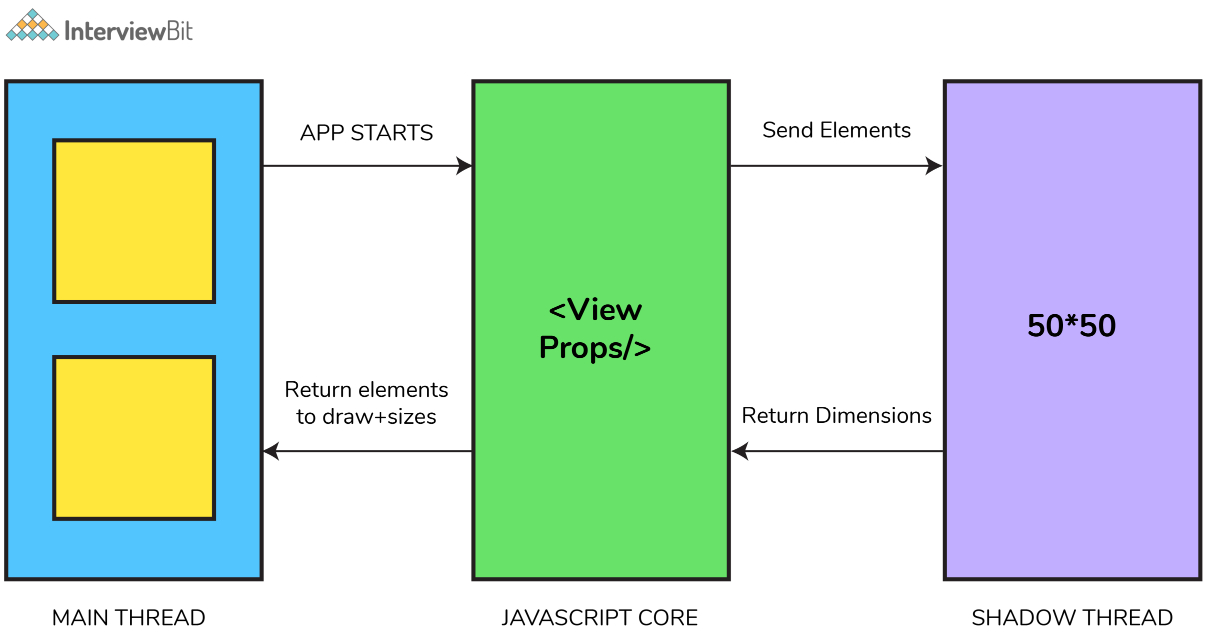
- **Live and Hot Reloading**
  Live reloading reloads or refreshes the entire app when a file changes. For example, if you were four links deep into your navigation and saved a change, live reloading would restart the app and load the app back to the initial route.
  Hot reloading only refreshes the files that were changed without losing the state of the app. For example, if you were four links deep into your navigation and saved a change to some styling, the state would not change, but the new styles would appear on the page without having to navigate back to the page you are on because you would still be on the same page.

- **Additional Third-Party Plugins**
  If the existing modules do not meet the business requirement in React Native we can also use Third Party plugins which may help to speed up the development process.

You can download a PDF version of React Native Interview Questions.

Download PDF

4. What are threads in General ? and explain Different Threads in ReactNative with Use of Each ?



The single sequential flow of control within a program can be controlled by a thread.

React Native right now uses 3 threads:

- **MAIN/UI Thread** — This is the main application thread on which your Android/iOS app is running. The UI of the application can be changed by the Main thread and it has access to it .
- **Shadow Thread** — layout created using React library in React Native can be calculated by this and it is a background thread.
- **JavaScript Thread** — The main Javascript code is executed by this thread.

5. Are default props available in React Native and if yes for what are they used and how are they used ?

Yes, default props available in React Native as they are for React, If for an instance we do not pass props value, the component will use the default props value.

import React, {Component} from 'react';

```
import {View, Text} from 'react-native';
class DefaultPropComponent extends Component {
   render() {
       return (
           <View>
             <Text>
              {this.props.name}
            </Text>
          </View>
       )
   }
}
Demo.defaultProps = {
   name: 'BOB'
}

export default DefaultPropComponent;
```

6. How is user Input Handled in React Native ?

TextInput is a Core Component that allows the user to enter text. It has an onChangeText prop that takes a function to be called every time the text changes, and an onSubmitEditing prop that takes a function to be called when the text is submitted.

```
import React, { useState } from 'react';
import { Text, TextInput, View } from 'react-native';

const PizzaTranslator = () => {
 const [text, setText] = useState('');
 return (
   <View style={{padding: 10}}>
     <TextInput
       style={{height: 40}}
       placeholder="Type here to translate!"
       onChangeText={text => setText(text)}
       defaultValue={text}
     />
     <Text style={{padding: 10, fontSize: 42}}>
       {text.split(' ').map((word) => word && '🍕').join(' ')}
     </Text>
   </View>
 );
}

export default PizzaTranslator;
```

7. What is State and how is it used in React Native?

It is used to control the components. The variable data can be stored in the state. It is mutable means a state can change the value at any time.

```
import React, {Component} from 'react';
import { Text, View } from 'react-native';
export default class App extends Component {
 state = {
myState: 'State of Text Component'
              }
updateState = () => this.setState({myState: 'The state is updated'})
render() {
return (
<View>
<Text onPress={this.updateState}> {this.state.myState} </Text>
</View>
); } }
```

Here we create a Text component with state data. The content of the Text component will be updated whenever we click on it. The state is updated by event onPress .

8. What is Redux in React Native and give important components of Redux used in React Native app ?

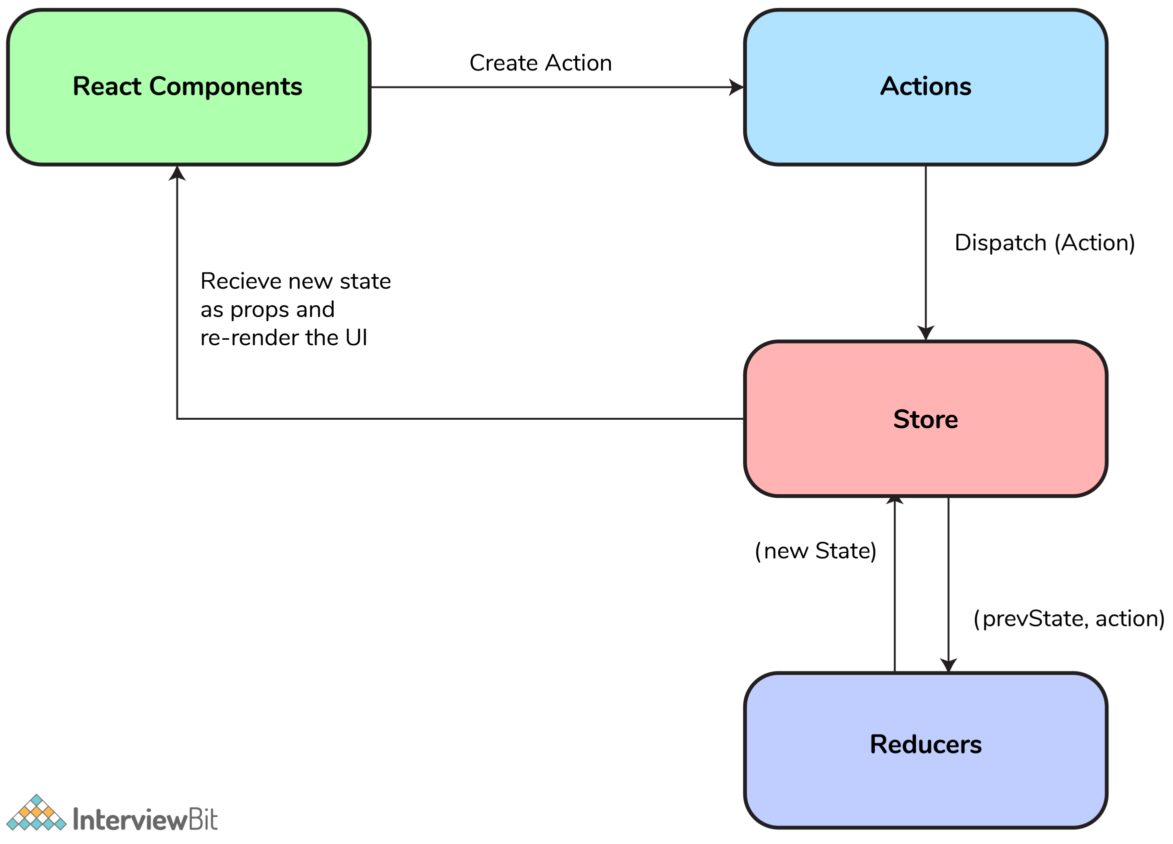
Redux is a predictable state container for JavaScript apps. It helps write applications that run in different environments. This means the entire data flow of the app is handled within a single container while persisting previous state.

Actions: are payloads of information that send data from your application to your store. They are the only source of information for the store. This means if any state change is necessary the change required will be dispatched through the actions.

Reducers: “Actions describe the fact that something happened, but don’t specify how the application’s state changes in response. This is the job of reducers.” when an action is dispatched for state change its the reducers duty to make the necessary changes to the state and return the new state of the application.

Store: a store can be created with help of reducers which holds the entire state of the application. The recommended way is to use a single store for the entire application rather than having multiple stores which will violate the use of redux which only has a single store.

Components: this is where the UI of the application is kept.



9. Describe Timers in React Native Application ?

Timers are an important and integral part of any application and React Native implements the browser timers.

Timers

setTimeout, clearTimeout

There may be business requirements to execute a certain piece of code after waiting for some time duration or after a delay setTimeout can be used in such cases, clearTimeout is simply used to clear the timer that is started.

```
setTimeout(() => {
yourFunction();
}, 3000);
```

setInterval, clearInterval

setInterval is a method that calls a function or runs some code after specific intervals of time, as specified through the second parameter.

```
setInterval(() => {
console.log('Interval triggered');
}, 1000);
```

A function or block of code that is bound to an interval executes until it is stopped. To stop an interval, we can use the clearInterval() method.

setImmediate, clearImmediate

Calling the function or execution as soon as possible.

```
var immediateID = setImmediate(function);
// The below code displays the alert dialog immediately.
var immediateId = setImmediate(
    () => {    alert('Immediate Alert');
}
```

clearImmediate is used for Canceling the immediate actions that were set by setImmediate().

requestAnimationFrame, cancelAnimationFrame

It is the standard way to perform animations.

Calling a function to update an animation before the next animation frame.

```
var requestID = requestAnimationFrame(function);
// The following code performs the animation.
var requestId = requestAnimationFrame(
    () => { // animate something}
)
```

cancelAnimationFrame is used for Canceling the function that was set by requestAnimationFrame().

10. How to debug React Native Applications and Name the Tools used for it ?

In the React Native world, debugging may be done in different ways and with different tools, since React Native is composed of different environments (iOS and Android), which means there's an assortment of problems and a variety of tools needed for debugging.

The Developer Menu:

Reload: reloads the app
Debug JS Remotely: opens a channel to a JavaScript debugger
Enable Live Reload: makes the app reload automatically on clicking Save
Enable Hot Reloading: watches for changes accrued in a changed file
Toggle Inspector: toggles an inspector interface, which allows us to inspect any UI element on the screen and its properties, and presents an interface that has other tabs like networking, which shows us the HTTP calls, and a tab for performance.

- **Chrome's DevTools:**

  Chrome is possibly the first tool to think of for debugging React Native. It's common to use Chrome's DevTools to debug web apps, but we can also use them to debug React Native since it's powered by JavaScript.To use Chrome's DevTools with React Native, first make sure to connect to the same Wi-Fi, then press command + R if you're using macOS, or Ctrl + M on Windows/Linux. In the developer menu, choose Debug Js Remotely. This will open the default JS debugger.

- **React Developer Tools**
  For Debugging React Native using React's Developer Tools, you need to use the desktop app. In can installed it globally or locally in your project by just running the following command:
  `yarn add react-devtools`
  Or npm:
  `npm install react-devtools --save`

  React's Developer Tools may be the best tool for debugging React Native for these two reasons:
  It allows for debugging React components.
  There is also a possibility to debug styles in React Native. There is also a new version that comes with this feature that also works with the inspector in the developer menu. Previously, it was a problem to write styles and have to wait for the app to reload to see the changes. Now we can debug and implement style properties and see the effect of the change instantly without reloading the app.

- **React Native Debugger**
  While using Redux in your React Native app, React Native Debugger is probably the right debugger for you. This is a standalone desktop app that works on macOS, Windows, and Linux. It even integrates both Redux's DevTools and React's Developer Tools in one app so you don't have to work with two separate apps for debugging.

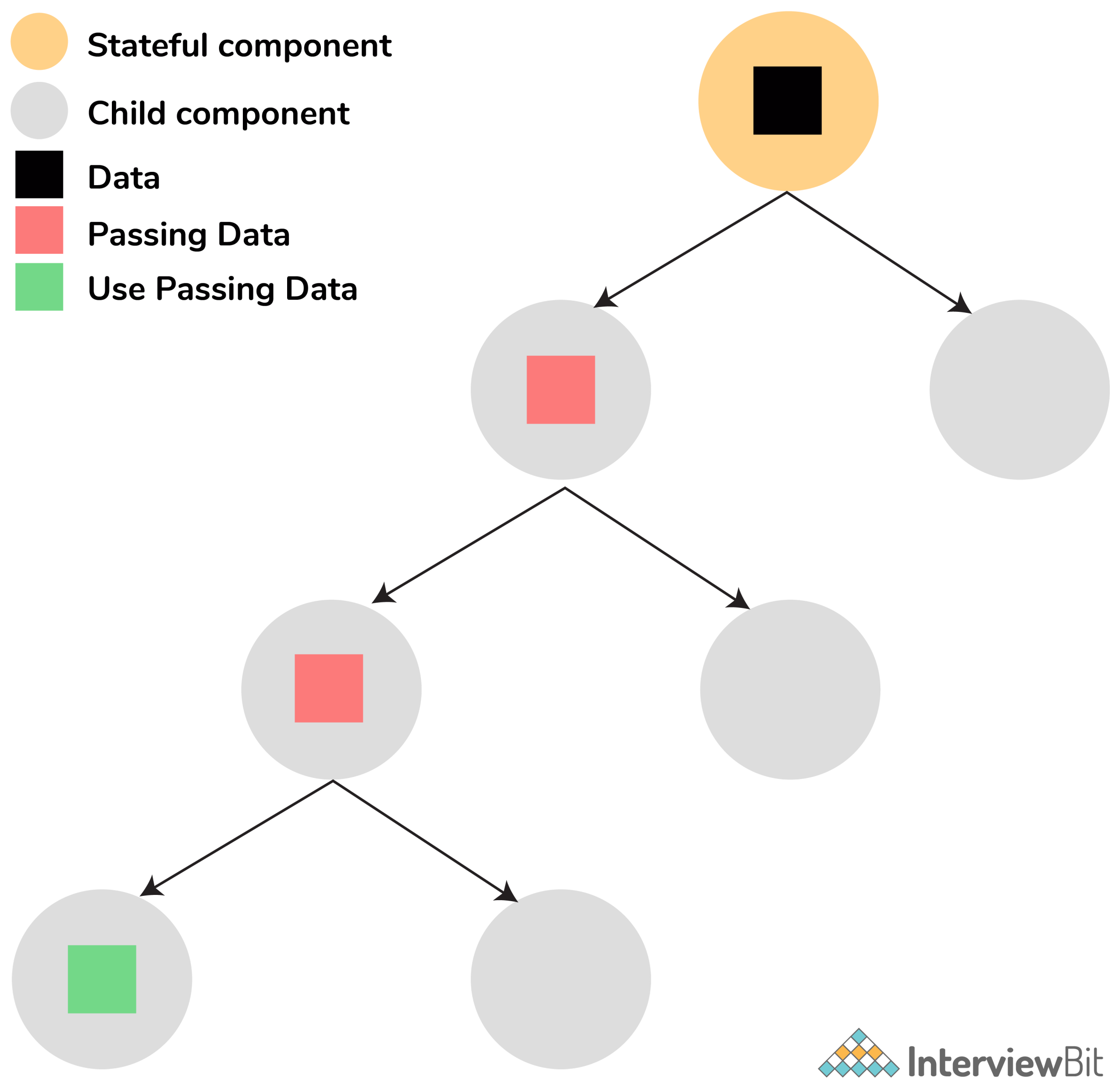
React Native CLI

You can use the React Native CLI to do some debugging as well. It can also be used for showing the logs of the app. Hitting react-native log-android will show you the logs of db logcat on Android, and to view the logs in iOS you can run react-native log-ios, and with console.log you can dispatch logs to the terminal:

```
console.log("some error🛑")
```

11. What is Props Drilling and how can we avoid it ?

Props Drilling (Threading) is a concept that refers to the process you pass the data from the parent component to the exact child Component BUT in between, other components owning the props just to pass it down the chain.



Steps to avoid it

1. React Context API.
2. Composition
3. Render props
4. HOC
5. Redux or MobX

12. Describing Networking in React Native and how to make AJAX network calls in React Native?

React Native provides the Fetch API for networking needs.
To fetch content from an arbitrary URL, we can pass the URL to fetch:

```
fetch('https://mywebsite.com/endpoint/', {
 method: 'POST',
 headers: {
   Accept: 'application/json',
   'Content-Type': 'application/json'
 },
 body: JSON.stringify({
   firstParam: 'yourValue',
   secondParam: 'yourOtherValue'
 })
});
```

Networking is an inherently asynchronous operation. Fetch methods will return a Promise that makes it straightforward to write code that works in an asynchronous manner:

```
const getMoviesFromApi = () => {
 return fetch('https://reactnative.dev/movies.json')
   .then((response) => response.json())
   .then((json) => {
     return json.movies;
   })
   .catch((error) => {
     console.error(error);
   });
};
```

The XMLHttpRequest API is built in to React Native Since frisbee and Axios use XMLHttpRequest we can even use these libraries.

```
var request = new XMLHttpRequest();
request.onreadystatechange = (e) => {
 if (request.readyState !== 4) {
   return;
 }

 if (request.status === 200) {
   console.log('success', request.responseText);
 } else {
   console.warn('error');
 }
};

request.open('GET', 'https://mywebsite.com/endpoint/');
request.send();
```

13. List down Key Points to integrate React Native in an existing Android mobile application

Primary points to note to integrating React Native components into your Android application are to:

- Set up React Native dependencies and directory structure.
- Develop your React Native components in JavaScript.
- Add a ReactRootView to your Android app. This view will serve as the container for your React Native component.
- Start the React Native server and run your native application.
- Lastly, we need to Verify that the React Native aspect of your application works as expected.

React Native Intermediate Interview Questions

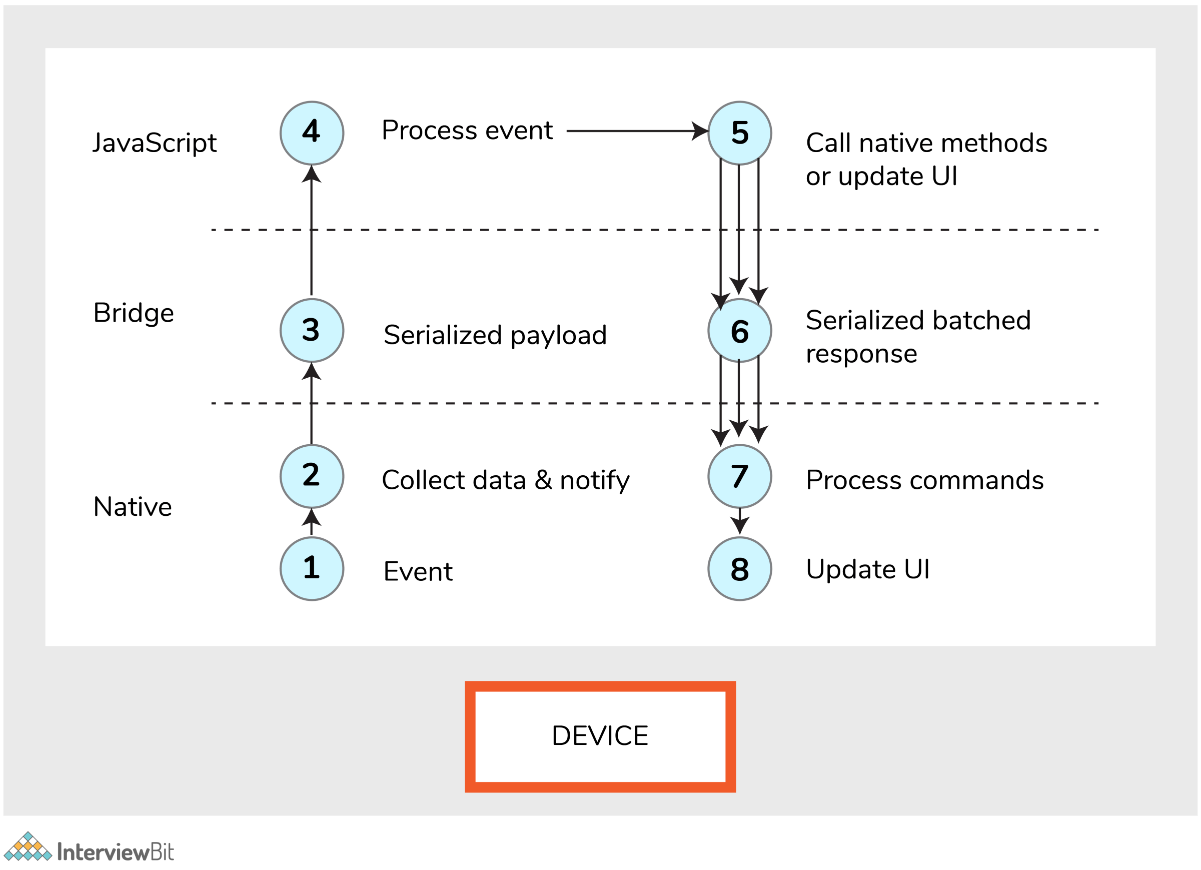
14. How is the entire React Native code processed to show the final output on a mobile screen

- At the first start of the app, the main thread starts execution and starts loading JS bundles.
- When JavaScript code has been loaded successfully, the main thread sends it to another JS thread because when JS does some heavy calculations stuff the thread for a while, the UI thread will not suffer at all times.
- When React starts rendering, Reconciler starts “diffing”, and when it generates a new virtual DOM(layout) it sends changes to another thread(Shadow thread).
- Shadow thread calculates layout and then sends layout parameters/objects to the main(UI) thread. (Here you may wonder why we call it “shadow”? It’s because it generates shadow nodes)
- Since only the main thread is able to render something on the screen, the shadow thread should send the generated layout to the main thread, and only then UI renders.

15. What is a bridge and why is it used in React Native ? Explain for both android and IOS ?

Bridge in ReactNative is a layer or simply a connection that is responsible for gluing together Native and JavaScript environments.

Consider Below diagram:

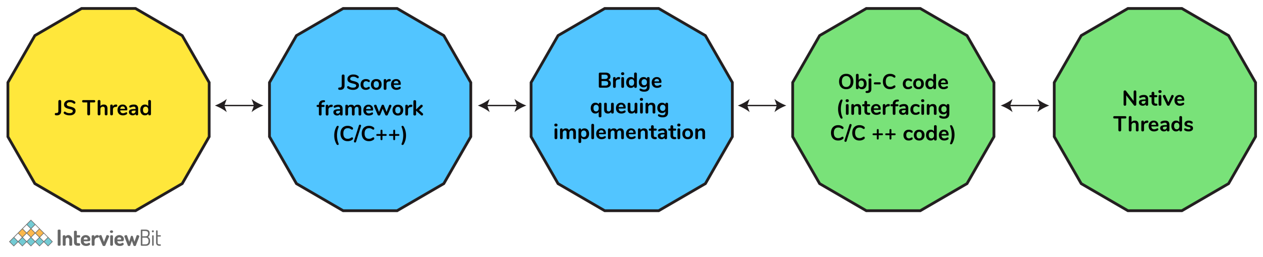


- The layer which is closest to the device on which the application runs is the Native Layer.

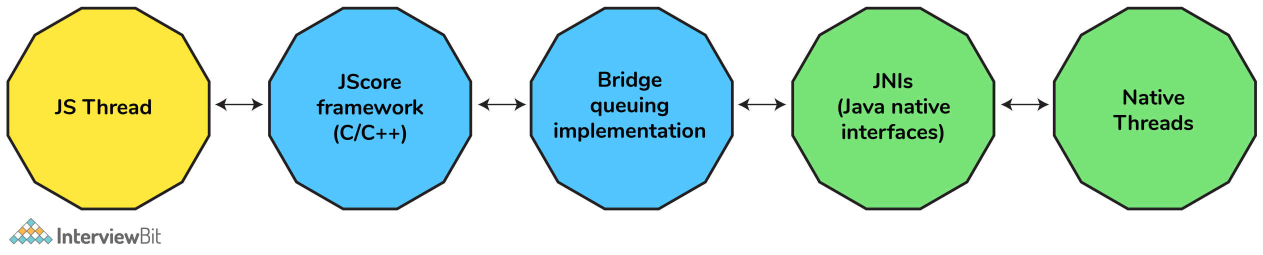
- The bridge is basically a transport layer which acts as a connection between Javascript and Native modules, it does the work of transporting asynchronous serialized batched response messages from JavaScript to Native modules.

  Now for an example, there is some state change that happens, because of which React Native will batch Update UI and send it to the Bridge. The bridge will pass this Serialized batched response to the Native layer, which will process all commands that it can distinguish from a serialized batched response and will update the User Interface accordingly.

  <u>IOS Platform:</u>


JS Thread
JScore framework (C/C++)
Bridge queuing implementation
Obj-C code (interfacing C/C ++ code)
Native Threads
InterviewBit

Android Platform:



16. Name core Components in React Native and the analogy of those components when compared with the web .

The core components used in React Native are <View> , <Text> , <Image> , <ScrollView> , <TextInput>

And analogy when compared Web can be explained by below diagram:

REACT NATIVE UI COMPONENT	ANDROID VIEW	IOS VIEW	WEB ANALOG	DESCRIPTION
`<View>`	`<ViewGroup>`	`<UIView>`	A non-scrolling `<div>`	A container that supports layout with flexbox style, some touch handling, and accessibility controls.
`<Text>`	`<TextView>`	`<UITextView>`	`<p>`	Displays, styles, and nests strings of text and even handles touch events.
`<Image>`	`<ImageView>`	`<UIImageView>`	`<img>`	Displays different types of images
`<ScrollView>`	`<ScrollView>`	`<UIScrollView>`	`<div>`	A generic scrolling container that can contain multiple components and views.
`<TextInput>`	`<EditText>`	`<UITextField>`	`<input type="text">`	Allows the user to enter text

17. What is ListView and describe its use in React Native ?

React Native ListView is a view component that contains the list of items and displays it in a vertically scrollable list.

```
export default class MyListComponent extends Component {
constructor() {
super();
const ds = new ListView.DataSource({rowHasChanged: (r1, r2) => r1 !== r2});
this.state = {
dataSource: ds.cloneWithRows(['Android','iOS', 'Java','Php', 'Hadoop', 'Sap',
'Python','Ajax', 'C++']),
};
}
render() {
return (
<ListView
dataSource={this.state.dataSource}
renderRow={
(rowData) =>
<Text style={{fontSize: 30}}>{rowData}</Text>} />
); }
}
```

18. How can you write different code for IOS and Android in the same code base ? Is there any module available for this ?

The platform module detects the platform in which the app is running.

```
import { Platform, Stylesheet } from 'react-native';
const styles = Stylesheet.create({
height: Platform.OS === 'IOS' ? 200 : 400
})
```

Additionally Platform.select method available that takes an object containing Platform.OS as keys and returns the value for the platform you are currently on.

```
import { Platform, StyleSheet } from 'react-native';
const styles = StyleSheet.create({
 container: {
flex: 1,
   ...Platform.select({
     ios: {
       backgroundColor: 'red',
     },
     android: {
       backgroundColor: 'green',
     },
     default: {
       // other platforms, web for example
       backgroundColor: 'blue',
     },    }),
},
});
```

19. What are Touchable components in react Native and which one to use

when ?

Tapping gestures can be captured by Touchable components and can display feedback when a gesture is recognized.

Depending on what kind of feedback you want to provide we choose Touchable Components.

Generally, we use TouchableHighlight anywhere you would use a button or link on the web. The background of the view will be darkened when the user presses down on the button.

We can use TouchableNativeFeedback on Android to display ink surface reaction ripples that respond to the user's touch.

TouchableOpacity can be used to provide feedback by reducing the opacity of the button, allowing the background to be seen through while the user is pressing down.

If we need to handle a tap gesture but you don't want any feedback to be displayed, use TouchableWithoutFeedback.

```
import React, { Component } from 'react';
import { Platform, StyleSheet, Text, TouchableHighlight, TouchableOpacity,
TouchableNativeFeedback, TouchableWithoutFeedback, View } from 'react-native';
export default class Touchables extends Component {
_onPressButton() {
   alert('You tapped the button!')  }
 _onLongPressButton() {
   alert('You long-pressed the button!')
 }
render() {
return (
<View style={styles.container}>
<TouchableHighlight onPress={this._onPressButton} underlayColor="white">
<View style={styles.button}>
<Text style={styles.buttonText}>TouchableHighlight</Text>
</View>
</TouchableHighlight>
);}
}
```

20. Explain FlatList components, what are its key features along with a code sample ?

The FlatList component displays similarly structured data in a scrollable list. It works well for large lists of data where the number of list items might change over time.

Key Feature:

The FlatList shows only those rendered elements which are currently displaying on the screen, not all the elements of the list at once.

```
import React, { Component } from 'react';
import { AppRegistry, FlatList,
   StyleSheet, Text, View,Alert } from 'react-native';

export default class FlatListBasics extends Component {

   renderSeparator = () => {
       return (
           <View
               style={{
                   height: 1,
                   width: "100%",
                   backgroundColor: "#000",
               }}
           />
       );
   };
   //handling onPress action
   getListViewItem = (item) => {
       Alert.alert(item.key);
   }

   render() {
       return (
           <View style={styles.container}>
               <FlatList
                   data={[
                       {key: 'Android'},{key: 'iOS'}, {key: 'Java'},{key:
'Swift'},
                       {key: 'Php'},{key: 'Hadoop'},{key: 'Sap'},
                   ]}
                   renderItem={({item}) =>
                       <Text style={styles.item}
                             onPress={this.getListViewItem.bind(this, item)}>
{item.key}</Text>}
                   ItemSeparatorComponent={this.renderSeparator}
               />
           </View>
       );
   }
}
AppRegistry.registerComponent('AwesomeProject', () => FlatListBasics);
```

21. How To Use Routing with React Navigation in React Native ?

One of the popular libraries for routing and navigation in a React Native application is React Navigation.

This library helps solve the problem of navigating between multiple screens and sharing data between them.

```
import * as React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createStackNavigator } from '@react-navigation/stack';

const Stack = createStackNavigator();

const MyStack = () => {
 return (
   <NavigationContainer>
     <Stack.Navigator>
       <Stack.Screen
         name="Home"
         component={HomeScreen}
         options={{ title: 'Welcome' }}
       />
       <Stack.Screen name="Profile" component={ProfileScreen} />
     </Stack.Navigator>
   </NavigationContainer>
 );
};
```

22. What are the Different Ways to style React Native Application ?

React Native give us two powerful ways by default to style our application :

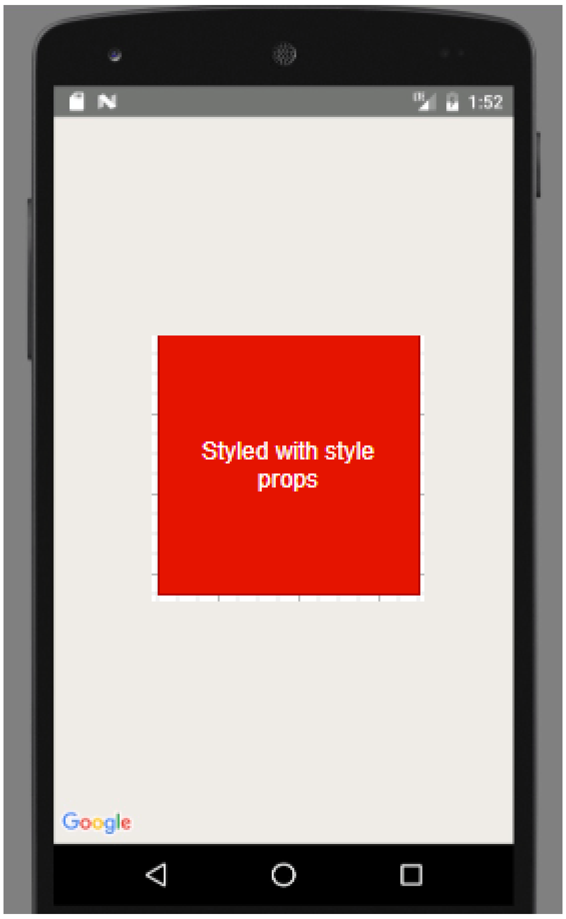
1) Style props

You can add styling to your component using style props. You simply add style props to your element and it accepts an object of properties.

```
import React, {Component} from 'react';
import {Platform, StyleSheet, Text, View} from 'react-native';
export default class App extends Component<Props> {
render() {
return (
<View style={{flex:1,justifyContent:"center",backgroundColor:"#fff",
alignItems:"center"}}>
<View style={{width:200,height:150,backgroundColor:"red",padding:10}}>
<Text style={{fontSize:20, color:"#666"}}>Styled with style props</Text>
</View>
</View>
);
}
}
```

2) Using StyleSheet

For an extremely large codebase or you want to set many properties to your elements, writing our styling rules directly inside style props will make our code more complex that's why React Native give us another way that let us write a concise code using the StyleSheet method:



```
import { StyleSheet} from 'react-native';
const styles = StyleSheet.create({
container: {
flex:1,
justifyContent:"center",
backgroundColor:"#fff",
alignItems:"stretch"
},
title: {
fontSize:20,
color:"#fff"
},
item1: {
backgroundColor:"orange",
flex:1
},
item2: {
backgroundColor:"purple",
flex:1
},
item3: {
backgroundColor:"yellow",
flex:1
},

});
```

We then pass the styles object to our component via the style props:

```
<View style={styles.container}>
<View style={styles.item1}>
<Text style={{fontSize:20, color:"#fff"}}>Item number 1</Text>
</View>
<View style={styles.item2}>
<Text style={{fontSize:20, color:"#fff"}}>Item number 1</Text>
</View>
<View style={styles.item3}>
<Text style={{fontSize:20, color:"#fff"}}>Item number 1</Text>
</View>
<View style={styles.item4}>
<Text style={{fontSize:20, color:"#fff"}}>Item number 1</Text>
</View>
</View>
```

3) styled-components in React Native

We can also use styled-components with React native so you can write your styles in React Native as you write normal CSS. It is very easy to include it in your project and it doesn’t need any linking just run this following command inside the root directory of your app to install it:

```
yarn add styled-components
```

```
import React, {Component} from 'react';
import { StyleSheet,Text, View} from 'react-native';
import styled from 'styled-components'
const Container=styled.View`
   flex:1;
   padding:50px 0;
   justify-content:center;
   background-color:#f4f4f4;
   align-items:center
`
const Title=styled.Text`
font-size:20px;
text-align:center;
color:red;
`
const Item=styled.View`
flex:1;
border:1px solid #ccc;
margin:2px 0;
border-radius:10px;
box-shadow:0 0 10px #ccc;
background-color:#fff;
width:80%;
padding:10px;

`


export default class App extends Component {
 render() {
   return (
     <Container>
            <Item >
            <Title >Item number 1</Title>
            </Item>
            <Item >
            <Title >Item number 2</Title>
            </Item>
            <Item >
            <Title >Item number 3</Title>
            </Item>
            <Item >
            <Title >Item number  4</Title>
            </Item>
     </Container>
   );
 }
```

23. Explain Async Storage in React Native and also define when to use it and when to not?

- Async Storage is the React Native equivalent of Local Storage from the web.

- Async Storage is a community-maintained module for React Native that provides an asynchronous, unencrypted, key-value store. Async Storage is not shared between apps: every app has its own sandbox environment and has no access to data from other apps.

DO USE ASYNC STORAGE WHEN..	DON'T USE ASYNC STORAGE FOR..
Persisting non-sensitive data across app runs	Token storage
Persisting Redux state	Secrets
Persisting GraphQL state	
Storing global app-wide variables	

React Native Advanced Interview Questions

24. What’s the real cause behind performance issues in React Native ?

The real cause behind React Native performance issues is that each thread (i.e Native and JS thread) is blazingly fast. The performance bottleneck in React Native app occurs when you’re passing the components from one thread to another unnecessarily or more than required. A major thumb rule to avoid any kind of performance-related issue in React Native is to keep the passes over the bridge to a minimum.

- Native thread built for running Java/ Kotlin, Swift/ Objective C
- Javascript thread is the primary thread that runs everything from javascript-based animations to other UI components
- The bridge as the name suggests acts as an intermediate communication point for the native and JS thread

25. List down some of the steps to optimize the application.

- Use Proguard to minimize the application size.(It does this by stripping parts of the React Native Java bytecode (and its dependencies) that your app is not using)
- Create reduced-sized APK files for specific CPU architectures. When you do that, your app users will automatically get the relevant APK file for their specific phone’s architecture. This eliminates the need to keep JSCore binaries that support multiple architectures and consequently reduces the app size.
- Compress images and other graphic elements. Another option to reduce image size is using file types like APNG in place of PNG files.
- Don’t store raw JSON data, eIther we need to Compress it or convert it into static object IDs.
- Optimize native libraries.
- Optimize the number of state operations and remember to use pure and memoized components when needed

- Use Global State wisely for example worst-case scenario is when state change of single control like TextInput or CheckBox propagates render of the whole application. Use libraries like Redux or Overmind.js to handle your state management in a more optimized way.
- Use key attribute on list items, it helps React Native to pick which list to update when rendering a long list of data
- Use VirtualizedList, FlatList, and SectionList for large data sets.
- Clear all the active timers which may lead to heavy memory leakage issues.

26. Describe Memory leak Issue in React Native , how can it be detected and resolved ?

In JavaScript memory is managed automatically by Garbage Collector (GC). In short, Garbage Collector is a background process that periodically traverses the graph of allocated objects and their references. If it happens to encounter a part of the graph that is not being referenced directly or indirectly from root objects (e.g., variables on the stack or a global object like window or navigator) that whole part can be deallocated from the memory.

In React Native world each JS module scope is attached to a root object. Many modules, including React Native core ones, declare variables that are kept in the main scope (e.g., when you define an object outside of a class or function in your JS module). Such variables may retain other objects and hence prevent them from being garbage collected.

Some Causes of Memory Leak:

- Unreleased timers/listeners added in componentDidMount
- Closure scope leaks

Detecting memory leaks for IOS:

In Xcode,

Go to XCode → Product → Profile (⌘ + i)

After that shows you all templates choose leaks.

Detecting memory leaks for Android :

Run React Native app normally (react-native run-android)
Run Android Studio

On the menu,
click Tools → Android → Enable ADB Integration
Click Tools → Android → Android Device Monitor
When Android Device Monitor shows up, click Monitor → Preferences

There is also one more way in Android
Perf Monitor (Performance Monitor) is a good choice to use for android leak monitoring.

```
Import PerfMonitor from 'react-native/Libraries/Performance/RCTRenderingPerf';
PerfMonitor.toggle();
PerfMonitor.start();
setTimeout(() => {
 PerfMonitor.stop();
}, 20000);
}, 5000);
```

27. Is there any out of the box way storing sensitive data in React ? If yes which and if not how can this be achieved ?

React Native does not come bundled with any way of storing sensitive data. However, there are pre-existing solutions for Android and iOS platforms.

iOS - Keychain Services
Keychain Services allows you to securely store small chunks of sensitive info for the user. This is an ideal place to store certificates, tokens, passwords, and any other sensitive information that doesn't belong in Async Storage.

Android - Secure Shared Preferences#
Shared Preferences is the Android equivalent for a persistent key-value data store. Data in Shared Preferences is not encrypted by default, but Encrypted Shared Preferences wraps the Shared Preferences class for Android, and automatically encrypts keys and values.

Android - Keystore
The Android Keystore system lets you store cryptographic keys in a container to make it more difficult to extract from the device. In order to use iOS Keychain services or Android Secure Shared Preferences, you can either write a bridge yourself or use a library that wraps them for you and provides a unified API at your own risk. Some libraries to consider:

- Expo-secure-store
- React-native-keychain
- react-native-sensitive-info - secure for iOS, but uses Android Shared Preferences for Android (which is not secure by default). There is however a branch that uses Android Keystore.

28. What is Network Security and SSL Pinning?

Understanding of SSL:

SSL (Secure Sockets Layer) and its successor, TLS (Transport Layer Security), are protocols for establishing authenticated and encrypted links between networked computers.
SSL/TLS works by binding the identities of entities such as websites and companies to

cryptographic key pairs via digital documents known as X.509 certificates. Each key pair consists of a private key and a public key. The private key is kept secure, and the public key can be widely distributed via a certificate.

Understanding of pinning

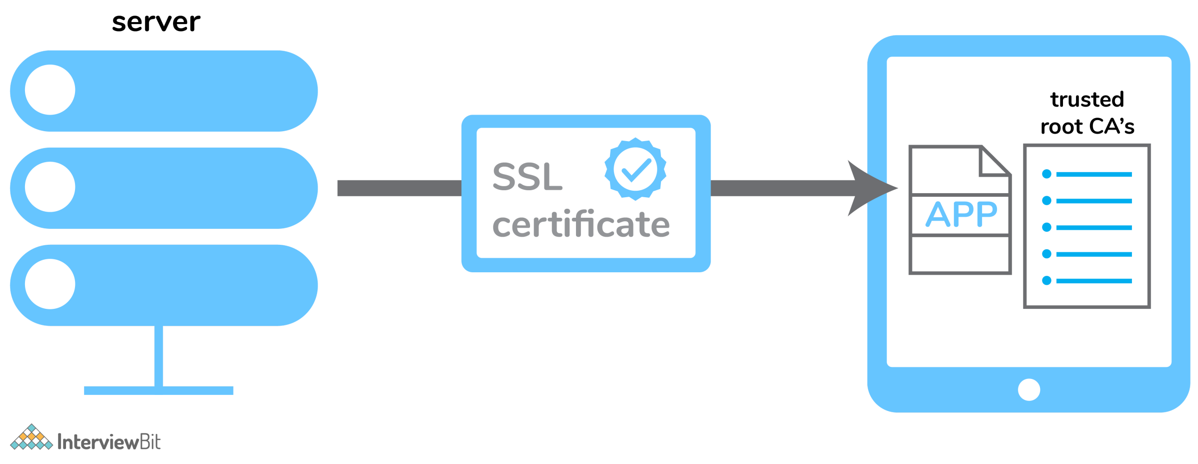
Pinning is an optional mechanism that can be used to improve the security of a service or site that relies on SSL Certificates. Pinning allows specifying a cryptographic identity that should be accepted by users visiting site/app

Why do we need SSL pinning?

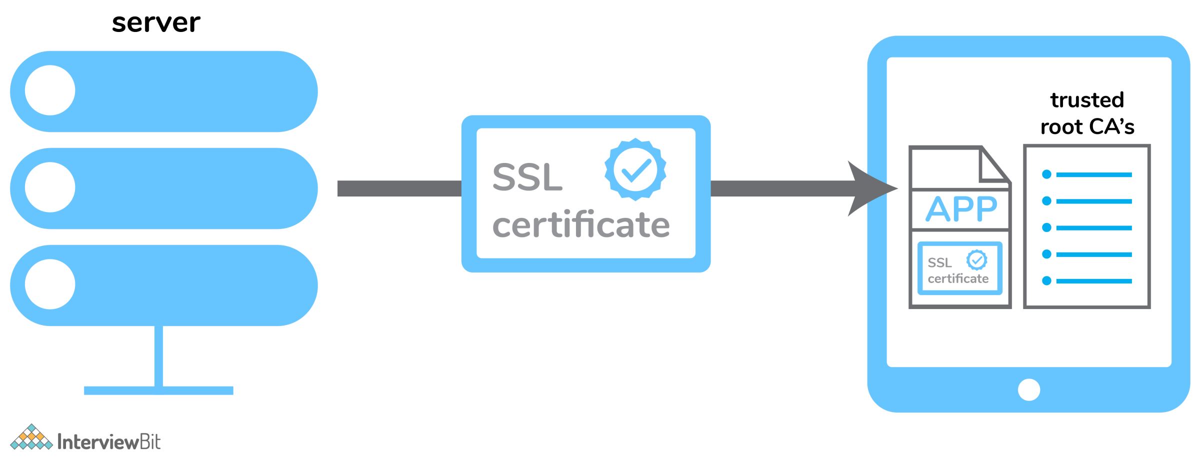
One of the inherent risks to the SSL ecosystem is mis-issuance. This is when an unauthorized certificate is issued for a domain/host you control. This can happen with both public and private PKIs (Public Key Infrastructure)

How is SSL pinning used in Mobile applications?

When mobile applications communicate with the server, they typically use SSL to protect the transmitted data against tampering. By default SSL implementations used, apps trust any server with a certificate trusted by the Operating systems trust store, This store is a list of certificate authorities that are shipped with the operating system.



With SSL pinning, however, the application is configured to reject all but one or few predefined certificates, whenever the application connects to a server, it compares the server certificate with the pinned certificate(s) , if and only if they match the server is trusted and SSL connection is established.



29. Explain setNativeProps. Does it create Performance issues and how is it used ?

It is sometimes necessary to make changes directly to a component without using state/props to trigger a re-render of the entire subtree. When using React in the browser, for example, you sometimes need to directly modify a DOM node, and the same is true for views in mobile apps. setNativeProps is the React Native equivalent to setting properties directly on a DOM node.
Use setNativeProps when frequent re-rendering creates a performance bottleneck.

Direct manipulation will not be a tool that you reach for frequently; you will typically only be using it for creating continuous animations to avoid the overhead of rendering the component hierarchy and reconciling many views. setNativeProps is imperative and stores state in the native layer (DOM, UIView, etc.) and not within your React components, which makes your code more difficult to reason about. Before you use it, try to solve your problem with setState and shouldComponentUpdate.

30. How to make your React Native app feel smooth on animations ?

The primary reason and an important one why well-built native apps feel so smooth are by avoiding expensive operations during interactions and animations. React Native has a limitation that there is only a single JS execution thread, but you can use InteractionManager to make sure long-running work is scheduled to start after any interactions/animations have completed.

Applications can schedule tasks to run after interactions with the following:

```
InteractionManager.runAfterInteractions(() => {
 // ...long-running synchronous task...
});
```