

INTERVIEW QUESTIONS ON REACT NATIVE

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**

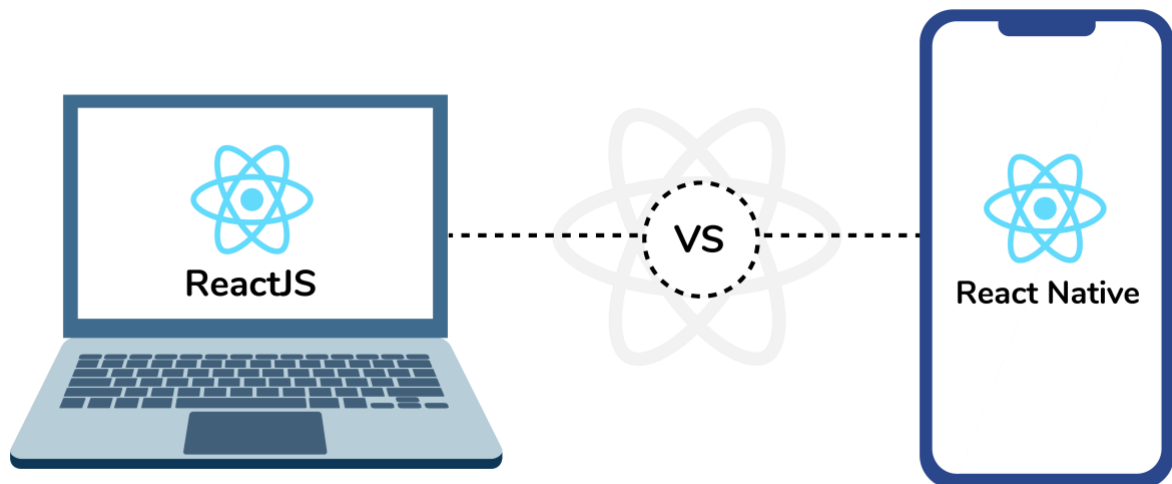
Both React and React Native uses JSX (JavaScript XML) syntax but React uses html tags like <div> <h1> <p> etc while React Native uses <view> <text> etc.

- **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.



 InterviewBit

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.

REACT JS	REACT NATIVE
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,

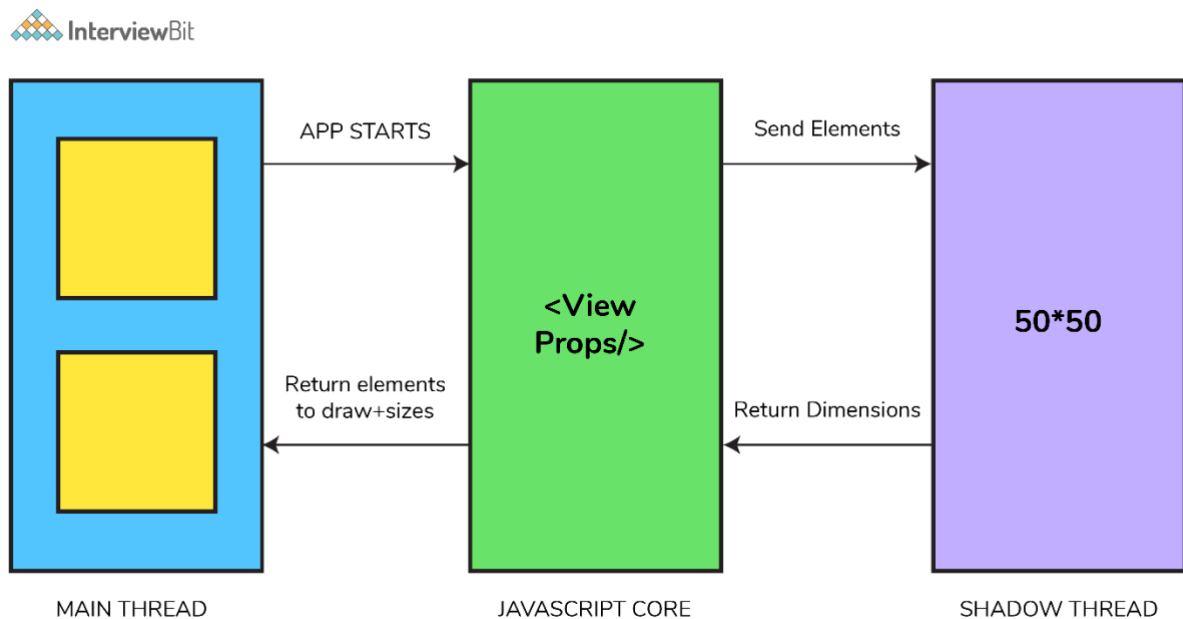
- Large Community**
 React Native is an Open Source Framework, it is completely community driven so any challenges can be resolved by getting online help from other developers.
- Reusability**
 Code can be written once and can be used for both IOS and ANDROID, which helps in maintaining and as well debugging 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.

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 ?

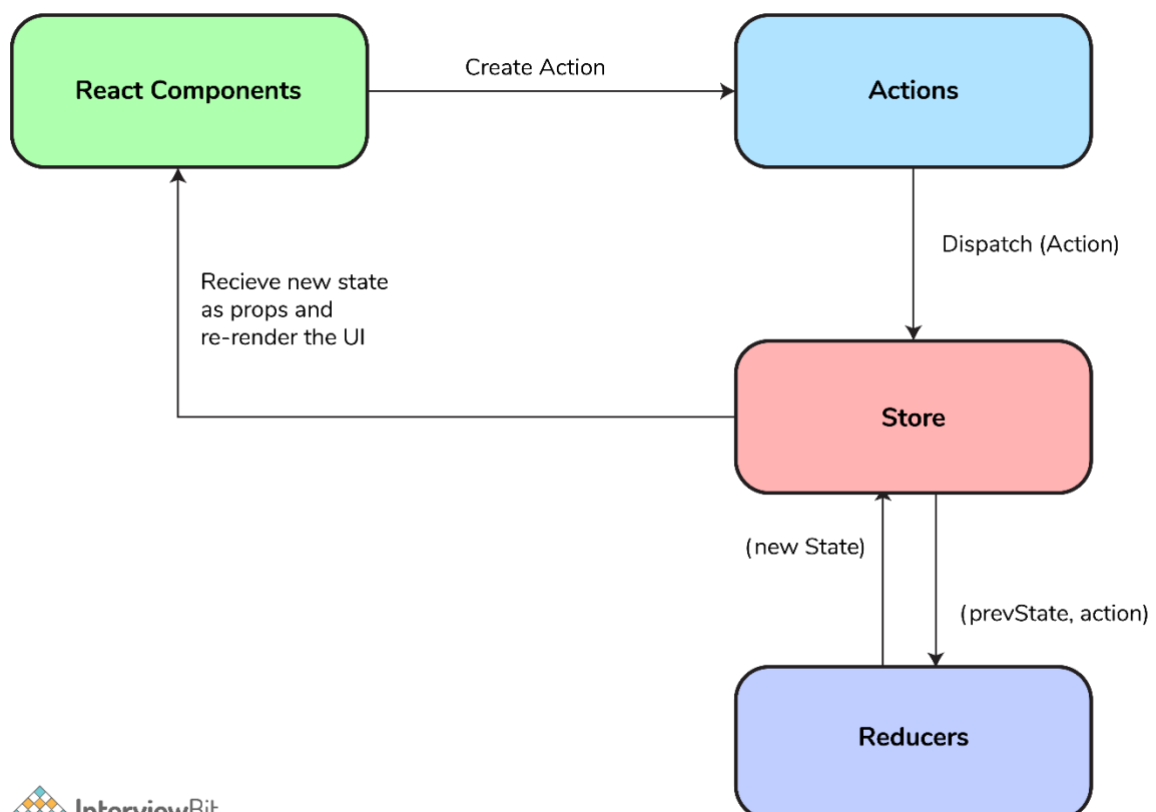
Redux is a predictable state container for JavaScript apps. It helps write applications that run in different environments. This means the entire data flow of the app is handled within a single container while persisting previous state.

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:**

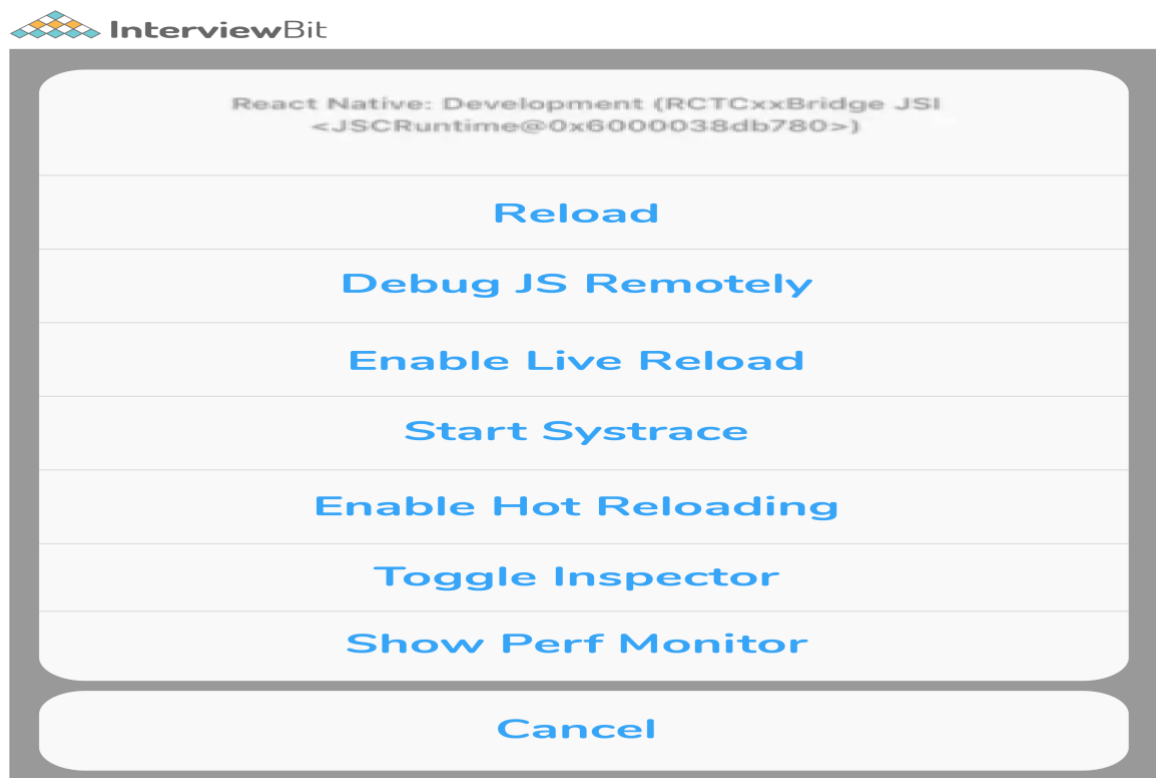
Reload: reloads the app

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.

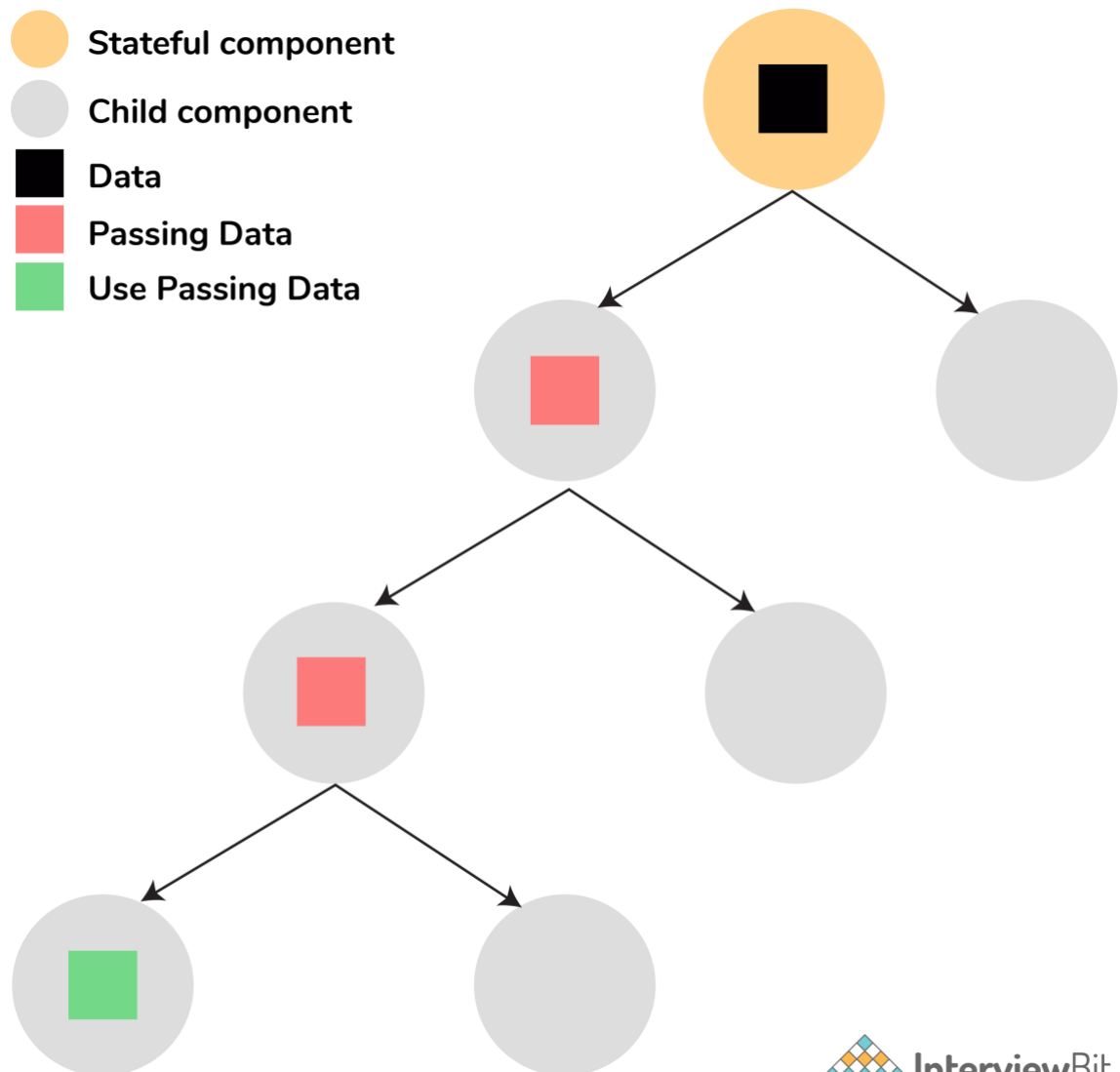
React Native CLI

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.

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

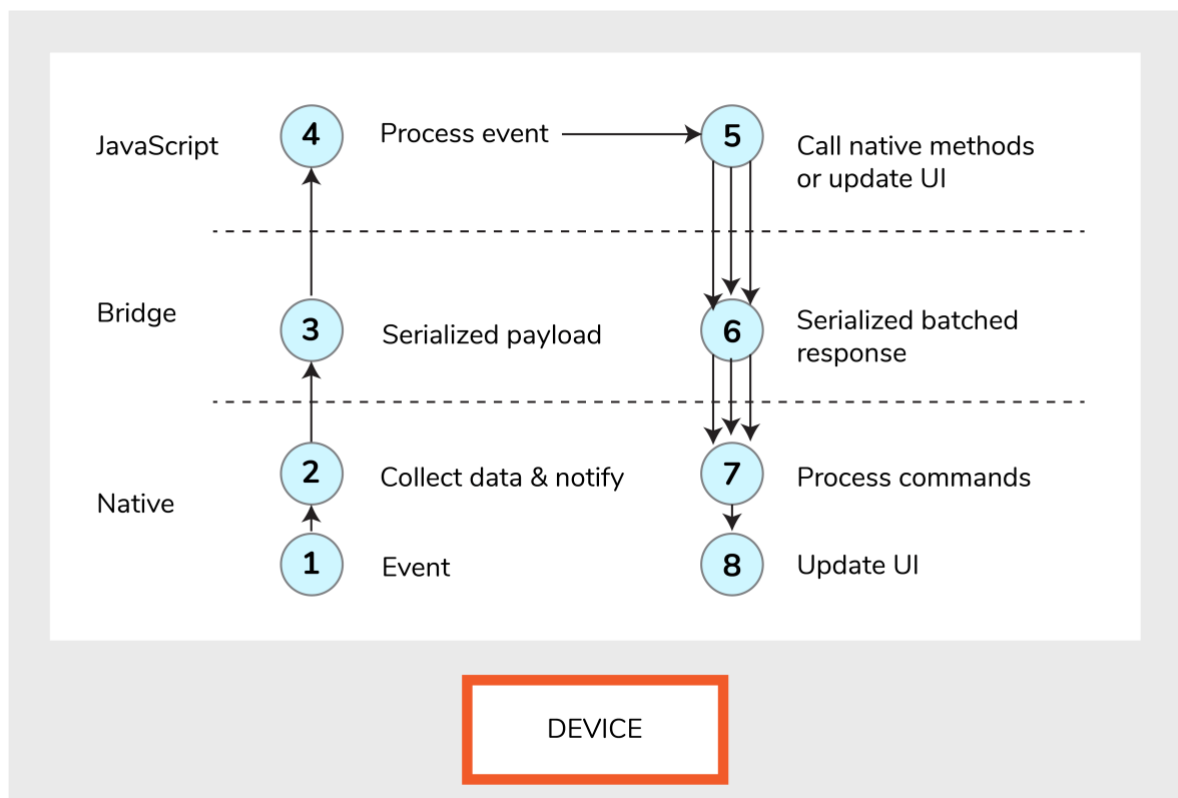
- At the first start of the app, the main thread starts execution and starts loading JS bundles.
- When JavaScript code has been loaded successfully, the main thread sends it to another JS thread because when JS does some heavy calculations stuff the thread for a while, the UI thread will not suffer 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:



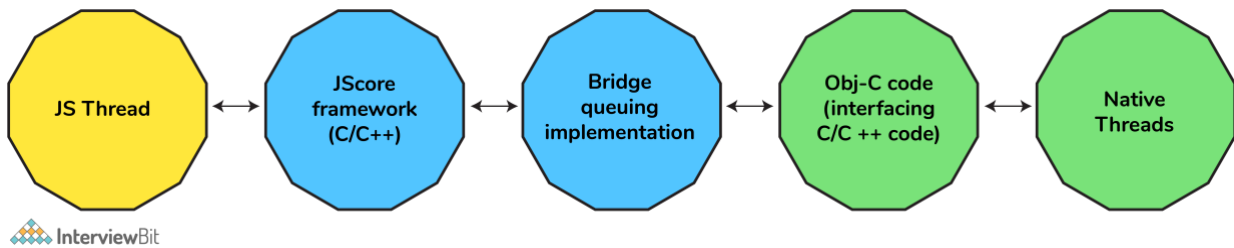
InterviewBit

- 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.

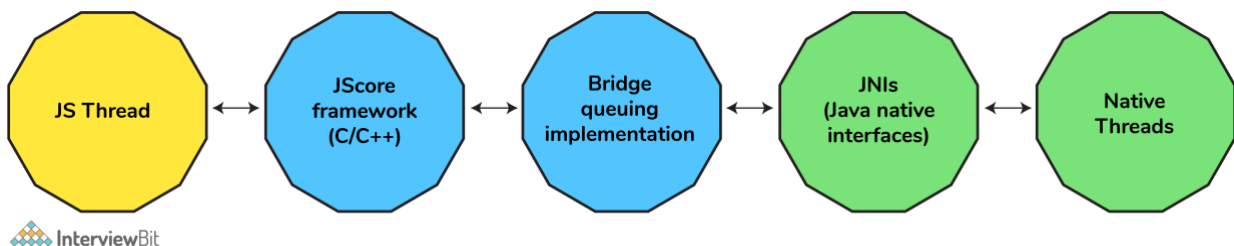
Now for an example, there is some state change that happens, because of which React Native will batch Update UI and send it to the Bridge. The bridge will pass

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.

IOS Platform:



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

REACT NATIVE UI COMPONENT	ANDROID VIEW	IOS VIEW	WEB ANALOG	DESCRIPTION
				handles touch events.
<Image>	<ImageView>	<UIImage View>		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>
      </View>
    );
  }
}
```

```
}
```

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 :

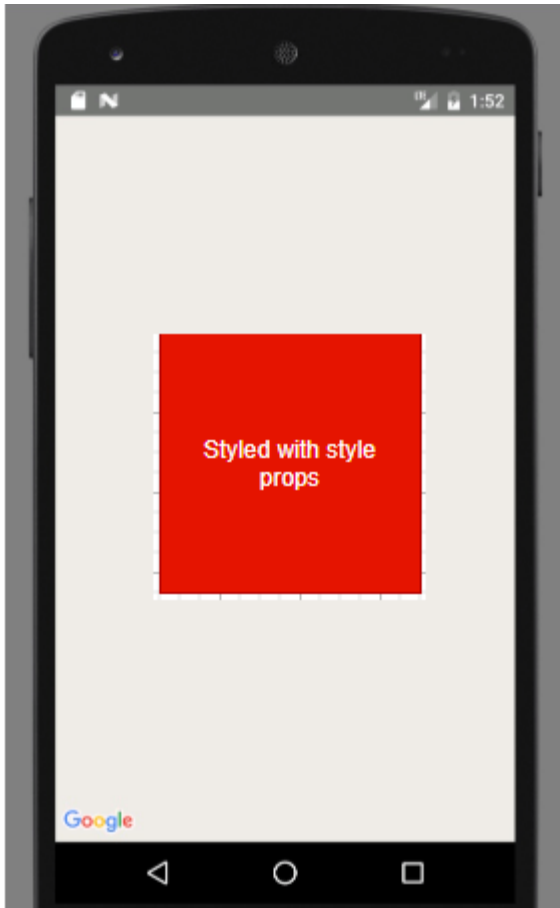
1) Style props

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

DO USE ASYNC STORAGE WHEN..	DON'T USE ASYNC STORAGE FOR..
Persisting Redux state	Secrets
Persisting GraphQL state	
Storing global app-wide variables	

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...  
});
```

1) Explain React Native?

React Native is an open-source JavaScript framework introduced by Facebook. It is used for developing a real, native mobile application for iOS and Android platforms. It uses only JavaScript to build a mobile app. It is like React, which uses native components rather than using web components as building blocks. It is cross-platform, which allows you to write code once and can run on any platform.

React Native application is based on React, a JavaScript library developed by Facebook and XML-Esque markup (JSX) for creating the user interface. It targets the mobile platform rather than the browser. It saves your development time as it allows you to build apps by using a single language JavaScript for both Android and iOS platforms.

2) What are the advantages of React Native?

React Native provides many advantages for building mobile applications. Some of the essential benefits of using React Native are given below:

- **Cross-Platform:** It offers the facility to "Write once and run everywhere." It is used to create apps for Android, iOS, and Windows platforms.
- **Performance:** The code written in React Native is compiled into native code, which enables it for all operating systems to provide closer native appearance and functions in the same way on all platforms.
- **Community:** React Native provides a large community of passionate developers who are always ready to help us to fix bugs, and issues occur at any instant.
- **Hot Reloading:** Making a few changes in your app's code immediately visible during development. If the business logic is changed, its reflection is live reloaded on screen.
- **Faster Development:** React Native helps to develop apps fast. It uses a common language to build an app for Android, iOS, and Windows platforms, which gives speedier app deployment, delivery, and quicker time-to-market.
- **JavaScript:** JavaScript knowledge is used to build native mobile apps.

3) What are the disadvantages of React Native?

Some of the big disadvantages of React Native for building mobile applications are given below:

- **React Native is still new and immature:** React Native is a new framework in Windows, Android, and iOS programming languages. It is still in the improvement stage, which can have a negative impact on the apps.

- **Learning is tough:** React Native cannot learn quickly, especially for a fresher in the app development field.
- **It Lacks the Security Robustness:** React Native is an open-source JavaScript framework, which is fragile and creates a gap in the security robustness. When you are creating banking and financial apps where data is highly confidential, experts advise not to choose React Native.
- **It Takes More Time to Initialize:** React Native takes a lot of time for initializing the runtime even if you are using the hi-tech gadgets and devices.
- **Existence is Uncertain:** As Facebook develop this framework, its presence is uncertain since it keeps all the rights to kill off the project anytime. As the popularity of React Native rises, it is unlikely to happen.

4) List the essential components of React Native.

These are the core components of React Native:

- **View:** It is the basic built-in component used to build UI of Mobile apps. The view is similar to the div in HTML. It is a content area where you can display your content.
- **States:** 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.
- **Props:** Props are used to pass data to the different components. It is immutable means props cannot change the value. It provides a connection between the container component and a presentation component.
- **Style:** It is an essential component in the web or mobile, which makes the application attractive. React Native does not require any special language or syntax for styling. It can style the application using the JavaScript object.
- **Text:** This component displays text in the app. It uses the basic component `textInput` to take text input from the user.
- **ScrollView:** It is a scrolling container used to host multiple views. It can be used to render the large list or content in view with a scroll bar.

5) How many threads run in React Native?

The React Native app contains the following thread:

- **React Native UI Thread (Main Thread):** This thread is used for the layout of the mobile app.
- **React Native JavaScript Thread:** It is a thread where our business logic will run. It means JS thread is a place where our JavaScript code is executed.
- **React Native Modules Thread:** Sometimes, our app needs access to platform API, which happens as a part of native module thread.
- **React Native Render Thread:** This thread is used to generate actual OpenGL commands used to draw the app UI.

6) What is React Native Apps?

React Native Apps are not web applications. These types of apps are running on mobile devices, and cannot load over the browser. Also, they are not hybrid apps that build over Ionic, Phonegap, etc. which can run over WebView component. They are the real native apps built in a single language JavaScript with the native components to run on mobile devices.

7) List Step to Create and start a React Native App?

The following steps are necessary to create and start a React Native app:

Step-1: Install Node.js

Step-2: Install react-native environments by using the following command.

1. \$ npm install -g create-react-native-app

Step-3: Create a project by using the following command.

1. \$ create-react-native-app MyProject

Step-4: Next, navigate in your project by using the following command.

1. \$ cd MyProject

Step-5: Now, run the following command to start the project.

1. \$ npm start

8) What are states 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.

Example

Here, we are going to create a Text component with state data. The content of the Text component will be updated whenever we click on it. The event **onPress** calls the **setState** function, which updates the state with "**myState**" text.

1. **import** React, {Component} from 'react';
2. **import** { Text, View } from 'react-native';
- 3.
4. export **default class** App **extends** Component {
5. state = {
6. myState: 'This is a text component, created using state data. It will change or updated on clicking it.'
7. }
8. updateState = () => **this**.setState({ myState: 'The state is updated'})
9. render() {
10. **return** (
11. <View>
12. <Text onPress={ **this**.updateState }> { **this**.state.myState } </Text>
13. </View>
14.);
15. }
16. }

9) What are props in React Native?

The properties of React Native components are pronounced as props. They are used to pass data to the different components. In React Native, several components are customized at the time of creation with different parameters, and these parameters are known as props. It is immutable means props cannot change the value. It provides a connection between the container component and a presentation component.

Example

Here, we have created a Heading component, with a message prop. The parent class App sends the prop to the child component Heading.

```
1. // Parent Component
2. export default class App extends Component {
3.   render () {
4.     return (
5.       <View style={{alignItems: 'center'} >
6.         <Heading message={'Custom Heading for Parent class?'}/>
7.       </View>
8.     )
9.   }
10. }
11.
12. // Child component
13. export default class Heading extends Component {
14.   render () {
15.     return (
16.       <View style={{alignItems: 'center'} >
17.         <Text>{this.props.message}</Text>
18.       </View>
19.     )
20.   }
21. }
22. const styles = StyleSheet.create({
23.   welcome: {
24.     fontSize: 30,
```

```
25. }  
26. });  
27. Heading.propTypes = {  
28. message: PropTypes.string  
29. }  
30. Heading.defaultProps = {  
31. message: 'Heading One'  
32. }
```

10) 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

11) Are all React components usable in React Native?

React web components use DOM elements (ex. div, h1, table, etc.) to display on UI. But, these components are not supported in React Native. You will need to find libraries or components which is made specifically for React Native. It is very hard to find that there are components available, which support both. But, it should be easy to figure out that the given components are made for React Native or not. Thus, it makes clear that all components are not usable in the React Native.

12) How Virtual DOM works in React Native?

Virtual DOM is a lightweight JavaScript object, which is an in-memory representation of a real DOM. It is an intermediary step between the render function being called and the displaying of elements on the screen. It is similar to a node tree, which lists the elements, their attributes, and content as objects and their properties. The render function creates a node tree of the React components and then updates this node tree in response to the mutations in the data model caused by various actions done by the user or by the system.

Virtual DOM works in three steps:

- Whenever any data changes in the React App, the entire UI is re-rendered in Virtual DOM representation.
- Now, the difference between the previous DOM representation and the new DOM is calculated.
- Once the calculations are completed, the real DOM updated with only those things which are changed

13) Can we combine native iOS or Android code in React Native?

Yes, we can combine the native iOS or Android code with React Native. It can combine the components written in Objective-C, Java, and Swift.

14) Do we use the same code base for Android and iOS?

Yes, we can use the same codebase for Android and iOS, and React takes care of all the native component translations. For example, a React Native ScrollView uses ScrollView on Android and UIScrollView on iOS.

15) What is the difference between an Element and a Component in React Native?

The difference between an Element and a Component in React Native are:

React Element	React Component
The React Element is a simple object, which describes a DOM node and its attributes or properties. It is an immutable object where you cannot apply any methods.	The React Component is a function or class that takes inputs and returns a React element. It contains references to its DOM nodes and the instances of the child components.
For Example:	For Example:

```
<button className = "green"></button>
```

```
const SignIn = () => (  
  <div>  
    <p>Sign In</p>  
    <button>Continue</button>  
    <button color='green'>Cancel</button>  
  </div>  
);
```

16) What is the difference between React and React Native?

The essential differences between React and React Native are:

- React is a JavaScript library, whereas React Native is a JavaScript framework based on React.
- Tags can be used differently in both platforms.
- React is used for developing UI and Web applications, whereas React Native can be used to create cross-platform mobile apps.

17) What is the difference between React Native and Ionic?

The essential differences between React Native and Ionic are:

- Ionic is a typical hybrid development framework. It mainly focuses on front-end user experience or UI interaction, which handles all the look and feel of your app. It is easy to learn and can integrate with other libraries or frameworks such as Angular, React, Cordova, etc. Its purpose is to write once and runs everywhere.
- React Native is an open-source JavaScript framework developed by Facebook to build a cross-platform mobile app. It is used for developing a mobile application for iOS, Android, and Windows. React Native is the same as React, but it uses native components instead of using web components as building blocks. It targets mobile platforms rather than the browser. Its purpose is to learn once and write anywhere.

18) What are the differences between React Native and Native (Android and iOS)?

React Native allows you to write once and runs everywhere. It means we can reuse the React Native code on both Android and iOS platforms. Since we can reuse most of the React Native code between both platforms, but Android and iOS are different systems. Here, we are going to see these differences.

Operating System

You can build applications for both Android and iOS with React Native, but it is not an easy task to check that the app works on both systems if you are working on Windows systems. Windows do not allow to run XCode and its simulator, which is a macOS app. There are other tools available, but they are not official.

Native elements

The elements perform different actions for the React Native and Native apps. React Native apps uses elements from React Native library, whereas Native apps do not use elements of React native libraries.

Specific Styles-Shadows

Shadows style is an essential term of differences between iOS and Android while working on cross-platform apps. Android does not support shadow; instead of this, it uses elevation property.

Linking libraries

Sometimes we want to use third-party libraries in our app. Most of the time, we add it as a dependency, but sometimes it requires manual linking for adding libraries. Linking libraries manually is not an easy task for developers, either web or native applications. Since the React Native is in the improvement stage, the libraries docs are not updated according to the latest framework.

React Native vs. Native (Android and iOS)

	React Native	Android	iOS
Language	JavaScript JSX	Java	Objective-C/Swift
Debugger	Text Editor, Chrome Debugger	Android Studio	XCode
Used By	Facebook, AirBnB	Airdroid, Chromer	GarageBand, iMovie

19) What is the difference between React Native and Xamarin?

The essential differences between React Native and Ionic are:

- React Native is an open-source JavaScript framework developed by Facebook to build a cross-platform mobile app for iOS, Android, and Windows. React Native is the same as React, but it uses native components instead of using web components as building blocks. It targets mobile platforms rather than the browser. Its purpose is to learn once and write anywhere.

- Xamarin is an open-source, cross-platform development framework, which offers you to build android, iOS, Windows, and Mac apps by using the C# language. It is first launched in May 2011 by Xamarin Company. In 2016, Microsoft had signed an agreement to acquire Xamarin.

20) What does a StyleSheet.create do?

In React native, the StyleSheet.create() ensures that the values are immutable and opaque. They are used to send the style only once through the bridge to avoid passing a new style object.

21) 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.

22) 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.

23) Which language is used in React Native?

The language used in React Native is Java for Android applications and Objective-C/Swift for iOS apps.

24) What is style in React Native?

It is an essential component in the web or mobile, which makes the application attractive. React Native does not require any special language or syntax for styling. It can style the application using the JavaScript object.

25) What are Refs in React Native?

React refs are useful features that allow you to access DOM elements or component's instance directly within React. It comes handy in situations where you want to change the child of a component without using props or re-rendering the whole component.

26) Why React Native app use keys?

Keys are a unique identifier. They are used to identify which items have changed, updated, or deleted from the lists. It should always use inside the array.

27) What is meant by HOC in React Native?

HOC Stands for Higher-Order Component. It is a technique, which allows you to reuse the component logic. It is a function that takes a component and gives back a new component.

Syntax

1. `const NewComponent = higherOrderComponent(WrappedComponent);`

28) 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.

29) What are the differences between Class and Functional Component?

The essential differences between the class component and functional component are:

Syntax: The declaration of both components is different. A functional component takes props, and returns React element, whereas the class components require to extend from React.

1. `//Functional Component`
2. `function WelcomeMessage(props) {`
3. `return <h1>Welcome to the , {props.name}</h1>;`
4. `}`
- 5.
6. `//Class Component`
7. `class MyComponent extends React.Component {`
8. `render() {`
9. `return (`
10. `<div>This is main component.</div>`

11.);

12. }

13. }

State: The class component has a state while the functional component is stateless.

Lifecycle: The class component has a lifecycle, while the functional component does not have a lifecycle.

30) When would you prefer a class component over functional components?

We prefer class component when the component has a state and lifecycle; otherwise, the functional component should be used.

31) How React Native handle different screen sizes?

React Native provides many ways to handle screen sizes. Some of them are given below:

1. Flexbox: It is used to provide a consistent layout on different screen sizes. It has three main properties:

- flexDirection
- justifyContent
- alignItems

2. Pixel Ratio: It is used to get access to the device pixel density by using the **PixelRatio** class. We will get a higher resolution image if we are on a high pixel density device.

3. Dimensions: It is used to handle different screen sizes and style the page precisely. It needs to write the code only once for working on any device.

4. AspectRatio: It is used to set the height or vice versa. The aspectRatio is present only in React-Native, not a CSS standard.

5. ScrollView: It is a scrolling container which contains multiple components and view. The scrollable items can be scroll both vertically and horizontally.

32) What is ListView?

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

33) What are the best UI Components for React Native?

The best UI component for React Native are:

- Material UI
- Semantic UI
- React Bootstrap
- React Toolbox
- Ant Design

34) What are the similarities between React and React Native?

The most common similarities between React and React Native are:

- React Lifecycle Methods
- React Components
- React States and Props
- Redux Libraries

35) What are animations in React Native?

The animation is a method in which images are manipulated to appear as moving objects. React Native animations allows you to add extra effects, which provide great user experience in the app. We can use it with React Native API, `Animated.parallel`, `Animated.decay`, and `Animated.stagger`.

React Native has two types of animation, which are given below.

- **Animated:** This API is used to control specific values. It has a start and stops methods for controlling the time-based animation execution.
- **LayoutAnimated:** This API is used to animate the global layout transactions.

36) How is data loaded on the server by React Native?

React Native uses Fetch API to fetched data for networking needs.

37) 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.

38) Can you integrate more features in the existing app by React Native?

Yes, we can add new features to existing applications in React Native.

39) What is meant by Gesture Responder System?

It is an internal system of React Native, which is responsible for managing the lifecycle of gestures in the system. React Native provides several different types of gestures to the users, such as tapping, sliding, swiping, and zooming. The responder system negotiates these touch interactions. Usually, it is used with Animated API. Also, it is advised that they never create irreversible gestures.

40) What does React Native Packager do in the React Native?

The React Native Packager performs the following functionalities:

- The React Native Packager combines all the JavaScript code of your application into a single file and then translate any of the JavaScript code that your device won't understand like JSX.
- It also converts the assets (e.g., PNG file) used in your project into objects, which can be displayed by an Image component.

41) Why 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 use 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.

42) How to update React Native with the latest version?

It is very important to upgrade the existing React Native with the latest version, which gives you access to more APIs, views, developer tools, and other latest features. The following steps need to be performed for upgrading the React Native with the latest versions.

1. Upgrade your **expo project** in **package.json** with the latest version of react-native, react, and expo package.
2. Set the latest version of SDK, which is compatible with the latest react-native in your **app.json** file.
3. Upgrade the React Native CLI to update the source file by using the following command.

1. \$ react-native upgrade

4. Install the upgrade helper web tool that provides you to upgrade your apps between any two versions.

5. Upgrade your project files by running the following command.

1. `$ react-native init`

6. Last, you need to perform the troubleshoot activity to upgrade with React Native CLI.

43) What is API in React Native?

An API or Application Programming Interface is a software intermediary that lets in two applications to communicate with each other without having to know how they are implemented. Sometimes it is thought of as a contract, with documentation that represents an agreement between two parties. **For example**, each time when you use an app in the mobile like Facebook, it sends a message, or when you see the weather on your phone, these are the usage of an API.

React Native use the Fetch networking API to suit our needs. It simply calls the URL through Fetch, and then make requests to the server as needed. The React Native API mainly uses three lifecycle methods, which are constructor, componentDidMount, and Render.

44) How to use Axios in the React Native?

Axios is a popular library for making HTTP requests from the browser. It allows us to make GET, POST, PUT, and DELETE requests from the browser. Therefore, React Native uses Axios to make requests to an API, return data from the API, and then perform actions with that data in our React Native app. We can use Axios by adding the Axios plugin to our project using the following command.

1. `# Yarn`

2. `$ yarn add axios`

3. `# npm`

4. `$ npm install axios --save`

Axios have several features, which are listed below:

- It makes XMLHttpRequests from the browser.
- It makes Http requests from the React Native framework.
- It supports most of the React Native API.
- It offers a client-side feature that protects the application from XSRF.
- It automatically transforms response and request data with the browser.

45) Which database is best for React Native?

The most popular database for React Native is an SQLite database.

46) How to use firebase in react native?

Firebase is a popular tool for mobile and web app development platform. It provides many services to help you in building fast and high-quality apps, grow your user base, and earn more money without managing infrastructure. It is a powerful Database as a Service (DBaaS) tool, which provides a scalable cloud database to store and sync data for client and server-side development. Some of the key features of firebase are authentication, Real-time database, cloud messaging, crash reporting, and analytics. Firebase is a type of freemium model, not an open-source model. However, you can use its services free until you don't pass the limits of its free tier.

We can get started with firebase by using following steps:

- First, login into the firebase console and then create a project.
- Retrieve apikey, authDomain, DatabaseURL, and storage bucket from the console.
- Next, you need to create a new React Native project
- Install firebase plugin from npm
- Add firebase plugin into the React Native project

1. How is React Native different from ReactJS?

React Native is a JavaScript framework that was developed by Facebook to meet the growing needs of mobile app development. It's open-source and based on JavaScript. It was designed to build native mobile apps with reusable components. It uses a lot of ReactJS components but implements them in a native way across different devices. It invokes the native rendering APIs in Objective-C (for IOS) and Java (for Android).

[ReactJS](#) was also developed by Facebook. It's an open-source JavaScript library used for developing responsive user interfaces for mobile and web applications. It has a library of reusable components that are meant to help developers build the foundation for their apps.

Let's take a look at some of their key differences:

- **Syntax:** React Native and ReactJS both use JSX, but ReactJS uses HTML tags, and React Native doesn't.
- **Navigation:** React Native uses its own built-in navigation library, while ReactJS uses a react-router.
- **Animation:** ReactJS uses [CSS](#) animations. React Native uses its animated API.
- **DOM:** ReactJS uses a virtual DOM with a partial refresh. React Native needs to use its native API when rendering UI components.

- **Usage:** ReactJS is mainly used for web app development, while React Native focuses on mobile applications.

2. What is JSX?

JavaScript XML, or JSX, is a XML/HTML template syntax used by React. It extends ECMAScript, which allows XML/HTML-like text to coincide with JavaScript and React code. It allows us to put HTML into JavaScript.

It's faster than normal JavaScript, makes it easier to create templates, and uses components. It comes with the full power of JavaScript, and you can use it with React to describe what the user interface should look like. Let's take a look at a **Hello World!** in JSX:

```
const element = <h1>Hello World!</h1>;
```

3. What are the core React Components and what do they do?

The core React components include:


- **Props:** You can use props to pass data to different React components. Props are immutable, which means props can't change their values.
- **ScrollView:** ScrollView is a scrolling container that's used to host multiple views. You can use it to render large lists or content.
- **States:** You use states to control components. The state is mutable in React, meaning that it can change the value at any time.
- **Style:** React Native doesn't require any special syntax for styling. It uses the JavaScript object.
- **Text:** The text components display text in your application. It uses **textInput** to take input from the user.
- **View:** View is used to build the UI for mobile applications. It's a place where you can display your content.

4. How do you install and create a React Native application?

Before you begin, make sure you have Node.js and NPM installed on your system.

To install a React Native application, you can use the following command:

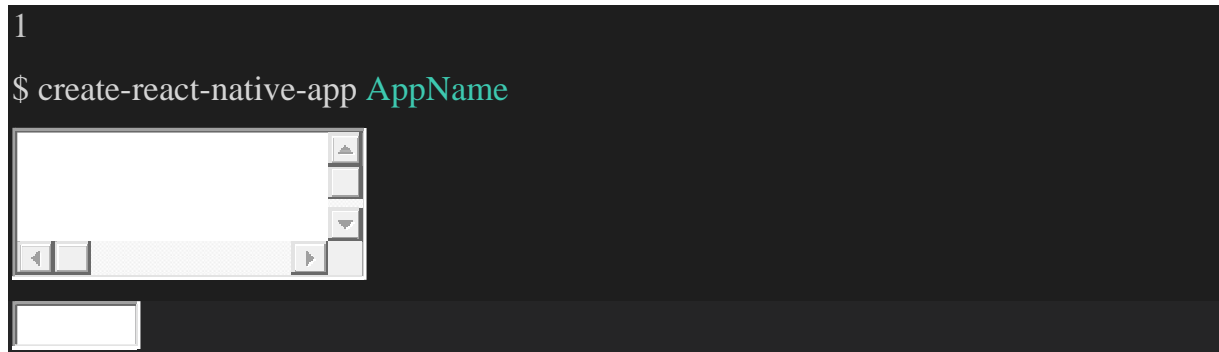
```
1
$ npm install -g create-react-native-app
```



A screenshot of a terminal window with a dark background. The prompt '\$' is visible, followed by the command 'npm install -g create-react-native-app'. Below the command, there is a small, light-colored rectangular window with a white border, which appears to be a placeholder or a small application window.

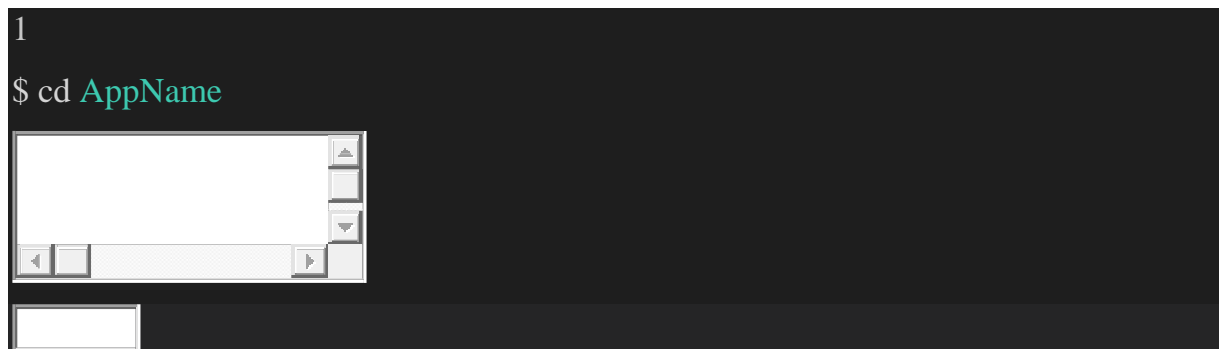
To create a React Native project, you can use the following command:

```
1
$ create-react-native-app AppName
```

A terminal window with a dark background. The first line shows a prompt character followed by the command 'create-react-native-app AppName'. A file explorer window is overlaid on the terminal, showing a file named 'AppName' in a directory. The file explorer has a search bar, a list of files, and a sidebar with navigation icons.

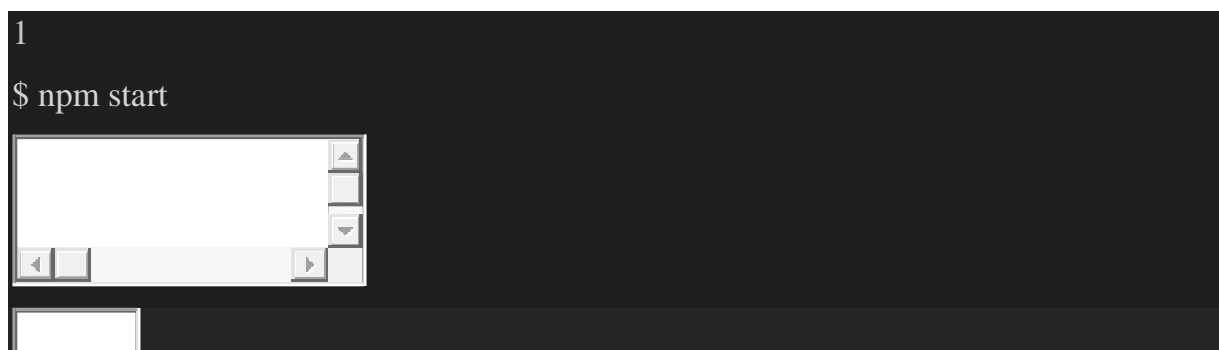
To navigate in your project, use the following command:

```
1
$ cd AppName
```

A terminal window with a dark background. The first line shows a prompt character followed by the command 'cd AppName'. A file explorer window is overlaid on the terminal, showing a file named 'AppName' in a directory. The file explorer has a search bar, a list of files, and a sidebar with navigation icons.

And to start your project, run this command:

```
1
$ npm start
```

A terminal window with a dark background. The first line shows a prompt character followed by the command 'npm start'. A file explorer window is overlaid on the terminal, showing a file named 'npm start' in a directory. The file explorer has a search bar, a list of files, and a sidebar with navigation icons.

5. What is Redux and when should you use it?

[Redux](#) is a state management tool for JavaScript applications. It helps you write apps that are consistent, apps that can be run in different environments, and apps that are easy to test.

Not all applications need Redux. It's designed to help you determine when you experience state changes. According to the official Redux documentation, here are some examples of when you'd want to use Redux:

- Your app state is updated frequently
- You have a large amount of app state and it's needed in many places within the app
- The logic to update your app state is complicated
- You want to see how the state is being updated over time

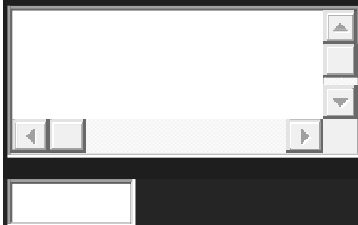
- Your app has a medium or large-sized codebase and will be worked on by multiple people

6. What is **state** and how do you use it?

In React Native, **state** handles data that is changeable. **state** is mutable, meaning that it can change the value at any time. You should initialize it in the constructor, and then call **setState** when you want to change it. Let's look at a code example of how to create a text class component using state data:

```
import React, {Component} from "react";
import {Text, StyleSheet} from "react-native";

class TextExample extends Component{
  constructor(props){
    super(props);
    this.state = {
      titleText: "What is React Native?",
      bodyText: "React Native is a JavaScript framework."
    };
  }
}
```



7. How do you debug React apps and what tools can you use?

There are many different ways to do your debugging in React Native applications. Since React Native has both IOS and Android environments, there's a wide range of different problems you can encounter and a wide range of different tools needed. We're going to explore a few different ways to debug. Let's start with outlining the dev menu:

Developer menu

The developer menu includes some different ways to debug and access debugging tools.

- **Reload:** reloads the app
- **Debug JS Remotely:** opens to a JavaScript debugger

- **Enable Live Reload:** causes the app to reload automatically after selecting “Save”
- **Enable Hot Reloading:** watches for changes
- **Toggle Inspector:** toggles the inspector interface so we can inspect UI elements and their properties
- **Show Perf Monitor:** monitors performance

Chrome DevTools

You can use these DevTools to debug React Native apps. You need to make sure that it's connected to the same WiFi. If you're using Windows or Linux, press **Ctrl + M**+, and if you're using macOS, press **Command + R**. In the developer menu, you select “Debug JS Remotely” and it will open the default debugger.

React Developer Tools

To use React's Developer Tools, you have to use the desktop app. These tools allow you to debug React components and styles.

React Native Debugger

If you're using Redux in your React app, this is a good debugger for you. It's a desktop app that integrates Redux's and React's developer tools in one app.

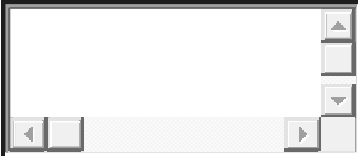
React Native CLI

You can use the React Native command-line interface to do debugging as well.

8. Build a React app that does nothing except say “Hello World!”

```
import React from "react";
import { Text, View } from "react-native";
const HelloWorldApp = () => {
  return (
    <View
      style={{
        flex: 1,
        justifyContent: "center",
        alignItems: "center"
      }}>
      <Text>Hello World!</Text>
    </View>
  )
}
```

```
}  
export default HelloWorldApp;
```



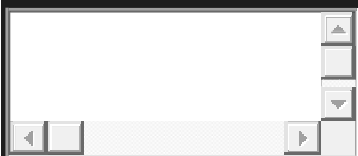
9. Can you write code for Android and IOS in the same codebase?

Yes, you can! React takes care of all of the native component translations.

10. Describe how to re-render a FlatList.

You can re-render a **FlatList** by using the **extraData** property. Let's look at a JavaScript code example:

```
<FlatList  
  data={data}  
  style={FlatListstyles}  
  extraData={this.state}  
  renderItem={this._renderItem}  
>
```



When we pass **extraData={this.state}** to the FlatList, we ensure it'll re-render itself when the selected state changes. Since **FlatList** is also a **PureComponent**, we need to set this prop so it knows to re-render items.

11. What happens when you call **SetState**?

When you call **SetState** in React, the object you passed into it will be merged into the current state of the component. This triggers something called *reconciliation*. Reconciliation aims to update the user interface in the most efficient way possible.

React does this by constructing a **tree** of React elements and compare it to the previous element tree. This shows React the exact changes that occurred so React can make updates in the necessary places.

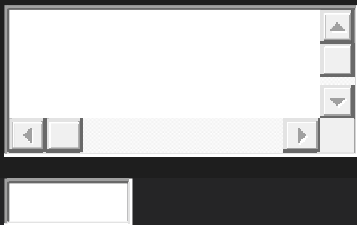
12. How do you style a component in React Native?

You use JavaScript. All of the core components of React accept a prop called `style`. This prop can be a simple JavaScript object. You can also pass an array of different styles.

If you have complex components, it's recommended to use `StyleSheet.create` to establish multiple styles in one place. Here's an example:

```
const styles = StyleSheet.create({
  container: {
    borderRadius: 4,
    borderWidth: 0.5,
    borderColor: '#d6d8da',
  },
  title: {
    fontSize: 19,
    fontWeight: 'bold',
  },
  activeTitle: {
    color: 'red',
  },
});

<View style={styles.container}>
  <Text style={[styles.title, this.props.isActive && styles.activeTitle]} />
</View>
```



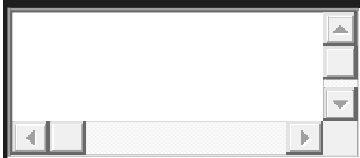
Keep the learning going.

Get started with React without scrubbing through videos or documentation. Educative's text-based learning paths are easy to skim and feature live coding environments, making learning quick and efficient.

13. What are Higher Order Components (HOC) and how do you use them?

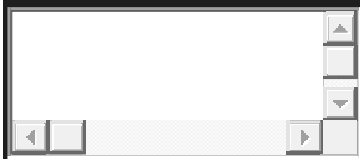
Higher-order components are pure functions that take components and return new components. Their main purpose is to condense and reuse stateful logic across different components. They're considered to be advanced techniques, and they aren't a part of the React API. Instead, they're patterns that emerge from React's compositional nature. Here's an example of a very simple HOC:

```
function simpleHOC(WrappedComponent) {  
  return class extends React.Component {  
    render() {  
      return <WrappedComponent {...this.props}/>;  
    }  
  }  
}
```



This simple React Higher Order Component takes **WrappedComponent** as a parameter, and then it returns a new React component. The new React component has **WrappedComponent** as its child. From this, we can create a new component like this:

```
const NewComponent = simpleHOC(Dog);  
<NewComponent/>
```



Our **NewComponent** can be used exactly like any other component.

14. How do you call a Web API in React Native?

The following code shows an example of how we can call a Web API in React Native:

```
fetch("http://**sampleurl**", {  
  method: "POST",  
  headers: {
```

```

"Accept": "application/json",
"Content-Type": "application/json",
},
body: JSON.stringify({
  username: "educative1",
  password: "educative987",
})
})

```

15. Describe how the Virtual DOM works.

In React Native, the Virtual DOM is a copy of the real DOM. It's a node tree that lists elements along with their attributions, contents, and properties. Whenever our underlying data changes, the Virtual DOM will re-render the UI. After that, the differences between other DOM representations and Virtual DOM representations will be counted, and the real DOM will update.

16. Describe Flexbox along with its most used properties.

Flexbox is a layout mode that enables elements to coordinate and distribute space within containers. It provides a consistent layout on different screen sizes.

The main properties in Flexbox are **flexDirection**, **justifyContent**, and **alignItems**. Let's discuss what each of these properties does:

- **flexDirection**: used to specify the alignment of elements (vertical or horizontal)
- **justifyContent**: used to decide how elements should be distributed inside a given container
- **alignItems**: used to specify the distribution of elements inside a given container along the secondary axis

17. What is the difference between a functional component and a class component?

Functional components are also known as stateless components. Functional components accept props and return HTML. They give solutions without using state, and they can be defined with or without arrow functions.

Here's an example of a functional component in React:

```

import React from "react";
const Friend = (props) => (
  <div>
    <h1> Hi, {props.name} </h1>
  </div>
)

```

```
</div>
```

```
);
```

```
export default Friend;
```

Class components are also known as stateful components. They're ES6 classes that extend the component class from the React library. They implement logic and state. Class components need to have `render()` method when returning HTML. You can pass props to them and access them with `this.props`.

Let's look at an example:

```
import React, {Component} from "react";
```

```
class Friend extends Component {
```

```
  constructor(props) {
```

```
    super(props)
```

```
    this.state = {
```

```
      name: "Erin";
```

```
    }
```

```
  }
```

```
  render() {
```

```
    return (
```

```
      <div>
```

```
        <hi> Hi {this.state.name}</h1>
```

```
      </div>
```

```
    );
```

```
  }
```

```
}
```

```
export default Friend;
```

18. How can you fetch data from a local JSON file in React Native?

There are a couple of ways to fetch data from a local JSON file in React Native. Let's take a look at two options:

Option 1:

```
const customData = require("./customData.json");
```

Option 2:

```
3
```

```
import * as data from "./example.json";  
const word = data.name;  
console.log(word);
```

19. List some ways you can optimize an application.

There are many different ways to optimize an application. Let's take a look at some of our options. We can:

- Compress or convert our raw JSON data instead of just storing it
- Make reduced-sized APK files for CPU architectures
- Optimize native libraries and the number of state operations
- Use key attributes on list items
- Compress images and other graphic elements
- Use Proguard to minimize app size and strip parts of our bytecode along with its dependencies

20. How do you create a stackNavigator in React Native?

Here's how to create stackNavigator in React Native:

```
const AppNavigator = createStackNavigator({  
  Home: {  
    Screen: HomeScreen,  
  },  
});
```

21. What are some causes of memory leaks and how can you detect them for IOS and Android?

Memory leaks can happen if unreleased timers or listeners are added in `componentDidMount` or with closure scope leaks.

To detect memory leaks for IOS, you go to Xcode, Product, then Profile.

To detect memory leaks for Android, you can use the Performance Monitor.

22. How do you install a specific version of React Native?

To install a specific version of React Native, we can use this command:

```
$ react-native init newproject --version react-native@VersionNumber
```

23. Give an example of props being used in React Native.

```
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: "Erin"
}
export default DefaultPropComponent;
```

24. How do you import components in React Native?

Here's how you can import components in React Native:

```
import React from "react";
import { AppRegistry } from "react-native";
import App from "../src/components/importcomponenttutorial";
const App = ( ) => (
  <Title/>
);
AppRegistry.registerComponent("ComponentDemo", ( ) => App);
```

25. How do you add React navigation to React Native?

We have a couple of options. Let's look at the first one:

```
yarn add react-navigation
```

Here's the second:

```
npm install react-navigation
```