**<https://www.educative.io/blog/top-react-native-interview-questions>**

**How is React Native different from ReactJS?**

React Native is a JavaScript framework

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 is a JavaScript library

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](https://www.educative.io/blog/css-interview-questions-cheat-sheet) 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.

### 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 Shift.

### 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.

### 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.

npm install -g create-react-native-app

create-react-native-app AppName

cd AppName

npm start

### 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.

### 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.

### 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."

      };

    }

}

**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.

import React, {Component} from 'react';

import { Text, View } from 'react-native';

export default class App extends Component {

state = {

myState: 'This is a text component, created using state data. It will change or updated on clicking it.'

}

updateState = () => this.setState({myState: 'The state is updated'})

render() {

return (

<View>

<Text onPress={this.updateState}> {this.state.myState} </Text>

</View>

);

}

}

### 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.

// Parent Component

export default class App extends Component {

render () {

return (

<View style={{alignItems: 'center' >

<Heading message={'Custom Heading for Parent class?}/>

</View>

)

}

}

// Child component

export default class Heading extends Component {

render () {

return (

<View style={{alignItems: 'center' >

<Text>{this.props.message}</Text>

</View>

)

}

}

const styles = StyleSheet.create({

welcome: {

fontSize: 30,

}

});

Heading.propTypes = {

message: PropTypes.string

}

Heading.defaultProps = {

message: 'Heading One'

}

### 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](https://www.educative.io/blog/data-structures-trees-java) 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.

### 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>

**What is Redux and when should you use it?**

[Redux](https://www.educative.io/blog/understanding-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

**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.

### 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 components 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.

### 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",

})

})

### 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.

### Describe Flexbox along with its most used properties.

[Flexbox](https://www.educative.io/blog/css-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

### 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>

);

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;

### 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**:

import \* as data from "./example.json";

const word = data.name;

console.log(word);

### 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

### How do you create a stackNavigator in React Native?

Here’s how to create stackNavigator in React Native:

const AppNavigator = createStackNavigator({

    Home: {

        Screen: HomeScreen,

    },

});

### 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.

### 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

## more interview questions to explore

* How do you use TypeScript in React Native?
* How do you store data in AsyncStorage?
* How do you load data from the server-side in React Native?
* What are hooks in React Native?
* What are the differences between Flexbox in the browser vs in React Native?
* What is props drilling?
* What is AppRegistry?
* What does StyleSheet.create do?
* How do you create a stack of screens in React Native?
* How do you import components?
* What is Axios and how does it work?
* What is InteractionManager and how is it used?
* What’s the difference between ScrollView and FlatList?
* What is Flux?
* Are all React components usable in React Native?
* What is the XHR module used for?
* How does React Native handle different screen sizes?
* What are animations in React Native?
* What is the storage system in React Native?
* How do you use Firebase in React Native?
* Explain setNativeProps. How is it used?
* What is Network Security and SSL Pinning?
* How is the entire React Native code processed to show final output on a mobile screen?
* Describe Timers in a React Native application.
* What are the key points to integrating React into an existing Android mobile application?

### What are the advantages of React Native?

* **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.

### 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.

## How are Hot Reloading and Live Reloading in React Native different?

* **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.

## What is the difference between ShadowDOM and VirtualDOM?

**Virtual DOM**

Virtual DOM is about avoiding unnecessary changes to the DOM, which are expensive performance-wise, because changes to the DOM usually cause re-rendering of the page. Virtual DOM also allows to collect several changes to be applied at once, so not every single change causes a re-render, but instead re-rendering only happens once after a set of changes was applied to the DOM.

**Shadow DOM**

Shadow dom is mostly about encapsulation of the implementation. A single custom element can implement more-or-less complex logic combined with more-or-less complex DOM. An entire web application of arbitrary complexity can be added to a page by an import and <body><my-app></my-app> but also simpler reusable and composable components can be implemented as custom elements where the internal representation is hidden in the shadow DOM like <date-picker></date-picker>.