**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.

### 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.

### 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 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",

})

})