**WEB vs NATIVE**

<https://youtu.be/R0Qf5a0Vxho>

Native applications look and "feel" different because they are fundamentally different. Even though we're using the same React principles that you've learned throughout this program, keep in mind that this is no longer the web! While some of these distinctions are more apparent (e.g., the development process, access to native features, how users get updates, etc.), there are some key differences that we'll be taking a deep dive into during this course.

For one, native apps often leverage **animations** to help create a great user experience. Animations such as button effects, screen transitions, and other visual feedback may be subtle, but they support continuity and guidance in the apps you build. They all function to dynamically tell a story about how your application works. Without animations, an application can feel like just a collection of static screens. For now, stay tuned; we'll be checking out animations in-depth during Lesson 5.

Another key difference between native and web applications is in **navigation**. Recall that React Router's Route component allows us to map a URL to a specific UI component. In React Native, routers function as a stack; that is, individual screens are "pushed" and "popped" as needed. We'll look at routing more closely later in Lesson 4.

## Android vs. iOS

Not only are there fundamental differences between native apps and web apps, you'll also find differences between how native platforms (iOS and Android) look and feel as well. Perhaps the most apparent are the distinct design philosophies on each platform: Android apps utilize Google's [Material Design](https://material.io/guidelines/material-design/introduction.html), while iOS apps take advantage of Apple's [Human Interface Design](https://developer.apple.com/ios/human-interface-guidelines/overview/themes/). When designing mobile applications, it's important to your users that an iOS app feels like an iOS app, and an Android app feelslike an Android app.

Navigation between screens feels distinct between Android and iOS as well. Android devices have access to a **navigation bar** at the bottom of the screen, which allows users to go back to the previous screen (among other features). On iOS, the approach is different: there is no such universal navigation bar! When building the UI for an iOS application, it is important to include a back button (perhaps on a custom [navigation bar](https://developer.apple.com/ios/human-interface-guidelines/bars/navigation-bars/)) to help guide users through your app.

One more key difference between Android and iOS involves tab navigation. iOS apps include [tab bars](https://developer.apple.com/ios/human-interface-guidelines/bars/tab-bars/)at the bottom of the app's screen, allowing for convenient access to different portions of the app. Likewise, Android apps include them as well; however tabs are distinctly located [at the top of the screen](https://material.io/guidelines/components/tabs.html). Both allow access to high-level content, and we'll explore React Native's **TabNavigator** in closer detail in Lesson 4.

## Summary

When developing your React Native projects, keep in mind that you're designing for a different experience than that of web applications. Mobile applications look and feel different due to fundamental differences, such as subtle animations that build a sense of continuity for your users. Differences exist between Android and iOS as well, especially in their design philosophies and navigation. We'll look at some fundamental components that make up React Natives apps in the next section!

### Further Research

* [iOS Interface Essentials](https://developer.apple.com/ios/human-interface-guidelines/overview/interface-essentials/)