Version: 6.x

Pitch & anti-pitch

It's useful when considering whether or not to use a project to understand the tradeoffs that the developers of the project made when building it. What problems does it explicitly try to solve for you, and which ones does it ignore? What are the current limitations of the project and common problems that people encounter? These are the kinds of questions that we believe you should have answers to when making an important technology decision for your project, and so we have documented answers to these questions as best we can here, in the form of a "pitch" (why you should use it) and "anti-pitch" (why you should not use it). Please submit a pull request if you believe we have omitted important information!

Pitch

- React Navigation doesn't include any native code in the library itself, but we use many native libraries such as Screens, Reanimated, Gesture Handler etc. to implement performant animations and gestures. Depending on the navigator, many UI components are written in JavaScript on top of React Native primitives. This has a lot of benefits:
  - Easy OTA updates
  - Debuggable
  - Customizable
- Most apps heavily customize navigation, to do this with an API that wraps native navigation you will need to write a lot of native code. In React Navigation, we provide navigators written fully with JavaScript (e.g. Stack Navigator) and navigators implemented on top of platform navigation primitives (e.g. Native Stack Navigator). This lets you pick the navigators suitable for your use case, depending on whether you want native platform behavior or full customizability.
- It's possible to write your own navigators that integrate cleanly with standard navigators, or to fork the standard navigators and create your own version of them with the exact look and feel you want in your app.

Anti-pitch

- Improvements may require breaking changes. We are working to make "easy things easy and hard things possible" and this may require us to change the API at times.
- Some navigators don't directly use the native navigation APIs on iOS and Android; rather, they use the lowest level pieces and then re-creates some subset of the APIs on top. This is a conscious choice in order to make it possible for users to customize any part of the navigation experience (because it's implemented in JavaScript) and to be able to debug issues that they encounter without needing to learn Objective C / Swift / Java / Kotlin.
  - If you need the exact platform behavior, you can choose to use the navigators that use native platform primitives (e.g. Native Stack Navigator), or use a different navigation library which provides fully native navigation APIs (e.g. React Native Navigation).
- There are other limitations which you may want to consider, see Limitations for more details.

Edit this page