Navigating Between Screens

Mobile apps are rarely made up of a single screen. Managing the presentation of, and transition between, multiple screens is typically handled by what is known as a navigator.

This guide covers the various navigation components available in React Native. If you are getting started with navigation, you will probably want to use React Navigation. React Navigation provides a straightforward navigation solution, with the ability to present common stack navigation and tabbed navigation patterns on both Android and iOS.

If you're integrating React Native into an app that already manages navigation natively, or looking for an alternative to React Navigation, the following library provides native navigation on both platforms: react-native-navigation.

React Navigation

The community solution to navigation is a standalone library that allows developers to set up the screens of an app with a few lines of code.

Installation and setup

First, you need to install them in your project:

```
npm install @react-navigation/native @react-navigation/native-stack
```

Next, install the required peer dependencies. You need to run different commands depending on whether your project is an Expo managed project or a bare React Native project.

• If you have an Expo managed project, install the dependencies with expo:

```
npx expo install react-native-screens react-native-safe-area-context
```

• If you have a bare React Native project, install the dependencies with npm:

```
npm install react-native-screens react-native-safe-area-context
```

For iOS with bare React Native project, make sure you have <u>CocoaPods</u> installed. Then install the pods to complete the installation:

```
cd ios
pod install
cd ..
```

(i) NOTE

You might get warnings related to peer dependencies after installation. They are usually caused by incorrect version ranges specified in some packages. You can safely ignore most warnings as long as your app builds.

Now, you need to wrap the whole app in NavigationContainer. Usually you'd do this in your entry file, such as index.js or App.js:

Now you are ready to build and run your app on the device/simulator.

Usage

Now you can create an app with a home screen and a profile screen:

```
import * as React from 'react';
import {NavigationContainer} from '@react-navigation/native';
import {createNativeStackNavigator} from '@react-navigation/native-stack';
const Stack = createNativeStackNavigator();
const MyStack = () => {
  return (
    <NavigationContainer>
      <Stack.Navigator>
        <Stack.Screen
          name="Home"
          component={HomeScreen}
          options={{title: 'Welcome'}}
       />
        <Stack.Screen name="Profile" component={ProfileScreen} />
      </Stack.Navigator>
    </NavigationContainer>
  );
};
```

In this example, there are 2 screens (Home and Profile) defined using the Stack.Screen component. Similarly, you can define as many screens as you like.

You can set options such as the screen title for each screen in the options prop of Stack.Screen.

Each screen takes a component properthat is a React component. Those components receive a properalled navigation which has various methods to link to other screens. For example, you can use navigation.navigate to go to the Profile screen:

```
);
};
const ProfileScreen = ({navigation, route}) => {
  return <Text>This is {route.params.name}'s profile</Text>;
};
```

This native-stack navigator uses the native APIs: UINavigationController on iOS and Fragment on Android so that navigation built with createNativeStackNavigator will behave the same and have the same performance characteristics as apps built natively on top of those APIs.

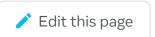
React Navigation also has packages for different kind of navigators such as tabs and drawer. You can use them to implement various patterns in your app.

For a complete intro to React Navigation, follow the React Navigation Getting Started Guide.

Is this page useful?







Last updated on Aug 24, 2023