



Fundamentals

Getting started

Version: 6.x

Getting started

What follows within the *Fundamentals* section of this documentation is a tour of the most important aspects of React Navigation. It should cover enough for you to know how to build your typical small mobile application, and give you the background that you need to dive deeper into the more advanced parts of React Navigation.

Pre-requisites

If you're already familiar with JavaScript, React and React Native, then you'll be able to get moving with React Navigation quickly! If not, we highly recommend you to gain some basic knowledge first, then come back here when you're done.

Here are some resources to help you out:

1. [React Native Express](#) (Sections 1 to 4)
2. [Main Concepts of React](#)
3. [React Hooks](#)
4. [React Context](#) (Advanced)

Minimum requirements

- `react-native` \geq 0.63.0
- `expo` \geq 41 (if you use [Expo](#))
- `typescript` \geq 4.1.0 (if you use [TypeScript](#))

Installation

Install the required packages in your React Native project:

npm **Yarn**

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

React Navigation is made up of some core utilities and those are then used by navigators to create the navigation structure in your app. Don't worry too much about this for now, it'll become clear soon enough! To frontload the installation work, let's also install and configure dependencies used by most navigators, then we can move forward with starting to write some code.

The libraries we will install now are `react-native-screens` and `react-native-safe-area-context`. If you already have these libraries installed and at the latest version, you are done here! Otherwise, read on.

Installing dependencies into an Expo managed project

In your project directory, run:

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

This will install versions of these libraries that are compatible.

You can now continue to "Hello React Navigation" to start writing some code.

Installing dependencies into a bare React Native project

In your project directory, run:

npm **Yarn**

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

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.

From React Native 0.60 and higher, [linking is automatic](#). So you **don't need to run** `react-native link`.

If you're on a Mac and developing for iOS, you need to install the pods (via [Cocoapods](#)) to complete the linking.

```
npx pod-install ios
```

`react-native-screens` package requires one additional configuration step to properly work on Android devices. Edit `MainActivity.java` file which is located in `android/app/src/main/java/<your package name>/MainActivity.java`.

Add the highlighted code to the body of `MainActivity` class:

```
public class MainActivity extends ReactActivity {  
    // ...  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(null);  
    }  
    // ...  
}
```

and make sure to add the following import statement at the top of this file below your package statement:

```
import android.os.Bundle;
```

This change is required to avoid crashes related to View state being not persisted consistently across Activity restarts.

Note: When you use a navigator (such as stack navigator), you'll need to follow the installation instructions of that navigator for any additional dependencies. If you're getting an error "Unable to resolve module", you need to install that module in your project.

Wrapping your app in `NavigationContainer`

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


```
import * as React from 'react';
import { NavigationContainer } from '@react-navigation/native';

export default function App() {
  return (
    <NavigationContainer>{/* Rest of your app code */}</NavigationContainer>
  );
}
```

Note: In a typical React Native app, the `NavigationContainer` should be only used once in your app at the root. You shouldn't nest multiple `NavigationContainer`s unless you have a specific use case for them.

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

Continue to "Hello React Navigation" to start writing some code.

 [Edit this page](#)