Setting up the development environment

This page will help you install and build your first React Native app.

If you are new to mobile development, the easiest way to get started is with Expo Go. Expo is a set of tools and services built around React Native and, while it has many features, the most relevant feature for us right now is that it can get you writing a React Native app within minutes. You will only need a recent version of Node.js and a phone or emulator. If you'd like to try out React Native directly in your web browser before installing any tools, you can try out Snack.

If you are already familiar with mobile development, you may want to use React Native CLI. It requires Xcode or Android Studio to get started. If you already have one of these tools installed, you should be able to get up and running within a few minutes. If they are not installed, you should expect to spend about an hour installing and configuring them.

Expo Go Quickstart **React Native CLI Quickstart**

Follow these instructions if you need to build native code in your project. For example, if you are integrating React Native into an existing application, or if you ran "prebuild" from Expo to generate your project's native code, you'll need this section.

The instructions are a bit different depending on your development operating system, and whether you want to start developing for iOS or Android. If you want to develop for both Android and iOS, that's fine - you can pick one to start with, since the setup is a bit different.

Development OS

macOS Windows Linux

Target OS

Android **iOS**

Installing dependencies

You will need Node, Watchman, the React Native command line interface, Xcode and CocoaPods.

While you can use any editor of your choice to develop your app, you will need to install Xcode in order to set up the necessary tooling to build your React Native app for iOS.

Node & Watchman

We recommend installing Node and Watchman using Homebrew. Run the following commands in a Terminal after installing Homebrew:

```
brew install node
brew install watchman
```

If you have already installed Node on your system, make sure it is Node 16 or newer.

Watchman is a tool by Facebook for watching changes in the filesystem. It is highly recommended you install it for better performance.

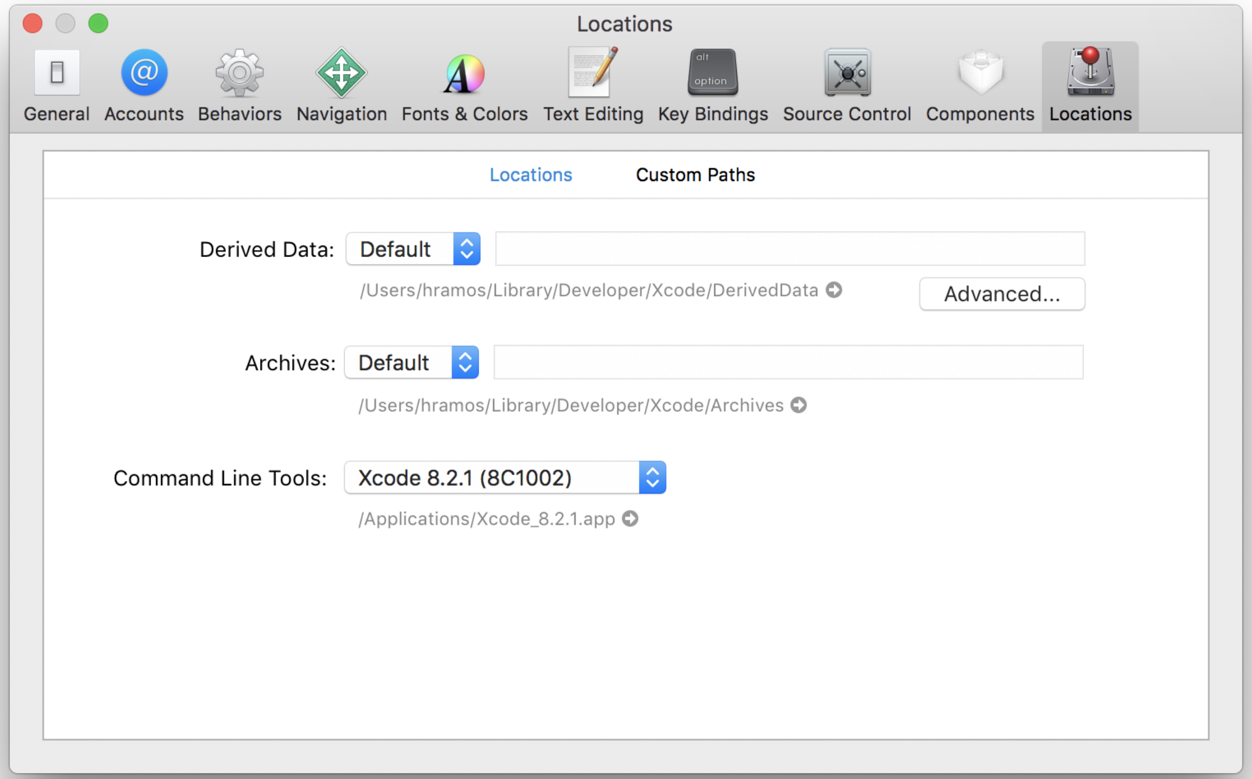
Xcode

The easiest way to install Xcode is via the Mac App Store. Installing Xcode will also install the iOS Simulator and all the necessary tools to build your iOS app.

If you have already installed Xcode on your system, make sure it is version 10 or newer.

Command Line Tools

You will also need to install the Xcode Command Line Tools. Open Xcode, then choose **Settings... (or Preferences...)** from the Xcode menu. Go to the Locations panel and install the tools by selecting the most recent version in the Command Line Tools dropdown.



Derived Data: Default
/Users/hramos/Library/Developer/Xcode/DerivedData

Archives: Default
/Users/hramos/Library/Developer/Xcode/Archives

Command Line Tools: Xcode 8.2.1 (8C1002)
/Applications/Xcode_8.2.1.app

Installing an iOS Simulator in Xcode

To install a simulator, open **Xcode > Settings... (or Preferences...)** and select the **Platforms (or Components)** tab. Select a simulator with the corresponding version of iOS you wish to use.

CocoaPods

CocoaPods is one of the dependency management system available for iOS. CocoaPods is a Ruby gem. You can install CocoaPods using the version of Ruby that ships with the latest version of macOS.

For more information, please visit CocoaPods Getting Started guide.

React Native Command Line Interface

React Native has a built-in command line interface. Rather than install and manage a specific version of the CLI globally, we recommend you access the current version at runtime using `npx`, which ships with Node.js. With `npx react-native <command>`, the current stable version of the CLI will be downloaded and executed at the time the command is run.

Creating a new application

> If you previously installed a global `react-native-cli` package, please remove it as it may cause unexpected issues:
>
> ```
> npm uninstall -g react-native-cli @react-native-community/cli
> ```

You can use React Native's built-in command line interface to generate a new project. Let's create a new React Native project called "AwesomeProject":

```
npx react-native@latest init AwesomeProject
```

This is not necessary if you are integrating React Native into an existing application, if you "ejected" from Expo, or if you're adding iOS support to an existing React Native project (see Integration with Existing Apps). You can also use a third-party CLI to init your React Native app, such as Ignite CLI.

> **INFO**
>
> If you are having trouble with iOS, try to reinstall the dependencies by running:
>
> 1. `cd ios` to navigate to the `ios` folder.
> 2. `bundle install` to install Bundler
> 3. `bundle exec pod install` to install the iOS dependencies managed by CocoaPods.

[Optional] Using a specific version or template

If you want to start a new project with a specific React Native version, you can use the `--version` argument:

```
npx react-native@X.XX.X init AwesomeProject --version X.XX.X
```

You can also start a project with a custom React Native template with the `--template` argument.

> **Note** If the above command is failing, you may have old version of `react-native` or `react-native-cli` installed globally on your pc. Try uninstalling the cli and run the cli using `npx`.

[Optional] Configuring your environment

Starting from React Native version 0.69, it is possible to configure the Xcode environment using the `.xcode.env` file provided by the template.

The `.xcode.env` file contains an environment variable to export the path to the `node` executable in the `NODE_BINARY` variable. This is the **suggested approach** to decouple the build infrastructure from the system version of `node`. You should customize this variable with your own path or your own `node` version manager, if it differs from the default.

On top of this, it's possible to add any other environment variable and to source the `.xcode.env` file in your build script phases. If you need to run script that requires some specific environment, this is the **suggested approach**: it allows to decouple the build phases from a specific environment.

> **INFO**
>
> If you are already using NVM (a command which helps you install and switch between versions of Node.js) and zsh, you might want to move the code that initialize NVM from your `~/.zshrc` into a `~/.zshenv` file to help Xcode find your Node executable:

```
export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"  # This loads nvm
```

> You might also want to ensure that all "shell script build phase" of your Xcode project, is using `/bin/zsh` as its shell.

Running your React Native application

Step 1: Start Metro

First, you will need to start Metro, the JavaScript bundler that ships with React Native. Metro "takes in an entry file and various options, and returns a single JavaScript file that includes all your code and its dependencies."—Metro Docs

To start Metro, run following command inside your React Native project folder:

npm **Yarn**

```
yarn start
```

> If you're familiar with web development, Metro is a lot like webpack—for React Native apps. Unlike Swift or Objective-C, JavaScript isn't compiled—and neither is React Native. Bundling isn't the same as compiling, but it can help improve startup performance and translate some platform-specific JavaScript into more widely supported JavaScript.

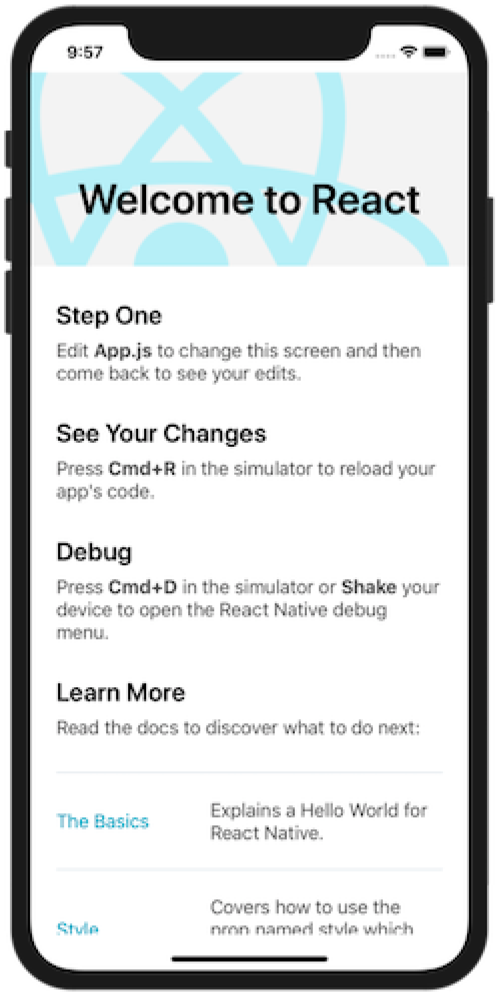
Step 2: Start your application

Let Metro Bundler run in its own terminal. Open a new terminal inside your React Native project folder. Run the following:

npm **Yarn**

```
yarn ios
```

You should see your new app running in the iOS Simulator shortly.



Welcome to React

Step One

Edit **App.js** to change this screen and then come back to see your edits.

See Your Changes

Press **Cmd+R** in the simulator to reload your app's code.

Debug

Press **Cmd+D** in the simulator or **Shake** your device to open the React Native debug menu.

Learn More

Read the docs to discover what to do next:

The Basics	Explains a Hello World for React Native.
Style	Covers how to use the prop named style which

This is one way to run your app. You can also run it directly from within Xcode.

> If you can't get this to work, see the Troubleshooting page.

Running on a device

The above command will automatically run your app on the iOS Simulator by default. If you want to run the app on an actual physical iOS device, please follow the instructions here.

Modifying your app

Now that you have successfully run the app, let's modify it.

- Open `App.tsx` in your text editor of choice and edit some lines.
- Hit `Cmd ⌘` + `R` in your iOS Simulator to reload the app and see your changes!

That's it!

Congratulations! You've successfully run and modified your first React Native app.

Now what?

- If you want to add this new React Native code to an existing application, check out the Integration guide.

If you're curious to learn more about React Native, check out the Introduction to React Native.

Is this page useful? 👍 👎

Edit this page

Last updated on ***Aug 29, 2023***