Running On Device

It's always a good idea to test your app on an actual device before releasing it to your users. This document will guide you through the necessary steps to run your React Native app on a device and to get it ready for production.

> **INFO**
>
> If you used `create-expo-app` to set up your project, you can run your app on a device in Expo Go by scanning the QR code that is displayed when you run `npm start`. Refer to the Expo guide for running your project on your device for more information.

Android | iOS

Running your app on Android devices

Development OS

macOS | Windows | Linux

1. Enable Debugging over USB

Most Android devices can only install and run apps downloaded from Google Play, by default. You will need to enable USB Debugging on your device in order to install your app during development.

To enable USB debugging on your device, you will first need to enable the "Developer options" menu by going to **Settings** → **About phone** → **Software information** and then tapping the `Build number` row at the bottom seven times. You can then go back to **Settings** → **Developer options** to enable "USB debugging".

2. Plug in your device via USB

Let's now set up an Android device to run our React Native projects. Go ahead and plug in your device via USB to your development machine.

Next, check the manufacturer code by using `lsusb` (on mac, you must first install lsusb). `lsusb` should output something like this:

```
$ lsusb
Bus 002 Device 002: ID 8087:0024 Intel Corp. Integrated Rate Matching Hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 22b8:2e76 Motorola PCS
Bus 001 Device 002: ID 8087:0024 Intel Corp. Integrated Rate Matching Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

These lines represent the USB devices currently connected to your machine.

You want the line that represents your phone. If you're in doubt, try unplugging your phone and running the command again:

```
$ lsusb
Bus 002 Device 002: ID 8087:0024 Intel Corp. Integrated Rate Matching Hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 002: ID 8087:0024 Intel Corp. Integrated Rate Matching Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

You'll see that after removing the phone, the line which has the phone model ("Motorola PCS" in this case) disappeared from the list. This is the line that we care about.

```
Bus 001 Device 003: ID 22b8:2e76 Motorola PCS
```

From the above line, you want to grab the first four digits from the device ID:

```
22b8:2e76
```

In this case, it's `22b8`. That's the identifier for Motorola.

You'll need to input this into your udev rules in order to get up and running:

```
echo 'SUBSYSTEM=="usb", ATTR{idVendor}=="22b8", MODE="0666", GROUP="plugdev"' | sudo tee
/etc/udev/rules.d/51-android-usb.rules
```

Make sure that you replace `22b8` with the identifier you get in the above command.

Now check that your device is properly connecting to ADB, the Android Debug Bridge, by running `adb devices`.

```
$ adb devices
List of devices attached
emulator-5554 offline   # Google emulator
14ed2fcc device         # Physical device
```

Seeing `device` in the right column means the device is connected. You must have **only one device connected** at a time.

3. Run your app

From the root of your project, type the following in your command prompt to install and launch your app on the device:

npm **Yarn**

```
yarn android
```

> If you get a "bridge configuration isn't available" error, see Using adb reverse.

> Hint: You can also use the `React Native CLI` to generate and run a `release` build (e.g. from the root of your project: `yarn android --mode release`).

Connecting to the development server

You can also iterate quickly on a device by connecting to the development server running on your development machine. There are several ways of accomplishing this, depending

on whether you have access to a USB cable or a Wi-Fi network.

Method 1: Using adb reverse (recommended)

You can use this method if your device is running Android 5.0 (Lollipop) or newer, it has USB debugging enabled, and it is connected via USB to your development machine.

Run the following in a command prompt:

```
$ adb -s <device name> reverse tcp:8081 tcp:8081
```

To find the device name, run the following adb command:

```
$ adb devices
```

You can now enable Live reloading from the Dev Menu. Your app will reload whenever your JavaScript code has changed.

Method 2: Connect via Wi-Fi

You can also connect to the development server over Wi-Fi. You'll first need to install the app on your device using a USB cable, but once that has been done you can debug wirelessly by following these instructions. You'll need your development machine's current IP address before proceeding.

Open a terminal and type `/sbin/ifconfig` to find your machine's IP address.

1. Make sure your laptop and your phone are on the **same** Wi-Fi network.
2. Open your React Native app on your device.
3. You'll see a red screen with an error. This is OK. The following steps will fix that.
4. Open the in-app Dev Menu.
5. Go to **Dev Settings** → **Debug server host & port for device**.
6. Type in your machine's IP address and the port of the local dev server (e.g. `10.0.1.1:8081`).
7. Go back to the **Dev Menu** and select **Reload JS**.

You can now enable Live reloading from the Dev Menu. Your app will reload whenever your JavaScript code has changed.

Building your app for production

You have built a great app using React Native, and you are now itching to release it in the Play Store. The process is the same as any other native Android app, with some additional considerations to take into account. Follow the guide for generating a signed APK to learn more.

Is this page useful? 

Edit this page

*Last updated on **Aug 10, 2023***