# Setting up ReactNative Dev environment on Windows

## Setting up reactnative / android dev environment

### software dependencies:

Install the following:

**node.js**

ReactNative CLI: **npm install -g react-native-cli**

**Watchman.exe**

**see note below regarding troubleshooting watchman errors*

http://bit.ly/watchmanwinalpha

https://facebook.github.io/watchman/docs/install.html

(copy the watchman.exe file to somewhere in your Path).

*Note: Watchman's job is to watch for files which have changed, so that Hot-reloading can rebuild relevant parts of your project when you make changes to your source. I have had some weird problems with Watchman. Hot reloading (an important feature of the reactnative development process) is a bit intermittent. When it works - it's great. when it doesn't, it's extremely frustrating. I am still trying to determine why it works sometimes and not other times on my system. I will document my findings here when and if I get to the bottom of the problem*

Android Studio (requires Java, and takes about 400 years to install !). Android Studio will put a copy of the Android SDK in your user profile !

### setting up environment variables:

*(this will tell the reactiveNative build tools where the SDK is located. You can do this via Control Panel / System and Security / Advanced / Environment Variables. Note: edit the system-wide variables, not the user ones )*

set **ANDROID_HOME** environment variable to **C:\Users\username\AppData\Local\Android\sdk** , or wherever the Android SDK is installed.

(by default, Android Studio puts the SDK inside user profile. It's about 5GB. I moved it off my C: drive to **E:\AndroidSDK** . There is a menu item inside Andoid Studio to set the location: Settings->Appearance & Behavior -> System Settings -> Android SDK)

Then, add:

**%ANDROID_HOME%\tools**

and

**%ANDROID_HOME%\platform-tools**

to your **Path** variable.

*(this enables you to use the "Android" command and other tools from console)*

## Setting up an Android simulator:

Create an AVD (Android Virtual Device) in **Android Virtual Device Manager**:

**android avd**

you may need to install a system image for the device you created. This is done in Android sdk manager (**android sdk**). For example, if you wanted to emulate an older Android 4.2.2 device, you need to install the system images for that device version. (Usually, SDK will only have the system images for the latest android version by default).

**android sdk**

OR alternatively, use the GUI in **Android Studio**

Note: Although the devices you are targeting will likely have ARM cpus in them, when developing on an intel computer (which is pretty much all computers these days) you will get much better emulator performance when emulating an android device with an Intel processor. To do this, you need to install Intel HAXM

(you will probably be asked to switch-off Hyper-V and reboot)

*Note: the "rage shake" menu can be accessed by **Ctrl-M** (this works on AVD emulator and GenyMotion)*

---

**You can only run one android device (virtual or Real) at a time !**

---

## Running on a real device:

Enable USB debugging on your device by going to Settings > Developer options.

Note: On Android 4.2 and newer, Developer options is hidden by default. To make it available, go to Settings > About phone and **tap Build number seven times**. Return to the previous screen to find Developer options.

The phone will need network connectivity via wireless to the development server.

To allow incoming connections, ensure that the dev server port is open on the dev computer's firewall (eg allow 192.168.15.100:8081)

To set the phone to use the IP and port of the dev server, **shake the phone while app is running** to open dev settings

---

## Setting up a GenyMotion Android emulator

The Genymotion Android Emulator is faster than the AVD emulator. However, it uses it's own adb server which causes a conflict with the SDK adb server. To fix this, in Genymotion, open Settings/ADB and choose 'Use custom Android SDK Tools' then set the path to the Android SDK

## Creating and running a project:

(*from* ReactNative GsG ):

**react-native init AwesomeProject**

**cd AwesomeProject**

**react-native run-android**

If everything is set up correctly, you should see your new app running in your Android emulator shortly.

A common issue is that the packager is not started automatically when you run react-native run-android. You can start it manually using react-native start.

If you hit a ERROR Watcher took too long to load on Windows, try increasing the timeout in this file (under your node_modules/react-native/).

If you get this error:

```
failed to find Build Tools revision 23.0.1
```

... then Android SDK manager (**android sdk**) and install the correct version of build tools (it's quite fussy about having the right version installed).

## Forcing a rebundle of debug build

This may be necessary if **react-native run-android** does not "notice" that you have made changes to your code, (and keeps skipping the **bundleDebugJsAndAssets** stage) ! It runs the **react-native bundle** command with the correct parameters to bundle the app.

```
react-native bundle --platform android --dev true --reset-cache --entry-file
index.android.js --bundle-output
"AbsolutePathToProject\app\build\intermediates\assets\debug\index.android.bundle" -
-assets-dest
"AbsolutePathToProject\android\app\build\intermediates\res\merged\debug"
```

---

## *Troubleshooting watchman erors

As noted above, file-watching with watchman.exe (which hot-reloading depends upon ...) can be a bit temperamental. I have noticed that it is often necessary to start the packager a few times before watchman will work properly. The following output snippets from **react-native start** show the difference between a good start and a bad one:

Bad Start:

```
React packager ready.

FileWatcher {
  domain: null,
  _events: { all: [Function: bound _onFileChange] },
  _eventsCount: 1,
  _maxListeners: undefined,
  _watcherByRoot: {},
  _loading: Promise { <pending> } }
Failed to build DependencyGraph: Watchman error: Watcher took too long to load
(WatchmanWatcher)
Try running `watchman version` from your terminal
https://facebook.github.io/watchman/docs/troubleshooting.html. Make sure watchman
is running for this project. See
https://facebook.github.io/watchman/docs/troubleshooting.html.
Error: Watchman error: Watcher took too long to load (WatchmanWatcher)
Try running `watchman version` from your terminal
https://facebook.github.io/watchman/docs/troubleshooting.html. Make sure watchman
is running for this project. See
https://facebook.github.io/watchman/docs/troubleshooting.html.
```

Good Start:

```
React packager ready.

FileWatcher {
  domain: null,
  _events: { all: [Function: bound _onFileChange] },
  _eventsCount: 1,
  _maxListeners: undefined,
  _watcherByRoot: {},
  _loading: Promise { <pending> } }
[11:27:00 AM] <END>   Crawling File System (1678ms)
[11:27:00 AM] <START> Building in-memory fs for JavaScript
[11:27:01 AM] <END>   Building in-memory fs for JavaScript (634ms)
[11:27:01 AM] <START> Building in-memory fs for Assets
[11:27:02 AM] <END>   Building in-memory fs for Assets (583ms)
[11:27:02 AM] <START> Building Haste Map
[11:27:02 AM] <START> Building (deprecated) Asset Map
[11:27:02 AM] <END>   Building (deprecated) Asset Map (173ms)
[11:27:02 AM] <END>   Building Haste Map (428ms)
[11:27:02 AM] <END>   Building Dependency Graph (3342ms)
```

```
[11:27:06 AM] <START> request:/index.android.bundle?
platform=android&dev=true&hot=true&minify=false
[11:27:06 AM] <START> find dependencies
transformed 634/634 (100%)
[11:27:08 AM] <END>   find dependencies (1699ms)
[11:27:08 AM] <END>   request:/index.android.bundle?
platform=android&dev=true&hot=true&minify=false (1850ms)
[Hot Module Replacement] Client connected
[Hot Module Replacement] File change detected (13:4:25:262)
[1:04:58 PM] <START> find dependencies
[1:04:58 PM] <END>   find dependencies (2ms)
[Hot Module Replacement] Sending HMR update to client (13:4:58:985)
```