

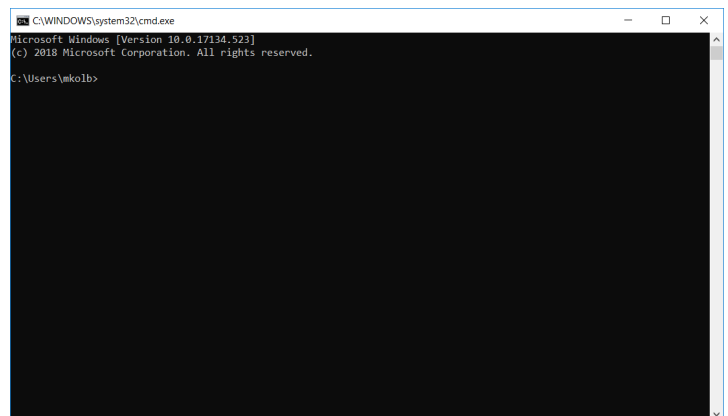
Mobile Applications

Cordova Install Lab 4

This lab contains a checkpoint at the end. Please contact any lecturing staff when you reach this point to receive your credit.

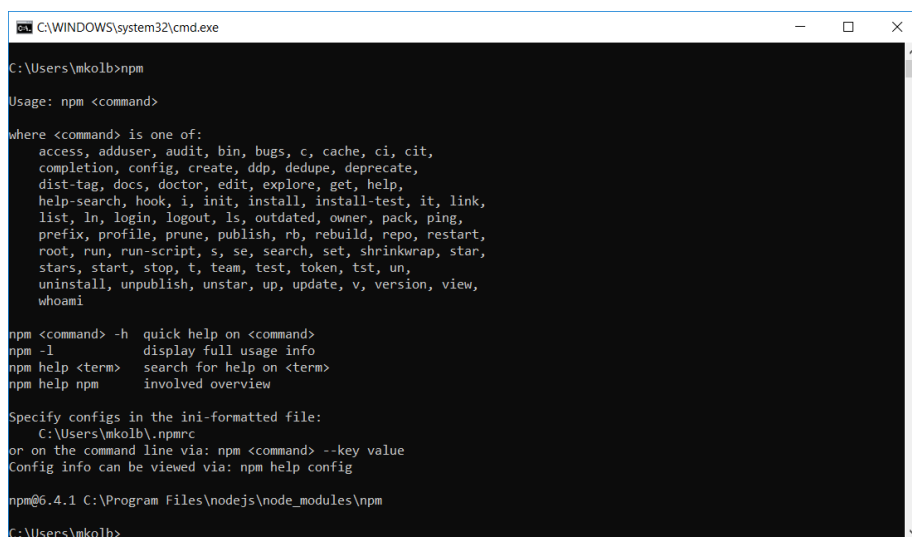
The basic aim of this lab is to set up a working Cordova environment and to run a basic Cordova app in the browser and on an Android phone emulator.

Most of the tools and commands for Cordova are executed from the command line interface. You can start a command prompt from the Windows Start menu by typing `cmd`. This should open up a command prompt window (see example at right).



As you may remember from the lectures Cordova relies on Node.js. Node.js is a JavaScript runtime engine built on Chrome's V8 JavaScript engine. As an asynchronous event driven JavaScript runtime, Node is designed to build scalable network applications. Luckily, Node.js is already installed for you on the lab machines. However, if you wish to install it on your own machine, you can download it from <https://nodejs.org/en/download/>.

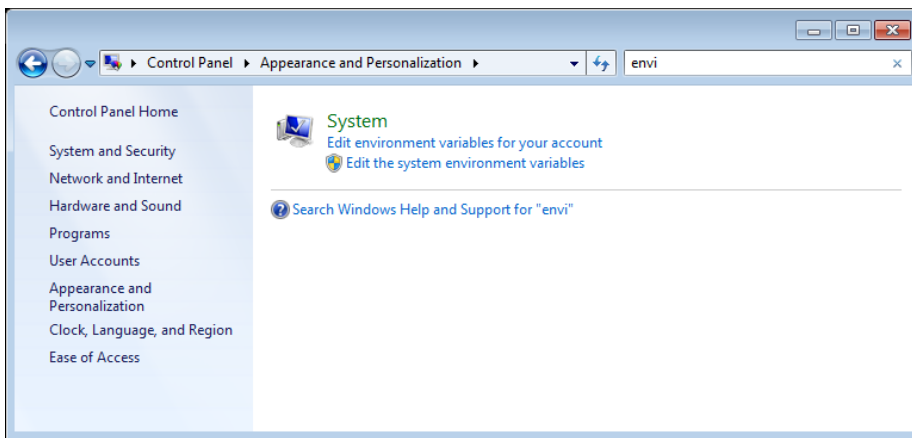
Just to check that Node.js is indeed installed correctly, try and execute the `npm` command (Node Package Manager). You should get a response similar to the screenshot below.



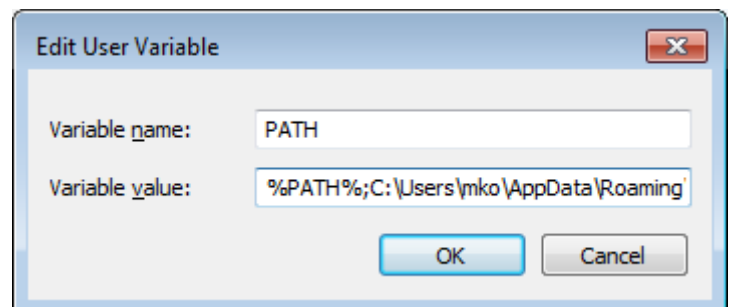
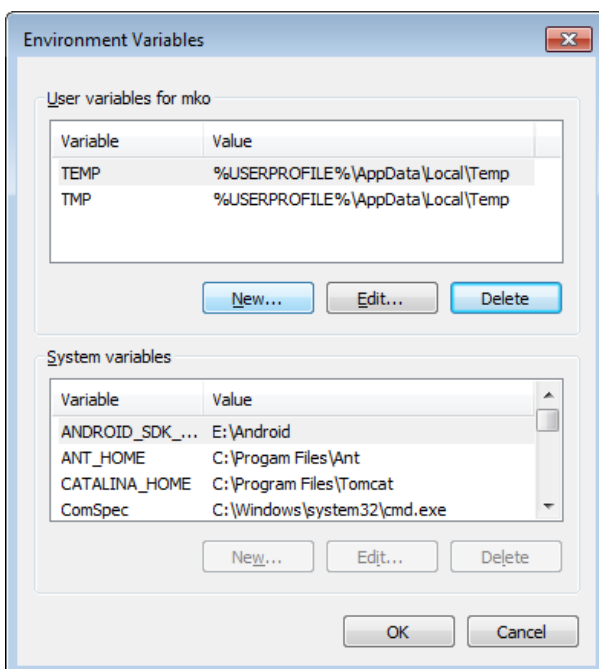
The next step is to install Cordova itself. Install the cordova module using npm utility of Node.js. The cordova module will automatically be downloaded by the npm utility. The full command is:
`npm install -g cordova`

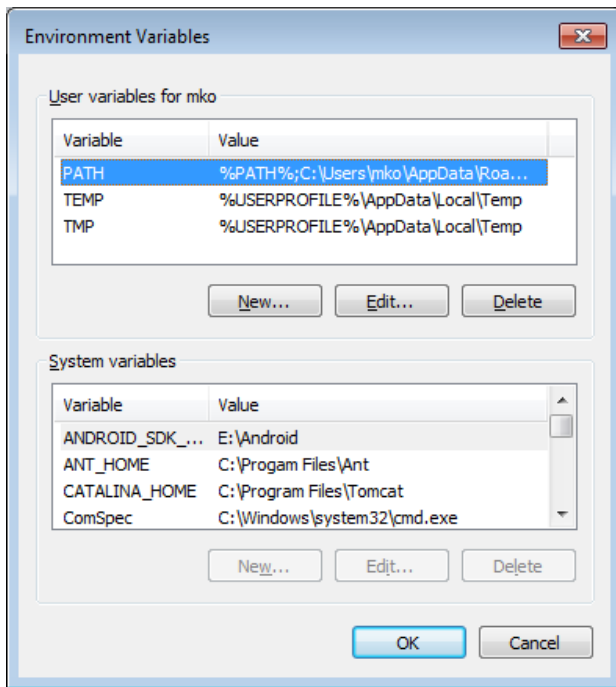
This will take just a minute or two and once finished you will get the prompt back. For convenience, you probably want to set the PATH variable so that the cordova command is automatically found by Windows. (At the moment if you type in cordova in the prompt you will get a not very helpful error message).

Again from the Windows Start menu, execute the Control Panel (at the right of the window popping up). In the Control Panel, use the search box and type in 'Environmental'. You probably don't need all the letters until a single item appears.



Click on 'Edit environmental variables for your account'. And in the window popping up click on New in the top half of the window (User variables for account ...). Add a variable PATH. And the value: %PATH%;C:\Users\mko\AppData\Roaming\npm (replace mko with your own username). Then click OK and verify that the variable has been added.





Unfortunately, newly set environmental variables do not apply to already open command windows. So you should close the window you already have (type `exit` at the prompt). And then start a new one. You should now be able to type in `cordova` and get an output similar to the window below.

```

C:\Windows\system32\cmd.exe

Aliases
  build -> cordova prepare && cordova compile
  emulate -> cordova run --emulator

Options
  -v, --version ..... prints out this utility's version
  -d, --verbose ..... debug mode produces verbose log output
for all activity,
  --no-update-notifier ..... disables check for CLI updates
  --nohooks ..... suppress executing hooks
                        (taking RegExp hook patterns as parameters)

Examples
  cordova create myApp org.apache.cordova.myApp myApp
  cordova plugin add cordova-plugin-camera
  cordova platform add android
  cordova plugin add cordova-plugin-camera --nosave
  cordova platform add android --nosave
  cordova requirements android
  cordova build android --verbose
  cordova run android
  cordova build android --release -- --keystore="..\android.keystore" --storePassword=android --alias=mykey
  cordova config ls

C:\Users\mko>

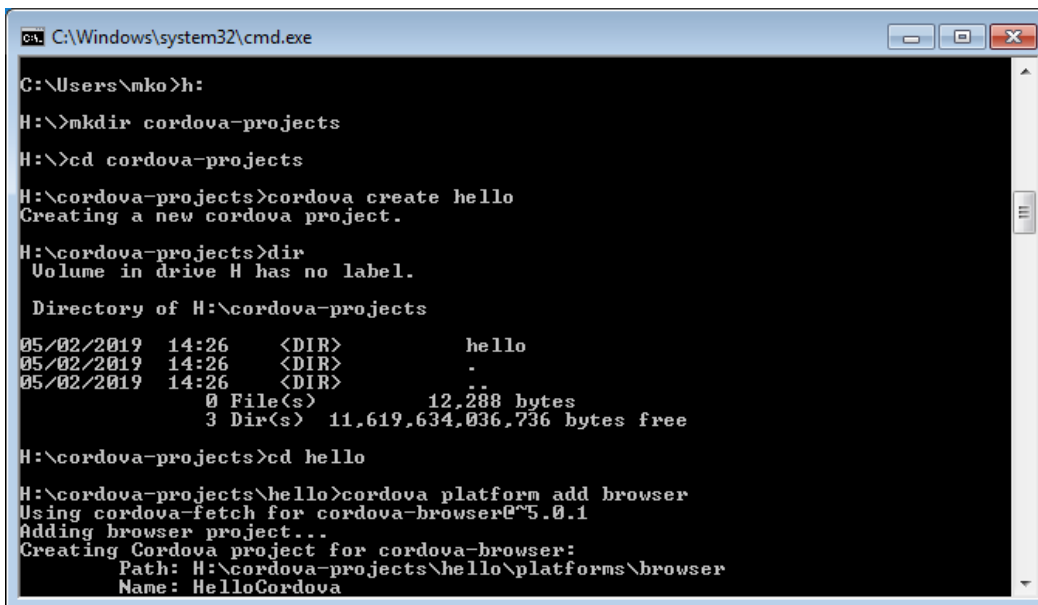
```

Success! You have successfully installed Cordova! Next let's try it out.

Let's create a new cordova project. You want to do this in a directory you have write access too and which is not local to the machine you are using. Your user drive would be a good location. Change to this directory in the command window. (H: changes the drive to the h: drive. `cd` changes the directory.)

You may want to create a folder for all your cordova projects (such as cordova-projects). Then type `cordova create hello` into the command prompt to create your new project called hello. Check that a new directory called hello has been created, then change into it. You should find the directory structure in the folder as discussed in the lectures. The www folder should have an `index.html` file and directories for Javascript and CSS (amongst others).

As Cordova is a cross-platform development framework, we will need to tell it what platforms we want supported for a project. Let's run the project in a browser first (this is the easiest and fastest way to execute a Cordova app) We can add the platform by typing the command `cordova platform add browser`.



```
C:\Windows\system32\cmd.exe

C:\Users\mko>h:
H:\>mkdir cordova-projects
H:\>cd cordova-projects
H:\cordova-projects>cordova create hello
Creating a new cordova project.
H:\cordova-projects>dir
Volume in drive H has no label.

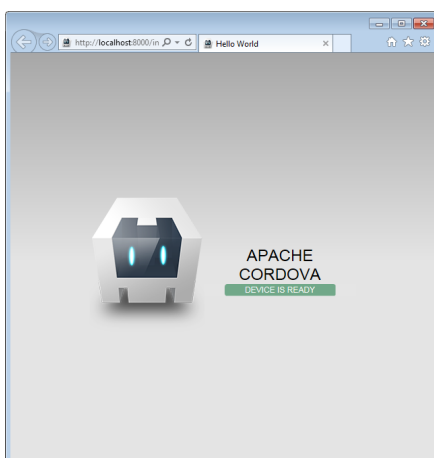
Directory of H:\cordova-projects

05/02/2019  14:26    <DIR>        hello
05/02/2019  14:26    <DIR>        .
05/02/2019  14:26    <DIR>        ..
               0 File(s)      12,288 bytes
               3 Dir(s)  11,619,634,036 bytes free

H:\cordova-projects>cd hello
H:\cordova-projects\hello>cordova platform add browser
Using cordova-fetch for cordova-browser@5.0.1
Adding browser project...
Creating Cordova project for cordova-browser:
  Path: H:\cordova-projects\hello\platforms\browser
  Name: HelloCordova
```

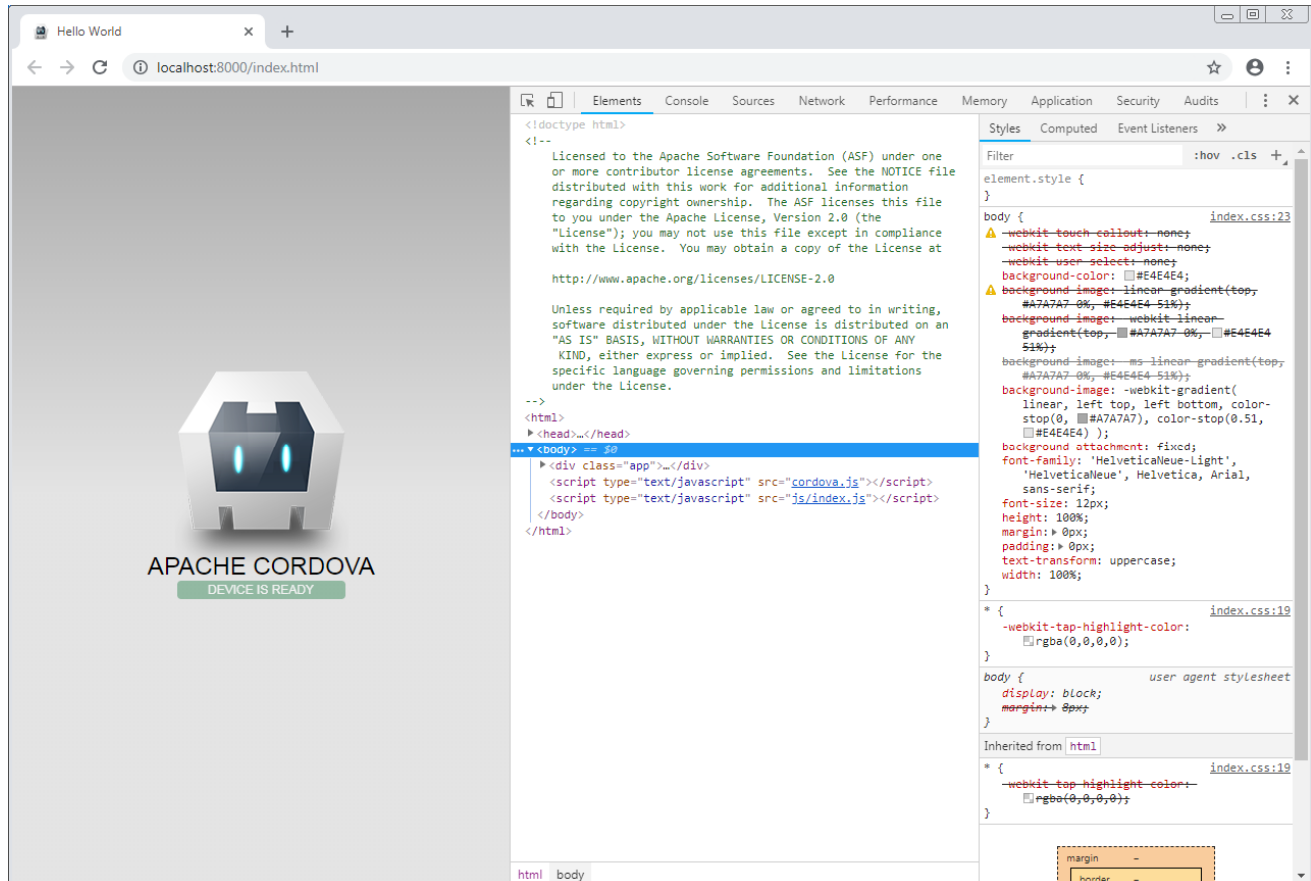
Executing this will take a couple of minutes. See below screenshot for what to expect.

Afterwards you should be able to execute the app in a browser by executing the command `cordova run browser`.



This will execute the app in the default browser. That might still be Internet Explorer for most of you. However, Chrome enjoys better support and also offers some interesting debugging facilities. So I'd recommend you use Chrome for these labs.

Once you have the app running in Chrome, try hitting Ctrl-Shift-I. This should open up the debug interface which allows you to spot errors in your HTML/Javascript (of which you will produce quite a few as you start developing your app).



Finally let's setup the Android environment too. This involves a much bigger tool apparatus. You will need an Android development environment (Android Studio) and SDK (software development kit) together with an Android emulator (or even a real phone). We will use the Android emulator. Some of this apparatus is already installed on the lab machines (Android Studio, Java 1.8) and some is provided on the memory sticks (Android SDK).

First put the memory stick into the USB port. Make sure that the pen is coming up as drive E:
The execute cordova platform add android.
Again, this will take a few minutes to complete.

```
C:\Windows\system32\cmd.exe

H:\cordova-projects\hello>cordova platform add android
Using cordova-fetch for cordova-android@7.1.1
Adding android project...
Creating Cordova project for the Android platform:
  Path: platforms\android
  Package: io.cordova.hellocordova
  Name: HelloCordova
  Activity: MainActivity
  Android target: android-27
Android project created with cordova-android@7.1.4
Android Studio project detected
Android Studio project detected
Installing "cordova-plugin-whitelist" for android

  This plugin is only applicable for versions of cordova-android gr
eater than 4.0. If you have a previous platform version, you do *not* need this
plugin since the whitelist will be built in.

--save flag or autosave detected
Saving android@7.1.4 into config.xml file ...

H:\cordova-projects\hello>
```

You can check afterwards what platforms are installed for your project:

```
cordova platform list
```

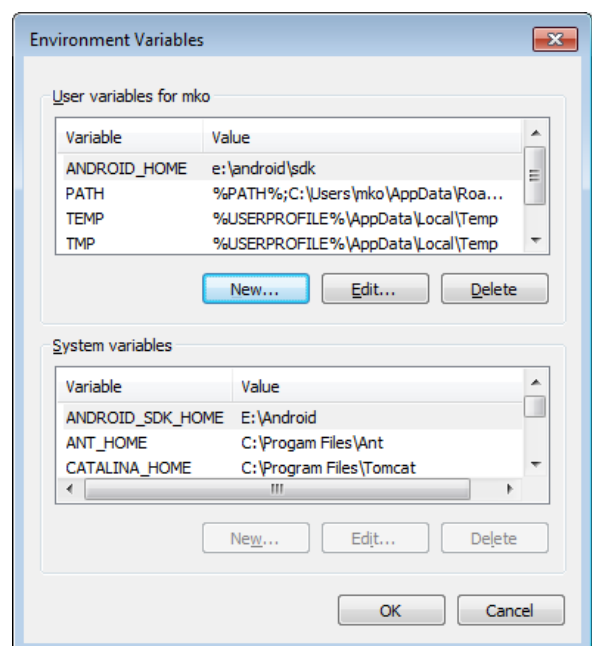
We will also need to set a new environmental variable called `ANDROID_HOME` to point to `e:\android\sdk`. Set this in the same way as you did before. This variable will tell Android Studio where the SDK and Android emulators are located (on the USB stick). Without this variable, Android Studio will insist to on downloading gigabytes worth of libraries and tools.

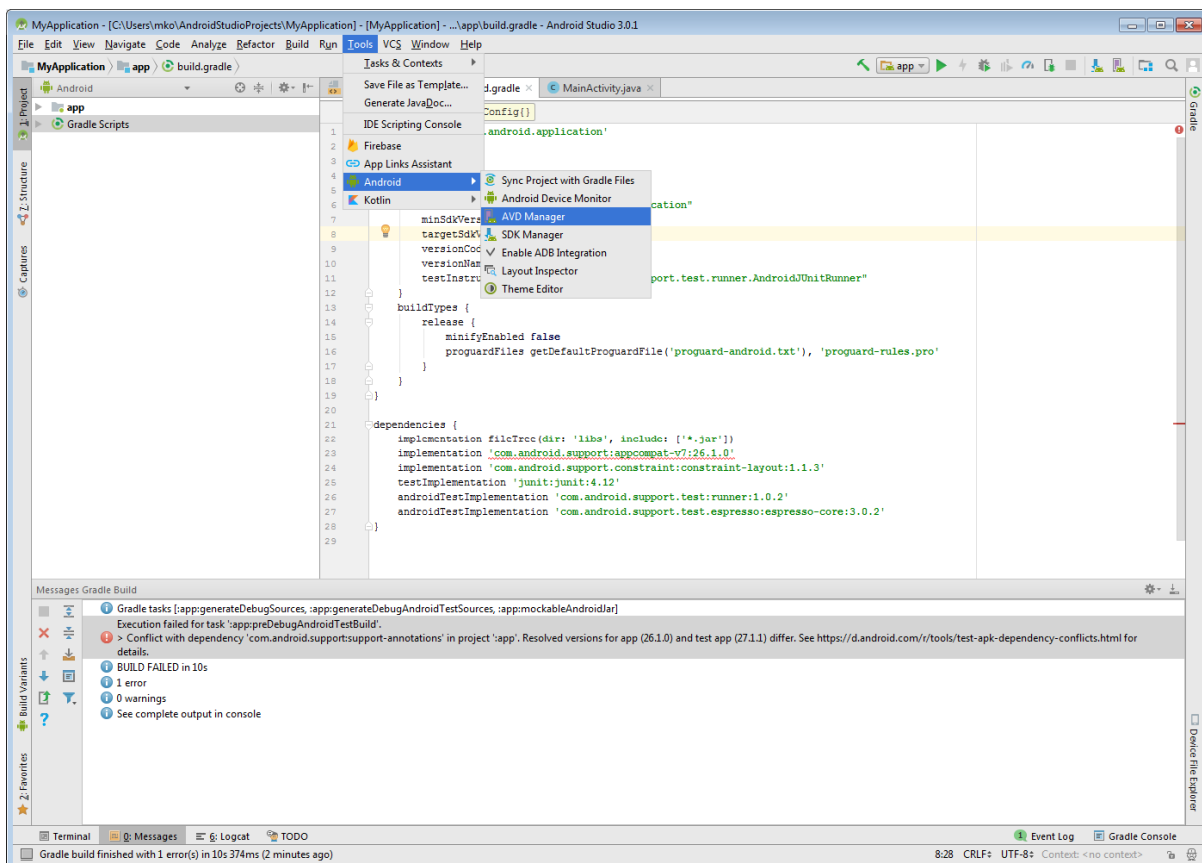
Then you can build the Android project by executing `cordova build android`

This will take some time to complete (about 5min). It will also download and install some files.

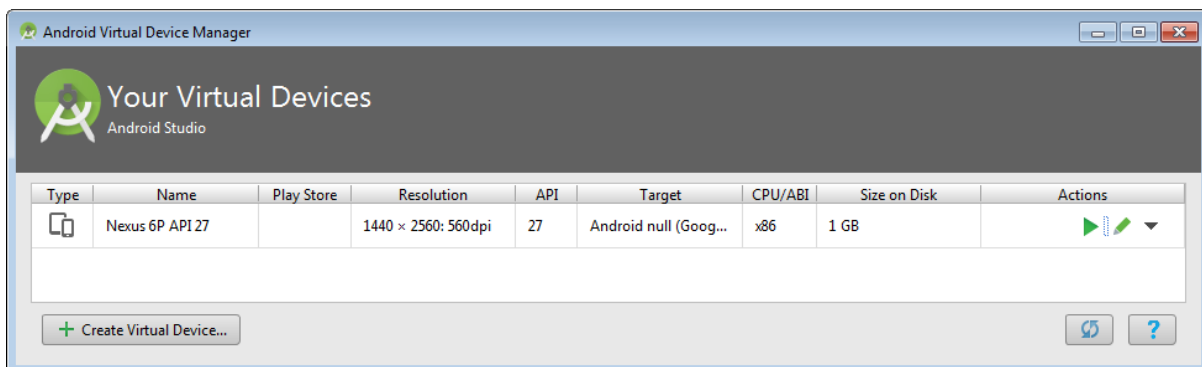
You are now ready to execute the Android app - well almost. We need an Android emulator (an application which pretends to be an Android phone).

Unfortunately, the software for the emulator has changed recently in a way that means that we cannot simply run the emulator. We will need to start the emulator from within Android Studio, the main Android development environment. But don't worry we will not be using this tool for anything else! Start up Android Studio from the Start Menu. Create a new project with an empty Activity. Once this is created and compiled, Android Studio will offer the AVD Manager (Android Virtual Device Manager).





Start this. You should find one virtual device (Nexus 6P) existing. Start this by clicking the green arrow under Actions.



After a short while you should see the Android phone started up. Experiment with the emulator a bit looking for the apps installed.



Finally, you can launch your app on the emulator by executing:
`cordova emulate android`

Alternatively, you can also use
`cordova run android`

With the second option Cordova is looking for a real phone to execute the app on first. As it does not find any, it reverts to the emulator. So both command will have the same effect in the lab.

Please also note that the `build` command we used above is automatically executed if you made changes and then execute `run` or `emulate`.

Checkpoint

In the next lab we will edit the app and add jQuery to the project. You may want to edit the `index.html` file and re-execute your app. You should be able to verify your changes.