**Contents**:

* [What you should already KNOW](#alreadyknow)
* [What you will LEARN](#tolearn)
* [What you will DO](#youwilldo)
* [App overview](#appintro)
* [Task 1. Install Android Studio](#task1intro)
* [Task 2: Create “Hello World” app](#task2intro)
* [Task 3: Explore the project structure and layout](#task3intro)
* [Task 4: Create a virtual device](#task4intro)
* [Task 5: Run your app on an emulator](#task5intro)
* [Task 6. Add log statements to your app](#task6intro)
* [Task 7: Explore the AndroidManifest.xml file](#task7intro)
* [Task 8. Explore the build.gradle file](#task8intro)
* [Task 9. Run your app on a device](#task9intro)
* [Coding challenge](#codingchallenge)
* [Conclusion](#conclusion)
* [Resources](#resources)

<a name="chapterstart" />

# Create and run your first Android app

Welcome to the practical exercises. In this series of exercises you will:

* Set up your Android Studio development environment.
* Create and explore the Hello World app.
* Add logging to your app for debugging.

<a name="alreadyknow" />

## What you should already KNOW

For this practical you should be familiar with:

* The software development process for Object Oriented applications using an IDE, ideally with background in Java language syntax. (These practicals will not explain OO or the Java language.)

<a name="tolearn" />

## What you will LEARN

* How to set up a computer with the Android Studio integrated development environment (IDE).
* The development process for building Android apps.
* How to create a project from an app template.

<a name="youwilldo" />

## What you will DO

* Install the Android Studio development environment.
* Create a virtual device to run your app on your computer.
* Create and run the Hello World app on the virtual and physical devices.
* Explore the project layout.
* Generate and view log statements from your app.
* Explore the AndroidManifest.xml file.

<a name="appintro">

## App Overview

After installing Android Studio, you will use it to create a project for the Hello World app from a template. This minimalist app consists of one activity that displays the greeting “Hello World”.

Here's what the finished app will look like:



IMAGEINFO: pv\_helloworld.png, Hello World App

<a name="task1intro" />

## Task 1. Install Android Studio

Android Studio is Google’s development environment for Android apps. Android Studio gives you an intelligent code editor and app templates, as well as tools for development, debugging, testing, and performance that make it faster and easier to develop apps. You can test your apps with a large range of preconfigured emulators or on your own mobile device, and build production APKs for publication.

**Why:** Android Studio is Google's IDE for Android development and used for this course.

<div class="note">

<strong>Note: </strong> Android Studio is continually being improved. For the latest information on system requirements and installation instructions, refer to the documentation at [developer.android.com](http://developer.android.com/sdk/index.html).

</div>

To get up and running with Android Studio:

* You may need to install the Java Development Kit
* Install Android Studio

Android Studio is available for Windows, Mac, and Linux computers. The installation is mostly the same for all platforms; differences will be called out.

<a name="task1steps1" />

### 1.1. Installing the Java Development Kit

1. On your computer, open a terminal window.
2. Type java -version

The output includes a line:

Java(™) SE Runtime Environment (build1.**X**.0\_05-b13)

**X** is the version number to look at.

* If this is 7 or greater, you can move on to installing Android Studio.
* Otherwise, you need to install the latest version of the Java SE development kit.

To download the Java SE Development Kit:

1. Go to the [Oracle Java SE downloads page](http://www.oracle.com/technetwork/java/javase/downloads/index.html).
2. Click the Java SE Downloads icon to open the [Java SE Development Kit 8 Downloads page](http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html).
3. In the box for the latest Java SE Development kit, accept the License Agreement and then download the version appropriate for your computer. Do not go to the demos and samples (the menus look very similar, so make sure to read the heading at the top).
4. Install the development kit.
5. Type Type java -version again to verify that installation has been successful.
6. Set the JAVA\_HOME environment variable to the installation directory of the JDK.

<div class="note">

<strong>Important: </strong> Do not move on with Android Studio install until after you have installed the JDK. Without a working copy of Java, the rest of the process will not work. If you can't get the download to work, look for error messages, and search online to find a solution.

</div>

<a name="task1steps2" />

### 1.2. Installing Android Studio

1. Navigate to the [Android developers site](https://developer.android.com/sdk/index.html) and follow the instructions to download and install Android Studio.
   * Accept the default configurations for all steps.
   * Make sure that all components are selected for installation.
2. After finishing the install, the Setup Wizard will download and install some additional components. Be patient, this might take some time depending on your internet speed.
3. When you are finished, Android Studio will start, and you are ready to create your first project.

<div class="note">

<strong>Troubleshooting: </strong> If you run into problems with your installation, check the latest documentation, programming forums, or get help from you instructors. </div>

<a name="task2intro" />

## Task 2: Create “Hello World” app

In this practical, you will create and configure a project for the “Hello World” app.

**Why:** This makes sure your installation is correct, and familiarizes you with the Android Studio Workflow.

<a name="task2steps1" />

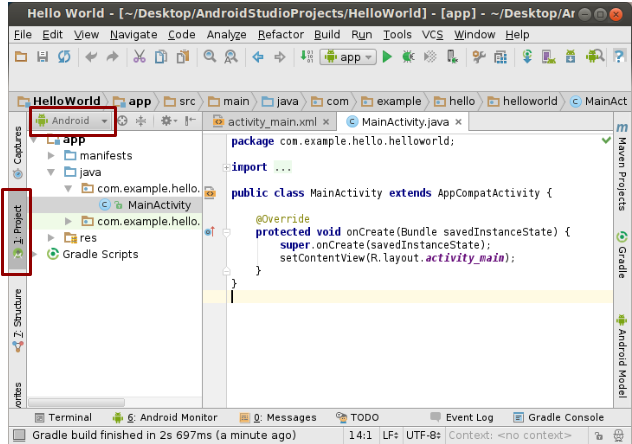
### 2.1 Create the “Hello World” app

1. Start Android Studio if it is not already running.
2. In the **Welcome to Android Studio** window, click “Start a new Android Studio project”.
3. In the **New Project** window, give your application an **Application Name**, such as Hello World.
4. Choose a unique **Company Domain**.
   * Apps published to the Google Play Store must have a unique package name. Since domains are unique, prepending your app's name with your or your company's domain name is going to result in a unique package name.
   * If you are not planning to publish your app, you can accept the default example domain. Be aware that changing the package name of your app later is extra work.
5. Verify that the default **Project location** is where you want to store your Android Studio projects, or change it to your preferred directory.
6. On the **Target Android Devices** screen, verify that API15: Android 4.0.3 Ice CreamSandwich is set as the **Minimum SDK**. (Fix this if necessary.)
   * At the writing of this book, choosing this API level makes the app compatible with 97% of devices active on the Google Play Store.
   * These are the settings used by the examples in this book.
7. Click **Next**.
8. If your installation requires additional components for your chosen target SDK, Android Studio will install them automatically. Click **Next**.
9. Every app needs at least one activity, and Android Studio provides templates to help you get started. For the Hello World project, choose the simplest template (as of this writing, the "Empty Activity" project template) available and click **Next**.
10. It is customary, but not required, to call your main activity MainActivity. Make sure the **Generate Layout file** box is checked. It is also customary to name layouts after the activity they belong to. Accept the defaults and click **Finish**.

After these steps, Android Studio::

* Creates a folder for your Android Studio Projects.
* Builds your project with gradle (this may take a few moments).
* Opens the code editor with your project.
* Displays a tip of the day. Android Studio offers many keyboard shortcuts, and reading the tips is a great way to learn them over time.

Your Android Studio window should look similar to this:

IMAGEINFO: as\_projectview.png, Android Studio Project View

In Project view, you can view your app and file hierarchy in multiple ways.

1. Click on the **Android** menu.
2. Explore the different view options for your project.

<div class="note">

<strong>Note: </strong> This book uses the <b>Android</b> view of the project files, unless specified otherwise.

</div>

<a name="task3intro" />

## Task 3: Explore the project structure and layout

In this practical, you will explore how the project and project files are structured and organized in Android Studio.

**Why:** Being familiar with the project structure makes it easier to find and change files for later exercises.

These steps assume that your project layout starts out as shown above in the Project view, after you created your new Hello World project. Take a look at the files.

<a name="task3steps1" />

### 3.1 Explore the project structure and layout

In the Project/Android view (see previous task), there are three top-level folders below your **app** folder: **manifests**, **java**, and **res**.

1. Expand the **manifests** folder.

This folder contains your **AndroidManifest.xml** file, which connects all the different parts of your app into one application.

1. Expand the **java** folder. All your Java language files are organized in this folder. The **java** folder contains two subfolders:
   * **com.example.hello.helloworld:** All the files for your packages are in folders named after your packages. Currently, there is one package for Hello World and it only contains MainActivity.java (the file extension may be omitted in the Project view).
   * **com.example.hello.helloworld(AndroidTest):** This second folder is for your tests and starts out with a skeleton ApplicationTest.java file.
   * **com.example.hello.helloworld(test):** This folder is for your unit tests and starts out with a skeleton unit test file.
2. Expand the **res** folder. This folder contains all the resources for your app, including images, layout files, strings, icons, and styling. It includes these subfolders:
   * **drawable**. Store all your app’s images in this folder.
   * **layout**. Every activity has at least one layout file. For Hello World, this folder contains activity\_main.xml
   * **mipmap**. Store your icons in this folder. The sub-folders accommodate different resolutions. The ic\_launcher.png folder contains the default launcher icons.
   * **values**. Instead of hardcoding values into your XML files, it is best practice to define them in their respective values file. This makes it easier to change and be consistent across your app.
3. Expand the **values** folder. It includes these subfolders:
   * **colors.xml**. Shows the default colors for your theme, and you can add your own colors or change them.
   * **dimens.xml**. Store sizes of views and objects for different resolutions.
   * **strings.xml**. Create resources for all your strings. This makes it easy to translate them to other languages.
   * **styles.xml**. All the CSS for your app goes here. Styles help you have a consistent look for all UI elements in your app.
4. Expand the **Gradle Scripts** folder.

This folder contains all the files needed by the build system. Look for the **build.gradle(Module:app)** file. When you are adding app-specific dependencies, such as using additional libraries, they go into this file.

<a name="task4intro" />

## Task 4: Create a virtual device

In this practical, you will use the [Android Virtual Device (AVD) manager](http://developer.android.com/tools/devices/managing-avds.html) to create a virtual device. The virtual device is used by the emulator to simulate a particular device on which you can run your app.

**Why:** With virtual devices, you can test your app on different devices (tablets, phones) with different API levels to make sure it looks good and works for the most users. You do not depend on having a physical device available for app development.

<a name="task4steps1" />

### 4.1 Create a virtual device

1. In Android Studio, select **Tools > Android > AVD Manager**,   
   or click the AVD Manager icon  in the toolbar.
2. Click **Create Virtual Device…**

The **Select Hardware** screen appears with all the available preconfigured hardware devices.

1. Choose a hardware device, for example, Nexus 5. Click **Next**.

Note that the **Density** for this device is xxhdpi, which means your app uses the icons in the xxhdpi folder of the mipmap folder. Likewise, you app will use layouts and drawables from folders defined for that density as well.

1. On the **System Image** screen, from the **Recommended** tab, choose one of the versions of the Android system to run on the virtual device. Choosing your target SDK is a good place to start, which is **Marshmallow 23 X86\_64** at the time of writing this tutorial.
2. If the **Download** link is visible, you need to download the system image. Click the link to start the download.
3. Click **Finish**.
4. Return to the **System Image** screen, choose the system image and click **Next**.
5. Verify your configuration, and click **Finish**. (You can close the AVD Manager if you wish.)

<a name="task5intro" />

## Task 5. Run your app on an emulator

In this practical, you will finally run your Hello World app.

**Why:** Goal achieved!

<a name="task5steps1" />

### 5.1 Run your app on an emulator

1. In Android Studio, select **Run > Run app** or click the **Run icon**  in the toolbar.
2. In the Device Chooser, select **Launch emulator** and select **Nexus 5 API 23** as the Android virtual device.
3. Click **OK**.

The emulator starts. Your app builds. Wait for the emulator to start and show your app.

IMAGEINFO: pv\_helloworld.png, Hello World App

<div class="note">

<strong>Note: </strong> If you are testing on an emulator, it is good practice to start it up once at the very beginning of your session, and not to close it until you are done so it doesn't have to go through the boot process again.

</div>

**Challenge:** You can fully customize your virtual devices. Study the [AVD Manager documentation](http://developer.android.com/tools/devices/managing-avds.html). Create one or several custom virtual devices. You may notice that not all combinations of devices and system versions work when you run your app. This is because not all system images can run on all hardware devices.

<a name="task6intro" />

## Task 6. Add log statements to your app

In this practical, you will add log statements to your app, which are displayed in the logging window of the Android Monitor.

**Why:** Log messages are a powerful debugging tool that you can use to check on values, execution paths, and report exceptions.

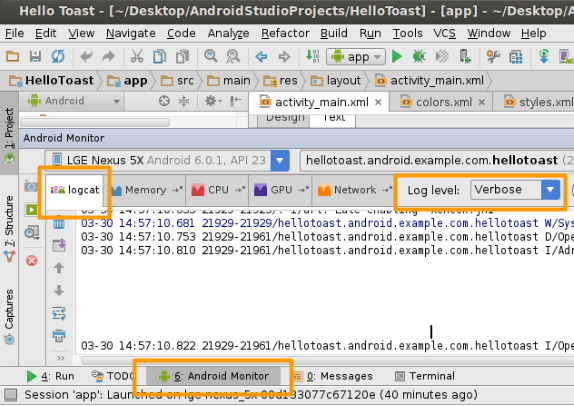
The **Android Monitor** displays information about your app.

1. Click the **Android Monito**r button at the bottom of Android Studio to open the Android Monitor.

By default, this opens to the **logcat** tab, which displays information about your app as it is running. If you add log statements to your app, they are printed here as well.

You can also monitor the Memory, CPU, GPU, and Network performance of your app from the other tabs of the Android Monitor. This can be helpful for debugging and performance tuning your code.

1. The default log level is **Verbose**. In the drop-down menu, change the log level to **Debug** for this practical.

IMAGEINFO: as\_logcat.png, Android Studio Logcat

Log statements that you add to your app code print a message specified by you in the logcat tab of the Android Monitor. For example:

Log.d("MainActivity", "Hello World");

The parts of the message are:

* Log – The [Log class](http://developer.android.com/reference/android/util/Log.html). API for sending log messages.
* d – The Log level. Used to filter log message display in logcat. “d” is for debug. Other log levels are “e” for error, “w” for warning, and “i” for info.
* “MainActivity” – The first argument is a tag which can be used to filter messages in logcat. This is commonly the name of the activity from which the message originates. However, you can make this anything that is useful to you for debugging.

By convention, log tags are defined as constants:

private static final String LOG\_TAG = MainActivity.class.getSimpleName();

* “Hello world” – The second argument is the actual message.

<a name="task6steps1" />

### 6.1 Add log statements to your app

1. Open your Hello World app in Android studio, and open MainActivity file.
2. **File > Settings > Editor > General >Auto Import (**Mac: **Android Studio > Preferences > Editor > General >Auto Import)**. Check all boxes and set **Insert imports** on paste to **All**. Unambiguous imports are now added automatically to your files. (Note that The "add unambiguous imports on the fly" option is important for some Android features such as NumberFormat. If not checked, NumberFormat shows an error.)
3. In the onCreate method, add the following log statement:

Log.d("MainActivity", "Hello World");

1. If the Android Monitor is not already open, click the Android Monitor tab at the bottom of Android Studio to open it. (See screenshot.)
2. Make sure that the Log level in the Android Monitor logcat is set to Debug or Verbose (default).
3. Run your app.

**Solution Code:**

package com.example.hello.helloworld;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

import android.util.Log;

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

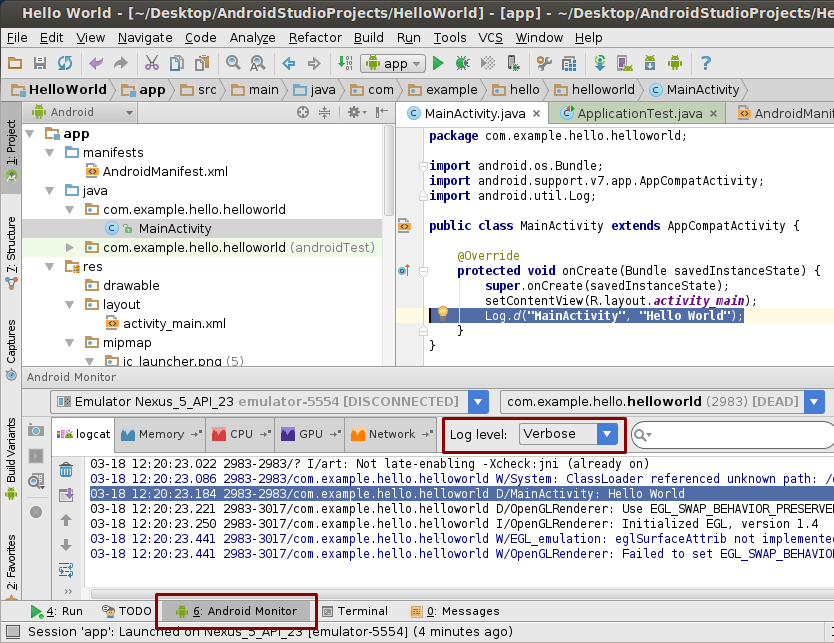
Log.d("MainActivity", "Hello World");

}

}

**Solution Log Message:**

03-18 12:20:23.184 2983-2983/com.example.hello.helloworld D/MainActivity: Hello World

IMAGEINFO: as\_hellolog.png, HelloWorld Log Result

**Resources:** See [Reading and Writing Logs](http://developer.android.com/tools/debugging/debugging-log.html) for additional information.

**Challenge:** Use the [Reading and Writing Logs](http://developer.android.com/tools/debugging/debugging-log.html) and [Log class documentation](http://developer.android.com/reference/android/util/Log.html) to find out how to include an exception with a log message. Then, write code in the main activity to trigger and log an exception.

<a name="task7intro" />

## Task 7: Explore the AndroidManifest.xml file

Every app includes an Android Manifest file (AndroidManifest.xml).The manifest file presents essential information about your app to the Android system. Android must have this information before it can run any of your app's code.

In this practical you will find and read the AndroidManifest.xml file for the Hello World app.

**Why:** As your apps become richer, you add information to the Android manifest.In later lessons, you will change this file to add additional features and permissions.

<a name="task7steps1" />

### 7.1 Explore the AndroidManifest.xml file

1. Open your Hello World app in Android studio, and in the **manifests** folder, open **AndroidManifest.xml**.
2. Read the file and consider what each line of code indicates.

**Solution:**

<!-- XML version and character encoding -->

<?xml version="1.0" encoding="utf-8"?>

<!-- Required starting tag for the manifest -->

<manifest

<!-- Defines the android namespace. Do not change. -->

xmlns:android="<http://schemas.android.com/apk/res/android>"

<!-- Unique package name of your app. Do not change once app is

published. -->

package="com.example.hello.helloworld">

<!-- Required application tag -->

<application

<!-- Allow the application to be backed up and restored. –>

android:allowBackup="true"

<!-- Icon for the application as a whole,

and default icon for application components. –>

android:icon="@mipmap/ic\_launcher"

<!-- User-readable for the application as a whole,

and default icon for application components. –>

android:label="@string/app\_name"

<!-- Whether the app is willing to support right-to-left layouts.–>

android:supportsRtl="true"

<!-- Default theme for styling all activities. –>

android:theme="@style/AppTheme">

<!-- Declares an activity. One is required.

All activities must be declared,

otherwise the system cannot see and run them. –>

<activity

<!-- Name of the class that implements the activity;

subclass of Activity. –>

android:name=".MainActivity">

<!-- Specifies the intents that this activity can respond to.–>

<intent-filter>

<!-- The action and category together determine what

happens when the activity is launched. –>

<!-- Start activity as the main entry point.

Does not receive data. –>

<action android:name="android.intent.action.MAIN" />

<!-- Start this activity as a top-level activity in

the launcher . –>

<category android:name="android.intent.category.LAUNCHER" />

<!-- Closing tags –>

</intent-filter>

</activity>

</application>

</manifest>

**Challenge:**

There are many other elements that can be set in the Android Manifest. Explore the [Android Manifest documentation](http://developer.android.com/guide/topics/manifest/manifest-intro.html) and learn about additional elements in the Android Manifest.

<a name="task8intro" />

## Task 8. Explore the build.gradle file

Android Studio uses a build system called Gradle. Gradle does incremental builds, which allows for shorter edit-test cycles.

In this practical, you will explore the build.gradle file.

**Why:** When you add new libraries to your Android project, you may also have to update your **build.gradle file**. It's useful to know where it is and its basic structure.

<a name="task8steps1" />

### 8.1 Explore the build.gradle(Module app) file

1. In your project hierarchy, find **Gradle Scripts** and expand it.
2. Open **build.gradle (Module.app).**
3. Read the file and consider what each line of code indicates.

**Solution:**

// Add Android-specific build tasks

apply plugin: 'com.android.application'

// Configure Android specific build options.

android {

// Specify the target SDK version for the build.

compileSdkVersion 23

// The version of the build tools to use.

buildToolsVersion "23.0.2"

// Core settings and entries. Overrides manifest settings!

defaultConfig {

applicationId "com.example.hello.helloworld"

minSdkVersion 15

targetSdkVersion 23

versionCode 1

versionName "1.0"

}

// Controls how app is built and packaged.

buildTypes {

// Another common option is debug, which is not signed by default.

release {

// Code shrinker. Turn this on for production along with

// shrinkResources.

minifyEnabled false

// Use ProGuard, a Java optimizer.

proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'

}

}

}

// This is the part you are most likely to change as you start using

// other libraries.

dependencies {

// Local binary dependency. Include any JAR file inside app/libs.

compile fileTree(dir: 'libs', include: ['\*.jar'])

// Configuration for unit tests.

testCompile 'junit:junit:4.12'

// Remote binary dependency. Specify Maven coordinates of the Support

// Library needed. Use the SDK Manager to download and install such

// packages.

compile 'com.android.support:appcompat-v7:23.2.1'

}

**Resources:** To learn more about Gradle, start with the [Gradle Wikipedia page](https://en.wikipedia.org/wiki/Gradle).

**Challenge:** For a deeper look into Gradle check out the [Build System Overview](http://developer.android.com/sdk/installing/studio-build.html) and [Configuring Gradle Builds](http://developer.android.com/tools/building/configuring-gradle.html) documentation. Learn more about [ProGuard](http://developer.android.com/tools/help/proguard.html). Learn more about [shrinking resources](http://tools.android.com/tech-docs/new-build-system/resource-shrinking).

<a name="task9intro" />

## Task 9. [Optional] Run your app on a device

In this final task, you will run your app on a physical mobile device such as a phone or tablet.

**Why:** Your users will run your app on physical devices. You should always test your apps on both virtual and physical devices.

What you need:

* An Android device such as a phone or tablet.
* A data cable to connect your Android device to your computer via USB port.
* If you are using a Linux or Windows OS, you may need to perform additional steps to run on a hardware device. Check the [Using Hardware Devices](http://developer.android.com/tools/device.html) documentation. On Windows, you may need to install the appropriate USB driver for your device. See [OEM USB Drivers](http://developer.android.com/tools/extras/oem-usb.html).

<a name="task9steps1" />

### 9.1 [Optional] Run your app on a device

To let Android Studio communicate with your device, you must turn on USB Debugging in the Developer options settings. . (Note: This is not the same as rooting your device.)

On Android 4.2 and higher, the Developer options screen is hidden by default. To show Developer options and enable USB Debugging:

1. Open **Settings > About** phone and tap **Build number** seven times.
2. Return to the previous screen (**Settings**). **Developer options** appears at the bottom of the list. Click **Developer options**.
3. Choose **USB Debugging**.

Now you can connect your device and run the app from Android Studio.

1. Connect your device to your development machine with a USB cable.
2. Check in the Android Monitor that Android Studio recognizes your device.
3. Click the Run button in the toolbar. The **Choose Device** window appears.
4. Select the **Choose a running device** radio button, select your device, and click **OK**.
5. Android Studio installs the app on your connected device and starts it.

### Troubleshooting

If you Android Studio does not recognize your device, try the following:

* Unplug and replug your device.
* Restart Android Studio.
* If your computer still does not find the device or declares it "unauthorized":
  1. Unplug the device.
  2. Open **Developer Options**.
  3. Tap **Revoke USB Debugging authorizations**.
  4. Plug it in the device.
  5. When prompted, grant authorizations.
* You may need to install the appropriate USB driver for your device. See the [Using Hardware Devices documentation](http://developer.android.com/tools/device.html) document.
* Check the latest documentation, programming forums, or get help from you instructors.

<a name="codingchallenge" />

## Coding challenge

<div class="note">

<strong>Note: </strong> All coding challenges are optional and not a prerequisite for the material in the next chapter.

</div>

Now that you are set up and familiar with the basic development workflow, do the following:

1. Create a new project in Android Studio.
2. Change the greeting to "Happy Birthday to " and someone with a recent birthday.
3. Change the background of the app using a birthday-themed image.
4. Take a screenshot of your finished app and email it to someone whose birthday you forgot.

Explore Android Studio using the documentation at [developers.android.com](http://developer.android.com/tools/studio/index.html).

<a name="conclusion" />

## Conclusion

In this chapter, you

* installed Android Studio and deployed the Hello World app in the emulator and [optionally] on a device.
* acquired a basic understanding of the structure of an Android app.
* added log statements that give you a basic tool for debugging.
* obtained a basic understanding of the development workflow in Android Studio.

<a name="resources" />

## Resources

**Developer Documentation:**

* [Android Studio download page](http://developer.android.com/sdk/index.html)
* [Android Studio documentation](http://developer.android.com/tools/studio/index.html)