

A decorative graphic on the left side of the slide, consisting of white lines and circles on a blue gradient background, resembling a circuit board or a stylized tree structure.

INSIDE : ANDROID STUDIO

ROMI FADILLAH RAHMAT

DEVELOPMENT ENVIRONMENT

- The Android SDK – Contains the libraries and tools to develop Android Apps :
 - SDK Platform
 - SDK Tools -> tools to do debug and testing
 - Documentation -> we can see the latest API documentation offline
 - Android Support -> Extra API
 - Sample Apps -> Sample for practice to understand how to use some of API's
 - Google Play Billing -> Allow to integrate billing service in the APP

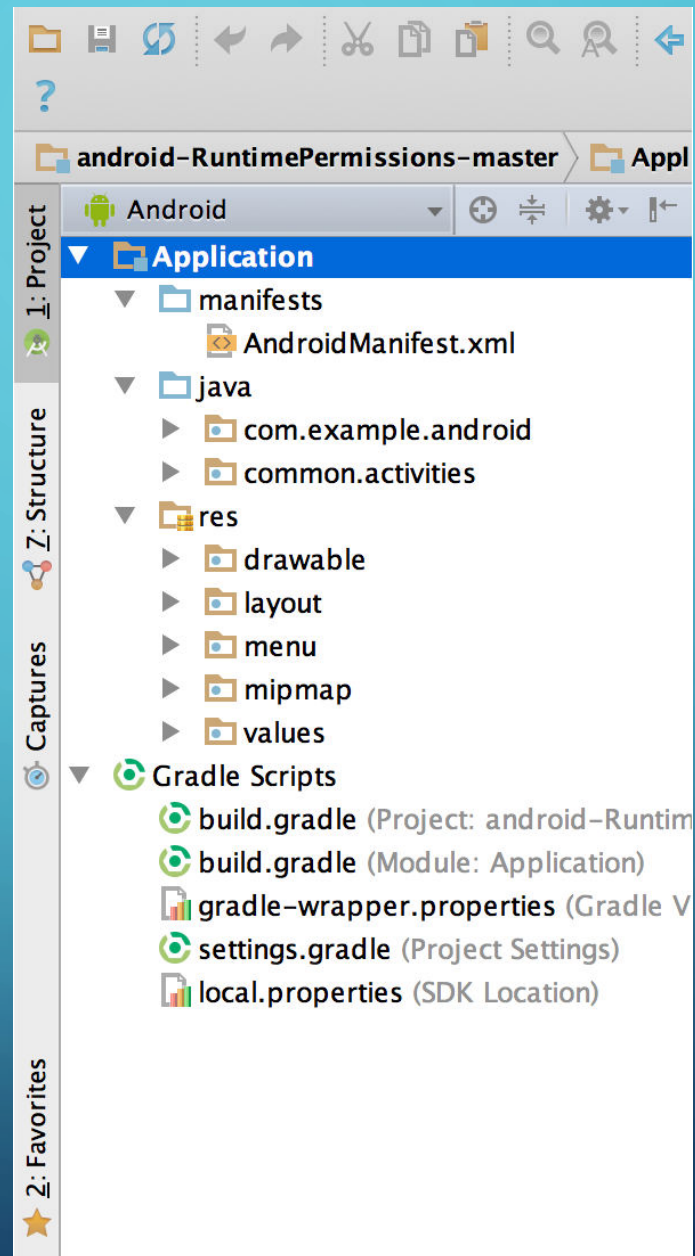
INSTALL ANDROID STUDIO

- The easiest way you can download from here :
<https://developer.android.com/studio/index.html>

PROJECT STRUCTURE

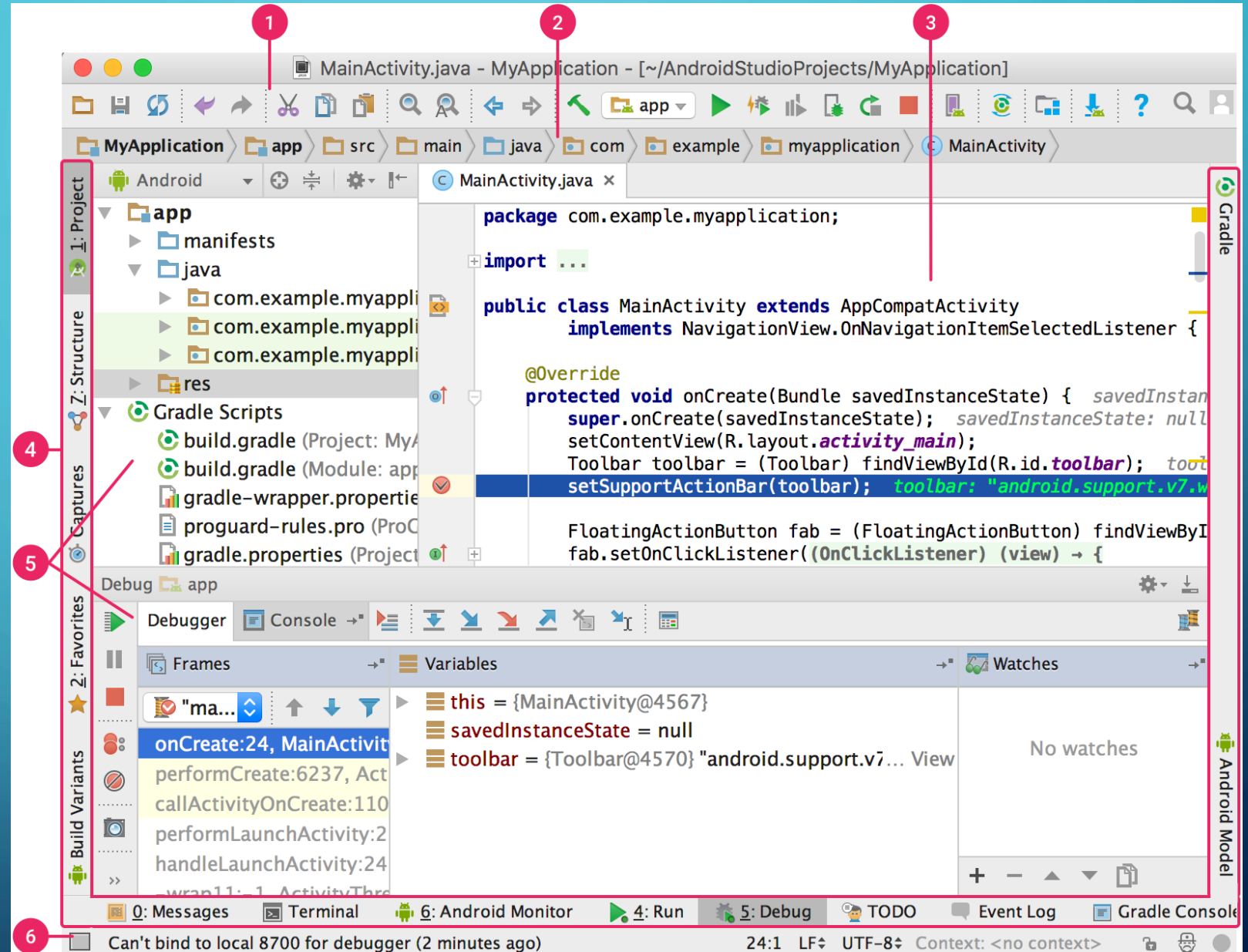
- Each project in Android Studio contains one or more modules with **source code files** and **resource files**. Types of modules include:
 - Android app modules
 - Library modules
 - Google App Engine modules
- By default, Android Studio displays your project files in the Android project view, as shown in figure 1.

FIGURE 1



- All the build files are visible at the top level under Gradle Scripts and each app module contains the following folders:
 - manifests: Contains the AndroidManifest.xml file.
 - java: Contains the Java source code files, including JUnit test code.
 - res: Contains all non-code resources, such as XML layouts, UI strings, and bitmap images.
- The Android project structure on disk differs from this flattened representation. To see the actual file structure of the project, select **Project** from the **Project** dropdown (in figure 1, it's showing as **Android**).

THE USER INTERFACE



- [1] The toolbar lets you carry out a wide range of actions, including running your app and launching Android tools.
- [2] The navigation bar helps you navigate through your project and open files for editing. It provides a more compact view of the structure visible in the Project window.
- [3] The editor window is where you create and modify code. Depending on the current file type, the editor can change. For example, when viewing a layout file, the editor displays the Layout Editor.
- [4] The tool window bar runs around the outside of the IDE window and contains the buttons that allow you to expand or collapse individual tool windows.
- [5] The tool windows give you access to specific tasks like project management, search, version control, and more. You can expand them and collapse them.
- [6] The status bar displays the status of your project and the IDE itself, as well as any warnings or messages.

ANDROID STUDIO EDITORS

- You view and edit files using the Android Studio editors. Double-click on the file you want to work with, and the file contents will appear in the middle of the Android Studio window.
- **The code editor** : Most files get displayed in the code editor. The code editor is just like a text editor, but with extra features such as color coding and code checking.
- The design editor : If you're editing a layout, you have an extra option. Rather than edit the XML, you can use the design editor. The design editor allows you to drag GUI components onto your layout, and arrange them how you want. The code editor and design editor give different views of the same file, so you can switch back and forth between the two.

INSIDE ACTIVITY_MAIN

activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp"
    tools:context=".MainActivity">
```

```
<TextView
    android:text="@string/hello_world"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

```
</RelativeLayout>
```

Add padding to the screen margins.

Include a `TextView` GUI component for displaying text.

Make the text wrap horizontally and vertically.

Display the text value of a string resource called `hello_world`.

Make the layout the same width and height as the screen size on the device.

INSIDE MAINACTIVITY.JAVA

MainActivity.java

```
package com.hfad.myfirstapp;

import android.os.Bundle;
import android.app.Activity;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

}
```

This is the package name.

These are Android classes
used in MainActivity.

Specifies which layout to use.

Implement the `onCreate()`
method from the `Activity`
class. This method is called
when the activity is first
created.

MainActivity extends the
Android class
`android.app.Activity`.

KEYBOARD SHORTCUTS

Tool Window	Windows and Linux	Mac
Project	Alt+1	Command+1
Version Control	Alt+9	Command+9
Run	Shift+F10	Control+R
Debug	Shift+F9	Control+D
Android Monitor	Alt+6	Command+6
Return to Editor	Esc	Esc
Hide All Tool Windows	Control+Shift+F12	Command+Shift+F12

SHORTCUTS

Description	Windows/Linux	Mac
General		
Save all	Control + S	Command + S
Synchronize	Control + Alt + Y	Command + Option + Y
Maximize/minimize editor	Control + Shift + F12	Control + Command + F12
Add to favorites	Alt + Shift + F	Option + Shift + F
Inspect current file with current profile	Alt + Shift + I	Option + Shift + I
Quick switch scheme	Control + ` (backquote)	Control + ` (backquote)
Open settings dialogue	Control + Alt + S	Command + , (comma)
Open project structure dialog	Control + Alt + Shift + S	Command + ; (semicolon)
Switch between tabs and tool window	Control + Tab	Control + Tab

[HTTPS://DEVELOPER.ANDROID.COM/STUDIO/INTRO/KEYBOARD-SHORTCUTS.HTML](https://developer.android.com/studio/intro/keyboard-shortcuts.html)

Navigating and Searching Within Studio		
Search everything (including code and menus)	Press Shift twice	Press Shift twice
Find	Control + F	Command + F
Find next	F3	Command + G
Find previous	Shift + F3	Command + Shift + G
Replace	Control + R	Command + R
Find action	Control + Shift + A	Command + Shift + A
Search by symbol name	Control + Alt + Shift + N	Command + Option + O
Find class	Control + N	Command + O
Find file (instead of class)	Control + Shift + N	Command + Shift + O
Find in path	Control + Shift + F	Command + Shift + F

CODE COMPLETION

- Android Studio has three types of code completion, which you can access using keyboard shortcuts.

Type	Description	Windows and Linux	Mac
Basic Completion	Displays basic suggestions for variables, types, methods, expressions, and so on. If you call basic completion twice in a row, you see more results, including private members and non-imported static members.	Control+Space	Control+Space
Smart Completion	Displays relevant options based on the context. Smart completion is aware of the expected type and data flows. If you call Smart Completion twice in a row, you see more results, including chains.	Control+Shift+Space	Control+Shift+Space
Statement Completion	Completes the current statement for you, adding missing parentheses, brackets, braces, formatting, etc.	Control+Shift+Enter	Shift+Command+Enter

FIND SAMPLE CODE

- The Code Sample Browser in Android Studio helps you find high-quality, Google-provided Android code samples based on the currently highlighted symbol in your project. For more information, see [Find Sample Code](https://developer.android.com/studio/write/sample-code.html).

<https://developer.android.com/studio/write/sample-code.html>

NAVIGATION

- Here are some tips to help you move around Android Studio.
 - Switch between your recently accessed files using the *Recent Files* action. Press **Control+E**(**Command+E** on a Mac) to bring up the Recent Files action. By default, the last accessed file is selected. You can also access any tool window through the left column in this action.
 - View the structure of the current file using the *File Structure* action. Bring up the File Structure action by pressing **Control+F12** (**Command+F12** on a Mac). Using this action, you can quickly navigate to any part of your current file.
 - Search for and navigate to a specific class in your project using the *Navigate to Class* action. Bring up the action by pressing **Control+N** (**Command+O** on a Mac). Navigate to Class supports sophisticated expressions, including camel humps, paths, line navigate to, middle name matching, and many more. If you call it twice in a row, it shows you the results out of the project classes.

NAVIGATION

- Navigate to a file or folder using the *Navigate to File* action. Bring up the *Navigate to File* action by pressing **Control+Shift+N** (**Command+Shift+O** on a Mac). To search for folders rather than files, add a `/` at the end of your expression.
- Navigate to a method or field by name using the *Navigate to Symbol* action. Bring up the *Navigate to Symbol* action by pressing **Control+Shift+Alt+N** (**Command+Shift+Alt+O** on a Mac).
- Find all the pieces of code referencing the class, method, field, parameter, or statement at the current cursor position by pressing **Alt+F7**

STYLE AND FORMATTING

- As you edit, Android Studio automatically applies formatting and styles as specified in your code style settings.
- You can customize the code style settings by programming language, including specifying conventions for tabs and indents, spaces, wrapping and braces, and blank lines.
- To customize your code style settings, click **File > Settings > Editor > Code Style** (**Android Studio > Preferences > Editor > Code Style** on a Mac.)

- Although the IDE automatically applies formatting as you work, you can also explicitly call the *Reformat Code* action by pressing **Control+Alt+L** (**Opt+Command+L** on a Mac), or auto-indent all lines by pressing **Control+Alt+I** (**Alt+Option+I** on a Mac).

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    mActionBar = getSupportActionBar();  
    mActionBar.setDisplayHomeAsUpEnabled(true);  
}
```

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    mActionBar = getSupportActionBar();  
    mActionBar.setDisplayHomeAsUpEnabled(true);  
}
```

Formatted 7 lines

Show reformat dialog: `\u2192\u2191\u2193\u2192L`

```
// Get reference to the drawer layout and set event listener
```

PROJECTS IN ANDROID

- A *project* in Android Studio contains everything that defines your workspace for an app, from source code and assets, to test code and build configurations.
- When you start a new project, Android Studio creates the necessary structure for all your files and makes them visible in the **Project** window on the left side of the IDE (click **View > Tool Windows > Project**).

MODULES

- A *module* is a collection of source files and build settings that allow you to divide your project into discrete units of functionality.
- Your project can have one or many modules and one module may use another module as a dependency. Each module can be independently built, tested, and debugged.
- Additional modules are often useful when creating code libraries within your own project or when you want to create different sets of code and resources for different device types, such as phones and wearables, but keep all the files scoped within the same project and share some code.
- You can add a new module to your project by clicking **File > New > New Module**.

TYPES OF MODULE

- ANDROID APP MODULE
- LIBRARY MODULE
- GOOGLE CLOUD MODULE

ANDROID APP MODULE

- Provides a container for your app's source code, resource files, and app level settings such as the module-level build file and Android Manifest file. When you create a new project, the default module name is "app". In the **Create New Module** window, Android Studio offers the following app modules:
 - Phone & Tablet Module
 - Android Wear Module
 - Android TV Module
 - Glass Module
- They each provide essential files and some code templates that are appropriate for the corresponding app or device type.

LIBRARY MODULE

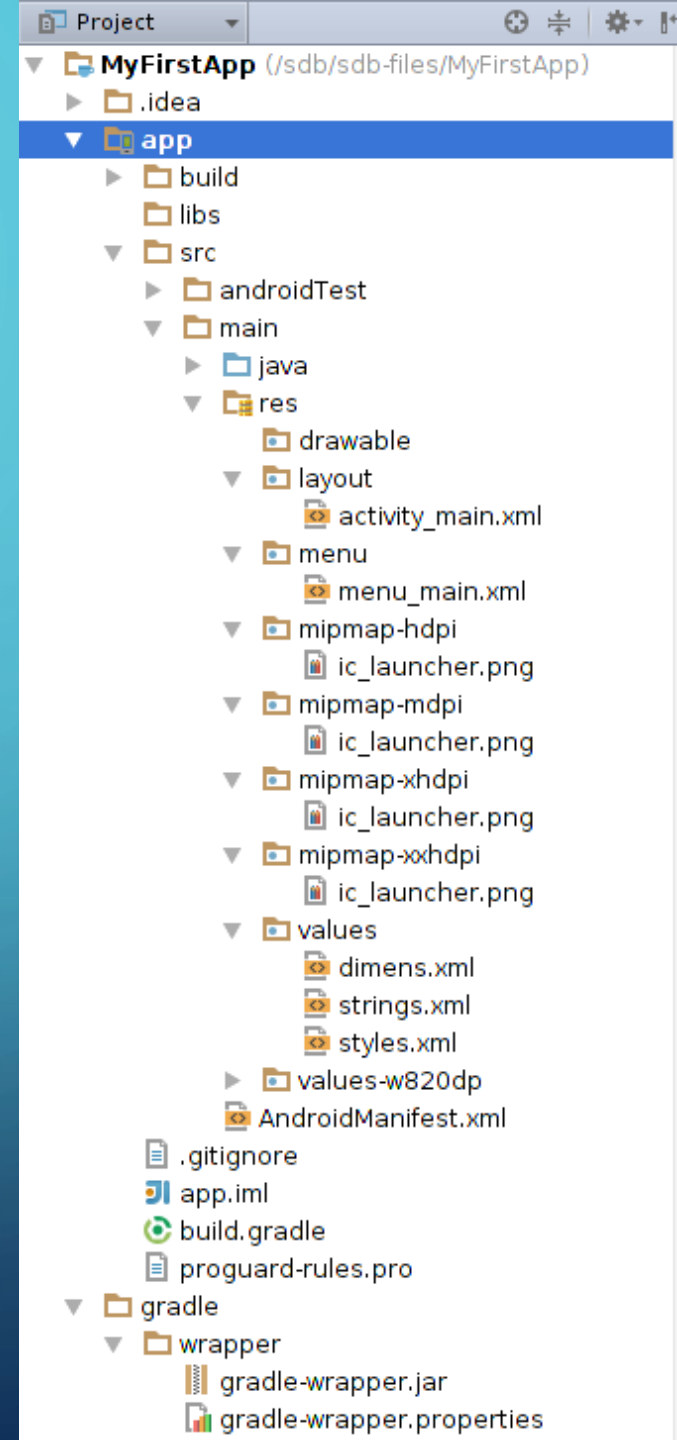
- Provides a container for your reusable code, which you can use as a dependency in other app modules or import into other projects. Structurally, a library module is the same as an app module, but when built, it creates a code archive file instead of an APK, so it can't be installed on a device.
- In the **Create New Module** window, Android Studio offers the following library modules:
 - **Android Library:** This type of library can contain all file types supported in an Android project, including source code, resources, and manifest files. The build result is an Android Archive (AAR) file that you can add as a dependency for your Android app modules.
 - **Java Library:** This type of library can contain only Java source files. The build result is an Java Archive (JAR) file that you can add as a dependency for your Android app modules or other Java projects.

GOOGLE CLOUD MODULE





- Provides a container for your Google Cloud backend code. This module has the required code and dependencies for a Java App Engine backend that uses simple HTTP, Cloud Endpoints, and Cloud Messaging to connect to your app. You can develop your backend to provide cloud services your app needs. Using Android Studio to develop your Google Cloud module lets you manage app code and backend code in the same project. You can also run and test your backend code locally, and use Android Studio to deploy your Google Cloud module.

THE ANDROID PROJECT VIEW

- To see the actual file structure of the project including all files hidden from the Android view, select **Project** from the dropdown at the top of the **Project** window.
- When you select **Project** view, you can see a lot more files and directories.



- `build/` Contains build outputs.
- `libs/` Contains private libraries.
- `src/` Contains all code and resource files for the module in the following subdirectories:
 - `androidTest/` Contains code for instrumentation tests that run on an Android device.
 - `main/` Contains the "main" sourceset files: the Android code and resources shared by all build variants
 - `AndroidManifest.xml` Describes the nature of the application and each of its components.
 - `java/` Contains Java code sources.
 - `jni/` Contains native code using the Java Native Interface (JNI).
 - `gen/` Contains the Java files generated by Android Studio, such as your `R.java` file and interfaces created from AIDL files.
 - `res/` Contains application resources, such as drawable files, layout files, and UI string.
 - `assets/` Contains file that should be compiled into an `.apk` file as-is. You can navigate this directory in the same way as a typical file system using URIs and read files as a stream of bytes using the `AssetManager`

- 
- 
- 
- 
- `test/` Contains code for local tests that run on your host JVM.
 - `build.gradle (module)` This defines the module-specific build configurations.
 - `build.gradle (project)` This defines your build configuration that apply to all modules. This file is integral to the project, so you should maintain them in revision control with all other source code.

PROJECT STRUCTURE SETTINGS

- To change various settings for your Android Studio project, open the Project Structure dialog by clicking File > Project Structure. It contains the following sections:
 - **SDK Location:** Sets the location of the JDK, Android SDK, and Android NDK that your project uses.
 - **Project:** Sets the version for Gradle and the Android plugin for Gradle, and the repository location name.
 - **Developer Services:** Contains settings for Android Studio add-in components from Google or other third parties. See Developer Services, below.
 - **Modules:** Allows you to edit module-specific build configurations, including the target and minimum SDK, the app signature, and library dependencies.

DEVELOPER SERVICES

- The *Developer Services* section of the *Project Structure* dialog box contains configuration pages for several services that you can use with your app. This section contains the following pages:
 - **AdMob:** Allows you to turn on Google's [AdMob](#) component, which helps you understand your users and show them tailored advertisements.
 - **Analytics:** Allows you to turn on [Google Analytics](#), which helps you measure user interactions with your app across various devices and environments.
 - **Authentication:** Allows users to use [Google Sign-In](#) to sign in to your app with their Google accounts.
 - **Cloud:** Allows you to turn on [Firebase](#) cloud-based services for your app.
 - **Notifications:** Allows you to use [Google Cloud Messaging](#) to communicate between your app and your server.
- Turning on any of these services may cause Android Studio to add necessary dependencies and permissions to your app. Each configuration page lists these and other actions that Android Studio takes if you enable the associated service.

ANDROID EMULATOR

- You have a couple of options when it comes to running your apps. The first option is to run them on a physical device. But what if you don't have one to hand, or you want to see how it looks on a type of device you don't have?
- An alternative option is to use the **Android emulator** that's built into the Android SDK. The emulator enables you to set up one or more **Android virtual devices** (AVDs) and then run your app in the emulator *as though it's running on a physical device*.
- The Android emulator allows you to run your app on an Android virtual device (AVD). The AVD behaves just like a physical Android device. You can set up numerous AVDs, each emulating a different type of device.

EMULATOR

- The emulator is an application that re-creates the exact hardware environment of an Android device: from its CPU and memory, through to the sound chips and the video display. The emulator is built on an existing emulator called QEMU, which is similar to other virtual machine applications you may have used, like VirtualBox or VMWare.
- The exact appearance and behavior of the AVD depends on how you've set up the AVD in the first place. The AVD here is set up to mimic a Nexus 4, so it will look and behave just like a Nexus 4 on your computer.

