Introduction to Android

Android Applications Development

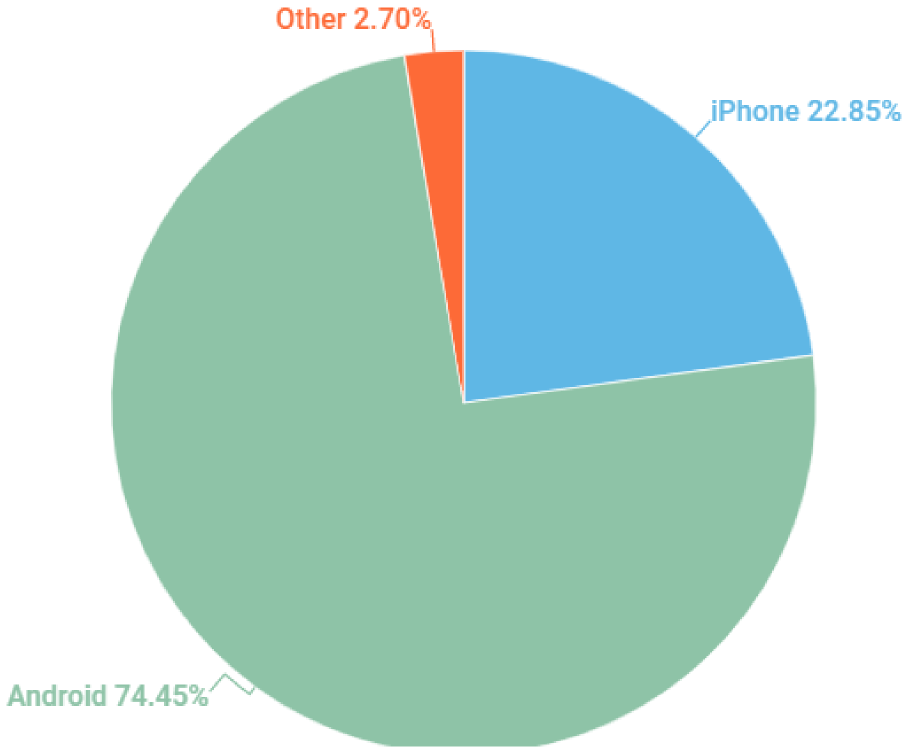
Programming apps

- Apps can be broadly classified into three distinct categories:

  native apps, dedicated web apps, and hybrid apps.

- Native applications are usually developed using higher level programming languages, such as Java for Android, Swift (Objective-C) for iOS.

iOS or Android ?



iPhone vs Android market share
Infogram

Android is..

Android is an operating system and programming platform developed by Google for smartphones and other mobile devices (such as tablets). It can run on many different devices from many different manufacturers.

Android includes a software development kit for writing original code and assembling software modules to create apps for Android users. It also provides a marketplace to distribute apps.

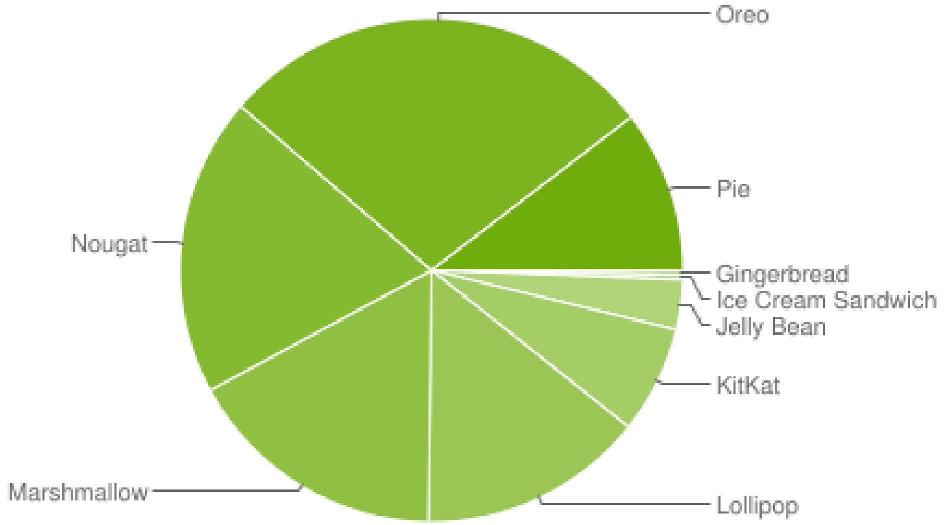
All together, Android represents an ecosystem for mobile apps.

History

- Android Inc. was founded in Palo Alto, California, in October **2003** by **Andy Rubin**, **Rich Miner**, **Nick Sears**, and **Chris White**. **Rubin** described the Android project as "tremendous potential in developing smarter mobile devices that are more aware of its owner's location and preferences".

- In **2005**, the next big chapter in Android's history was made when the original company was acquired by Google.
- In late **2014**, Rubin left Google altogether and launched a startup business incubator.
- Earlier in **2017**, Rubin officially revealed his return to the smartphone industry with his company's announcement of the Android-based Essential Phone.

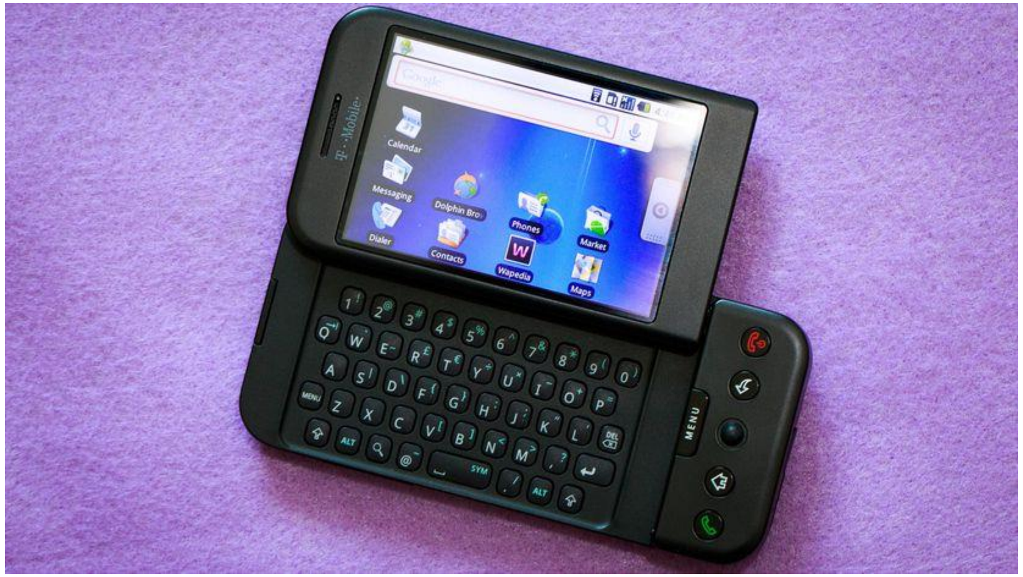
Android Versions



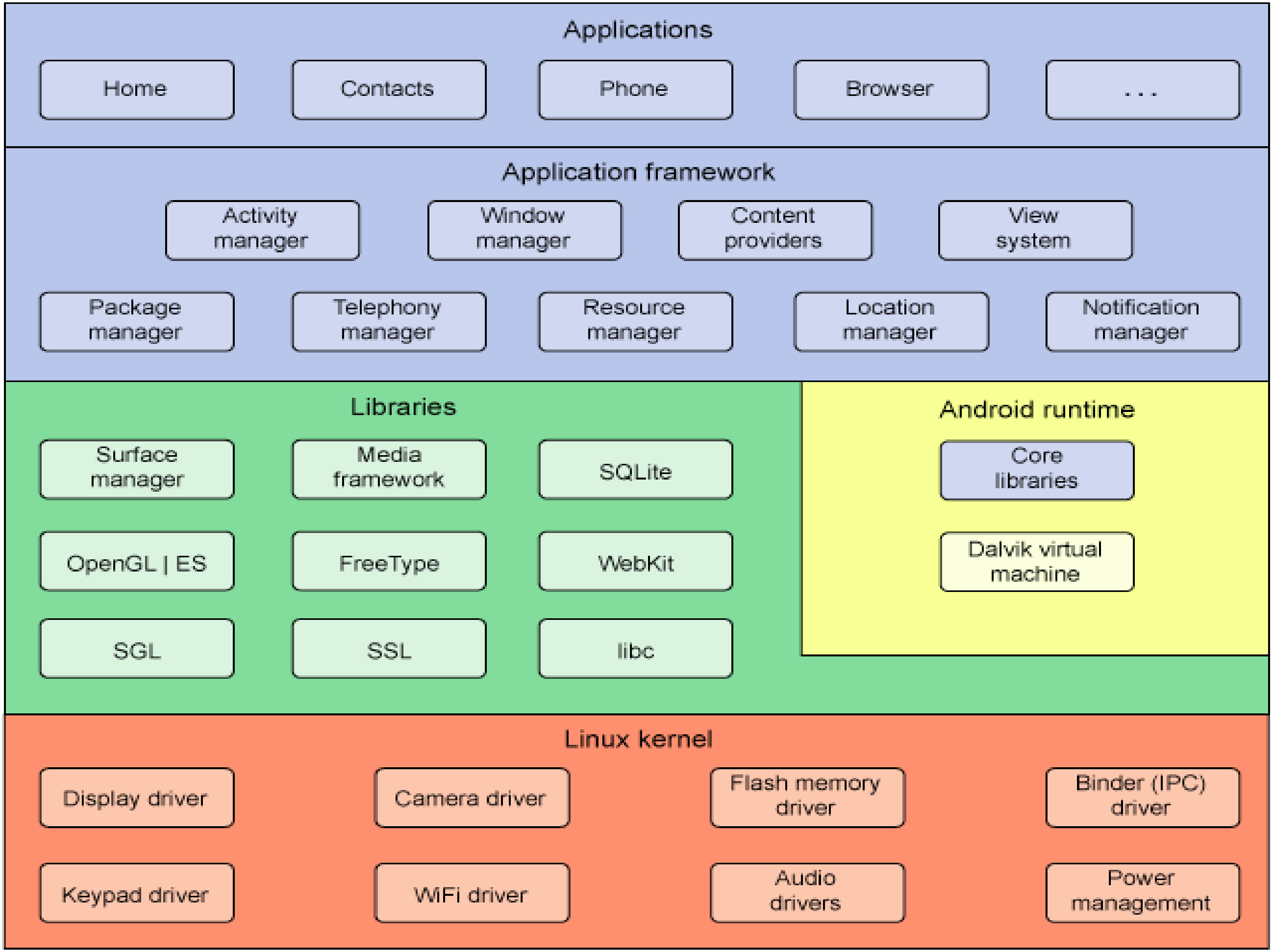
First Android Phone

First android phone was launched by HTC on 22nd October 2008.

HTC Dream which is also known as the T-**Mobile** G1 in the United States of America and some parts of Europe is the **first** commercially launched **device** to be powered by **Android** operating system



Android architecture



Applications

Home | Contacts | Phone | Browser | . . .

Application framework

Activity manager | Window manager | Content providers | View system

Package manager | Telephony manager | Resource manager | Location manager | Notification manager

Libraries

Surface manager | Media framework | SQLite

OpenGL | ES | FreeType | WebKit

SGL | SSL | libc

Android runtime

Core libraries

Dalvik virtual machine

Linux kernel

Display driver | Camera driver | Flash memory driver | Binder (IPC) driver

Keypad driver | WiFi driver | Audio drivers | Power management

Android architecture

1. **Linux kernel**
   - *This provides a level of abstraction between the device hardware and it contains all the essential hardware drivers like camera, keypad, display etc. Also, the kernel handles all the things that Linux is really good at such as networking and a vast array of device drivers, which take the pain out of interfacing to peripheral hardware.*
2. **Libraries**
   - *On top of Linux kernel there is a set of libraries including open-source Web browser engine WebKit, well known library libc, SQLite database which is a useful repository for storage and sharing of application data, libraries to play and record audio and video, SSL libraries responsible for Internet security etc.*
3. **Android Runtime**
   - *This is the third section of the architecture and available on the second layer from the bottom. This section provides a key component called Dalvik Virtual Machine which is a kind of Java Virtual Machine specially designed and optimized for Android. The Dalvik VM makes use of Linux core features like memory management and multi-threading, which is intrinsic in the Java language. The Dalvik VM enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine. The Android runtime also provides a set of core libraries which enable Android application developers to write Android applications using standard Java programming language.*

Android architecture

4. **Application Framework**

The Application Framework layer provides many higher-level services to applications in the form of Java classes. Application developers are allowed to make use of these services in their applications.

The Android framework includes the following key services −

- **Activity Manager** − Controls all aspects of the application lifecycle and activity stack.
- **Content Providers** − Allows applications to publish and share data with other applications.
- **Resource Manager** − Provides access to non-code embedded resources such as strings, color settings and user interface layouts.
- **Notifications Manager** − Allows applications to display alerts and notifications to the user.
- **View System** − An extensible set of views used to create application user interfaces.

5. **Applications**

You will find all the Android application at the top layer. You will write your application to be installed on this layer only. Examples of such applications are Contacts Books, Browser, Games etc.

IDE

Android Studio is an official IDE for developing android applications

- Free and Simple
- Open Source
- Written in java
- Integrated gradle build system

Setting up IDE

To be an Android developer, you need

- Android Studio , which in turn contains
  - Android SDK (tools, platform-tools/build-tools)
  - A version of the Android platform
  - A version of the Android system image for the emulator
  - Other tools (usb drivers etc.)

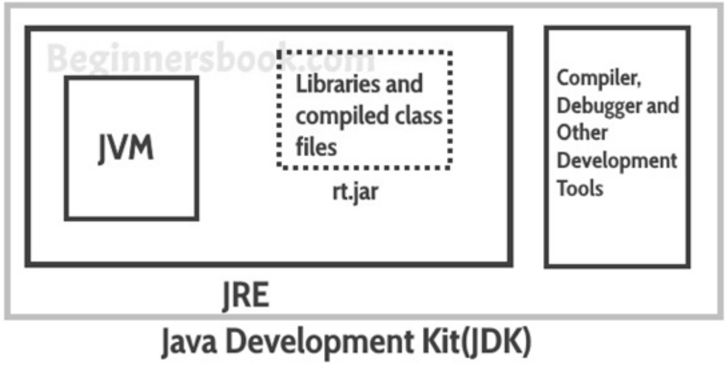
Step 1: Install java

Choose JDK

- Java Development Toolkit
- Contains complete JRE
- Compilers
- Debuggers

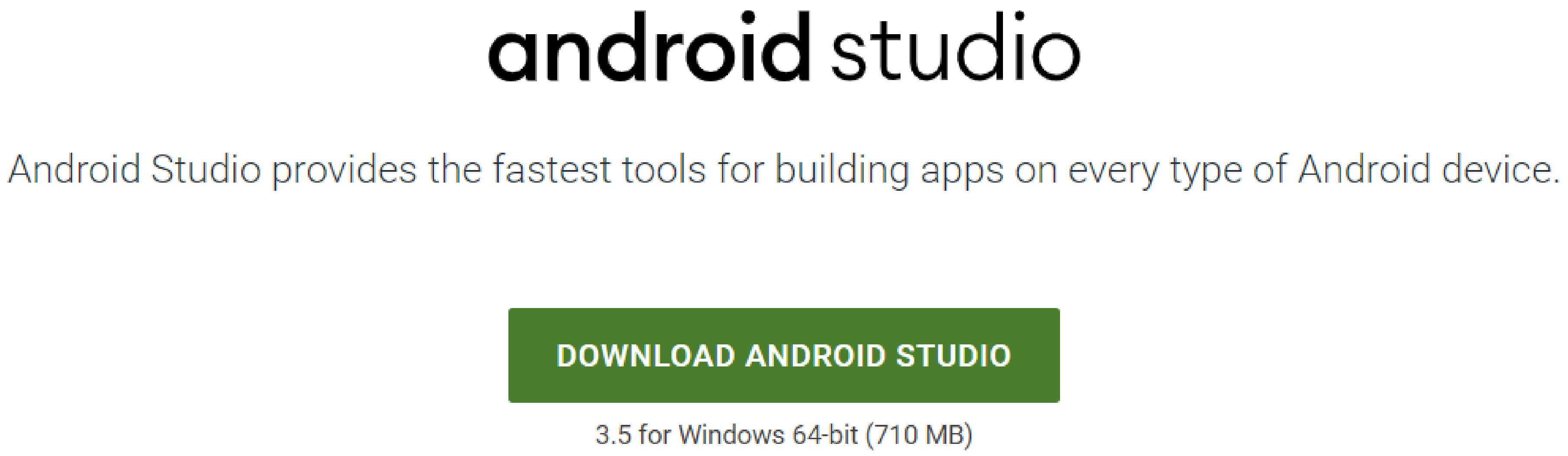
Not JRE

- Java Runtime Environment
- Libraries
- Java Virtual Machine
- Components to run applets


Beginnersbook.com
JVM
Libraries and compiled class files
rt.jar
Compiler, Debugger and Other Development Tools
JRE
Java Development Kit(JDK)

Step 2 : Install Android Studio

https://developer.android.com/studio/



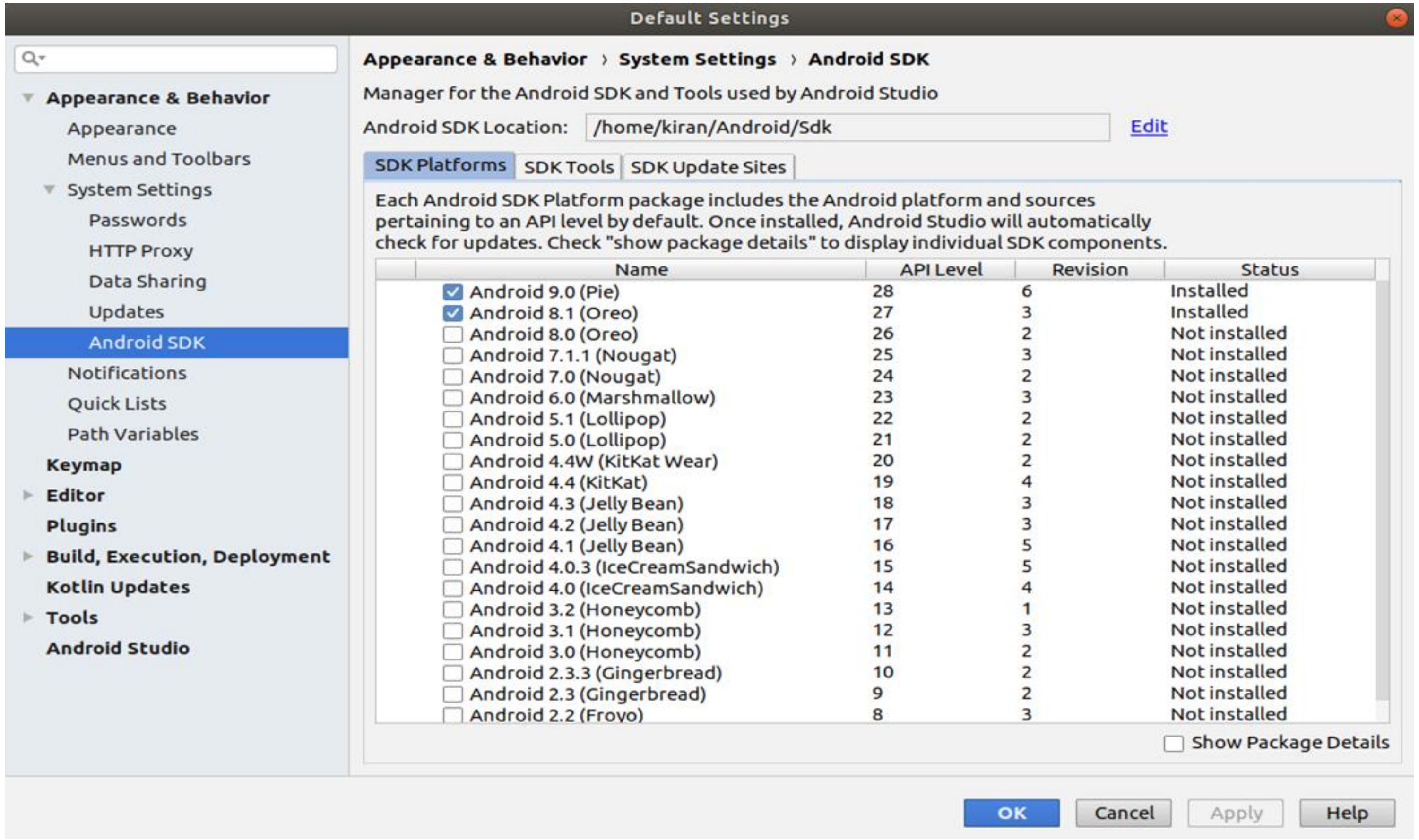
android studio

Android Studio provides the fastest tools for building apps on every type of Android device.

3.5 for Windows 64-bit (710 MB)

Step 3 : Update SDK

To open SDK Manager: **Android Studio -> Tools -> SDK Manager**



Appearance & Behavior › System Settings › Android SDK

Manager for the Android SDK and Tools used by Android Studio

Android SDK Location: /home/kiran/Android/Sdk

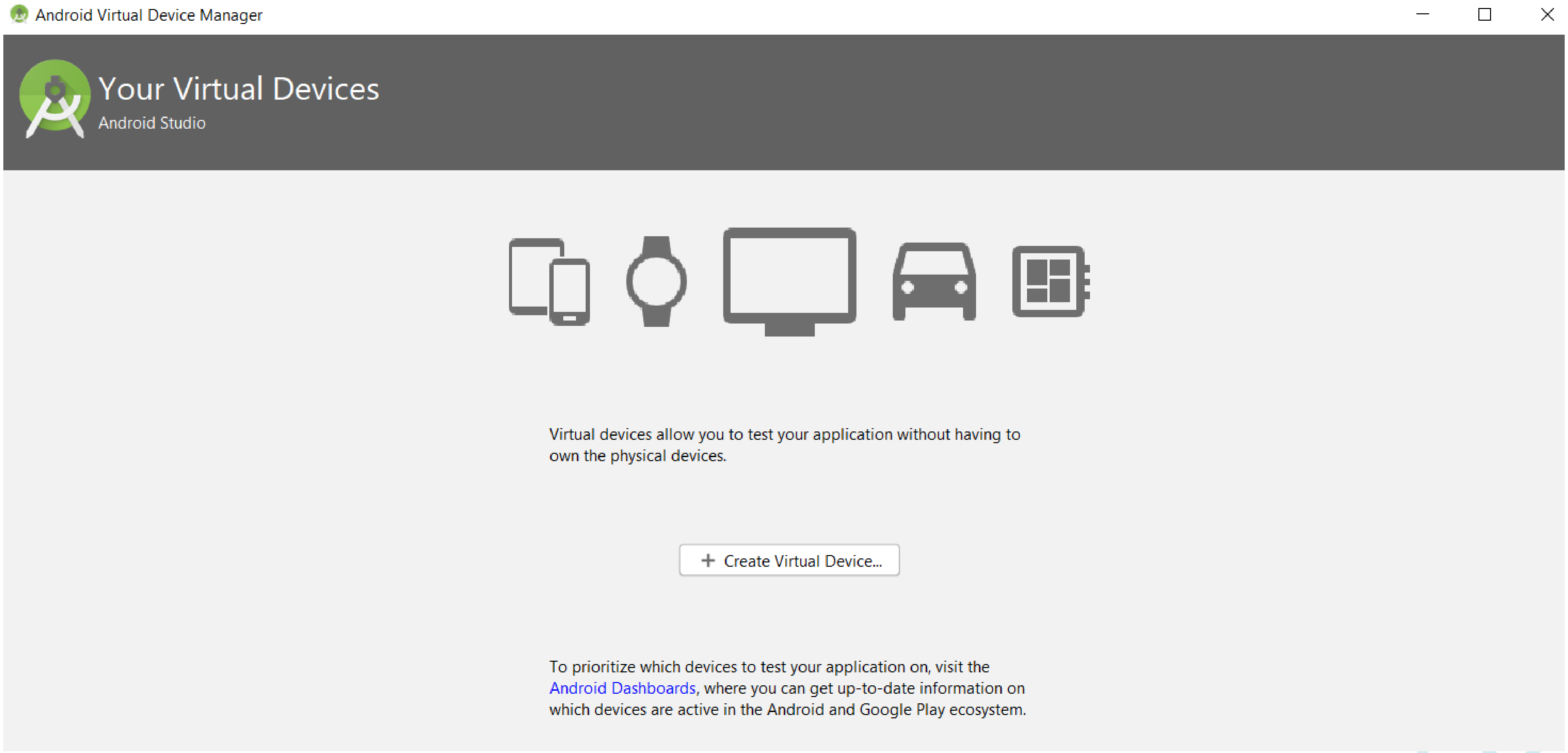
Each Android SDK Platform package includes the Android platform and sources pertaining to an API level by default. Once installed, Android Studio will automatically check for updates. Check "show package details" to display individual SDK components.

Name	API Level	Revision	Status
☑ Android 9.0 (Pie)	28	6	Installed
☑ Android 8.1 (Oreo)	27	3	Installed
□ Android 8.0 (Oreo)	26	2	Not installed
□ Android 7.1.1 (Nougat)	25	3	Not installed
□ Android 7.0 (Nougat)	24	2	Not installed
□ Android 6.0 (Marshmallow)	23	3	Not installed
□ Android 5.1 (Lollipop)	22	2	Not installed
□ Android 5.0 (Lollipop)	21	2	Not installed
□ Android 4.4W (KitKat Wear)	20	2	Not installed
□ Android 4.4 (KitKat)	19	4	Not installed
□ Android 4.3 (Jelly Bean)	18	3	Not installed
□ Android 4.2 (Jelly Bean)	17	3	Not installed
□ Android 4.1 (Jelly Bean)	16	5	Not installed
□ Android 4.0.3 (IceCreamSandwich)	15	5	Not installed
□ Android 4.0 (IceCreamSandwich)	14	4	Not installed
□ Android 3.2 (Honeycomb)	13	1	Not installed
□ Android 3.1 (Honeycomb)	12	3	Not installed
□ Android 3.0 (Honeycomb)	11	2	Not installed
□ Android 2.3.3 (Gingerbread)	10	2	Not installed
□ Android 2.3 (Gingerbread)	9	2	Not installed
□ Android 2.2 (Froyo)	8	3	Not installed

□ Show Package Details

Step 4 : Create AVDs (Android virtual devices)

To open SDK Manager: Android Studio -> Tools -> AVD Manager



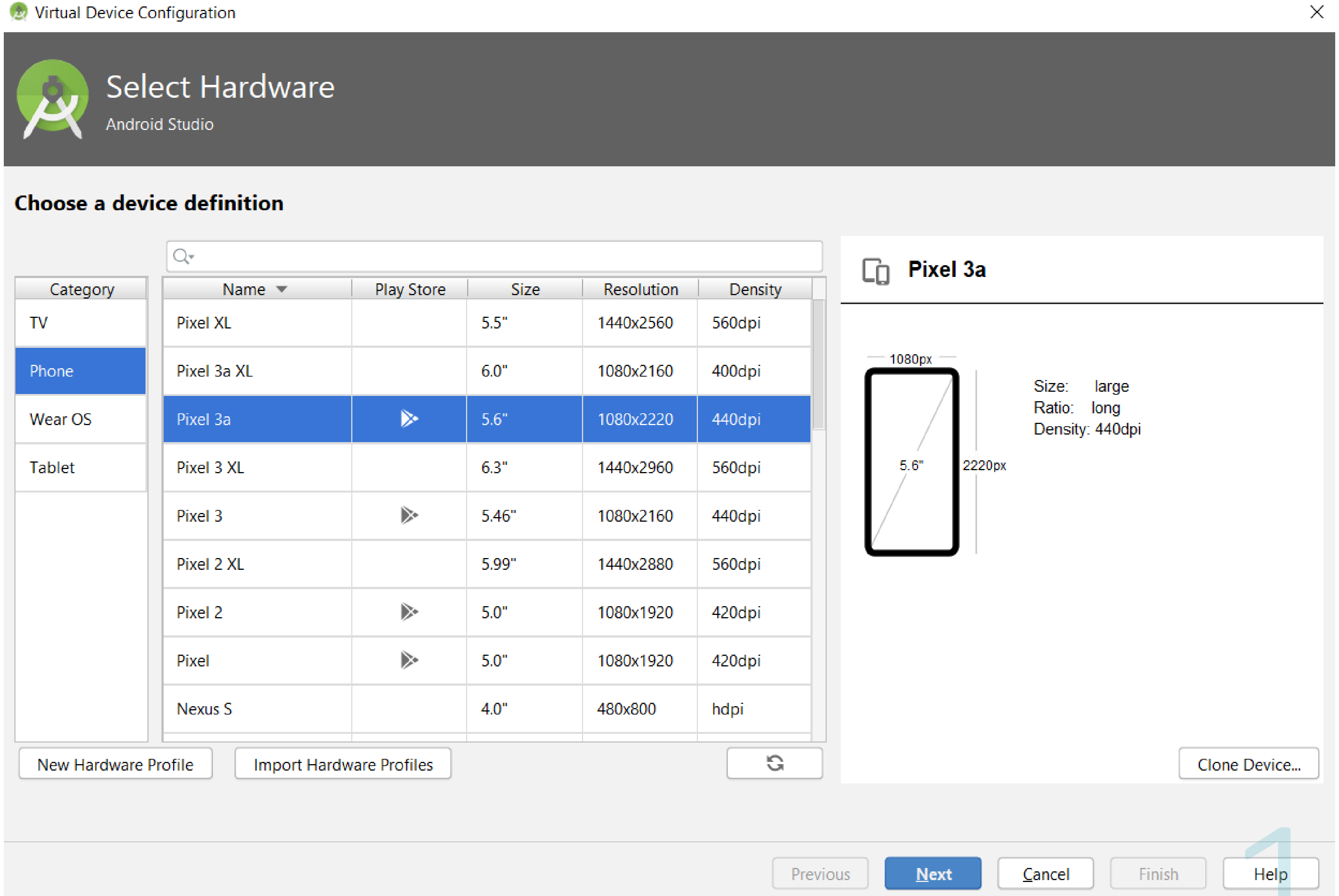
Your Virtual Devices

Android Studio

Virtual devices allow you to test your application without having to own the physical devices.

To prioritize which devices to test your application on, visit the Android Dashboards, where you can get up-to-date information on which devices are active in the Android and Google Play ecosystem.

Step 4 : Create AVDs (Android virtual devices)



Virtual Device Configuration

Select Hardware

Android Studio

Choose a device definition

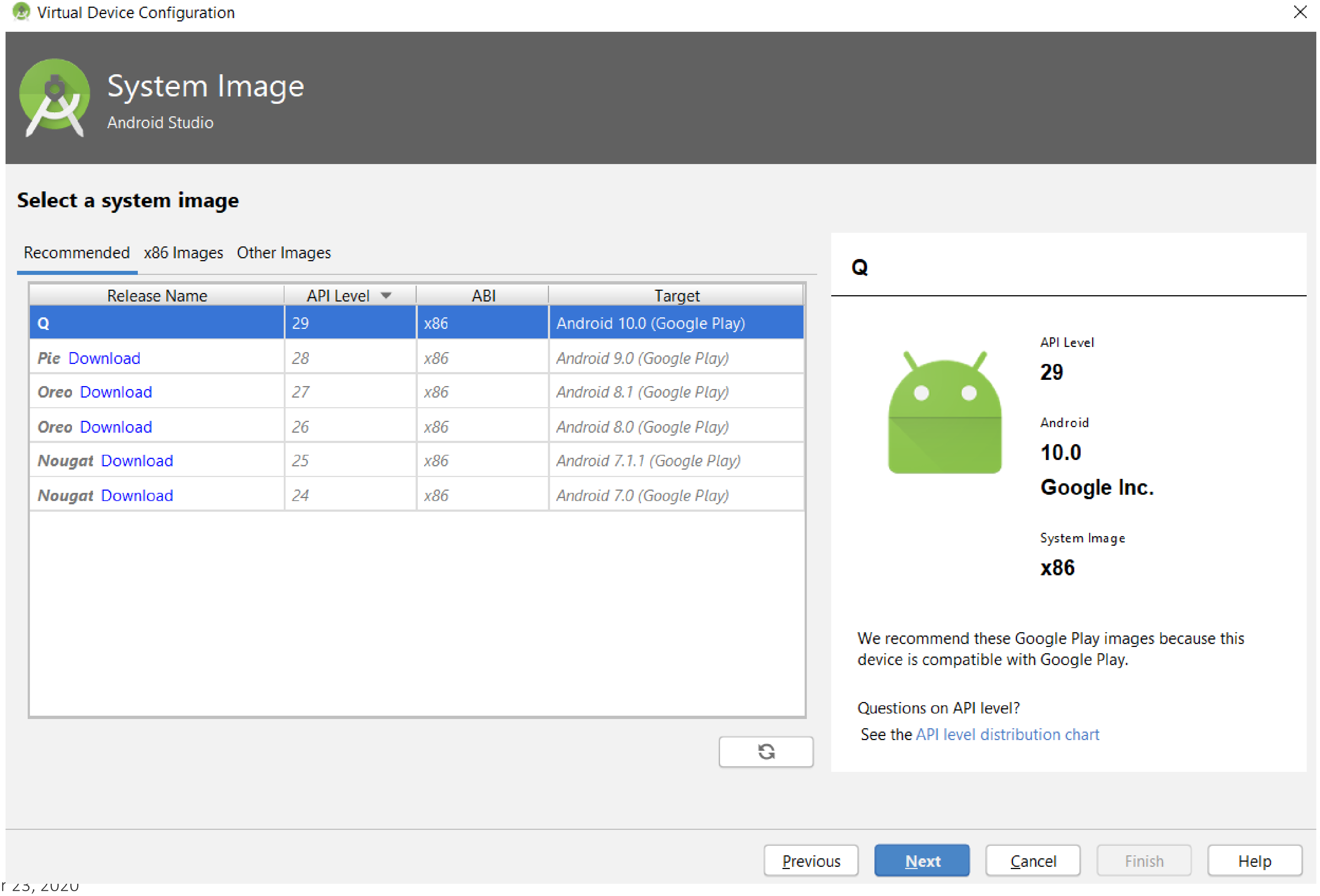
Category
TV
Phone
Wear OS
Tablet

Name	Play Store	Size	Resolution	Density
Pixel XL		5.5"	1440x2560	560dpi
Pixel 3a XL		6.0"	1080x2160	400dpi
Pixel 3a	▶	5.6"	1080x2220	440dpi
Pixel 3 XL		6.3"	1440x2960	560dpi
Pixel 3	▶	5.46"	1080x2160	440dpi
Pixel 2 XL		5.99"	1440x2880	560dpi
Pixel 2	▶	5.0"	1080x1920	420dpi
Pixel	▶	5.0"	1080x1920	420dpi
Nexus S		4.0"	480x800	hdpi

Pixel 3a

Size: large
Ratio: long
Density: 440dpi

Step 4 : Create AVDs (Android virtual devices)



System Image

Android Studio

Select a system image

Release Name	API Level	ABI	Target
Q	29	x86	Android 10.0 (Google Play)
Pie Download	28	x86	Android 9.0 (Google Play)
Oreo Download	27	x86	Android 8.1 (Google Play)
Oreo Download	26	x86	Android 8.0 (Google Play)
Nougat Download	25	x86	Android 7.1.1 (Google Play)
Nougat Download	24	x86	Android 7.0 (Google Play)

Q

API Level

29

Android

10.0

Google Inc.

System Image

x86

We recommend these Google Play images because this device is compatible with Google Play.

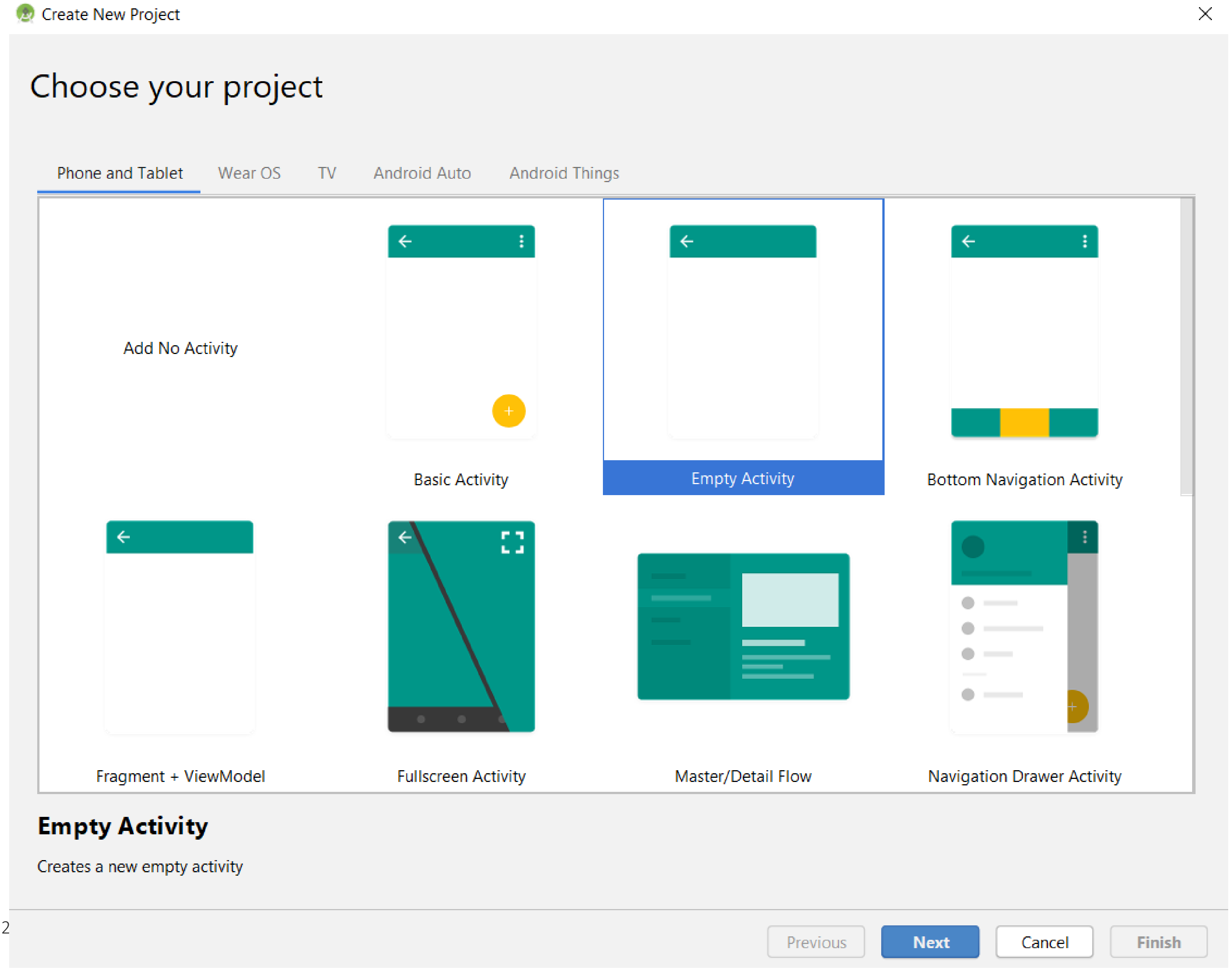
Questions on API level?

See the API level distribution chart

Step 5 : Testing device

- Enable USB debugging on your device.
- Set up your system to detect your device
  - Enable debug mode
- Setup your computer i.e. install drivers etc.
  - depends on your computer/phone
- Check here
  - https://developer.android.com/studio/run/device.html
  - https://developer.android.com/studio/run/oem-usb#Drivers

First Android app



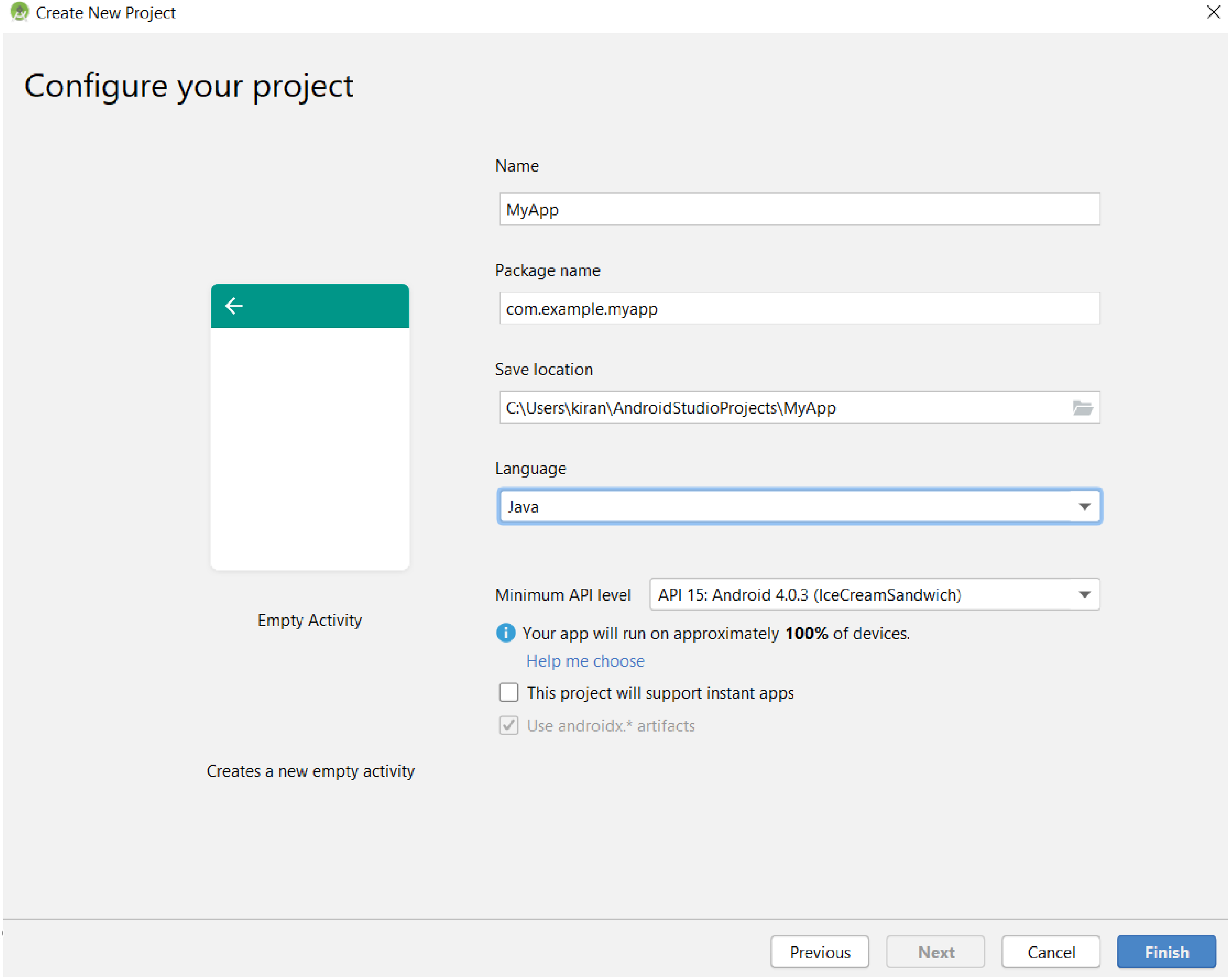
Create New Project

Choose your project

Empty Activity

Creates a new empty activity

First Android app



Create New Project

Configure your project

Name: MyApp

Package name: com.example.myapp

Save location: C:\Users\kiran\AndroidStudioProjects\MyApp

Language: Java

Minimum API level: API 15: Android 4.0.3 (IceCreamSandwich)

Your app will run on approximately **100%** of devices.

Help me choose

□ This project will support instant apps

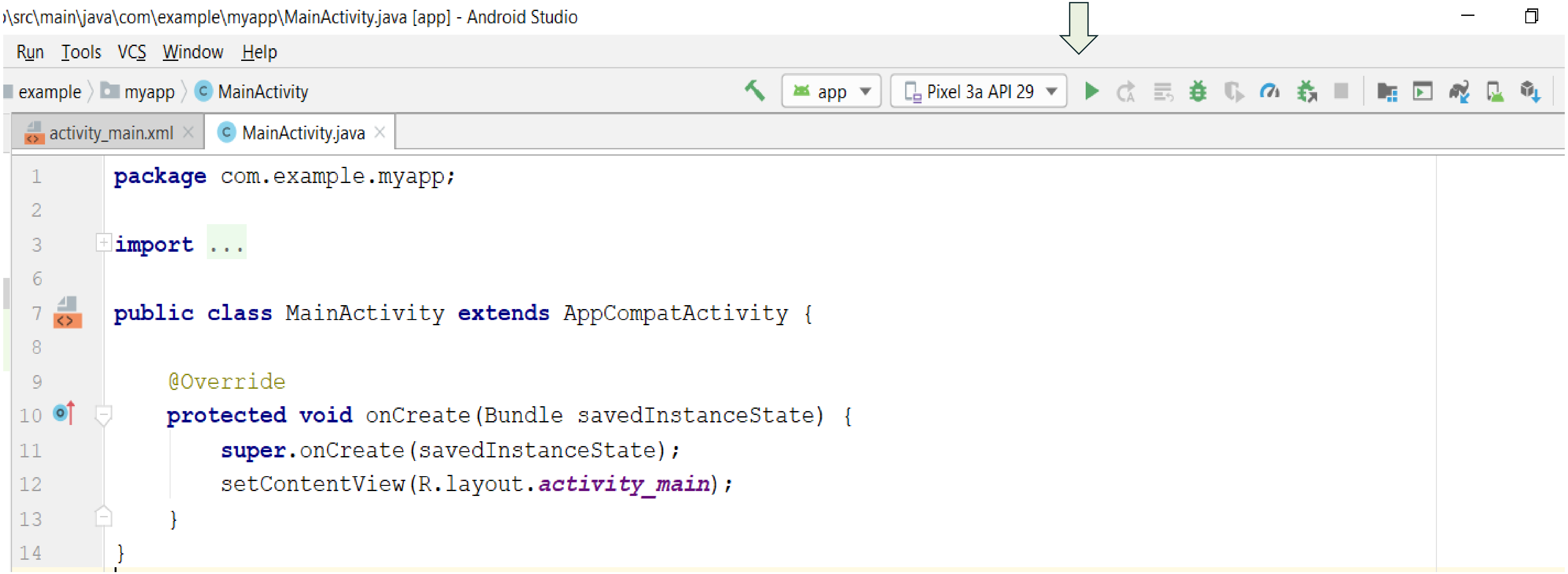
☑ Use androidx.* artifacts

Empty Activity

Creates a new empty activity

Running android app

Click here to run android app



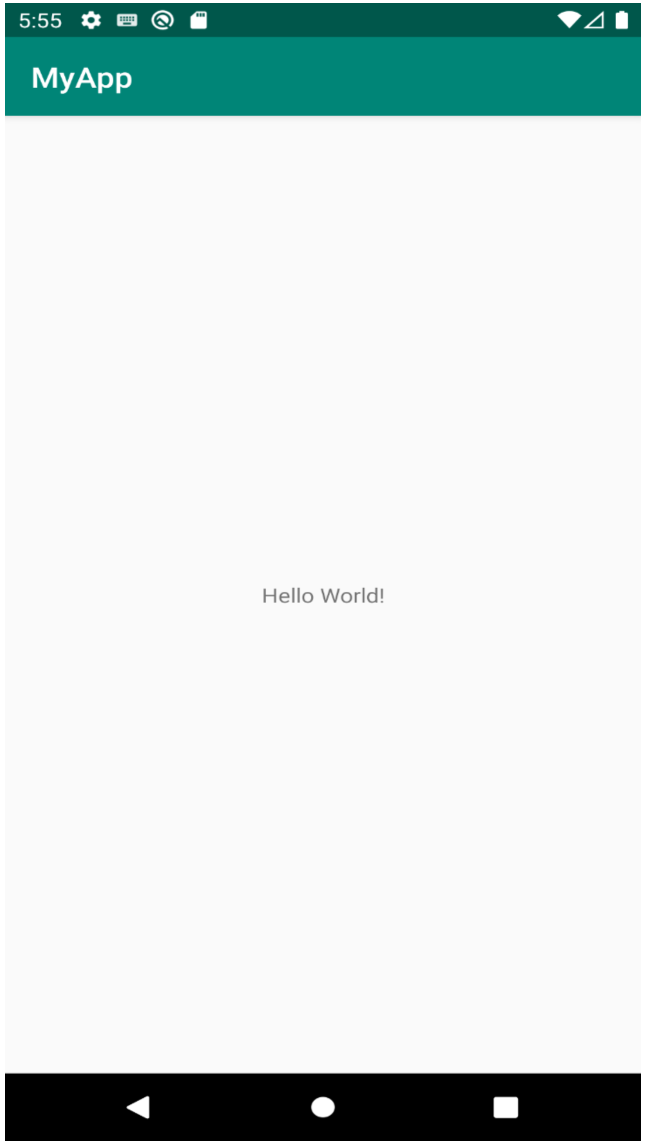
```java
package com.example.myapp;

import ...

public class MainActivity extends AppCompatActivity {

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

Output



Hello World!

build.gradle

Gradle is a build system (open source) which is used to automate building, testing, deployment etc. “Build.gradle” are scripts where one can automate the tasks.

For example, the simple task to copy some files from one directory to another can be performed by Gradle build script before the actual build process happens.

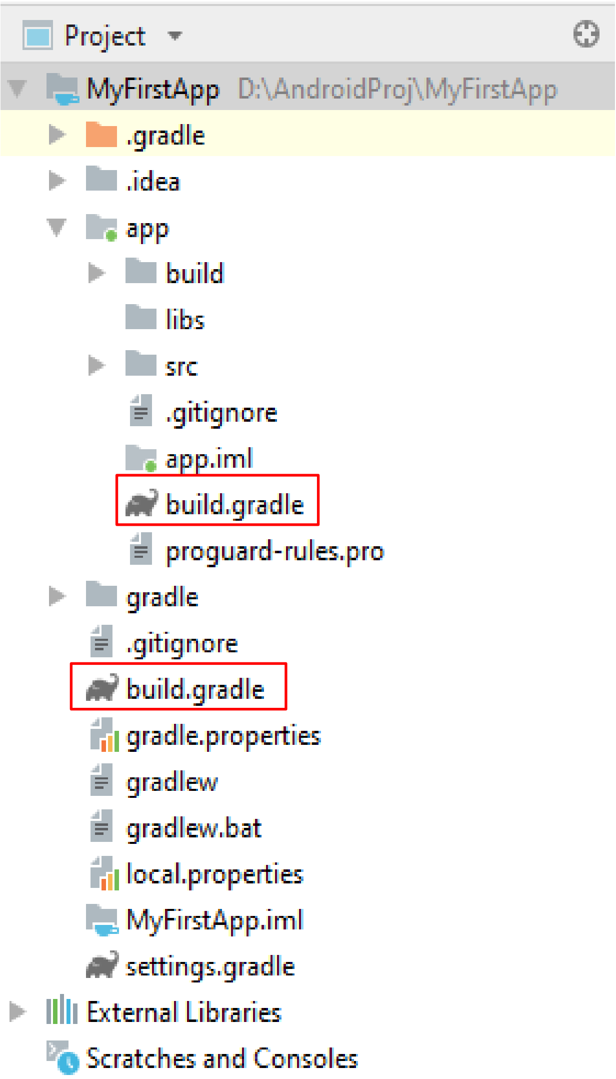
Every Android project needs a gradle for generating an apk from the *.java* and *.xml* files in the project.

Simply put, a gradle takes all the source files (java and XML) and apply appropriate tools, e.g., converts the java files into dex files and compresses all of them into a single file known as apk that is actually used.

Types of build.gradle scripts

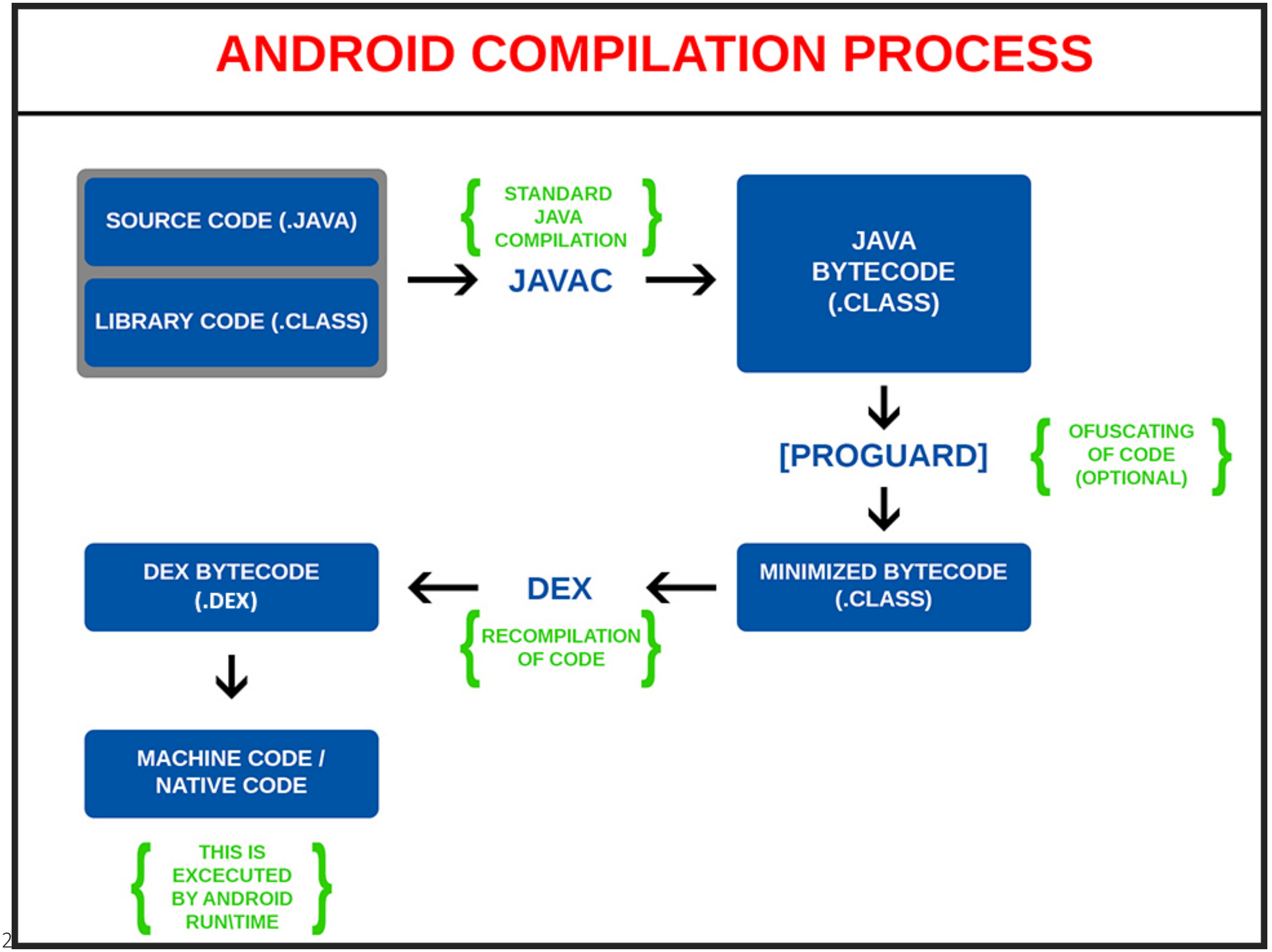
There are two types of build.gradle scripts

1. **Top-level** build.gradle
2. **Module-level** build.gradle



```
Project ▾
▼ MyFirstApp  D:\AndroidProj\MyFirstApp
  ▶ .gradle
  ▶ .idea
  ▼ app
    ▶ build
      libs
    ▶ src
      .gitignore
      app.iml
      build.gradle
      proguard-rules.pro
  ▶ gradle
    .gitignore
    build.gradle
    gradle.properties
    gradlew
    gradlew.bat
    local.properties
    MyFirstApp.iml
    settings.gradle
▶ External Libraries
  Scratches and Consoles
```

Android Compilation Process



ANDROID COMPILATION PROCESS

SOURCE CODE (.JAVA)

LIBRARY CODE (.CLASS)

STANDARD JAVA COMPILATION

JAVAC

JAVA BYTECODE (.CLASS)

[PROGUARD]

OFUSCATING OF CODE (OPTIONAL)

MINIMIZED BYTECODE (.CLASS)

DEX

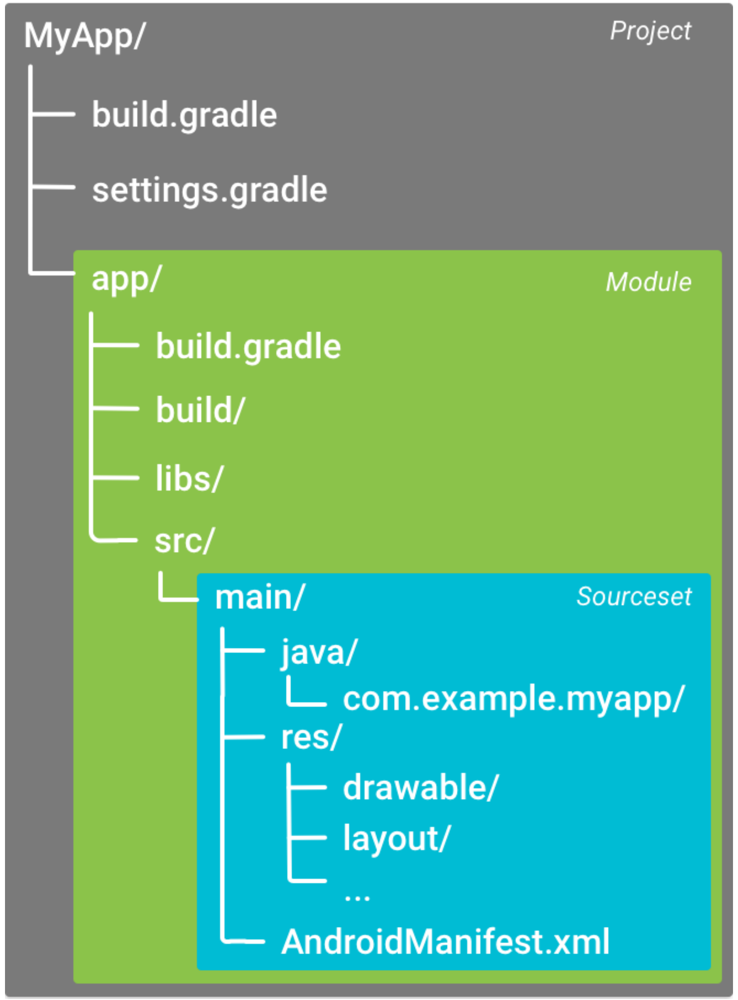
RECOMPILATION OF CODE

DEX BYTECODE (.DEX)

MACHINE CODE / NATIVE CODE

THIS IS EXCECUTED BY ANDROID RUN\TIME

The build process



MyApp/ *Project*

build.gradle

settings.gradle

app/ *Module*

build.gradle

build/

libs/

src/

main/ *Sourceset*

java/

com.example.myapp/

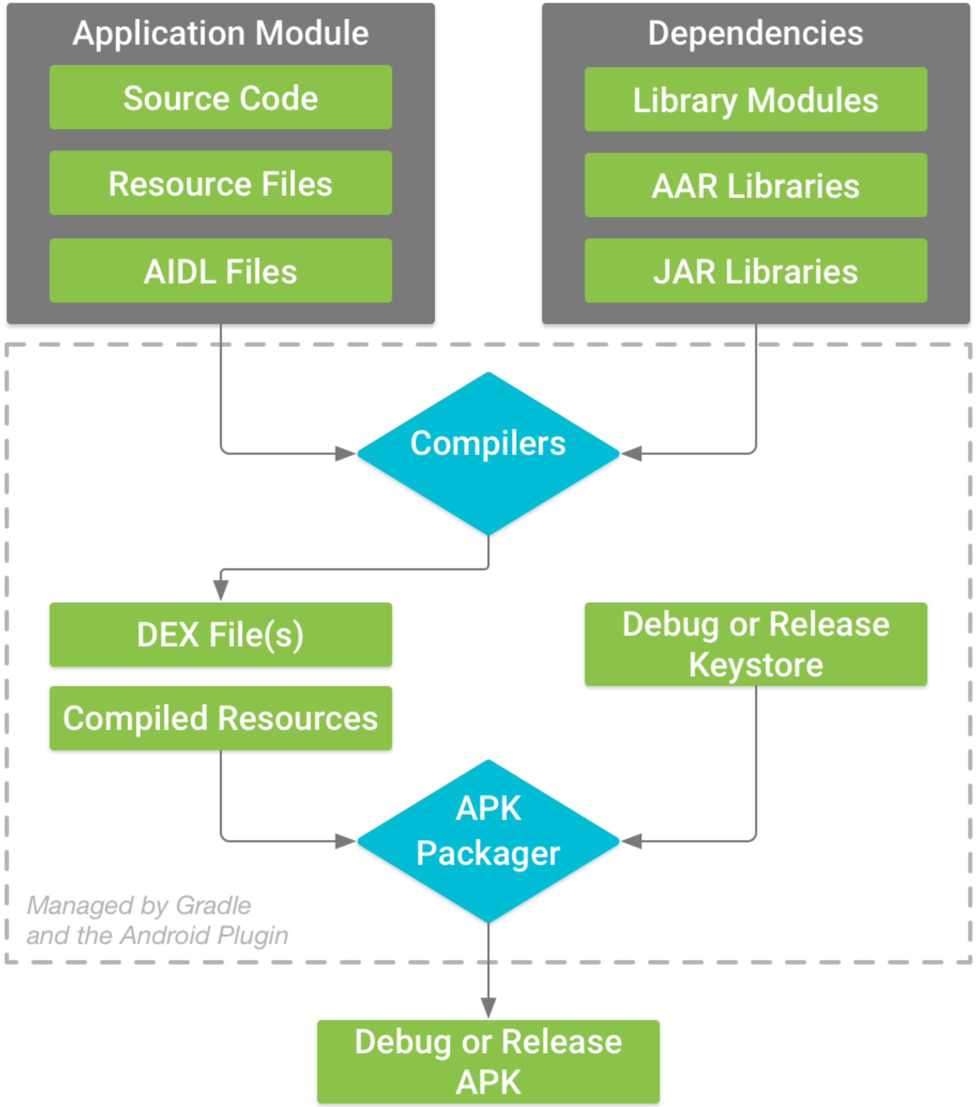
res/

drawable/

layout/

...

AndroidManifest.xml



Application Module

Source Code

Resource Files

AIDL Files

Dependencies

Library Modules

AAR Libraries

JAR Libraries

Compilers

DEX File(s)

Compiled Resources

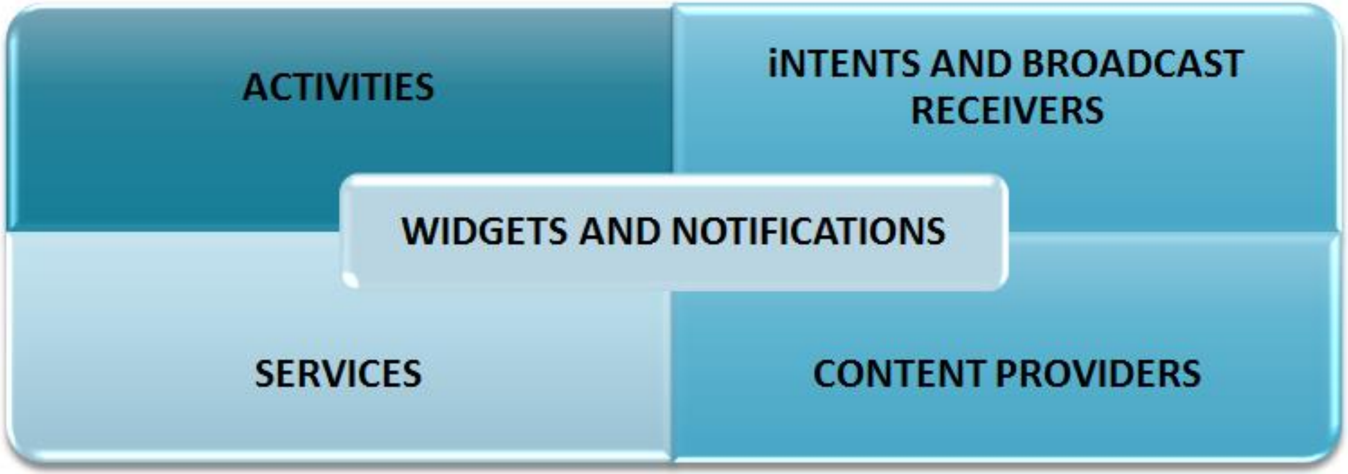
Debug or Release Keystore

APK Packager

Managed by Gradle and the Android Plugin

Debug or Release APK

Basic Parts of Android Application



ACTIVITIES

iNTENTS AND BROADCAST RECEIVERS

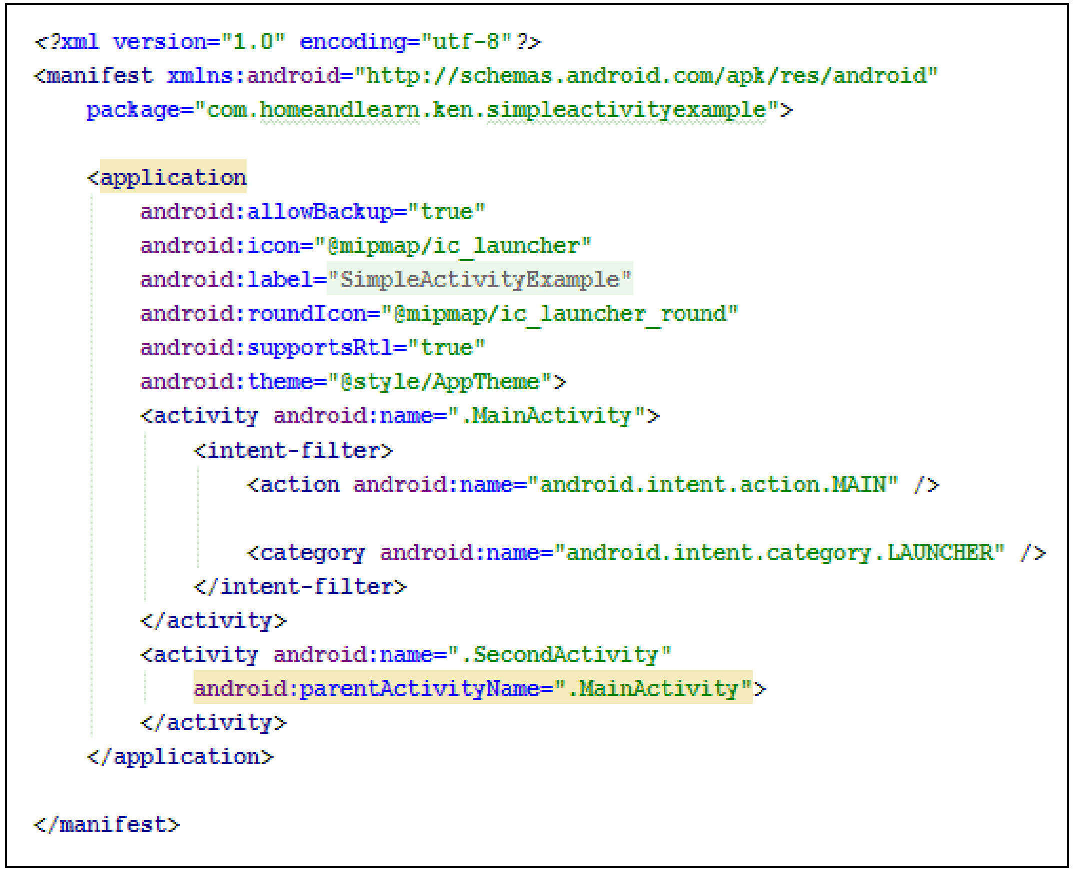
WIDGETS AND NOTIFICATIONS

SERVICES

CONTENT PROVIDERS

The Android Manifest.xml File

- Every application must have an AndroidManifest.xml file (with precisely that name) in its root directory. The manifest presents essential information about the application to the Android system, information the system must have before it can run any of the application's code. Among other things, the manifest does the following:
- It names the Java package for the application. The package name serves as a unique identifier for the application.
- It describes the components of the application — **the activities, services, broadcast receivers, and content providers** that the application is composed of.
- It declares which **permissions** the application must have in order to access protected parts of the API and interact with other applications.
- It determines which processes will host application components.



```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.homeandlearn.ken.simpleactivityexample">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="SimpleActivityExample"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".SecondActivity"
            android:parentActivityName=".MainActivity">
        </activity>
    </application>

</manifest>
```