ANDROID INTRODUCTION

Basics of Android

- What is Android?

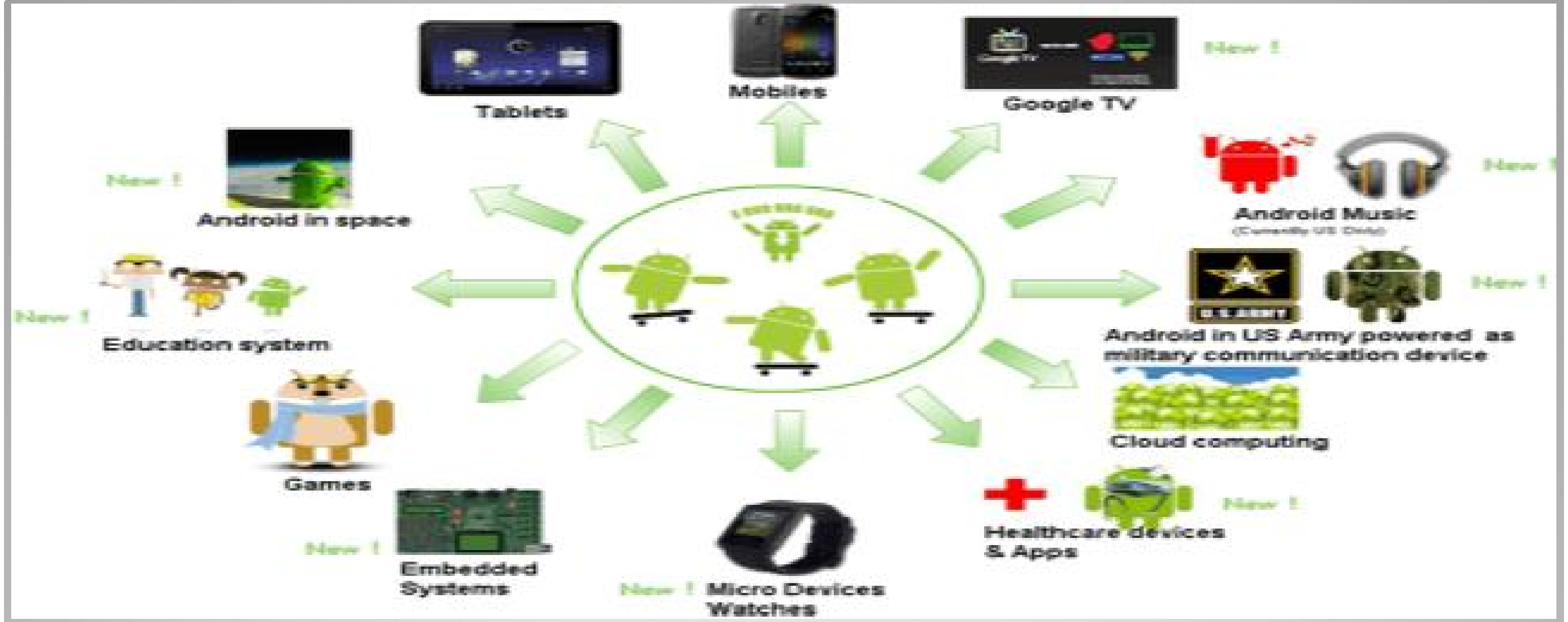
Android is a mobile **operating system** based on the linux and kernel. It's maintained by **Google**, and comes in a few different versions. At the time of writing, mobile phones run a variant of version 2 of Android, while most new tablets run a variant of version 3.

Android's standard layout is to have a series of Home screens, which can contain shortcuts to launch apps, or can contain widgets, which are small programs that serve a single function, such as controlling your music or displaying Facebook updates.

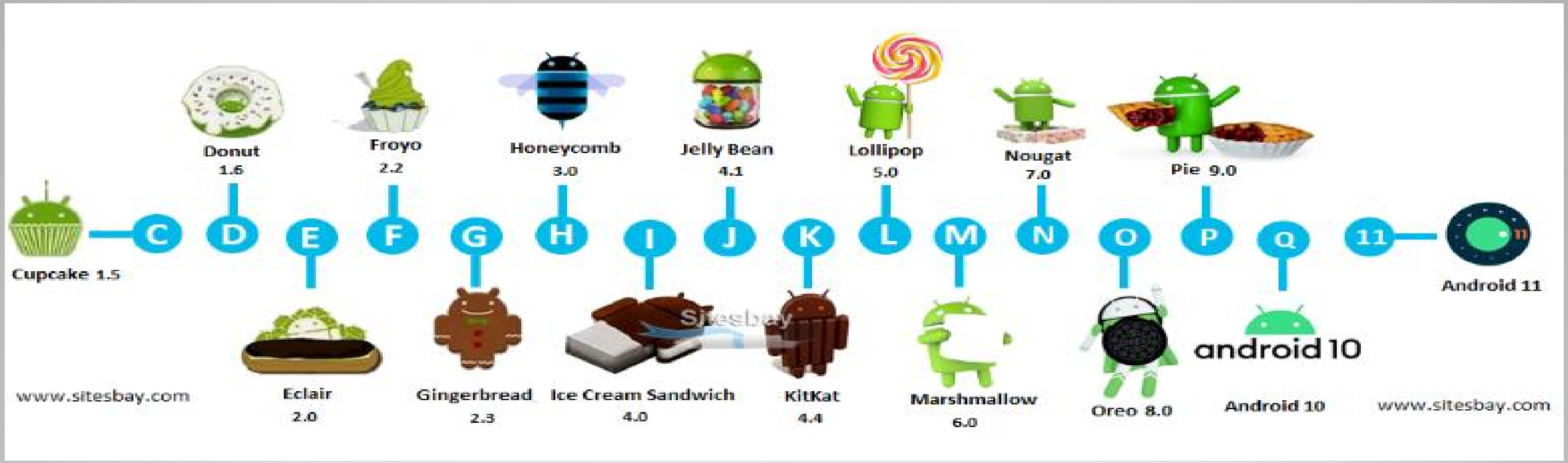
History and version of Android

It is very interesting to know history and version of android.

- Android was founded by **Andy Rubin**, **Rich Minner**, and **Charris White**.
- The early intentions of the company were to develop an advanced operating system for digital cameras, when it was realized that the market for the devices was not large enough, and diverted their efforts to producing a smart phone operating system to rival those of Symbian and Windows Mobile.
- Initially android was developed for Television, video games, digital cameras and electronics.



- Later android apps were shifted to mobiles.
- After 2 years Google bought the Android in 2005.
- In 2007 Google officially announced development of Android OS.
- Features are going to be added one version to another version.



VERSIONS OF ANDROID

Code name	Version numbers	API level	Release date
No codename	1.0	1	September 23, 2008
No codename	1.1	2	February 9, 2009
Cupcake	1.5	3	April 27, 2009
Donut	1.6	4	September 15, 2009
Eclair	2.0 – 2.1	5 – 7	October 26, 2009
Froyo	2.2 – 2.2.3	8	May 20, 2010
Gingerbread	2.3 – 2.3.7	9 – 10	December 6, 2010
Honeycomb	3.0 – 3.2.6	11 – 13	February 22, 2011
Ice Cream Sandwich	4.0 – 4.0.4	14 – 15	October 18, 2011
Jelly Bean	4.1 – 4.3.1	16 – 18	July 9, 2012
KitKat	4.4 – 4.4.4	19 – 20	October 31, 2013
Lollipop	5.0 – 5.1.1	21– 22	November 12, 2014
Marshmallow	6.0 – 6.0.1	23	October 5, 2015
Nougat	7.0	24	August 22, 2016
Nougat	7.1.0 – 7.1.2	25	October 4, 2016
Oreo	8.0	26	August 21, 2017
Oreo	8.1	27	December 5, 2017
Pie	9.0	28	August 6, 2018
Android 10	10.0	29	September 3, 2019
Android 11	11	30	September 8, 2020

Android versions, name, and API level

Android core building block

- A component is simply a piece of code that has a well defined life cycle. e.g.-Activity, Receiver, service etc.
- The **core building blocks** or **fundamental components** of android are activities, views, intents, services, content providers, fragments and AndroidManifest.xml.

❑ **Activity**

An activity is a class that represents a single screen. It is like a frame in AWT.

❑ **View**

A view is the UI element such as button, label, text, field etc.

Anything that you see is a view.

❑ **Intent**

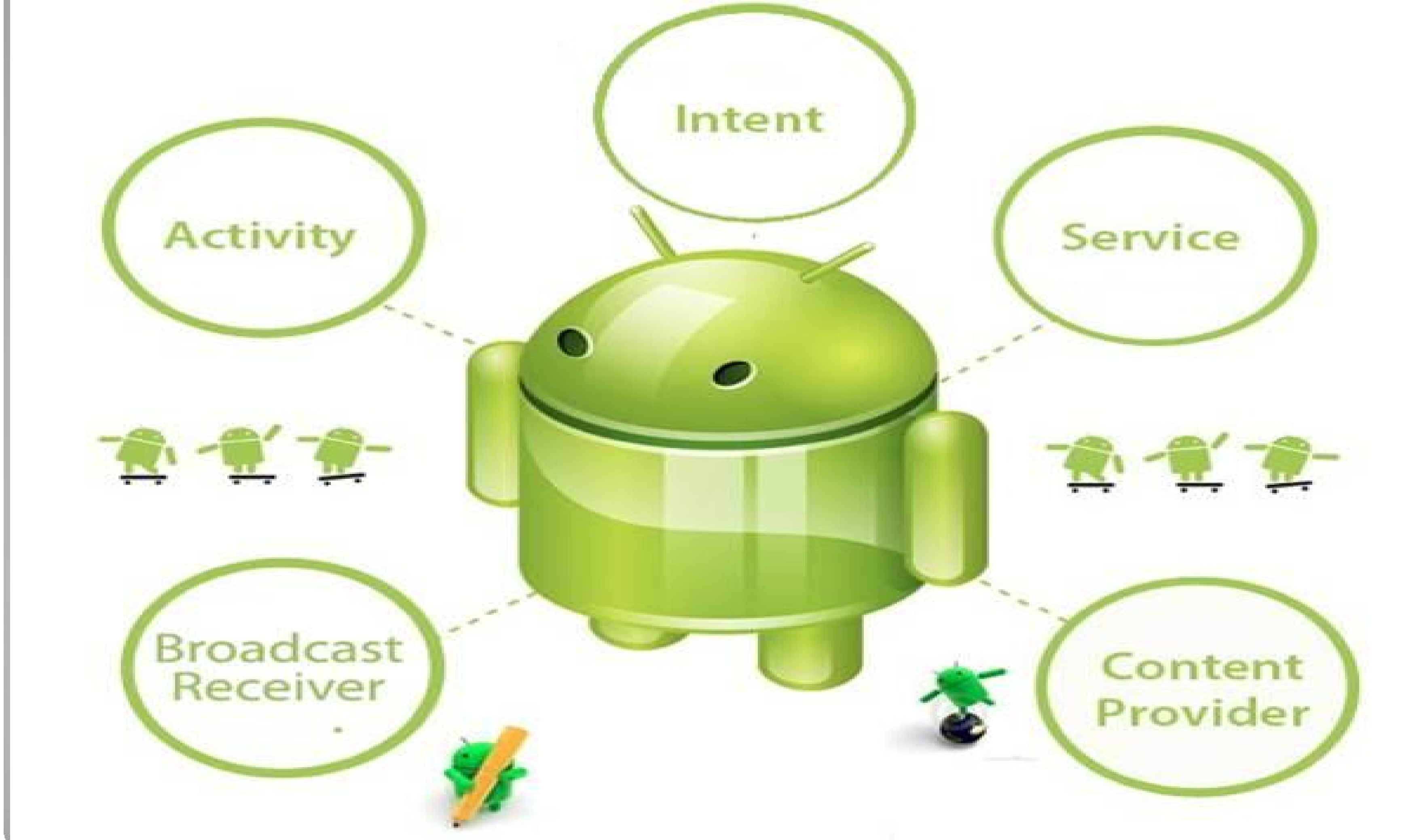
intent is used to invoke components. It is mainly is used to:

- Start the service
- Launch an activity
- Display a web page
- Display a list of contacts
- Broadcast a message
- Dial a phone call etc.

❑**Services**

Service is background process that can run for a long time.

Android core building block



- **Content provider**

Content provider are used to share data between the applications.

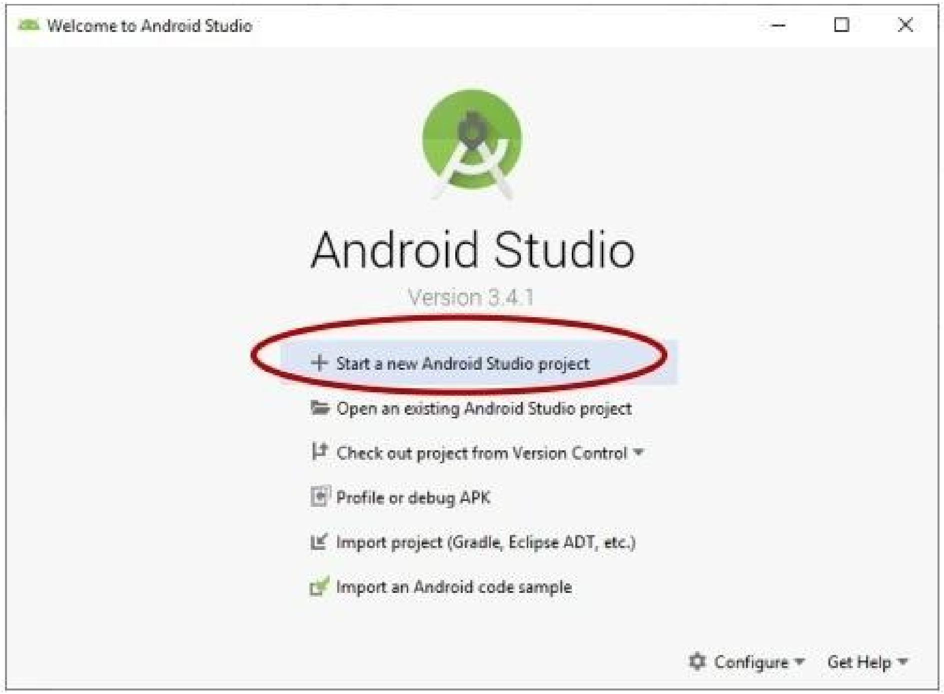
- **Fragment**

Fragment are like part of activity. An activity can display one or more fragments on the screen at the same time.

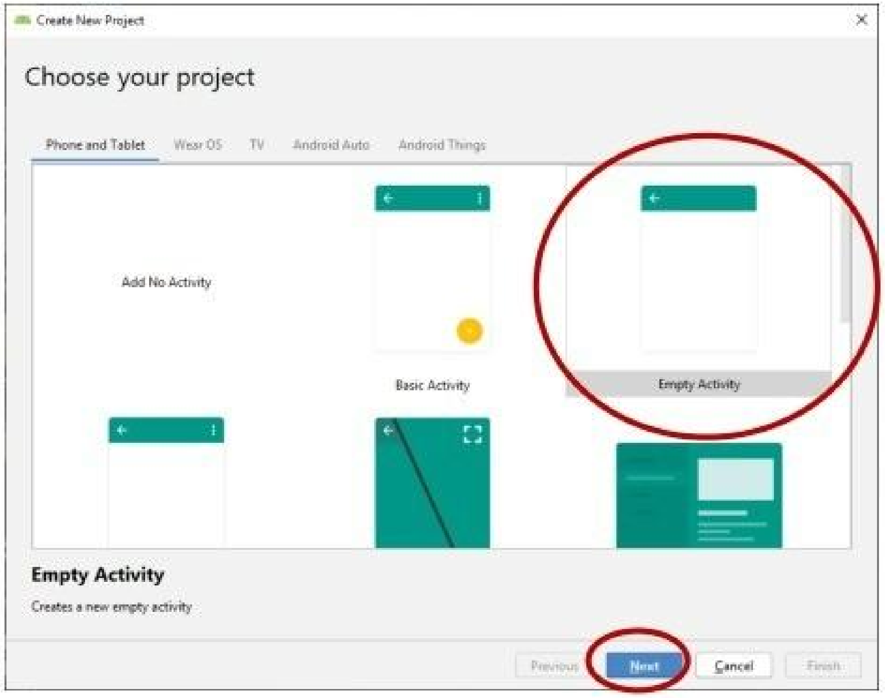
- **Androidmanifest.xml**

It contains information about activities, content provider, permissions etc.

Click on Start a new Android Studio project


Welcome to Android Studio
Android Studio
Version 3.4.1
+ Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
Profile or debug APK
Import project (Gradle, Eclipse ADT, etc.)
Import an Android code sample
Configure
Get Help

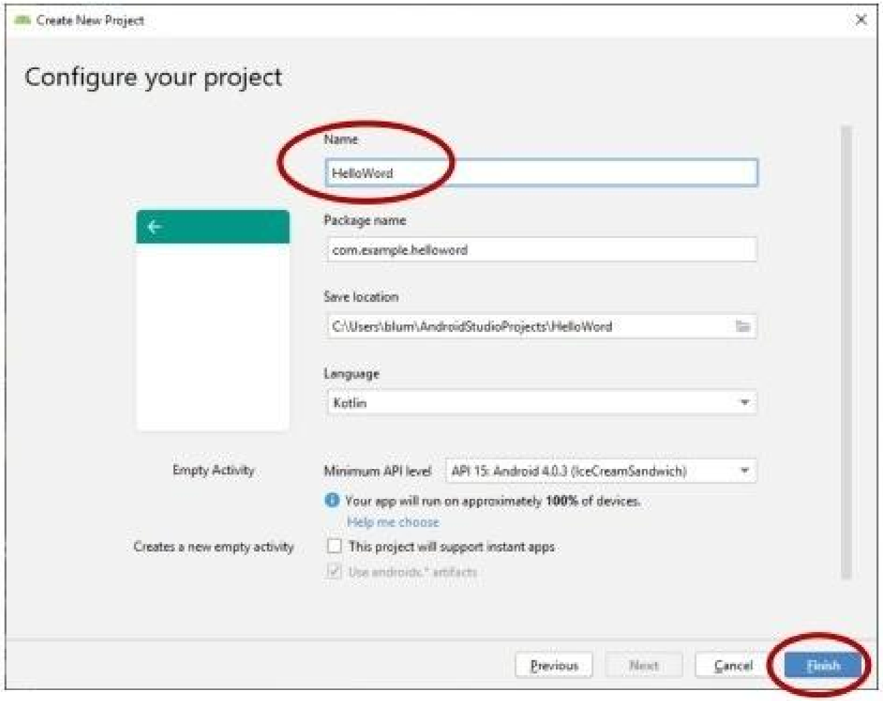
Choose an Empty Activity and click Next



Empty Activity

Creates a new empty activity

Name your App and click Finish


Create New Project

Configure your project

Name
HelloWord

Package name
com.example.helloword

Save location
C:\Users\blum\AndroidStudioProjects\HelloWord

Language
Kotlin

Empty Activity

Minimum API level API 15: Android 4.0.3 (IceCreamSandwich)

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

Creates a new empty activity

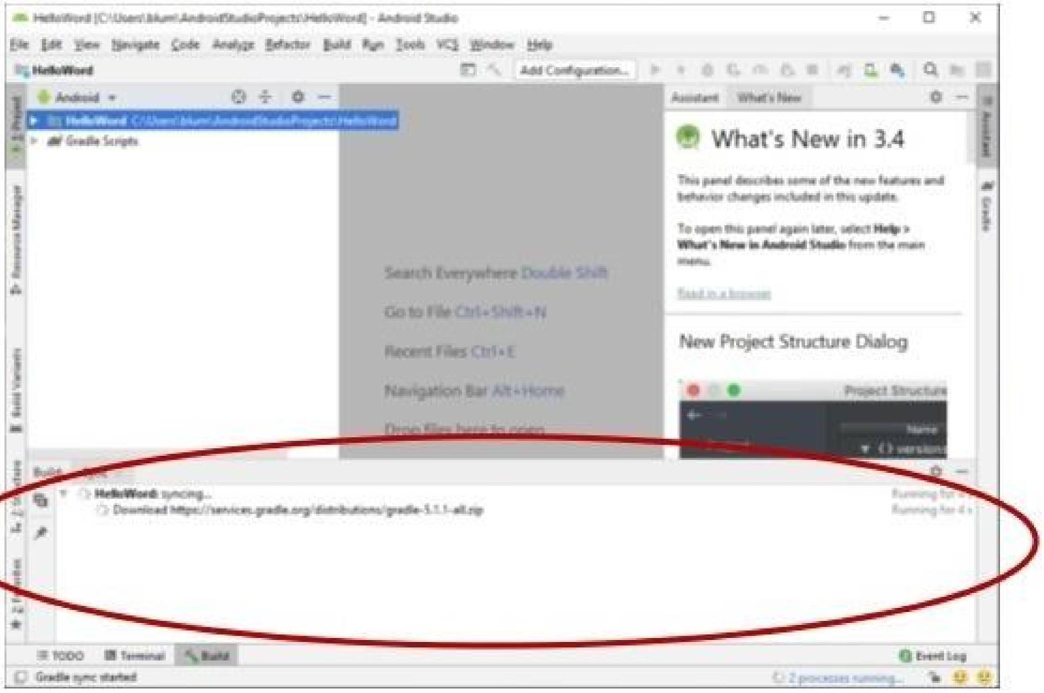
This project will support instant apps

Use androidx.* artifacts

Note that Kotlin is now the default language in Android Studio.

The other choice is Java

Wait until the Sync-ing has taken place – Sometimes it fails the first time and then re-syncs

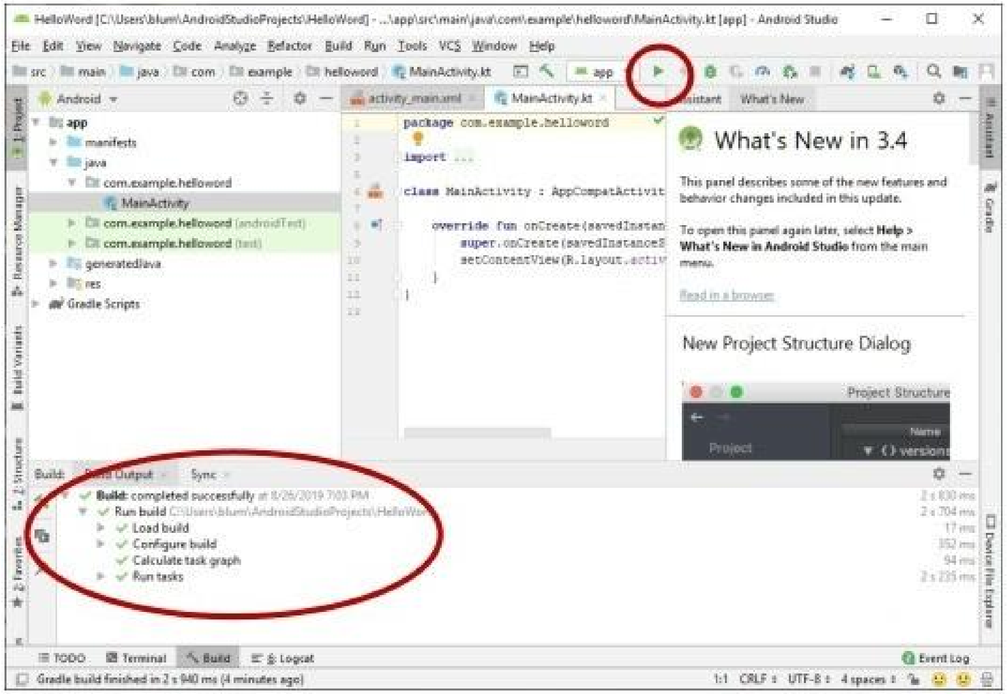


```
HelloWord: syncing...                                                   Running for 4 s
  Download https://services.gradle.org/distributions/gradle-5.1.1-all.zip   Running for 4 s
```

Gradle sync started

HelloWord [C:\Users\blum\AndroidStudioProjects\HelloWord] - Android Studio
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
HelloWord
Add Configuration...
Android
HelloWord C:\Users\blum\AndroidStudioProjects\HelloWord
Gradle Scripts
Search Everywhere Double Shift
Go to File Ctrl+Shift+N
Recent Files Ctrl+E
Navigation Bar Alt+Home
Drop files here to open
Assistant
What's New
What's New in 3.4
This panel describes some of the new features and behavior changes included in this update.
To open this panel again later, select Help > What's New in Android Studio from the main menu.
Read in a browser
New Project Structure Dialog
Project Structure
Build
TODO
Terminal
Build
Event Log
2 processes running...

When Sync area has green arrows, then click on Run arrow icon (or Run on menu)



HelloWord [C:\Users\blum\AndroidStudioProjects\HelloWord] - ...\app\src\main\java\com\example\helloword\MainActivity.kt [app] - Android Studio

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
src main java com example helloword MainActivity.kt app
Android
activity_main.xml MainActivity.kt
Assistant What's New
app
manifests
java
com.example.helloword
MainActivity
com.example.helloword (androidTest)
com.example.helloword (test)
generatedJava
res
Gradle Scripts

```
package com.example.helloword

import ...

class MainActivity : AppCompatActivit

    override fun onCreate(savedInstan
        super.onCreate(savedInstanceS
        setContentView(R.layout.activ
    }
}
```

What's New in 3.4

This panel describes some of the new features and behavior changes included in this update.

To open this panel again later, select **Help > What's New in Android Studio** from the main menu.

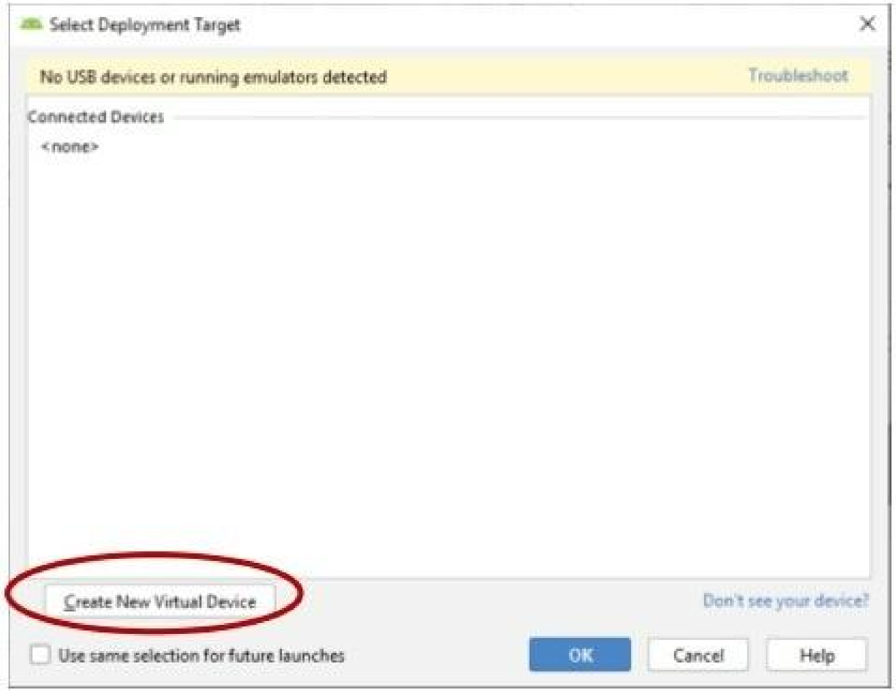
Read in a browser

New Project Structure Dialog

```
Build: completed successfully at 8/26/2019 7:03 PM          2 s 830 ms
  Run build C:\Users\blum\AndroidStudioProjects\HelloWor    2 s 704 ms
    Load build                                                  17 ms
    Configure build                                            352 ms
    Calculate task graph                                        94 ms
    Run tasks                                               2 s 235 ms
```

Gradle build finished in 2 s 940 ms (4 minutes ago)   1:1 CRLF UTF-8 4 spaces

Click on Create New Virtual Device button


Select Deployment Target

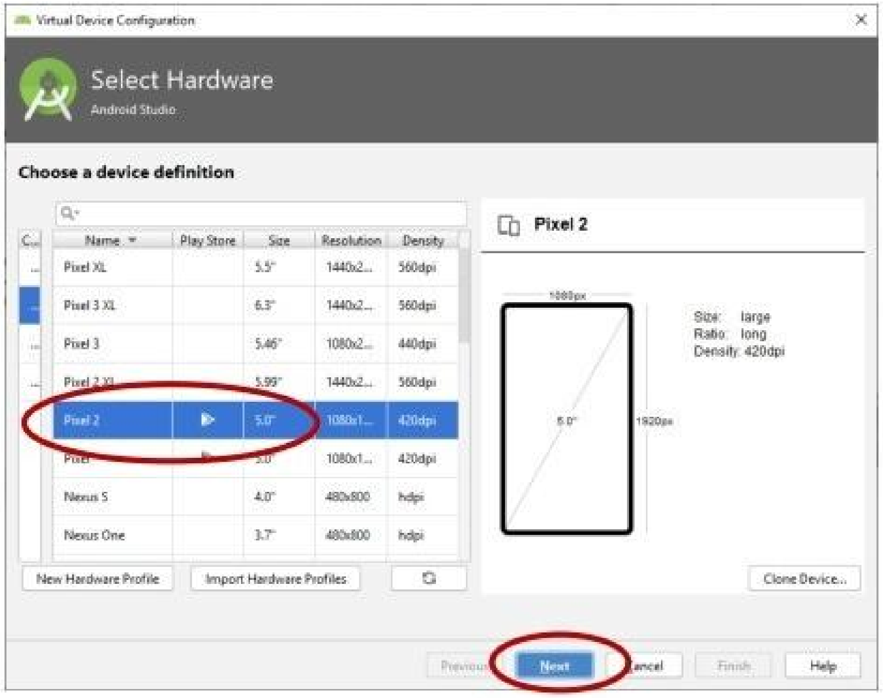
No USB devices or running emulators detected

Connected Devices

<none>

Create New Virtual Device
Don't see your device?
Use same selection for future launches
OK
Cancel
Help

I usually go with the default Pixel 2 choice and click Next


Virtual Device Configuration

Select Hardware

Android Studio

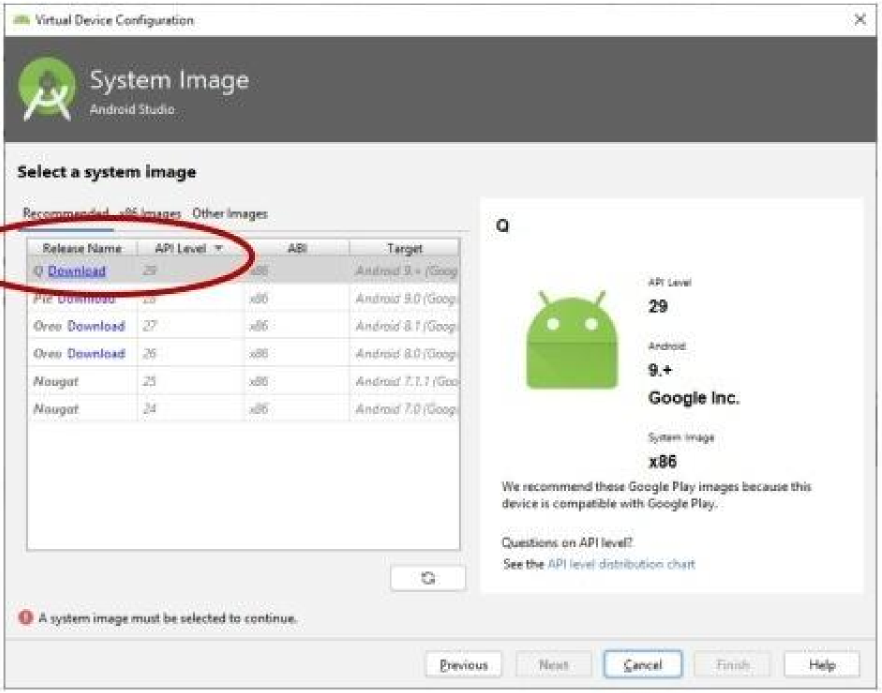
Choose a device definition

C...	Name	Play Store	Size	Resolution	Density
...	Pixel XL		5.5"	1440x2...	560dpi
...	Pixel 3 XL		6.3"	1440x2...	560dpi
...	Pixel 3		5.46"	1080x2...	440dpi
...	Pixel 2 XL		5.99"	1440x2...	560dpi
	Pixel 2	▶	5.0"	1080x1...	420dpi
	Pixel	▶	5.0"	1080x1...	420dpi
	Nexus S		4.0"	480x800	hdpi
	Nexus One		3.7"	480x800	hdpi

Pixel 2

Size: large
Ratio: long
Density: 420dpi

Choose a release Name (I go with Q) and click Download



Select a system image

Release Name	API Level	ABI	Target
Q Download	29	x86	Android 9.+ (Goog
Pie Download	28	x86	Android 9.0 (Goog
Oreo Download	27	x86	Android 8.1 (Goog
Oreo Download	26	x86	Android 8.0 (Goog
Nougat	25	x86	Android 7.1.1 (Goo
Nougat	24	x86	Android 7.0 (Goog

Q

API Level
29

Android
9.+
Google Inc.

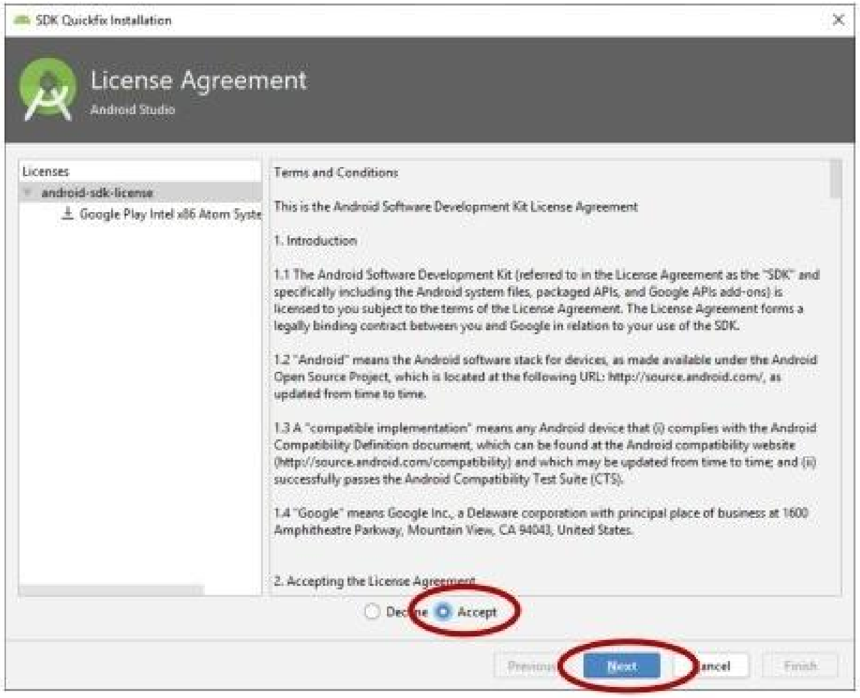
System Image
x86

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

Questions on API level?
See the API level distribution chart

A system image must be selected to continue.

Select Accept and click Next



Terms and Conditions

This is the Android Software Development Kit License Agreement

1. Introduction

1.1 The Android Software Development Kit (referred to in the License Agreement as the "SDK" and specifically including the Android system files, packaged APIs, and Google APIs add-ons) is licensed to you subject to the terms of the License Agreement. The License Agreement forms a legally binding contract between you and Google in relation to your use of the SDK.

1.2 "Android" means the Android software stack for devices, as made available under the Android Open Source Project, which is located at the following URL: http://source.android.com/, as updated from time to time.

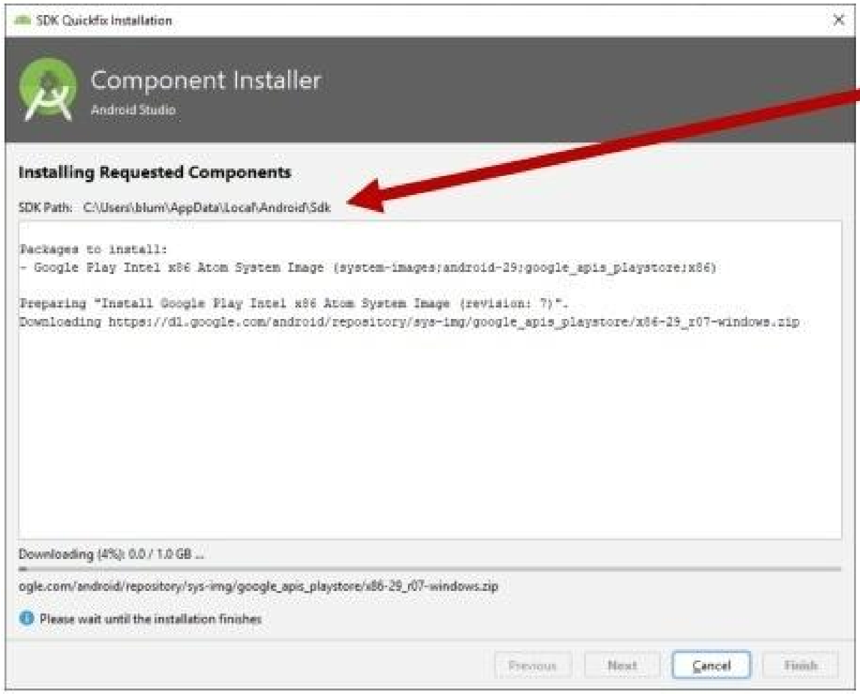
1.3 A "compatible implementation" means any Android device that (i) complies with the Android Compatibility Definition document, which can be found at the Android compatibility website (http://source.android.com/compatibility) and which may be updated from time to time; and (ii) successfully passes the Android Compatibility Test Suite (CTS).

1.4 "Google" means Google Inc., a Delaware corporation with principal place of business at 1600 Amphitheatre Parkway, Mountain View, CA 94043, United States.

2. Accepting the License Agreement

Decline [✓ Accept]

Wait for download to take place



Installing Requested Components

SDK Path: C:\Users\blum\AppData\Local\Android\Sdk

```
Packages to install:
- Google Play Intel x86 Atom System Image (system-images;android-29;google_apis_playstore;x86)

Preparing "Install Google Play Intel x86 Atom System Image (revision: 7)".
Downloading https://dl.google.com/android/repository/sys-img/google_apis_playstore/x86-29_r07-windows.zip
```

Downloading (4%): 0.0 / 1.0 GB ...

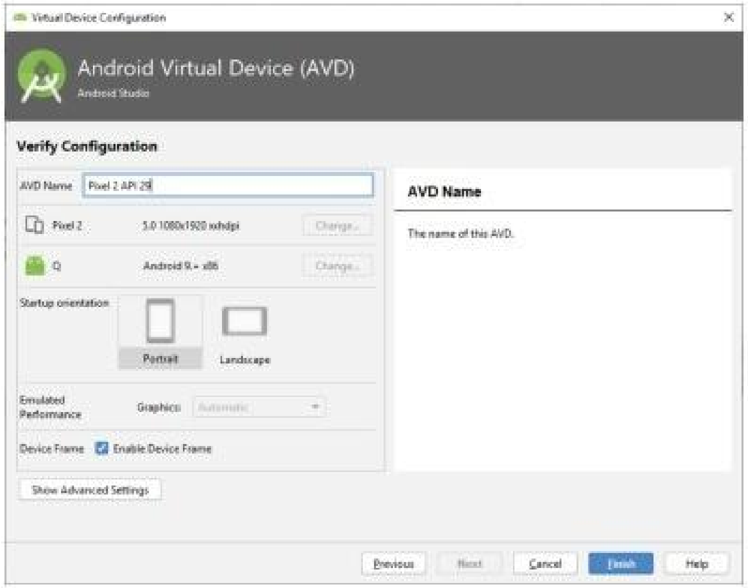
ogle.com/android/repository/sys-img/google_apis_playstore/x86-29_r07-windows.zip

Please wait until the installation finishes

It tells you where the SDK is in case you ever need to know.

AppData is typically a hidden location

Click Finish


Virtual Device Configuration

Android Virtual Device (AVD)
Android Studio

Verify Configuration

AVD Name: Pixel 2 API 29

Pixel 2 — 5.0 1080x1920 xxhdpi

Q — Android 9.+ x86

Startup orientation: Portrait / Landscape

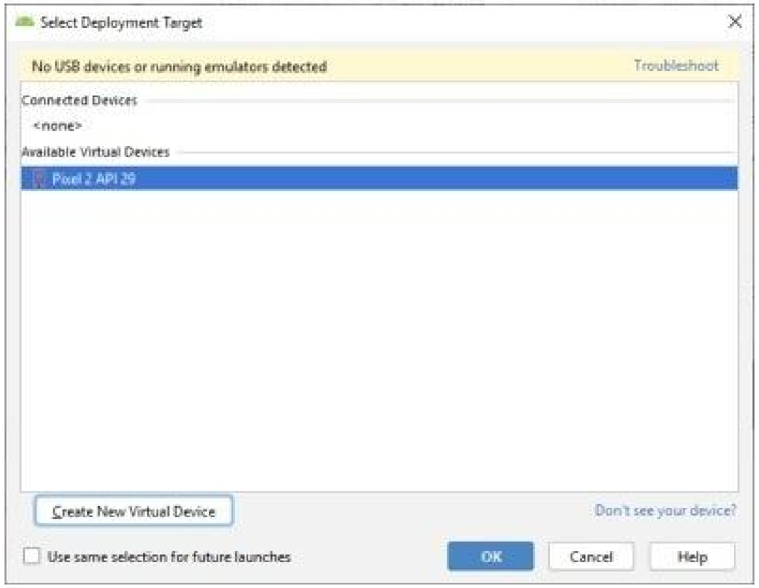
Emulated Performance — Graphics: Automatic

Device Frame: ☑ Enable Device Frame

AVD Name

The name of this AVD.

Click OK


Select Deployment Target

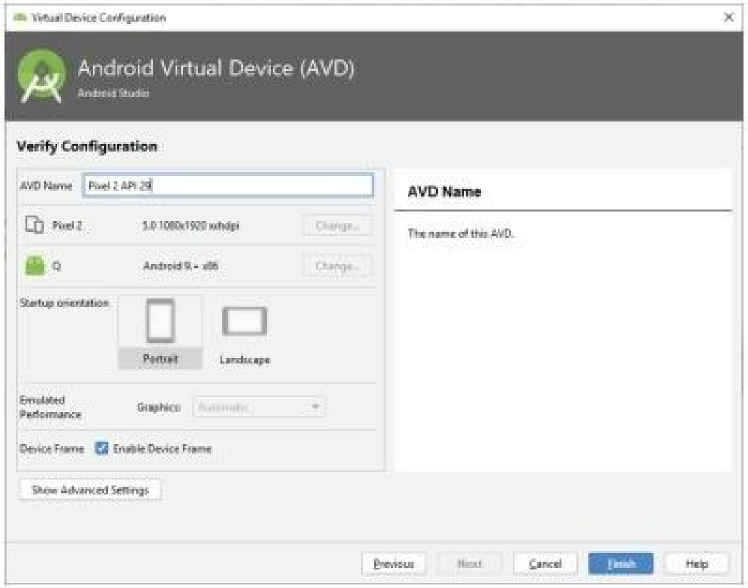
No USB devices or running emulators detected

Connected Devices
<none>

Available Virtual Devices
Pixel 2 API 29

☐ Use same selection for future launches

Click Finish



Virtual Device Configuration

Android Virtual Device (AVD)
Android Studio

Verify Configuration

AVD Name: Pixel 2 API 29

Pixel 2 — 5.0 1080x1920 xxhdpi

Q — Android 9.+ x86

Startup orientation: Portrait

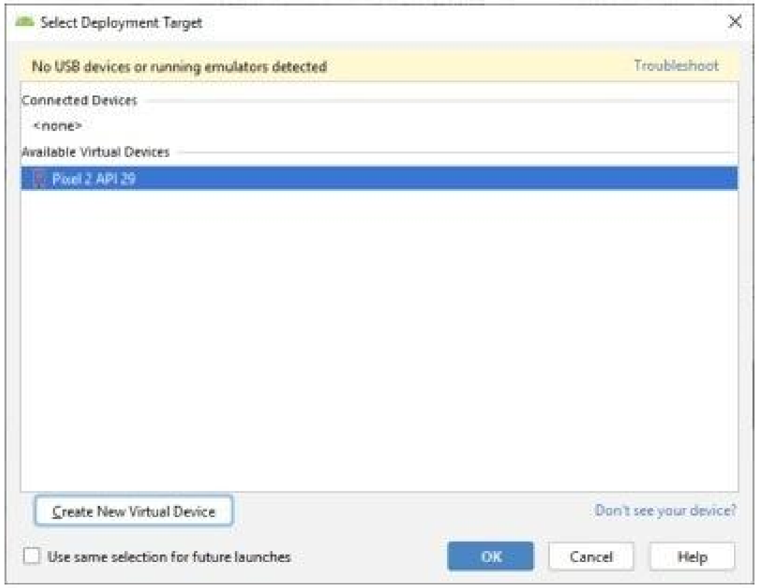
Emulated Performance — Graphics: Automatic

Device Frame: ☑ Enable Device Frame

AVD Name

The name of this AVD.

Click OK



Select Deployment Target

No USB devices or running emulators detected

Connected Devices

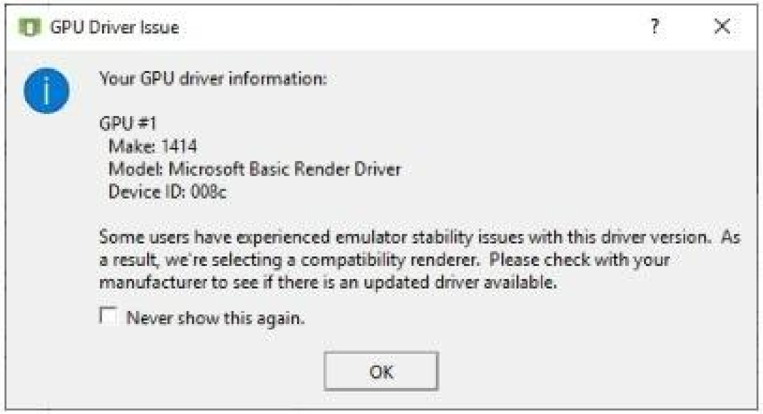
<none>

Available Virtual Devices

Pixel 2 API 29

☐ Use same selection for future launches

Emulating can take a little while.
And you might receive a message about the GPU driver.



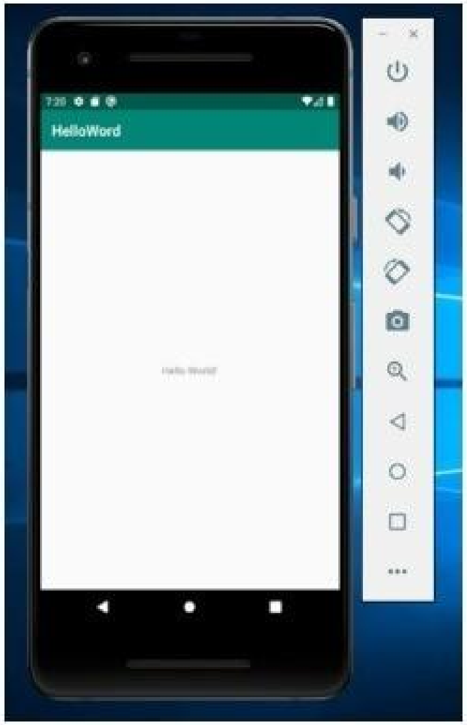
Your GPU driver information:

GPU #1
  Make: 1414
  Model: Microsoft Basic Render Driver
  Device ID: 008c

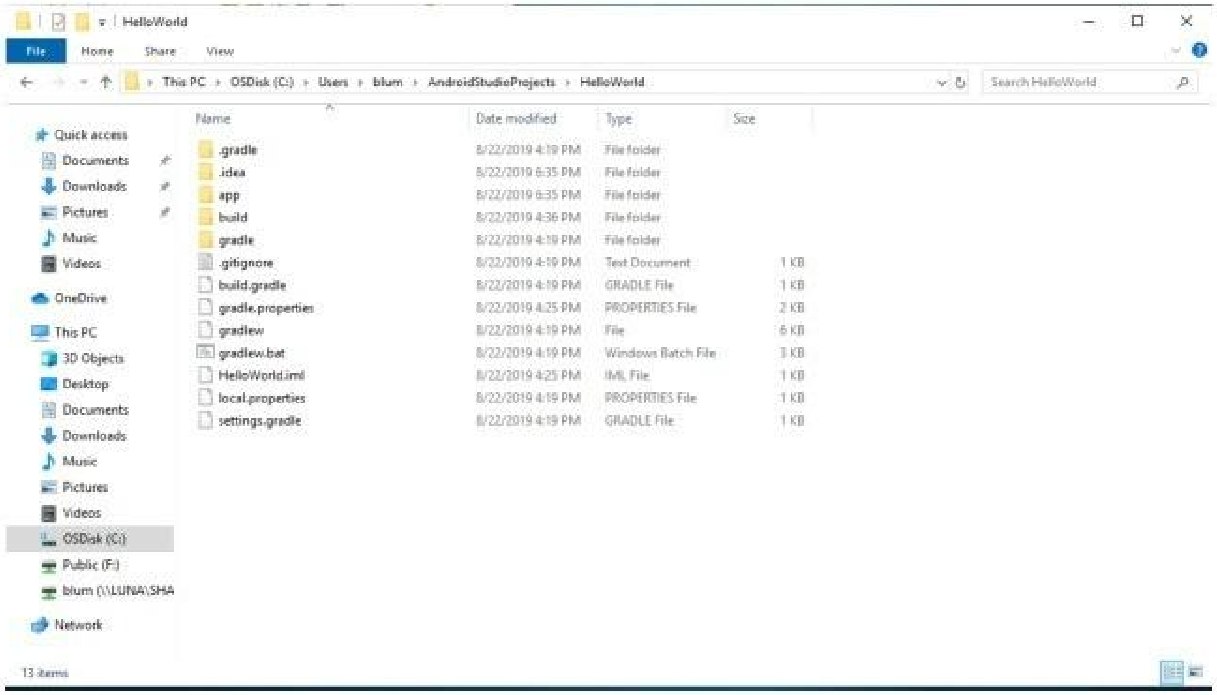
Some users have experienced emulator stability issues with this driver version. As a result, we're selecting a compatibility renderer. Please check with your manufacturer to see if there is an updated driver available.

☐ Never show this again.

After a couple of other screens, your Hello World should appear.



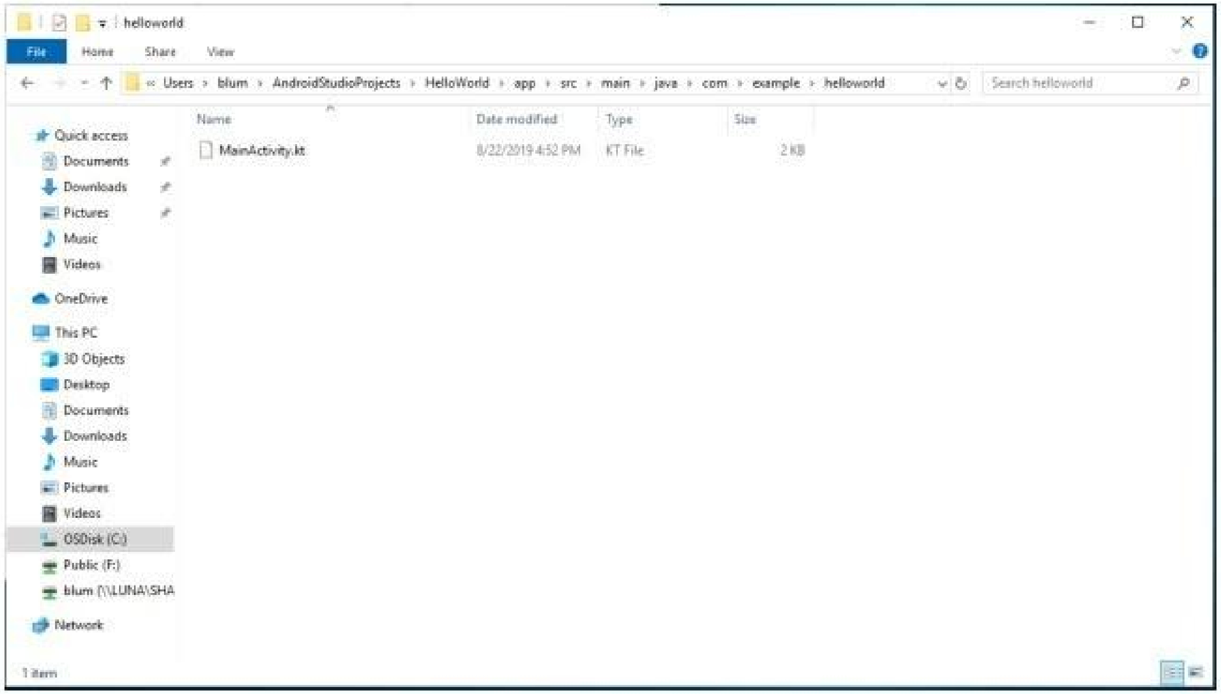
(Default) Location of Android Project



This PC › OSDisk (C:) › Users › blum › AndroidStudioProjects › HelloWorld

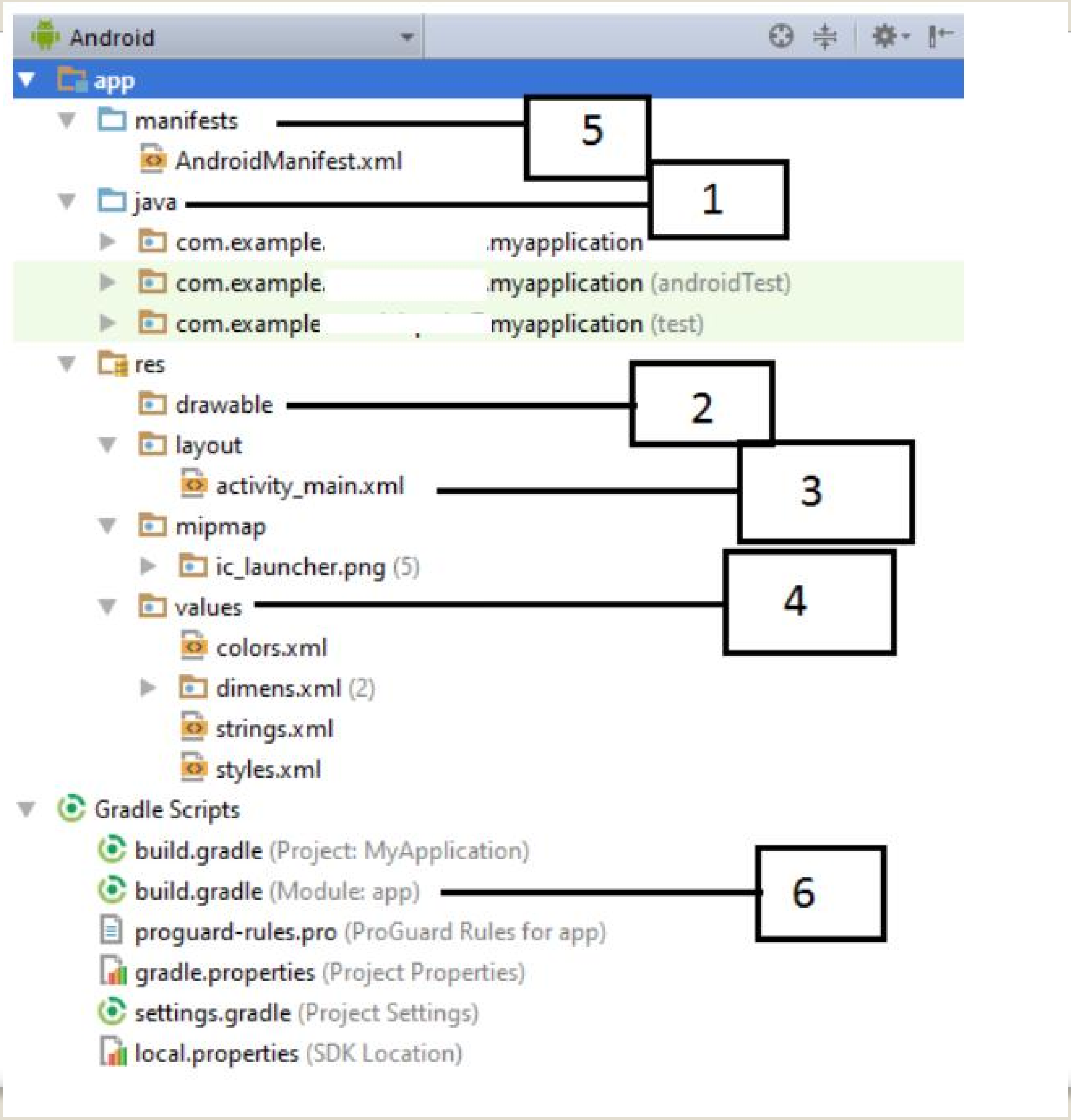
Name	Date modified	Type	Size
.gradle	8/22/2019 4:19 PM	File folder	
.idea	8/22/2019 6:35 PM	File folder	
app	8/22/2019 6:35 PM	File folder	
build	8/22/2019 4:36 PM	File folder	
gradle	8/22/2019 4:19 PM	File folder	
.gitignore	8/22/2019 4:19 PM	Text Document	1 KB
build.gradle	8/22/2019 4:19 PM	GRADLE File	1 KB
gradle.properties	8/22/2019 4:25 PM	PROPERTIES File	2 KB
gradlew	8/22/2019 4:19 PM	File	6 KB
gradlew.bat	8/22/2019 4:19 PM	Windows Batch File	3 KB
HelloWorld.iml	8/22/2019 4:25 PM	IML File	1 KB
local.properties	8/22/2019 4:19 PM	PROPERTIES File	1 KB
settings.gradle	8/22/2019 4:19 PM	GRADLE File	1 KB

Location of the Kotlin file



« Users › blum › AndroidStudioProjects › HelloWorld › app › src › main › java › com › example › helloworld

Name	Date modified	Type	Size
MainActivity.kt	8/22/2019 4:52 PM	KT File	2 KB



Android
app
manifests
AndroidManifest.xml
java
com.example. .myapplication
com.example. .myapplication (androidTest)
com.example myapplication (test)
res
drawable
layout
activity_main.xml
mipmap
ic_launcher.png (5)
values
colors.xml
dimens.xml (2)
strings.xml
styles.xml
Gradle Scripts
build.gradle (Project: MyApplication)
build.gradle (Module: app)
proguard-rules.pro (ProGuard Rules for app)
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)
5
1
2
3
4
6

1	**Java** This contains the **.java** source files for your project. By default, it includes an *MainActivity.java* source file having an activity class that runs when your app is launched using the app icon.
2	**res/drawable-hdpi** This is a directory for drawable objects that are designed for high-density screens.
3	**res/layout** This is a directory for files that define your app's user interface.
4	**res/values** This is a directory for other various XML files that contain a collection of resources, such as strings and colours definitions.
5	**AndroidManifest.xml** This is the manifest file which describes the fundamental characteristics of the app and defines each of its components.
6	**Build.gradle** This is an auto generated file which contains compileSdkVersion, buildToolsVersion, applicationId, minSdkVersion, targetSdkVersion, versionCode and versionName

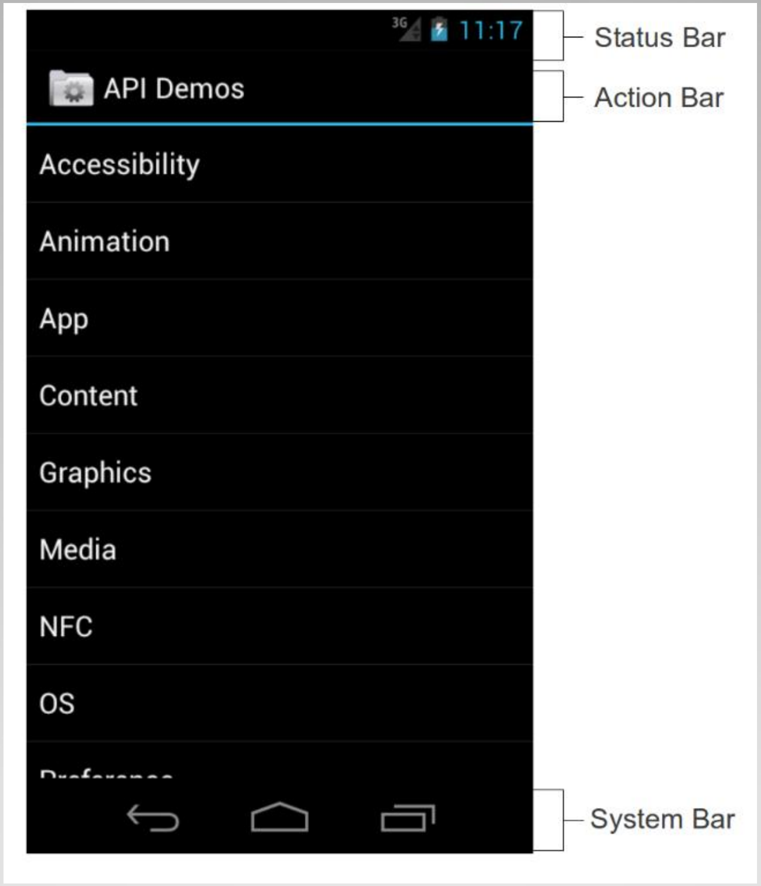
Steps in project

- Step 1: Create a new project
- Step 2: Modify the strings.xml file
- Step 3: Modify the activity_main.xml file
- Step 4: Accessing the components in the MainActivity file
-

- Bundle - Android Bundle is **used to pass data between activities**. The values that are to be passed are mapped to String keys which are later used in the next activity to retrieve the values.

MainActivity Explained – Nav Controller

- AppCompat is **a set of support libraries which can be used to make the apps developed with newer versions work with older versions. Backward compatibility.**
- **AppCompatActivity is the base class for activities with the support library action bar features. ActionBar can be added to your activity by extending this class for your activity and setting the activity theme to Theme when running on API level 7 or higher.**


11:17
API Demos
Accessibility
Animation
App
Content
Graphics
Media
NFC
OS
Status Bar
Action Bar
System Bar

Android UI widgets

There are many UI widgets with simple example such as button edittext, autocompletetextview, togglebutton, datepiker, timepicker, progressbar etc.

- **Working with buttons**

  Learn how to perform event handling on button click.
- **Android toast**

  Displays information for the short duration of time.
- **Custom toast**

  we can display the image on toast.
- **Check box**

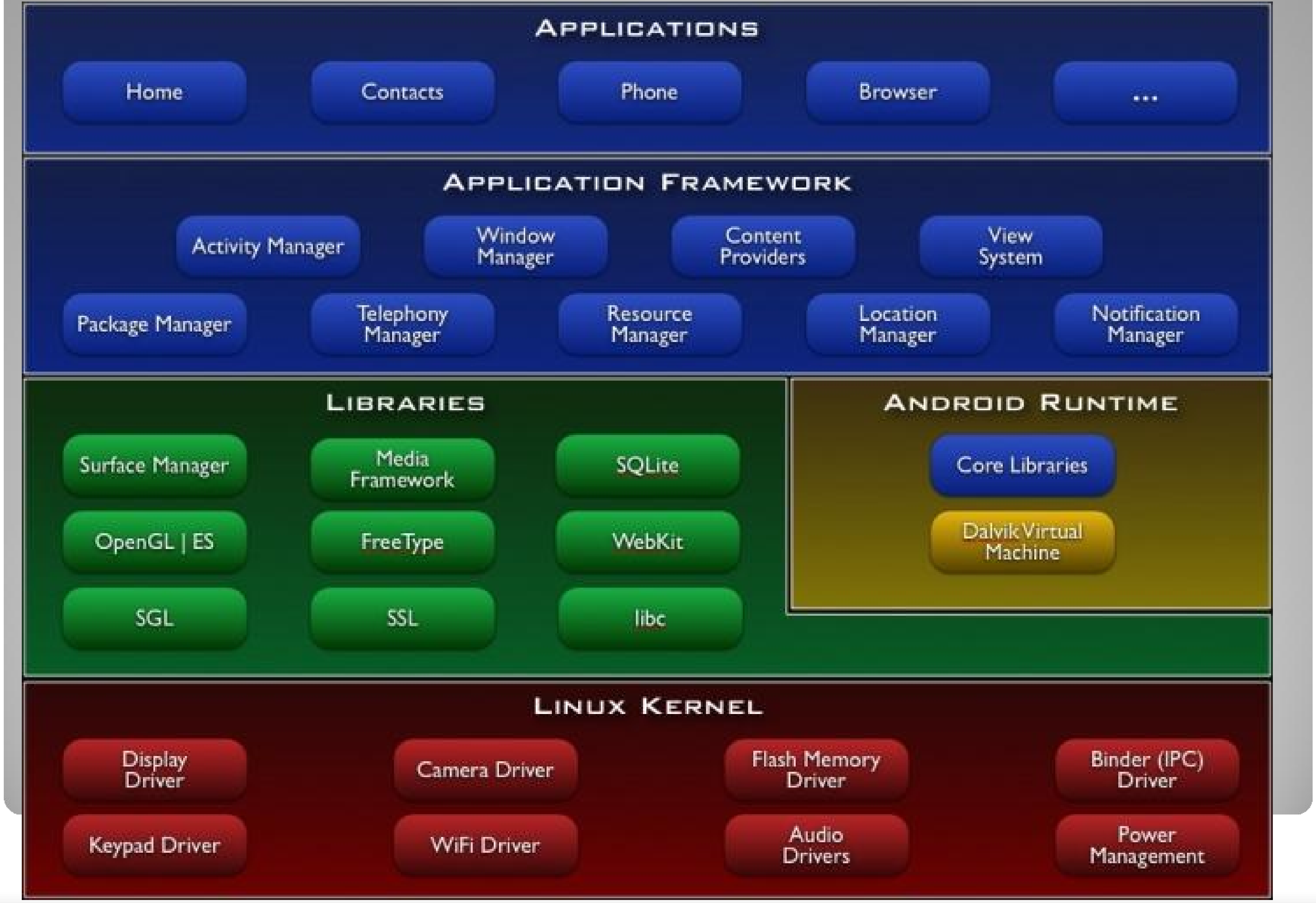
  application of simple food ordering.
- **Toggle button**

  it has two states on/off.

Applications

- Written in Kotlin (it’s possible to write native code)
- Good separation (and corresponding security) from other applications:
  - Each application runs in its own process
  - Each process has its own separate VM
  - Each application is assigned a unique Linux user ID – by default files of that application are only visible to that application

Android Architecture



- Activity Manager – Life Cycle / Navigation within and among applications
- Content Provider – encapsulate data
- Location Manager – aware of its physical location
- Notification Manager – users informed about events
- Package Manager – infn. about other appln. Pkgs
- Resource Manager – lets appln accesses its resources

- Telephony Manager – to learn about device telephony services
- View System – manages UI elements / events
- Window Manager – perform Window related operations

Activities

- Basic component of most applications
- Most applications have several activities that start each other as needed
- Each is implemented as a subclass of the base Activity class

Activities – The View

- Each activity has a default window to draw in
- The content of the window is a view or a group of views (derived from View or ViewGroup)
- Example of views: buttons, text fields, scroll bars, menu items, check boxes, etc.
- View(Group) made visible via Activity.setContentView() method.

Services

- Does not have a visual interface
- Runs in the background indefinitely
- Examples
  - Network Downloads
  - Playing Music
  - TCP/UDP Server
- You can bind to an existing service and control its operation

Broadcast Receivers

- Receive and react to broadcast announcements
- Extend the class BroadcastReceiver
- Examples of broadcasts:
  - Low battery, power connected, shutdown, timezone changed, etc.
  - Other applications can initiate broadcasts

Content Providers

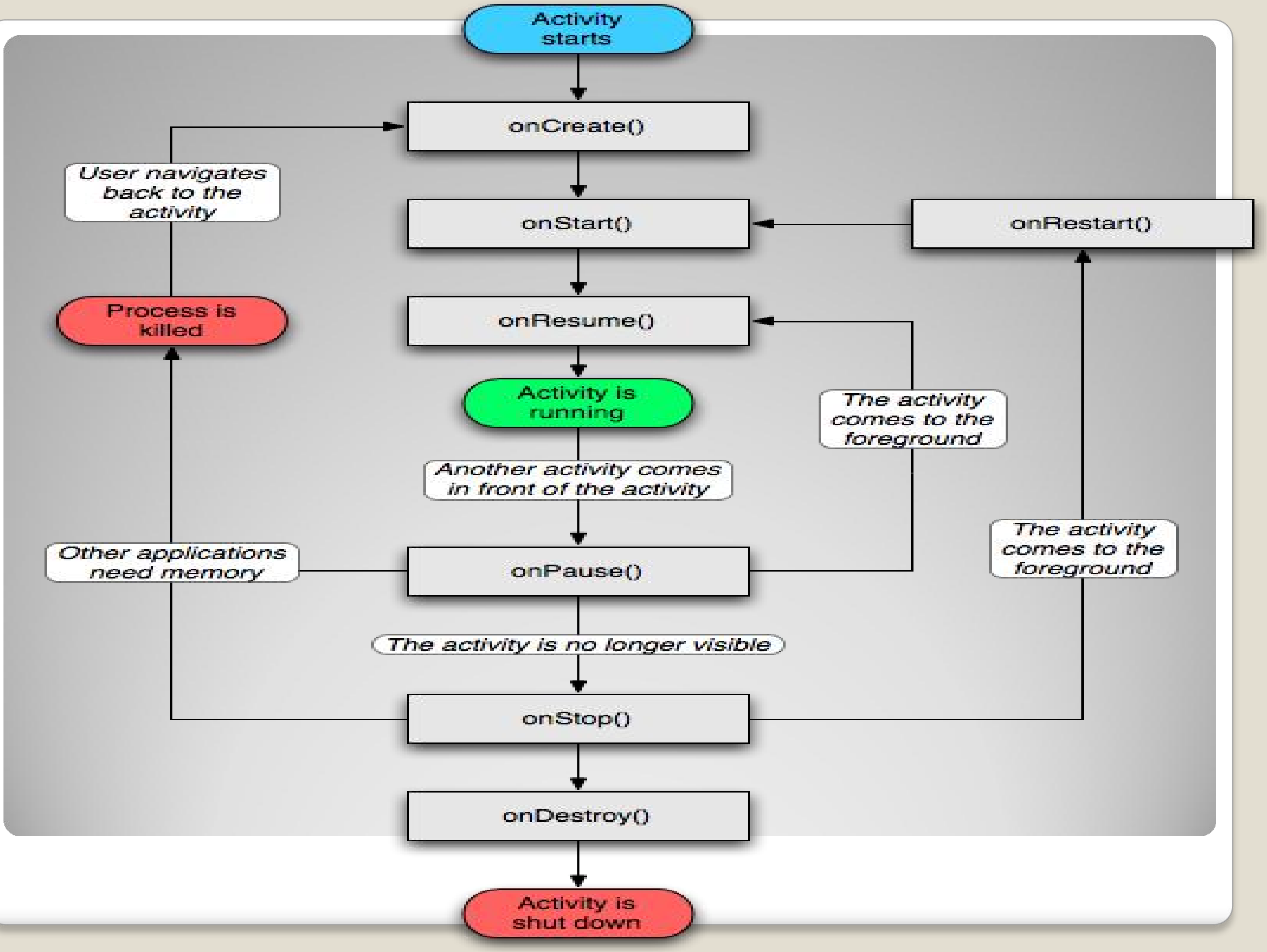
- Makes some of the application data available to other applications
- It’s the only way to transfer data between applications in Android (no shared files, shared memory, pipes, etc.)
- Extends the class ContentProvider;
- Other applications use a ContentResolver object to access the data provided via a ContentProvider

Shutting down components

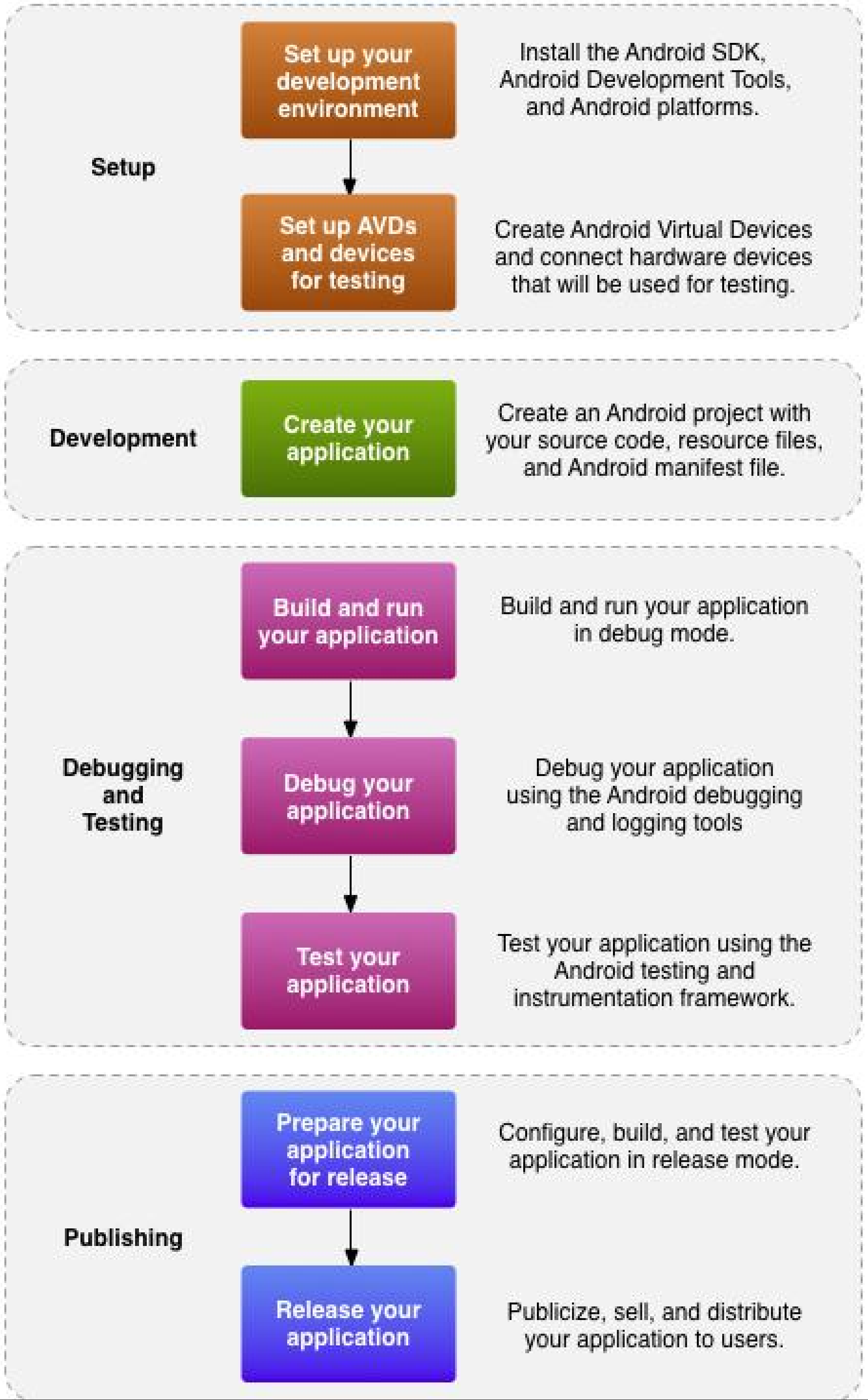
- Activities
  - Can terminate itself via finish();
  - Can terminate other activities it started via finishActivity();
- Services
  - Can terminate via stopSelf(); or Context.stopService();
- Content Providers
  - Are only active when responding to ContentResolvers
- Broadcast Receivers
  - Are only active when responding to broadcasts

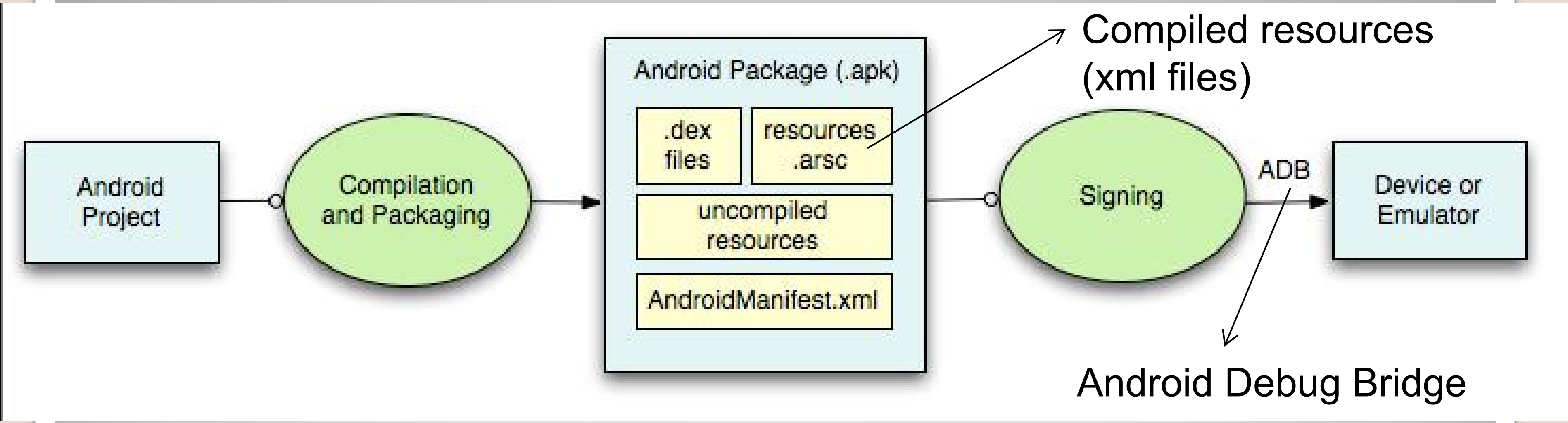
Activity

OnCreate – when activity is first created
OnStart – when it becomes visible
OnRestart – calls start
OnResume – just before the activity starts interacting
OnPause – when moving to another activity
OnStop – if no longer visible
OnDestroy – before it is destroyed


Activity starts
onCreate()
User navigates back to the activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
The activity comes to the foreground
Other applications need memory
onPause()
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down

Development process for an Android app


Setup
Set up your development environment
Install the Android SDK, Android Development Tools, and Android platforms.
Set up AVDs and devices for testing
Create Android Virtual Devices and connect hardware devices that will be used for testing.
Development
Create your application
Create an Android project with your source code, resource files, and Android manifest file.
Debugging and Testing
Build and run your application
Build and run your application in debug mode.
Debug your application
Debug your application using the Android debugging and logging tools
Test your application
Test your application using the Android testing and instrumentation framework.
Publishing
Prepare your application for release
Configure, build, and test your application in release mode.
Release your application
Publicize, sell, and distribute your application to users.



Building and running

- ADB is a client server program that connects clients on developer machine to devices/emulators to facilitate development.
- An IDE like Andriod Studio handles this entire process for you.

Building and running

LinearLayout - displays View-elements as a single row (if it is Horizontal) or a single column (if it is Vertical).

TableLayout - displays elements in the form of a table, with rows and columns.

RelativeLayout - each element's position is configured relatively to other elements.

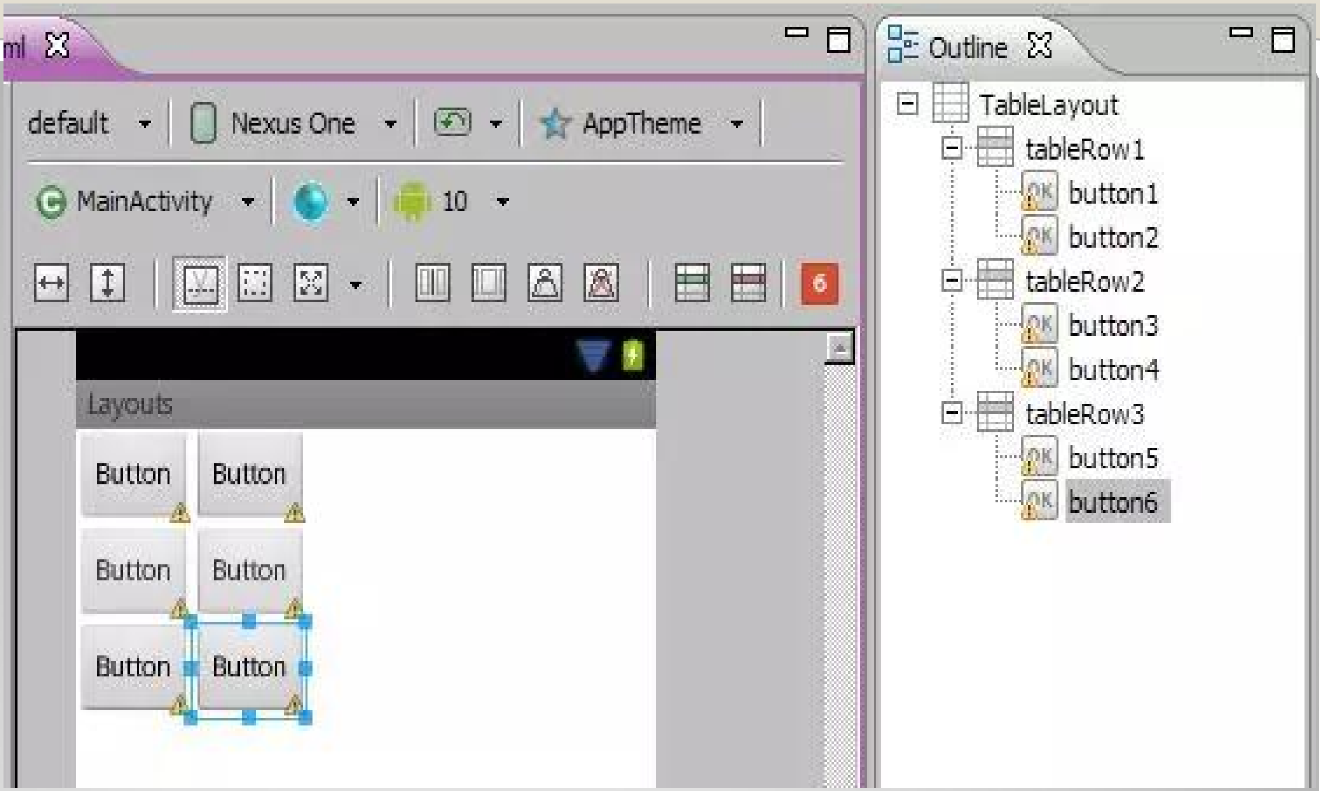
AbsoluteLayout - each element is specified an absolute position on the screen in the coordinate system (x, y)

ConstraintLayout - provides you the ability to completely design your UI with the drag and drop feature provided by the Android Studio design editor.

LinearLayout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/ap
k/res/android"
  xmlns:tools="http://schemas.android.com/tool
s"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical">
</LinearLayout>
```

TableLayout



```
TableLayout
  tableRow1
    button1
    button2
  tableRow2
    button3
    button4
  tableRow3
    button5
    button6
```

```
<TableRow
    android:layout_height="wrap_content"
    android:layout_width="fill_parent"
    android:gravity="center_horizontal">
    <TextView
      android:layout_width="match_parent"  android:layout_height="wrap_content"
      android:textSize="18dp"  android:text="Row 1"   android:layout_span="3"
      android:padding="18dip"  android:background="#b0b0b0"
      android:textColor="#000"/>
  </TableRow>
```

RelativeLayout

```
<TextView
  android:id="@+id/label"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:text="Type here:">
</TextView>
<EditText
  android:id="@+id/entry"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:layout_below="@+id/label"
  android:background="@android:drawable/editbox_background">
</EditText>
<Button
  android:id="@+id/ok"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_alignParentRight="true"
  android:layout_below="@+id/entry"
  android:layout_marginLeft="10dip"
  android:text="OK">
</Button>              (Density Independent pixels)
```

```
<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="42dp"
    android:layout_y="62dp"
    android:text="Button">
  </Button>
  <TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="142dp"
    android:layout_y="131dp"
    android:text="TextView">
  </TextView>
  <CheckBox
    android:id="@+id/checkBox1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="55dp"
    android:layout_y="212dp"
    android:text="CheckBox">  </CheckBox>
```

AbsoluteLayout

- **Alert dialog**

  Alert dialog containing the message with ok and cancel buttons.
- **Spinner**

  display the multiple options but only selected at a time.
- **Ratingbar**

  Ratingbar display rating bar.
- **Datepicker**

  datepicker displays datepicker dialog that can be used to pick the date.
- **Timepicker**

  it can be used to pick the time.

Android fragments

- A fragment is an independent component which can be used by an activity. Fragments represents multiple screen inside one activity.
- A fragment runs in the context of an activity, but has its own life cycle and typically its own user interface. It is also possible to define fragments without an user interface, i.e., headless fragments.

Advantages of using fragments

- Fragments make it easy to reuse components in different layouts, e.g., you can build single-pane layouts for handsets (phones) and multi-pane layouts for tablets.
- This is not limited to tablets; for example, you can use fragments also to support different layout for landscape and portrait orientation on a smartphone.