Android Introduction

Android is a software package and linux based operating system for mobile devices such as tablet computers and smartphones.
It is developed by Google and later the OHA (Open Handset Alliance). Java language is mainly used to write the android code even though other languages can be used.
The goal of android project is to create a successful real-world product that improves the mobile experience for end users.
There are many code names of android such as Lollipop, Kitkat, Jelly Bean, Ice cream Sandwich, Froyo, Ecliar, Donut etc which is covered in next page.

What is **Open Handset Alliance (OHA)**
It's a consortium of 84 companies such as google, samsung, AKM, synaptics, KDDI, Garmin, Teleca, Ebay, Intel etc.
It was established on 5th November, 2007, led by Google. It is committed to advance open standards, provide services and deploy handsets using the Android Plateform.

Features of Android
After learning what is android, let's see the features of android. The important features of android are given below:
1) It is open-source.
2) Anyone can customize the Android Platform.
3) There are a lot of mobile applications that can be chosen by the consumer.
4) It provides many interesting features like weather details, opening screen, live RSS (Really Simple Syndication) feeds etc.
It provides support for messaging services(SMS and MMS), web browser, storage (SQLite), connectivity (GSM, CDMA, Blue Tooth, Wi-Fi etc.), media, handset layout etc.

Categories of Android applications
There are many android applications in the market. The top categories are:

- Entertainment
- Tools
- Communication
- Productivity

•Personalization

•Music and Audio

•Social

•Media and Video

•Travel and Local etc.

Clearly there's a demand for Android app development, and it's turning the platform with the lovable green mascot into more and more of a strong first choice rather than just a secondary option to iOS.

With over *one billion* devices activated, Android is an exciting space to make apps to help you communicate, organize, educate, entertain or anything else you're passionate about.

If that's not enough, here are a few more reasons to learn Android:

•You'll be plugged into the open source platform with (at the time of press) the *largest market share* of smart devices worldwide.

•Android's policies on device provisioning and app submission are more open than Apple's, meaning that once you complete your first app—as you'll do in this tutorial—you and your friends can enjoy it on your devices right away!

•If you have experience developing for iOS, you can become well-versed in the ways that the two platforms coincide and differ (we'll discuss a few in this tutorial) and what you like about each. Then you'll have more tools at your disposal for your next mobile project.

•It's not just the iPhone anymore. There are so many smartphones, tablets, glasses, and watches out there, coming from so many manufacturers, and they're all trying to jump into the game. You don't have to be any sort of market analyst to know that there are a few important platforms and Android is one of them.

Android architecture

android architecture or Android software stack is categorized into five parts:

1.linux kernel

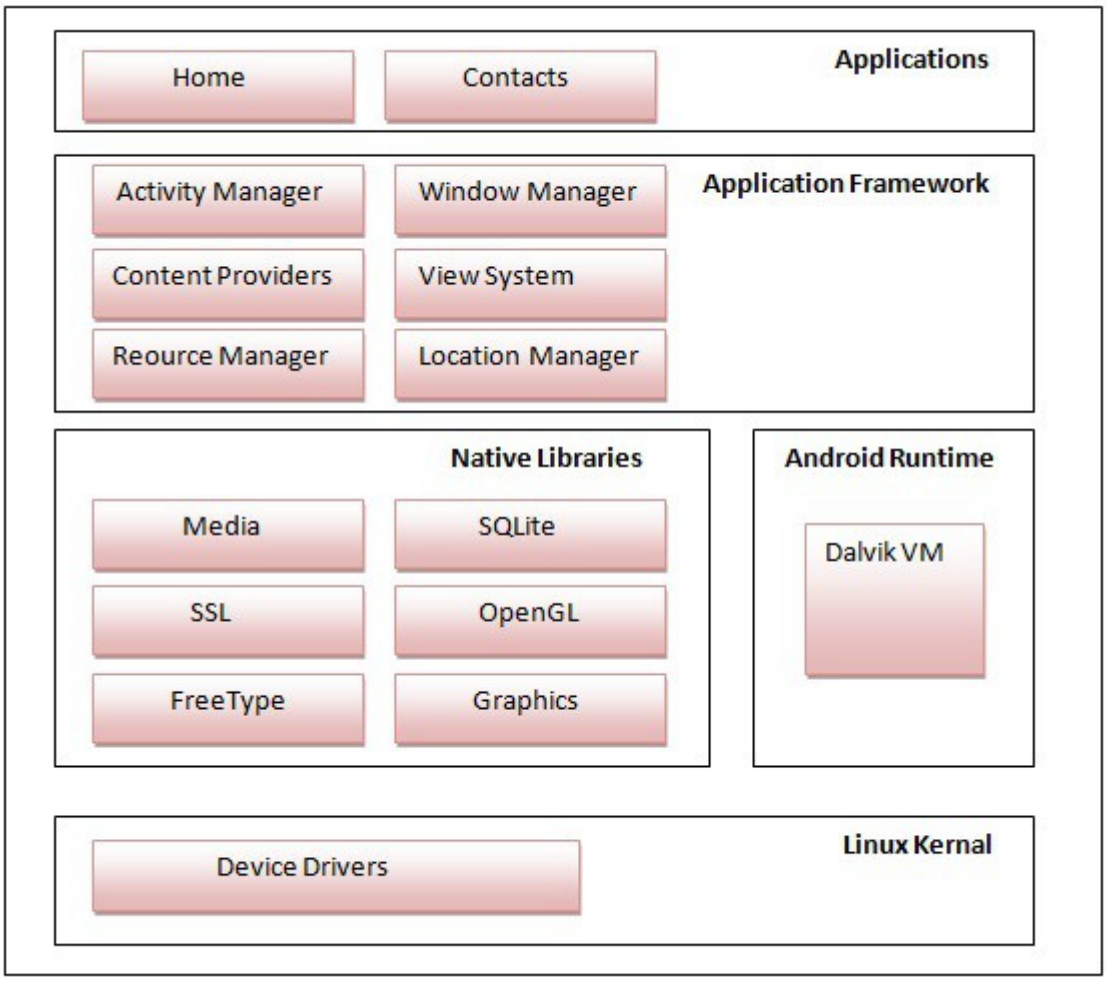
2.native libraries (middleware),

3.Android Runtime

4.Application Framework

5.Applications

Let's see the android architecture first.



1) Linux kernel

It is the heart of android architecture that exists at the root of android architecture. **Linux kernel** is responsible for device drivers, power management, memory management, device management and resource access.

2) Native Libraries

On the top of linux kernel, their are **Native libraries** such as WebKit, OpenGL, FreeType, SQLite, Media, C runtime library (libc) etc.

The WebKit library is responsible for browser support, SQLite is for database, FreeType for font support, Media for playing and recording audio and video formats.

3) Android Runtime

In android runtime, there are core libraries and DVM (Dalvik Virtual Machine) which is responsible to run android application. DVM is like JVM but it is optimized for mobile devices. It consumes less memory and provides fast performance.

4) Android Framework

On the top of Native libraries and android runtime, there is android framework. Android framework includes **Android API's** such as UI (User Interface), telephony, resources, locations, Content Providers (data) and package managers. It provides a lot of classes and interfaces for android application development.

5) Applications

On the top of android framework, there are applications. All applications such as home, contact, settings, games, browsers are using android framework that uses android runtime and libraries. Android runtime and native libraries are using linux kernal.

Getting Started

How does Android development go down? First, the zoomed-out basics:

- You'll write your programming—what you want your app to do—in Java files and design your layouts—how you want your app to look—in XML files.

- Once your app is ready, you'll use a build tool to compile all the project files and package them together into a **.apk** file that you can run on Android devices and/or submit to Google Play.

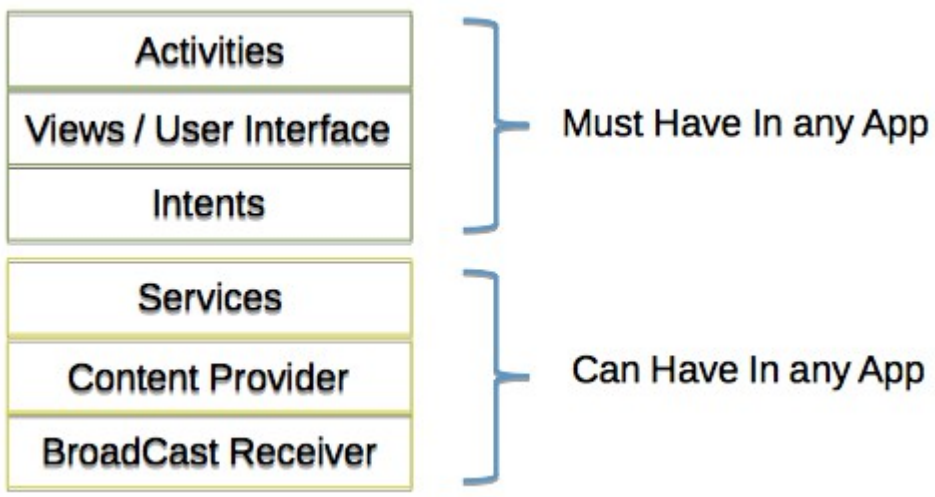
- All of the files you used to put your app together are managed by an Integrated Development Environment (IDE). The IDE is the program you will open to edit your code files and to manage your projects.
- The standard IDE for Android used to be Eclipse, but this is now being replaced by Google's ownAndroid Studio.

If you zoom in (metaphorically), you'll find more in-depth processes going on behind the scenes during all of the above steps. For example, advanced users will want to investigate the role of the Dalvik Virtual Machine and it's new replacement, ART.

Though a lot has changed since the launch of Android, the basic building blocks of Android app have remained the same. Let's look at some of the basic building blocks of an Android app in detail. Later, we will see how all these components come together to create an app, using the example of a Twitter application.

Android Bulding Blocks

An Android Application is coded in Java and compiled into a single distribution package called as APK. APK is the executable file which is installed on your phone as Android app. Let's walk through the various building blocks of the an Android application.



Building Blocks of Android

The basic building blocks of any Android app are - **Activities**, **Views** and **Intents**. These are a must in any Android app

Services, Content Provider and Broadcast are some of the advance features which add more functionality to your app

Let's go 1 by 1 into each of these components and get more details about them.

Activity

This is the first Android component you will encounter as soon as you open an Android app. An Android app should have at least one Activity in it.

- So what exactly is Activity ? Every screen in an Android app is an Activity. Activity will always have a User Interface (abbreviated as UI). It is not possible to have an Activity without a UI.

•Any Android app has one or more Activities. Out of these, one Activity will act as the entry point. The Activity will show up when the app starts. This is usually referred to as the launcher Activity or the main Activity too.

•Every Activity has its own lifecycle. You can read more about it in the Activity Concept Lesson .

•Every Android app has a Manifest.xml where all Activites are defined and one Activity is marked as Main Activity. One of the most common error developers commit when they start with Android development is forgetting to add a new Activity in Mainfest.xml file.

As a developer, Activity is a Java class file where you write the logic. Activity does not include the UI. Rather, one of the things you need to write in your Activity logic is - which UI to show.

Go to hands on tutorial on Acitivity

User Interface / Views

User Interface or UI is what the user sees on the screen. The Activity has the responsibility to *set* the UI for the screen. UI comprises primarily of two type of sub-components. **Views** & **Layouts or ViewGroups**.

View as the name suggest is the basic building block like buttons, label, input-box etc. Layout is the container for View elements. The Layout primarily defines the pattern in which the View element should show up. Eg. LinearLayout mandates that the View elements inside it either stack up horizontally or vertically while RelativeLayout lets each View element position itself relative to the parent or a sibling element.

UI is defined as XML. The top XML element is a Layout element . Inside it, there can be either View elements or Layout elements. An example XML file is shown below.

```
<LinearLayout 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:orientation="horizontal" >
   <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Button 1" />
   <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
```

```
        android:text="Button 2" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button 3" />
 </LinearLayout>
```

You may read more about Layouts & Views in our Layout Concept Lesson & View Concept Lesson.

Go to hands on tutorial on Android UI

Intents

To move from one Activity to another (or one screen to another), on user interaction like click of a button or click of a notification item, Intents are used. It is possible to pass data including whole objects with Intent. Using Intent you can also open another Android application.

Apart from launching an Activity, Intent can also launch a Service.

With Activity, Views and Intent; you can create a basic Android app. Lots of apps are made by using only these three concepts.

Let's look into some of the advanced Android concepts.

Go to hands on tutorial on Intents

Services

Have you ever listened to Music on your phone? Have you observed that the music continues to play in the background when you go to the home screen from the Music app or close the app?

This is achieved using Services. Services is the Android way of keeping an operation going on in the background. When you need to have long running tasks like playing music, downloading data or uploading photos; it is achieved through Service.

Service doesn't have any UI. To show any information to the user from Service, Notifications are used.

There are two ways in which you can create a Service. One way is to tie the Service with an Activity. In this case, the Service will end once Activity stops. The other way is to run a Service independent of any Application. This way, the Service keeps running in the background even after the application is stopped.

SQLite (Database) in Android

Android ships with SQLite database support. Android apps can store data locally in the SQLite

database. Every Android application can create its own private SQLite databases which it can use to store data for offline reference.

The purpose of storing data locally is primarily to provide a good user experience. Consider the example of the Android Twitter app. When you open the app, you immediately see the tweet list while there is a loading symbol shown which conveys that the tweets are still being fetched. The tweet list is stale, but you do not have to wait for the tweets to load because the app has stored the tweets locally.

There are other options apart from the database to store data in an Android app. Information of small sizes can be stored in Shared Preferences. You can also create local files & store data in it.

Content Providers

If your Android app wants to use data from another Android app, you may use Content Providers.

A simple example of Content Provider is the Contacts app. You can get contacts in multiple applications like your SMS application, Dialer Application etc.

If you are using SQLite/database in your app, you can either access it directly or through a Content Provider. Content Provider gives you encapsulated access.

Notifications

Earlier in the Services section, we came across Notifications. If you have used an Android phone, you would have seen small notifications in the top part of your phone for missed call, SMS or email received etc. These are Notifications.

- Notifications are simple messages which are used to display information to user that doesn't require user's immediate attention
- Typically notification comes with Text and Small icon, but newer version of Android support richer notification with buttons and images
- User can tap on notification and can interact with it

Notifications are the most widely used functionality in Android, from SMS to missed call, from wifi network connection to Twitter notifications. They are used everywhere.

Broadcast Receivers

Broadcast receiver is a way to listen to systemwide events happening in your phone or tablet. Using Broadcast Receivers, we can create some great applications like Call Number finder, SMS blocker etc, which works when some events happen in your device.

Many systemwide events broadcast their information. For example

- When SMS / Call is received
- Battery low
- Network state Changed
- Photo captured from camera

- Phone Starts

We can also generate custom broadcast from inside our application. Whenever you want to create any application which works with system events use broadcast receiver

Broadcast receivers work even if your application is not running. That means you can create functionality in your application which can be invoked automatically by Android if something happens without actually running any service or application in the background.