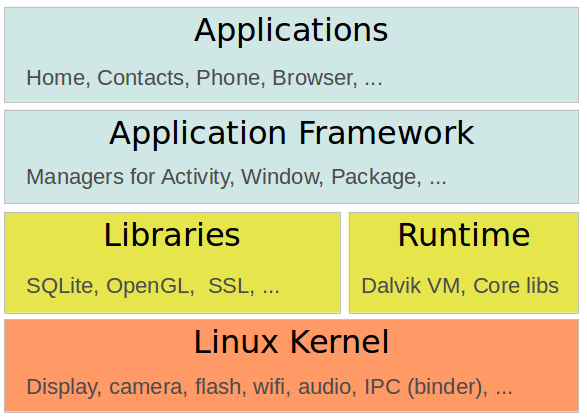
Android Development.

**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**.

### 1.1 Android platform components.

The Android system is a full software stack, which is typically divided into the four areas as depicted in the following graphic.



The levels can be described as:

* Applications - The Android Open Source Project contains several default application, like the Browser, Camera, Gallery, Music, Phone and more.
* Application framework - An API which allows high-level interactions with the Android system from Android applications.
* Libraries and runtime - The libraries for many common functions (e.g.: graphic rendering, data storage, web browsing, etc.) of the Application Framework and the Dalvik runtime, as well as the core Java libraries for running Android applications.
* Linux kernel - Communication layer for the underlying hardware.

The Linux kernel, the libraries and the runtime are encapsulated by the application framework. The Android application developer typically works with the two layers on top to create new Android applications.

Why Android?



# **History of Android**

The history and versions of android are interesting to know. The code names of android ranges from A to J currently, such as **Aestro**, **Blender**, **Cupcake**, **Donut**, **Eclair**, **Froyo**, **Gingerbread**, **Honeycomb**, **Ice Cream Sandwitch**, **Jelly Bean**, **KitKat** and **Lollipop**. Let's understand the android history in a sequence.

1) Initially, **Andy Rubin** founded Android Incorporation in Palo Alto, California, United States in October, 2003.

2) In 17th August 2005, Google acquired android Incorporation. Since then, it is in the subsidiary of Google Incorporation.

3) The key employees of Android Incorporation are **Andy Rubin**, **Rich Miner**, **Chris White** and **Nick Sears**.

4) Originally intended for camera but shifted to smart phones later because of low market for camera only.

5) Android is the nick name of Andy Rubin given by coworkers because of his love to robots.

6) In 2007, Google announces the development of android OS.

7) In 2008, HTC launched the first android mobile.

## **Android Versions, Codename and API**

Let's see the android versions, codenames and API Level provided by Google.

|  |  |  |
| --- | --- | --- |
| **Version** | **Code name** | **API Level** |
| 1.5 | Cupcake | 3 |
| 1.6 | Donut | 4 |
| 2.1 | Éclair | 7 |
| 2.2 | Froyo | 8 |
| 2.3 | Gingerbread | 9 and 10 |
| 3.1 and 3.3 | Honeycomb | 12 and 13 |
| 4.0 | Ice Cream Sandwitch | 15 |
| 4.1, 4.2 and 4.3 | Jelly Bean | 16, 17 and 18 |
| 4.4 | KitKat | 19 |
| 5.0 | Lollipop | 21 |

What is API level?

API Level is an integer value that uniquely identifies the framework API revision offered by a version of the Android platform.

Features of Android

Android is a powerful operating system competing with Apple 4GS and supports great features. Few of them are listed below:

|  |  |
| --- | --- |
| **Feature** | **Description** |
| Beautiful UI | Android OS basic screen provides a beautiful and intuitive user interface. |
| Connectivity | GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, NFC and WiMAX. |
| Storage | SQLite, a lightweight relational database, is used for data storage purposes. |
| Media support | H.263, H.264, MPEG-4 SP, AMR, AMR-WB, AAC, HE-AAC, AAC 5.1, MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, and BMP |
| Messaging | SMS and MMS |
| Web browser | Based on the open-source WebKit layout engine, coupled with Chrome's V8 JavaScript engine supporting HTML5 and CSS3. |
| Multi-touch | Android has native support for multi-touch which was initially made available in handsets such as the HTC Hero. |
| Multi-tasking | User can jump from one task to another and same time various application can run simultaneously. |
| Resizable widgets | Widgets are resizable, so users can expand them to show more content or shrink them to save space |
| Multi-Language | Supports single direction and bi-directional text. |
| GCM | Google Cloud Messaging (GCM) is a service that lets developers send short message data to their users on Android devices, without needing a proprietary sync solution. |
| Wi-Fi Direct | A technology that lets apps discover and pair directly, over a high-bandwidth peer-to-peer connection. |
| Android Beam | A popular NFC-based technology that lets users instantly share, just by touching two NFC-enabled phones together. |

ANDROID ARCHITECTURE



Android operating system is a stack of software components which is roughly divided into five sections and four main layers as shown above in the architecture diagram.

**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.

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. Native Libaries**

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.

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**

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.

**Android Core Building Blocks (or) Application Components**

An android **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.



Application components are the essential building blocks of an Android application. These components are loosely coupled by the application manifest file *AndroidManifest.xml* that describes each component of the application and how they interact.

There are following four main components that can be used within an Android application:

|  |  |
| --- | --- |
| **Components** | **Description** |
| Activities | They dictate the UI and handle the user interaction to the smart phone screen |
| Services | They handle background processing associated with an application. |
| Broadcast Receivers | They handle communication between Android OS and applications. |
| Content Providers | They handle data and database management issues. |

1. Activities.

An activity represents a single screen with a user interface,in-short Activity performs actions on the screen. For example, an email application might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. If an application has more than one activity, then one of them should be marked as the activity that is presented when the application is launched.

An activity is implemented as a subclass of **Activity** class as follows −

public class MainActivity extends Activity {

2. Services

A service is a component that runs in the background to perform long-running operations. For example, a service might play music in the background while the user is in a different application, or it might fetch data over the network without blocking user interaction with an activity.

A service is implemented as a subclass of **Service** class as follows −

public class MyService extends Service {

}

3. Broadcast Receivers

Broadcast Receivers simply respond to broadcast messages from other applications or from the system. For example, applications can also initiate broadcasts to let other applications know that some data has been downloaded to the device and is available for them to use, so this is broadcast receiver who will intercept this communication and will initiate appropriate action.

A broadcast receiver is implemented as a subclass of **Broadcast Receiver** class and each message is broadcaster as an **Intent** object.

public class MyReceiver extends BroadcastReceiver {

public void onReceive(context,intent){}

}

4. Content Providers

A content provider component supplies data from one application to others on request. Such requests are handled by the methods of the *ContentResolver* class. The data may be stored in the file system, the database or somewhere else entirely.

A content provider is implemented as a subclass of **ContentProvider** class and must implement a standard set of APIs that enable other applications to perform transactions.

public class MyContentProvider extends ContentProvider {

public void onCreate(){}

}

We will go through these tags in detail while covering application components in individual chapters.

5. Additional Components

There are additional components which will be used in the construction of above mentioned entities, their logic, and wiring between them. These components are –

|  |  |
| --- | --- |
| **Components** | **Description** |
| Fragments | Represents a portion of user interface in an Activity. |
| Views | UI elements that are drawn on-screen including buttons, lists forms etc. |
| Layouts | View hierarchies that control screen format and appearance of the views. |
| Intents | Messages wiring components together. |
| Resources | External elements, such as strings, constants and drawable pictures. |
| Manifest | Configuration file for the application. |

Building Blocks of Android

|  |
| --- |
| Activity |
| View /User Interface |
| Intents |
| Services |
| Broadcast Receivers |
| Content Providers |

1. Activity:-

* This is the first Component as soon as you open an android Application, An Android App should have at least one Activity.
* Every Screen in an Android App is an Activity, It always has User Interface.
* Any Android App has one or more Activities, One Activity (Main Activity) will act as a Entry Point.
* Every Activity has its Own Life Cycle.
* Every Android App has a **Manifest.xml** file where all Activities are Defined and one Activity marked as **MainActivity**.

2. View/User Interface:-

* User Interface has two types of Sub Components:-Views & Layouts (or) View Groups.
* **View** as the name suggest it is the basic building block like Buttons, Labels, Input Box , etc. **Layout** is Container for View Elements.
* The Layout primarily defines the pattern in which the elements should show up. Layouts consists of Linear , Relative , Grid layouts etc.

3. Intents:-

* It is used to Invoke Components Like
* Start the Service
* Launch an Activity
* Display Web page
* Display List of Contacts
* To move from One Activity to another (or ) one Screen to another, on user interaction like Click a button of a notification item, Intents are used.
* It is possible to pass data including the Objects with Intents. Using Intent we can also open another Android application.

4. 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, Uploading Photos etc. It is achieved through Services.
* 2 ways in which you create a services

1. One way is to tie the Service with an Activity .In this Case ,the Service will end once Activity Stops.
2. Other way is to Run a Service independently of any application. This way, the Service keeps Running in the background even after the application stopped.

5. Content Providers:-

* Contents providers help in sharing the data between Apps. If Android App wants to use data from another app, then we will use Content providers.
* Examples:-Contacts App, you can get contacts in multiple application like your SMS application,Dialer application,watsup etc.

6. Broadcast Receiver:-

* Broadcast Receiver is a way to listen System wide Events Happening in the Phones.
* Using Broadcast Receivers, we can create some great applications like Call Number finder, Sms Blocker etc. This works when some events happen in the device.
* Many System wide events broadcast their information

Examples:-1.When Sms/call is received

2. Battery Low

3. Network State Changed

4. Phone captured from Camera

5. Phone Starts.

**Using Android Studio**

Android supports java, C++, c# etc. language to develop android applications. Java is the officially supported language for android.

**Installing Android Studio**

1. **Install JDK in your OS.**

2**. Visit http://developer.android.com/sdk/index.html.**

3. **Click the download button for your operating system (Windows, Mac, or Linux).**

4. **Install SDK Packages.**

**Layouts**

**Layout** is Container for View Elements .The Layout primarily defines the pattern in which the elements should show up.

A layout defines the visual structure for a user interface, such as the UI for an activity or app widget. We can declare a layout in two ways:

**Declare UI elements in XML**:- Android provides a straight forward XML vocabulary that corresponds to the View classes and subclasses, such as those for widgets and layouts.

**Instantiate layout elements at runtime**:- our application can create View and ViewGroup Objects(and manipulate their properties) programmatically.

**Layout Types**

1. Relative layout

2.Linear layout 3.Table View 4.Frame View 5.Web View 6.Grid layout 7.List View

Activity:-

* An Activity is an application Component that provides a screen with which users can interact in order to do something, such as dial the phone, take a phone, send an email, or view a map.
* Each Activity is given a window in which to draw its user interface.
* An Android app can have multiple activities and one among them will be designated as main activity. If an android app has more than one activity, only one of them can be in active state at a time. These states of the activity are maintained by a stack called back stack. Currently running activity will be on top of the stack. When a new Activity starts the older activity is pushed down the stack and the current activity becomes the top element in the stack which is given user access.

2.1. Activity State and Callback Methods

Each activity is interlinked by calling other activities. These subsequent calls form activity life cycle which will be handled by activity manager by interacting. An android activity life cycle consists of three states namely,

* running (resumed)
* paused
* stopped

Activity Callback Methods.

When an activity transitions from one state to another, system calls the given callback methods. Following are the available callback methods,

* onCreate()
* onStart()
* onPause()
* onResume()
* onStop()
* onRestart()
* onDestroy()

### Declaring the activity in the manifest.xml

We must declare activity in the manifest file in order for it to be accessible to the system. To declare your activity, open your manifest file and add an <activity> element as a child of the <application> element.\

For examples:

<manifest…> <application….> <activity android:name=”.ExampleActivity”/>…….. </application>

…..

</manifest>

### Activity Life Cycle

**public** **class** **ExampleActivity** **extends** **Activity** {

@Override  
    **public** **void** [onCreate](http://developer.android.com/reference/android/app/Activity.html#onCreate(android.os.Bundle))(**Bundle** savedInstanceState) {  
        **super**.onCreate(savedInstanceState);  
        // The activity is being created.  
    }  
    @Override  
    **protected** **void** [onStart()](http://developer.android.com/reference/android/app/Activity.html#onStart()) {  
        **super**.onStart();  
        // The activity is about to become visible.  
    }  
    @Override  
    **protected** **void** [onResume()](http://developer.android.com/reference/android/app/Activity.html#onResume()) {  
        **super**.onResume();  
        // The activity has become visible (it is now "resumed").  
    }  
    @Override  
    **protected** **void** [onPause()](http://developer.android.com/reference/android/app/Activity.html#onPause()) {  
        **super**.onPause();  
        // Another activity is taking focus (this activity is about to be "paused").  
    }  
    @Override  
    **protected** **void** [onStop()](http://developer.android.com/reference/android/app/Activity.html#onStop()) {  
        **super**.onStop();  
        // The activity is no longer visible (it is now "stopped")  
    }  
    @Override  
    **protected** **void** [onDestroy()](http://developer.android.com/reference/android/app/Activity.html#onDestroy()) {  
        **super**.onDestroy();  
        // The activity is about to be destroyed.  
    }  
}