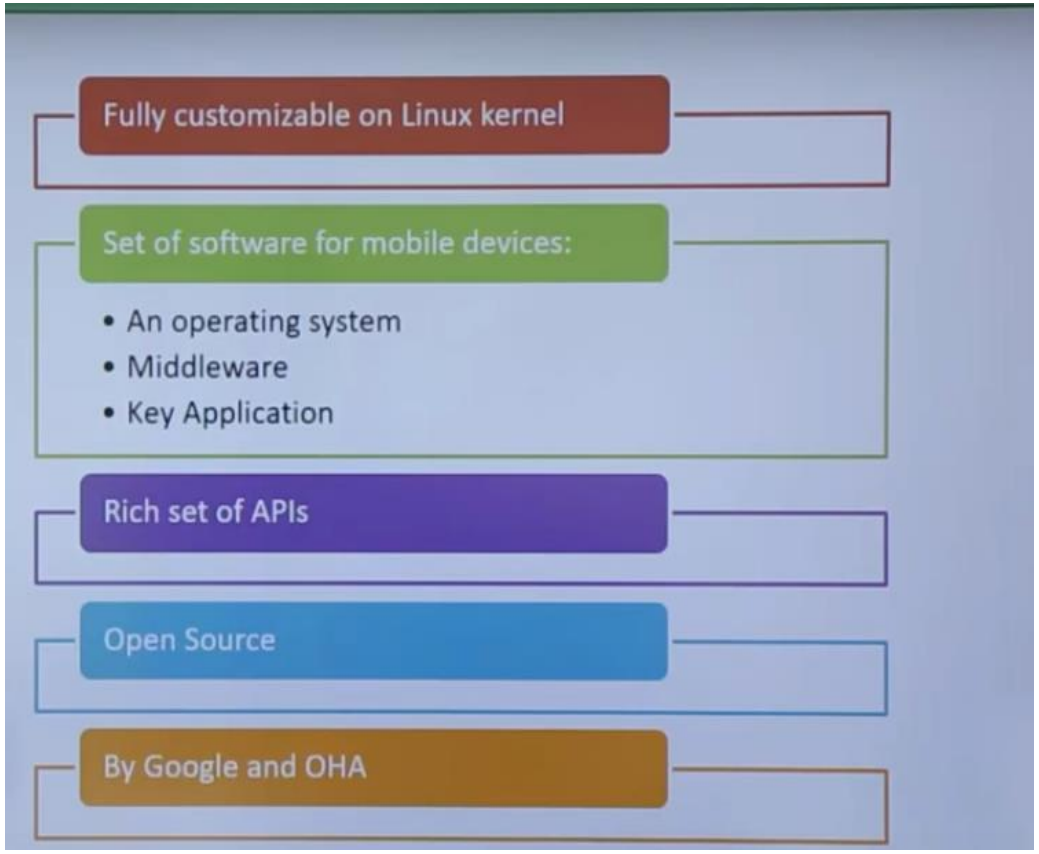
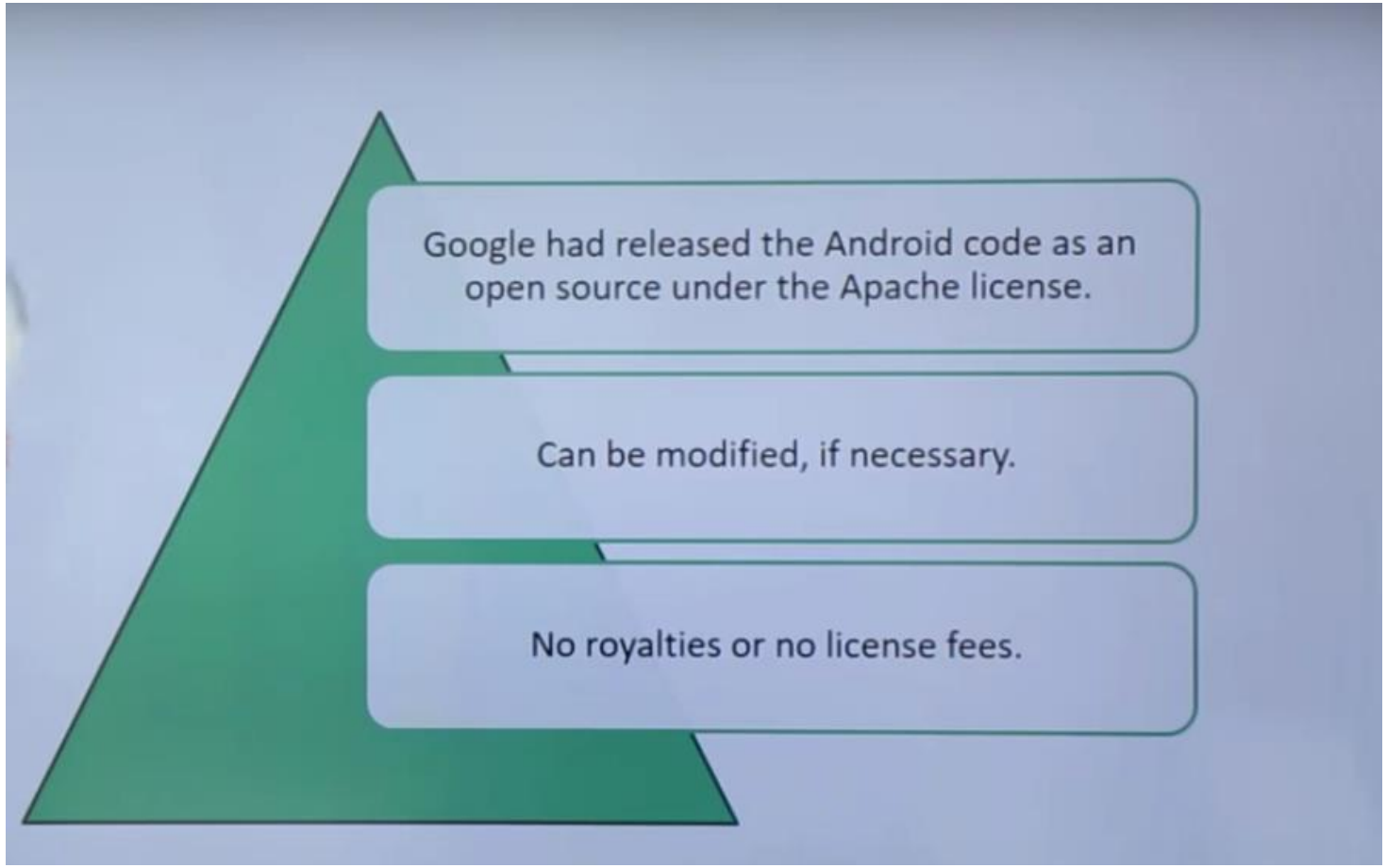
What is Android?

What is Android?

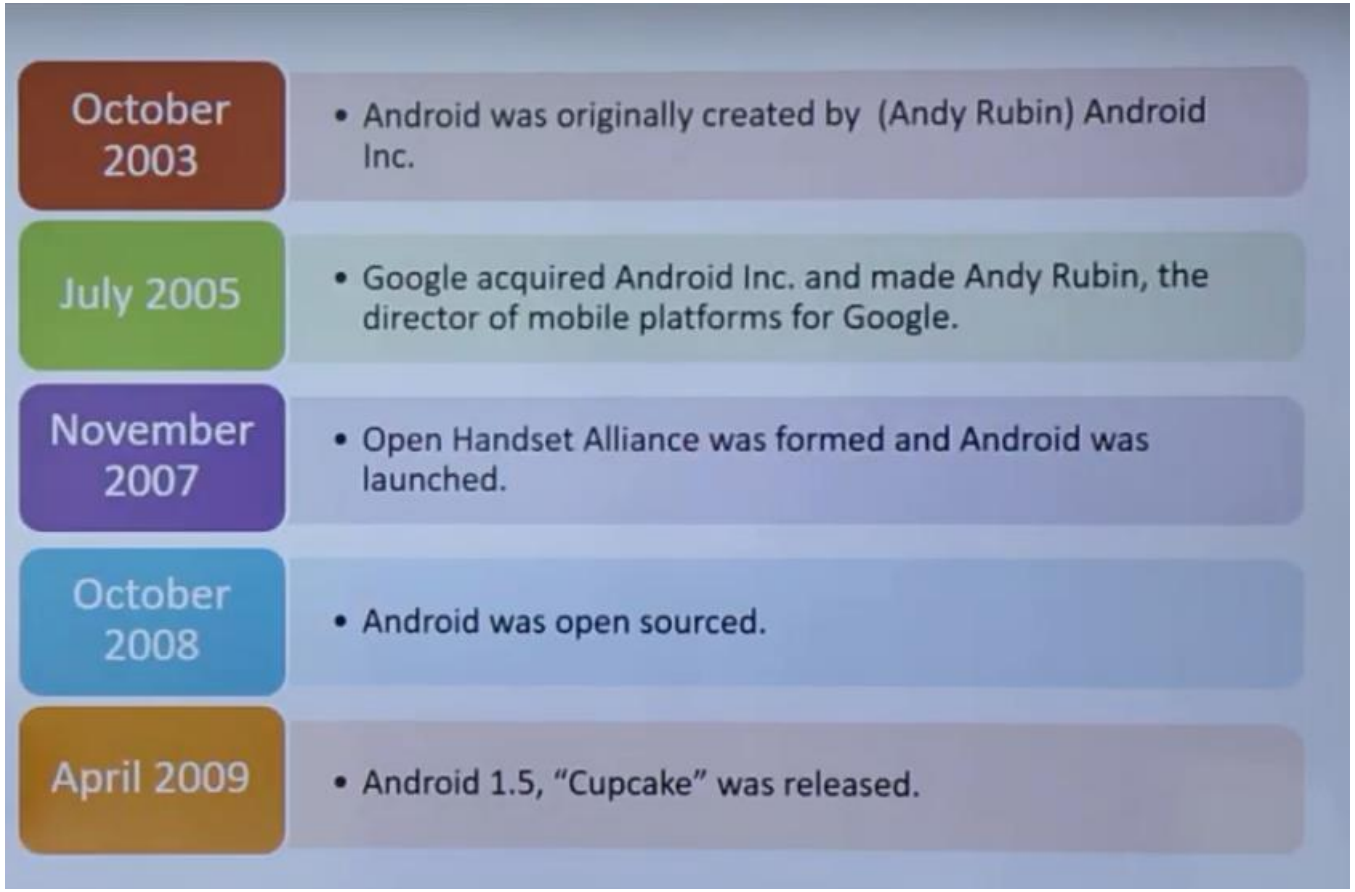
- Linux kernel - Android is based on Linux. The main work of Linux kernel is to get the work done from hardware.
- Operating System - is an interface between the hardware and the user. It's a System Software
- Middleware - Some set of libraries written together that make the developer easier to work on it. They are the glue between the kernel and the applications that enhance the ease of use for developer.
- Key Applications – Are nothing but the applications that are inbuilt in our Operating System. E.g. SMS, logs, Camera etc.
- Application Programming Interface (API) – They are already written and can be used by the developer for using any functionality. No need to begin from scratch.



What is Open Source Platform?



History Of Android

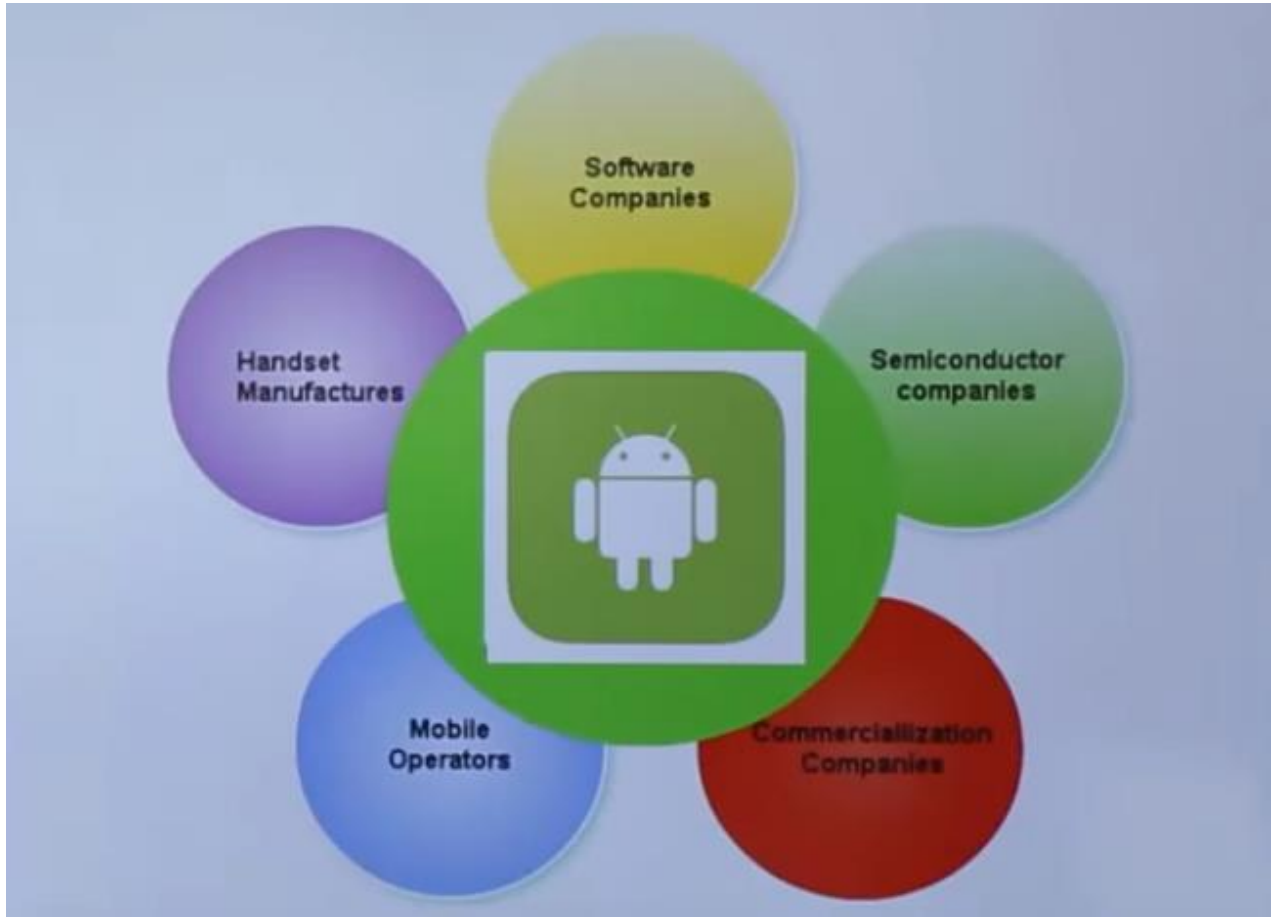


Inc. is the abbreviation for incorporated.

An incorporated company, or corporation, is a separate legal entity from the person or people forming it.

Android was developed by the Open Handset Alliance, led by Google, and other companies.

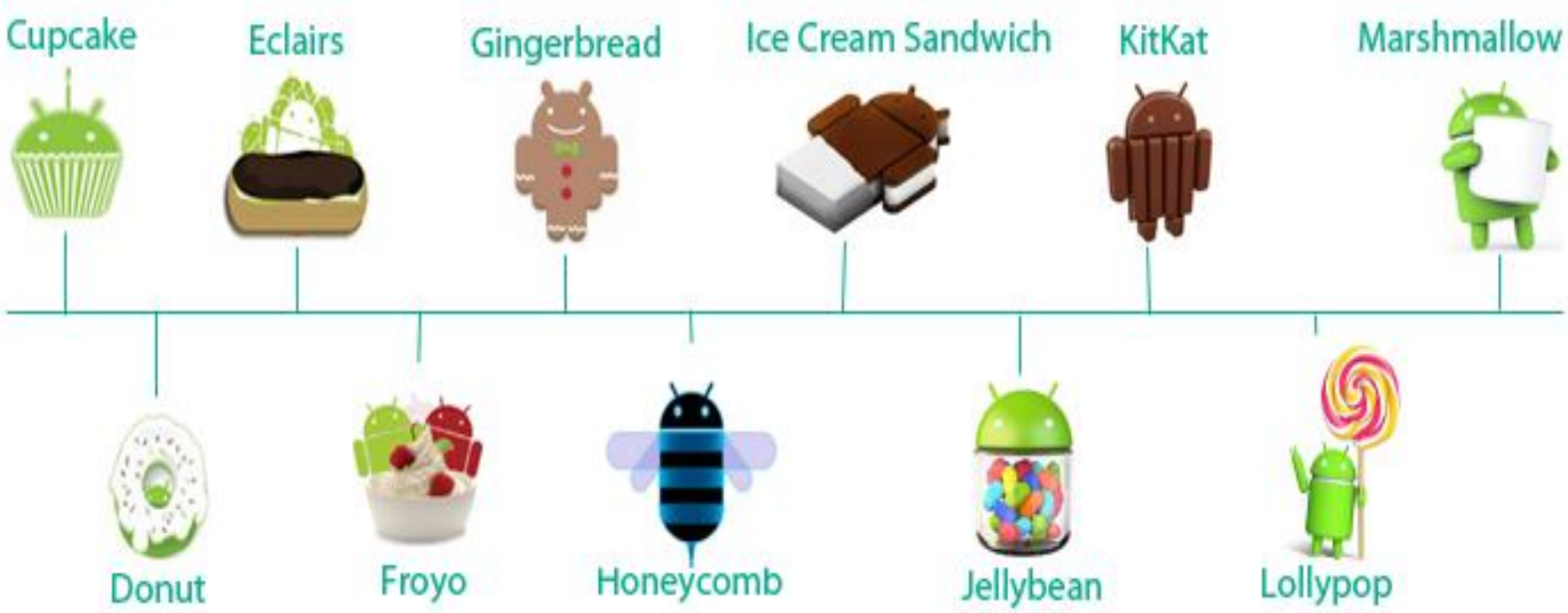
Open Handset Alliance (OHA)



Why Android ?


Open Source
Larger Developer and Community Reach
Increased Marketing
Inter App Integration
Reduced Cost of Development
Higher Success Ratio
Rich Development Environment
android

History of Android Operating System



Android 1.0 and 1.1 were not named after desserts

Android 1.0 was sometimes called “Alpha” or “Astro Boy.” Android 1.1 kind has a dessert name, but not in the order that we know today. It was known internally as “Petit Four,” which is a small French appetizer. Android 1.5 “Cupcake” was the first version to have an official dessert nickname.

Cupcake (1.5)

- Marks the emergence of a virtual keyboard on Android. Support for third-party keyboards.
- Five default widgets were made available – a calendar, a music player, an analog clock, a picture frame, search function.

Cupcake (1.5)

Donut(1.6)

- The camera and gallery functionality were improved in this version, now a user can select and delete multiple photos in the gallery.

Eclairs(2.0)

- Google Navigation was added in this version.
- Google maps proved a major change in the history of Android Operating System, it provides users directions and interactive maps.
- Support for multiple accounts. Contacts, emails, calendar synchronization were improved.
- Along with these functionalities, the browser was also improved.
- Now the user was able to save bookmarks with the thumbnails, can zoom by double tapping.
- The hardware keyboard was toned down as the facility to answer a call on-screen was provided.

Eclairs (2.0)

Android 2.2 (Froyo)

(Frozen Yogurt)

- Hotspot/Tethering started from Froyo version, it allowed you to connect other devices to the internet through one's phone.
- Apps can now be moved from phone memory to external SD card that will help to free up the device space.
- Flash support for full desktop browser experience on mobile.

Gingerbread(2.3)

The on-screen keyboard is redesigned to improve typing speed and accuracy, and suggestions are now available as you type.

HoneyComb(3.0/3.2)

- This time Android came with redesigned widgets. Basically, Honeycomb 3.0 was build for tablets and so the focus was on big screens. New widgets that were redesigned are the search box, date/time picker, calendar.
- Home Screen was customized. Now the user can swipe between 5 different screens along with a new 3D look for the home screen.

Android 3.0 is the only version to never run on phones

IceCream Sandwich(4.0)

- The UI was made easy with touch gestures for notifications and recent apps.
- From this version of Android, users were able to create folders on the home screen by simply dropping on the top of another

Ice Cream Sandwich (4.0)

JellyBean (4.1)

- A support for native language was provided for Hindi, Swahili, Zulu. An added support to languages such as Hebrew and Arabic was also provided.

Android 4.1 Jelly Bean

Kitkat(4.4)

- Just say “Ok Google” to start the service and Google will get the desired results you wish.
- Google Hangouts was also a part of KitKat- it dealt with SMS.
- Google Drive became a default app from KitKat.

KitKat (4.4)

Lollipop(5.0)

- Device protection was improved, even if your phone is lost or stolen, the thief can only factory reset the device but cannot unlock unless your Google account is entered.
- Dual Sim support now officially part of Android from Lollipop.

Marshmallow(6.0)

- Some of the features that are likely to be included in Marshmallow are Android pay, Google Now on tap, App permissions, standardized fingerprint support, Doze.
- When you go for an installation in Google Play Store, it will let you with a long list of things that developer want to access. From Marshmallow, it will be easier to track what phone features need to accessed and permissions will now be placed in certain categories like location, sensor, camera etc.

Marshmallow (6.0)

Nougat(7.0)

- Bring your words to life with updated and entirely new emoji.
- Multi window view
- Quick switch between apps
- High performance 3D Graphics

Android Oreo is our sweetest and newest release yet.

Oreo(8.0)

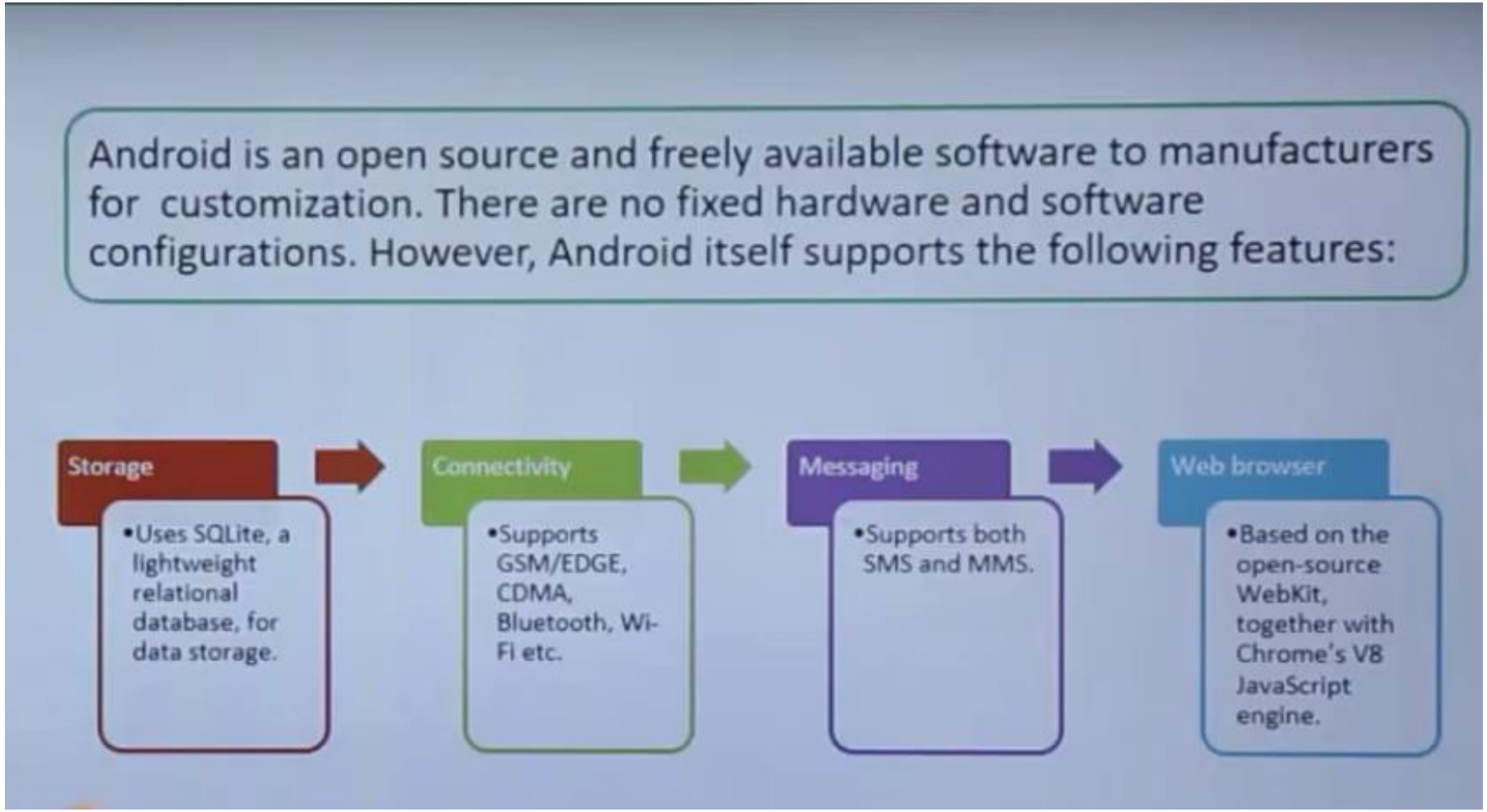
- Allows you to see two apps at once. It's like having super strength and laser vision.
- Press the notification dots to quickly see what's new, and easily clear them by swiping away
- With your permission, AutoFill remembers your logins to get you into your favourite apps at supersonic speed.

The first Android prototype looked like a Blackberry

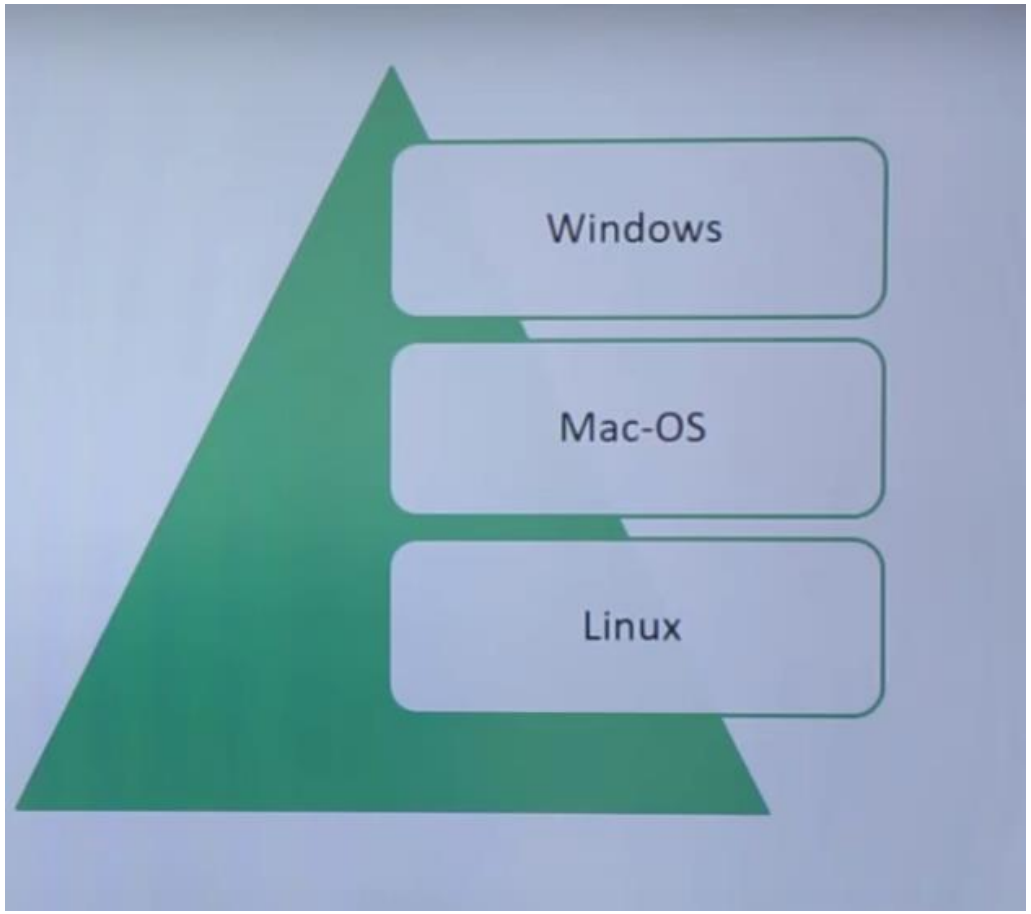
Categories of Android applications

- Entertainment
- Tools
- Communication
- Productivity
- Personalization
- Music and Audio
- Social
- Media and Video
- Travel and Local etc.

Android Features



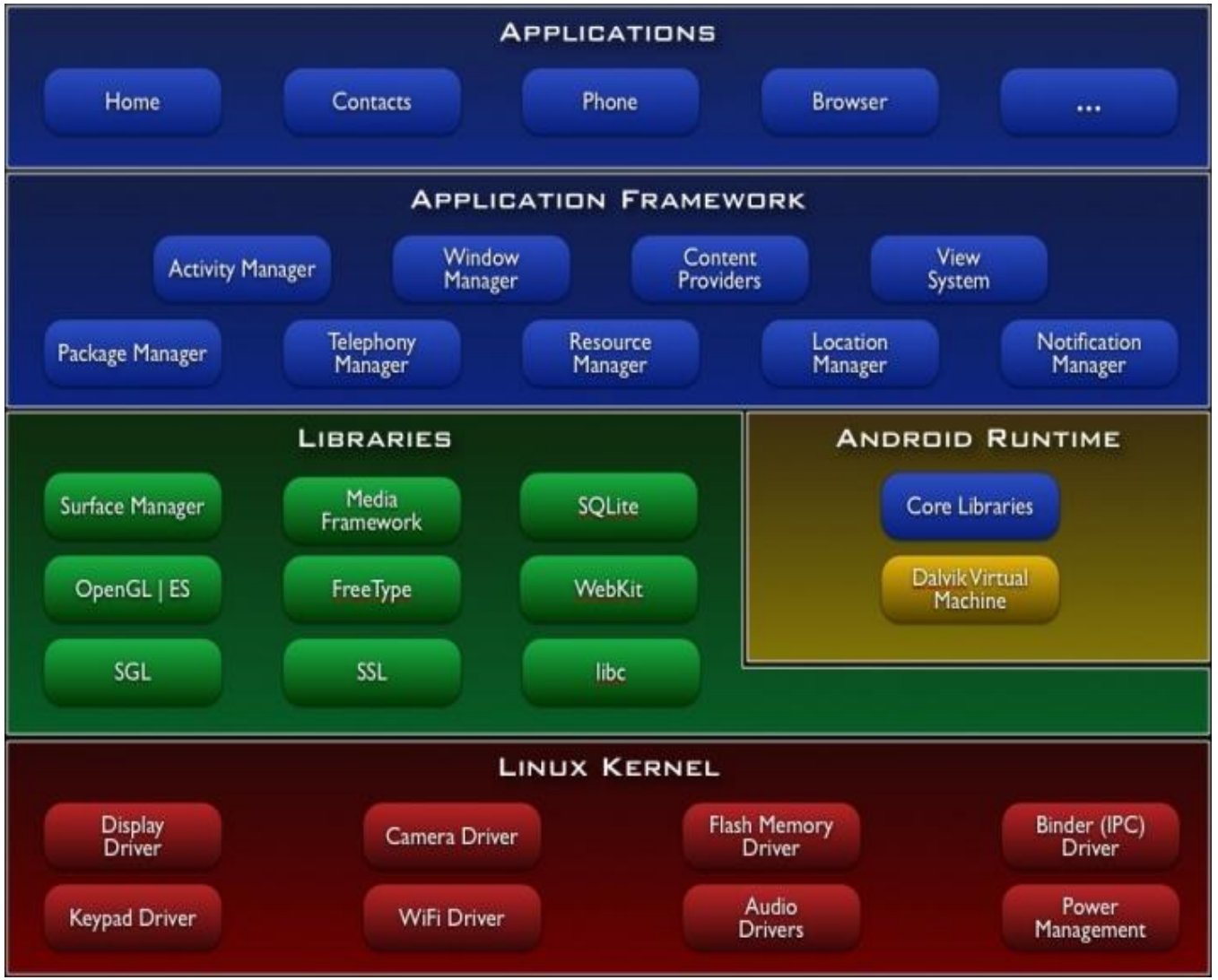
OS Supported for Android Development



Architecture of Android

5 Layer Architecture

- Linux kernel - Android is based on Linux. The main work of Linux kernel is to get the work done from hardware. Hardware could be anything camera, display or Wi-Fi
- Libraries – Are nothing but logical group of instructions that we want to give to the kernel to perform some action.
- Application Framework – Is a group of instances put together for purpose for developer to make things understandable. E.g Activity manager manages application's life cycle.
- Applications – Are built in our phone like browser, phone and camera etc



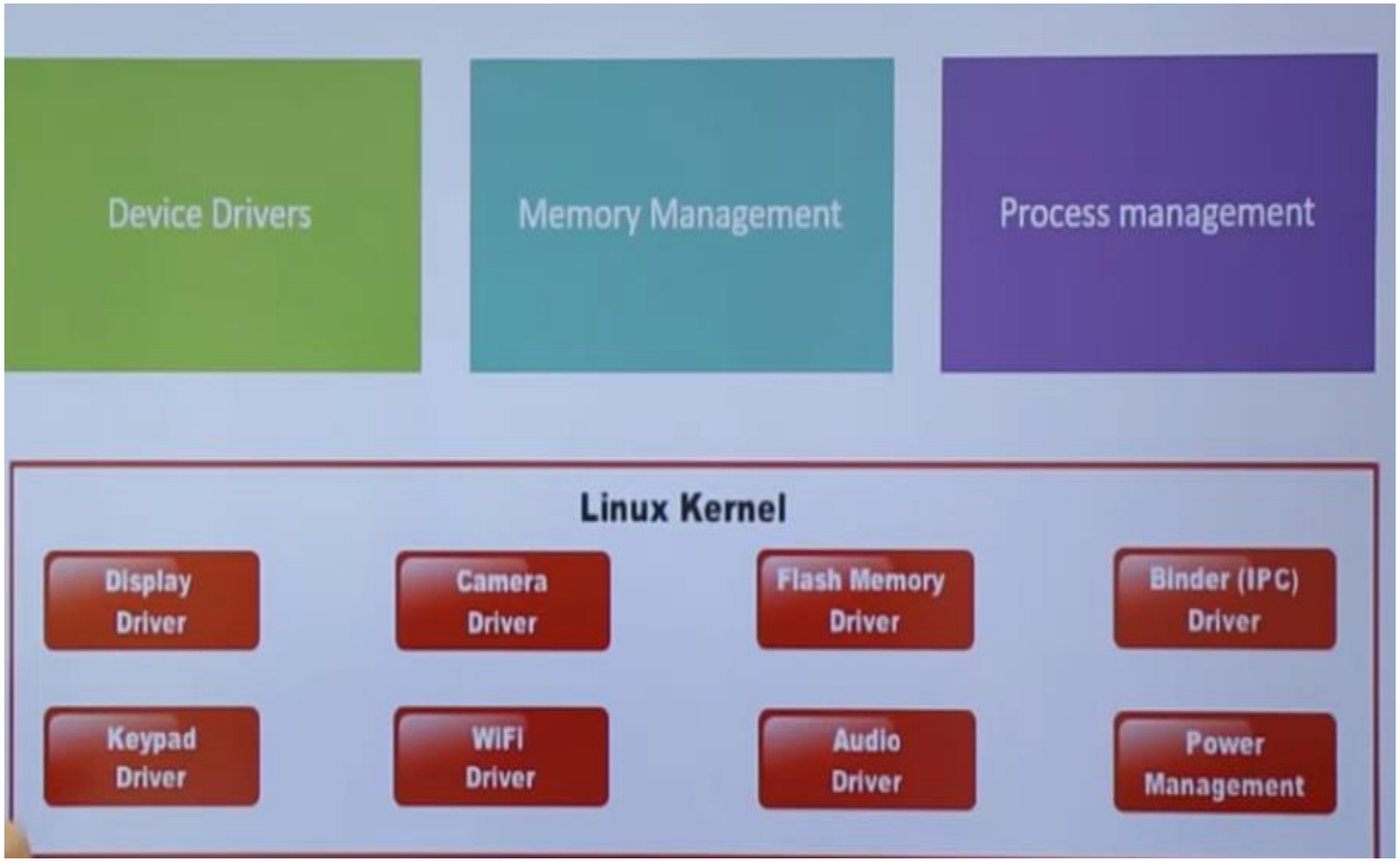
Linux Kernel

Kernel is involved in three main parts:

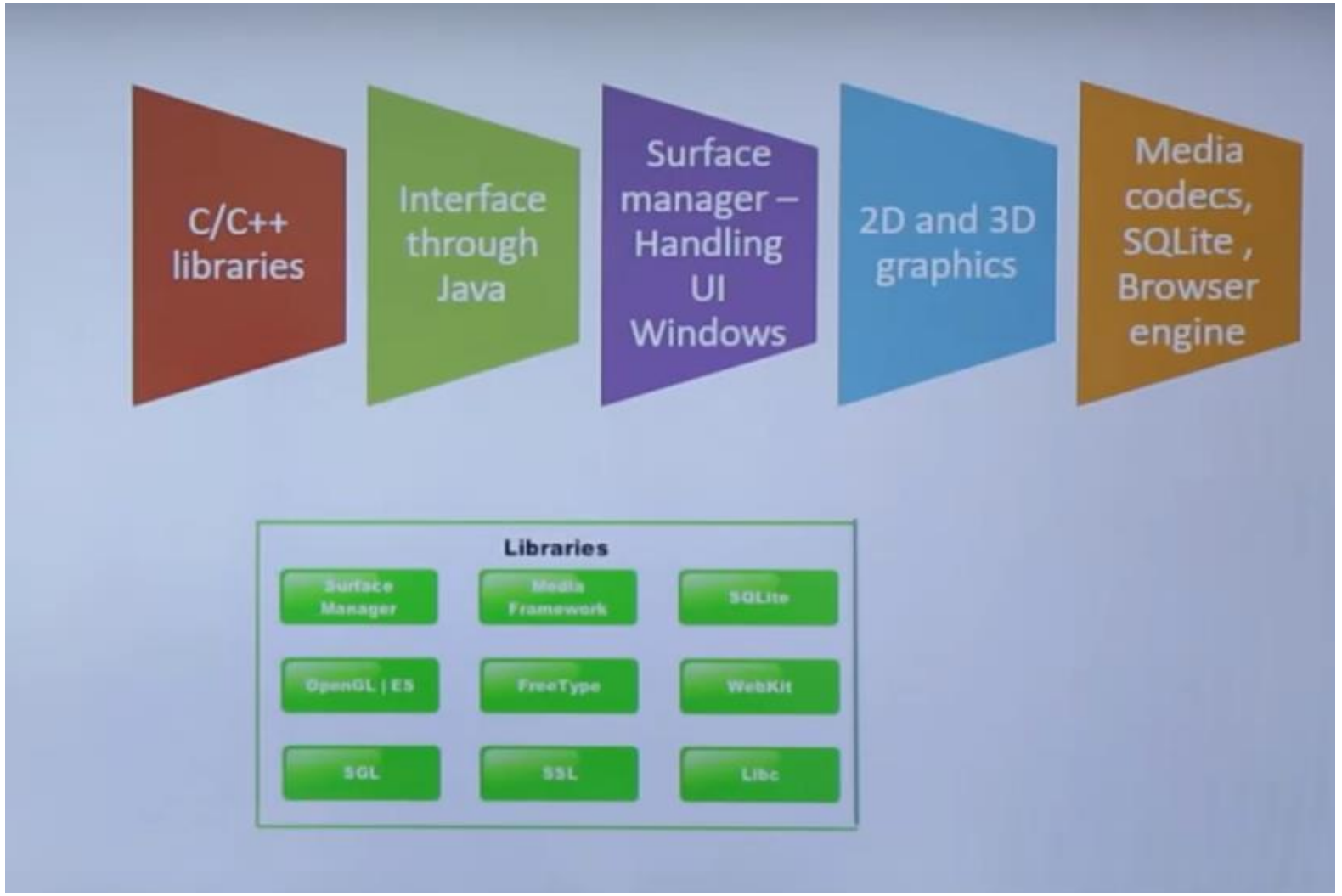
1. Device Drivers – Helps to get the work done from devices

2. Memory Management – How to manage memory

3. Process Management- A program in execution is a process E.g. Playing music, Turning Off/On the Bluetooth.

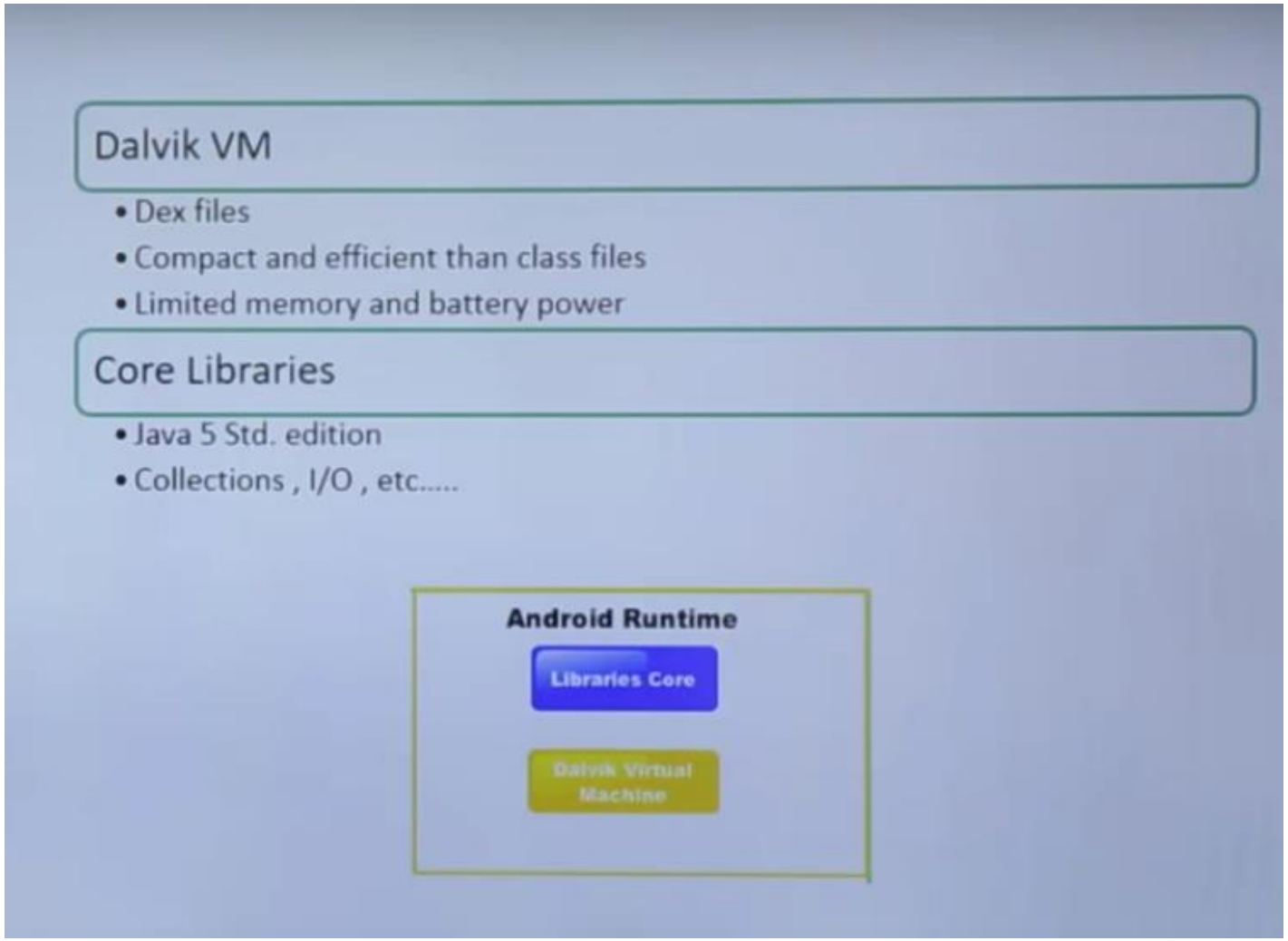


Libraries

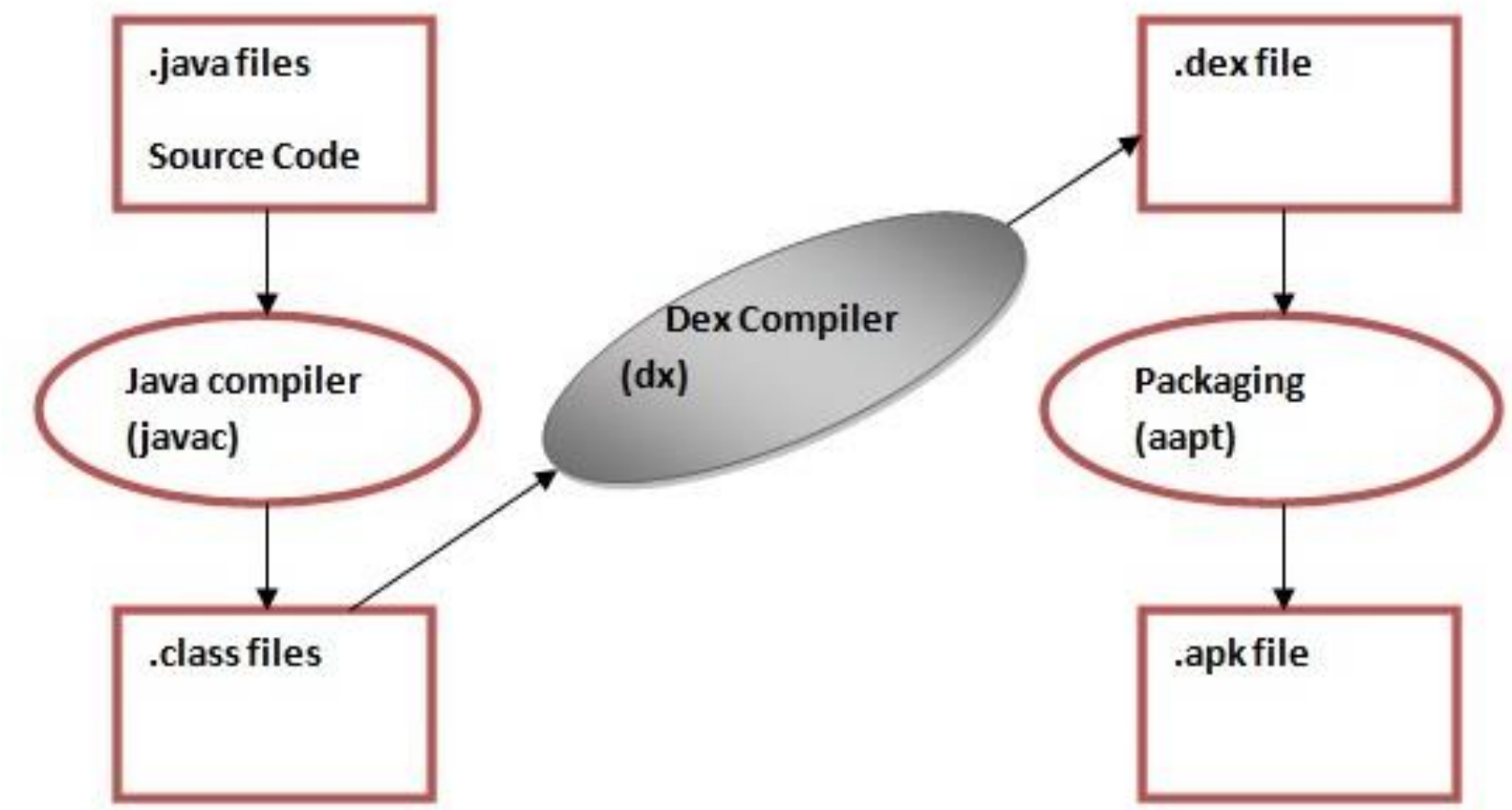

C/C++ libraries
Interface through Java
Surface manager – Handling UI Windows
2D and 3D graphics
Media codecs, SQLite , Browser engine
Libraries
Surface Manager
Media Framework
SQLite
OpenGL | ES
FreeType
WebKit
SGL
SSL
Libc

Android Runtime

1. Like for Java we have Java Virtual Machines(JVM), for android we have Dalvik Virtual Machines(DVM).
   The Dalvik Virtual Machine (DVM) is an android virtual machine optimized for mobile devices. It optimizes the virtual machine for memory, battery life and performance. Dalvik is a name of a town in Iceland. The Dalvik VM was written by Dan Bornstein.
2. Like .jar files we have dex files here. The Dex compiler converts the class files into the .dex file that run on the Dalvik VM. Multiple class files are converted into one dex file.
3. DVM and Core libraries together form Android Run Time.



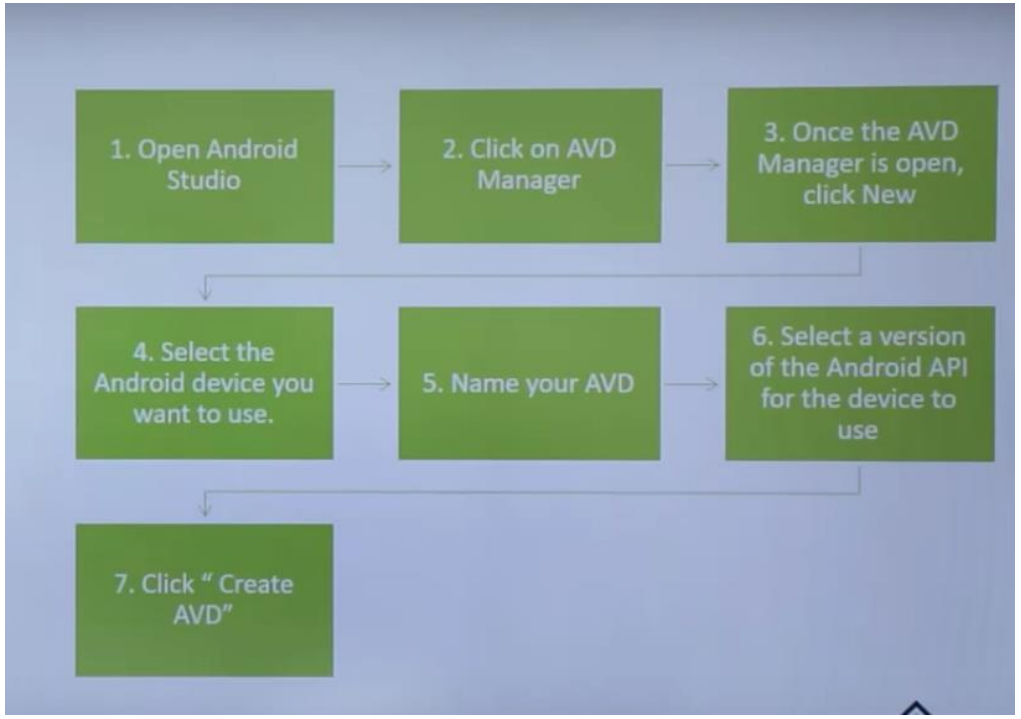
Compiling and Packaging Process



- The javac tool compiles the java source file into the class file.
- The dx tool takes all the class files of your application and generates a single .dex file. It is a platform-specific tool.
- The Android Assets Packaging Tool (aapt) handles the packaging process.

Setting Up the Android Virtual Device(AVD)/ Emulator

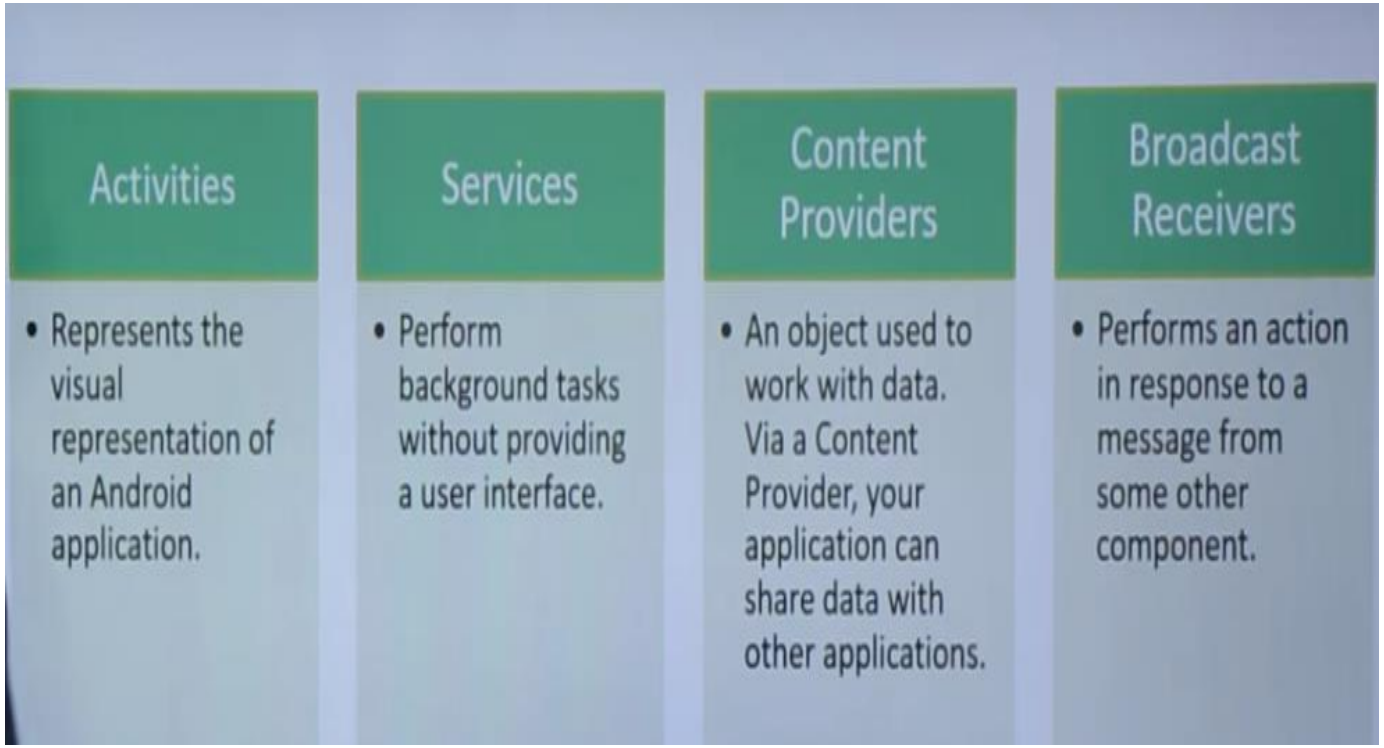
Being an Application Developer, if I need to check whether my application is working fine on Android device. I can use the Emulator known as Android Virtual Machine. This Virtual Device is the exact replica of your Mobile phone.




Intent
Activity
Service
Broadcast Receiver
Content Provider

Android Application Components

1. Activity - It is something visible to user. Each Activity has one UI. In Short, Android calls activity and it in turns calls an UI.
2. Service – Not visible to user. Runs on background.
3. Content Provider - helps data of other Applications to interact with our own application's data .Like SMS servicing
4. Broadcast Receivers – E.g WhatsApp verification message having OTP is itself known to WhatsApp. As Broadcast receiver throws an intent that this thing has happened.



Activity

- An activity is a class that represents a single screen.

View

- Anything that you see is a view. A view is the UI element such as button, label, text field etc.

Broadcast Receiver

- It is the component that responds to system conditions such as battery low of screen being turned off.
- Use it to initiate a response from a running application
- Implemented as a subclass of android BROADCAST RECIEVER CLASS

Manifest File

Applications can request services from the device's built in components such as camera and networking components , but the request to this services are added to manifest file at application design and development time

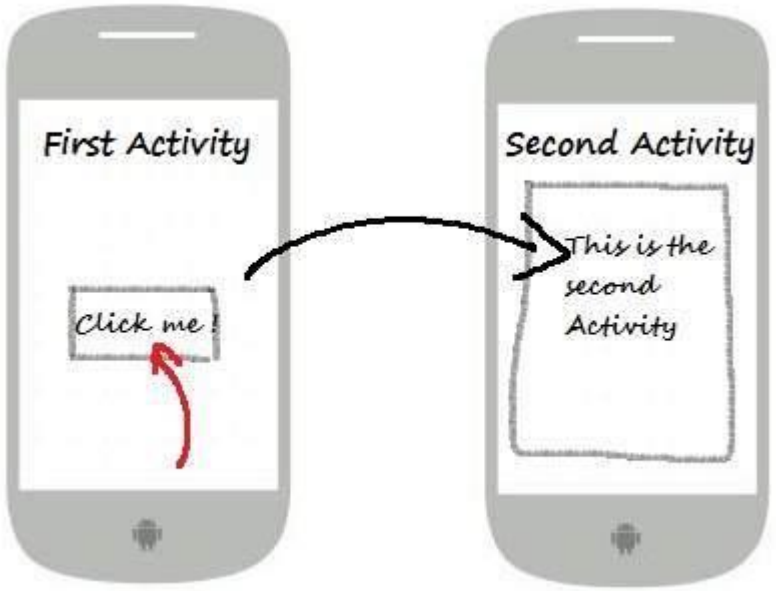
GRANT OR DENY REQUEST

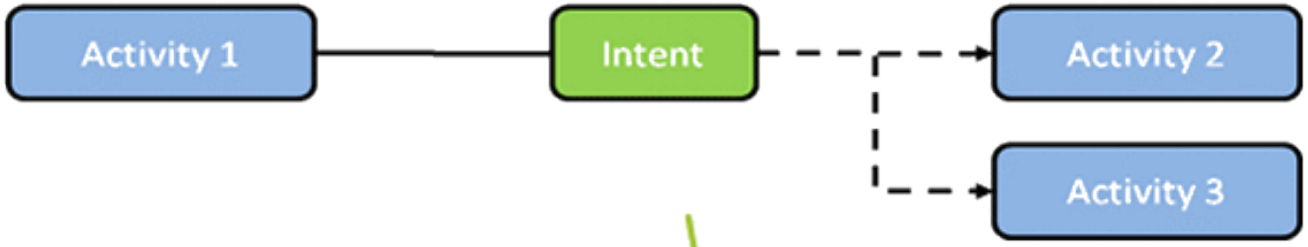
Intent

Intent is used to invoke components. An Android Intent is an abstract description of an operation to be performed. It is mainly 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

Intent





Intents

Service

- Service is a background process that can run for a long time.
- It do not provide user interfaces
- Developer creates service as a subclass of android SERVICE class
- E.g :- download data in background

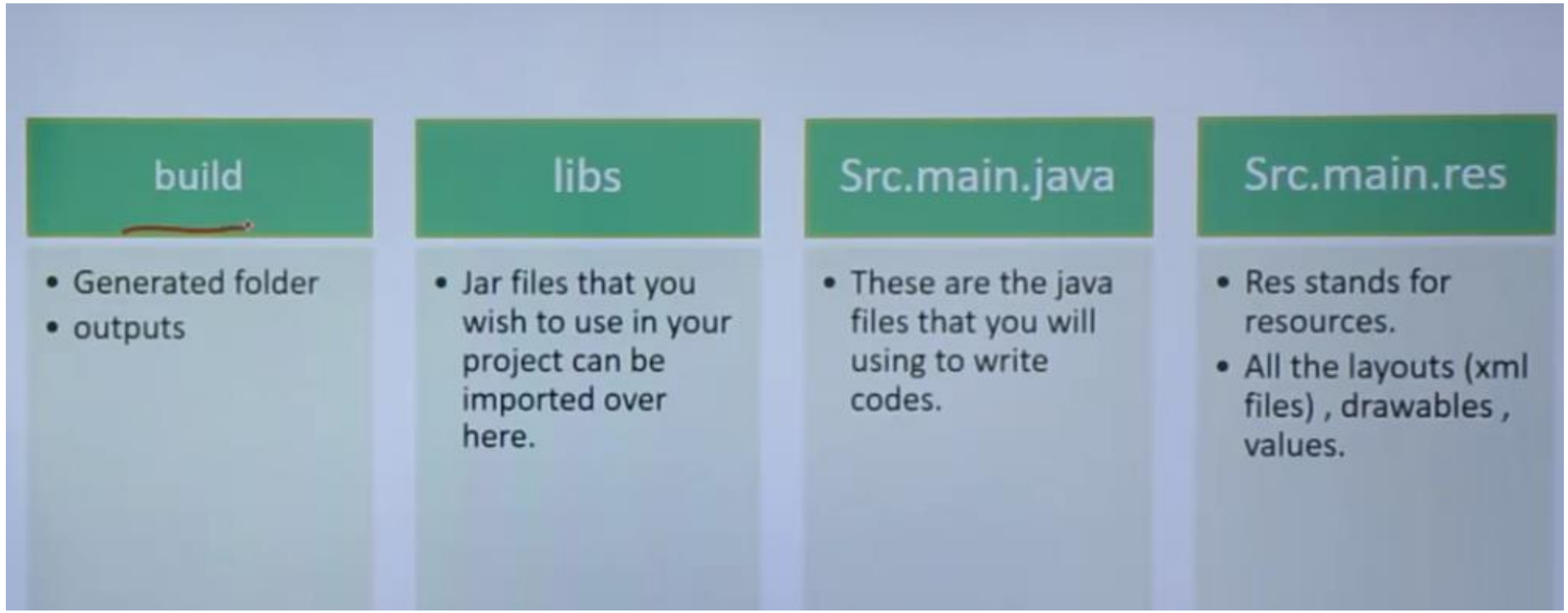
Content Providers

- Content Providers are used to share data between the applications
- Other applications can have access if they have the permission

Android Studio Project Structure

Following folders are there :

1. Build
2. Lib
3. Src
4. Res
5. And a Manifest file


build
• Generated folder
• outputs
libs
• Jar files that you wish to use in your project can be imported over here.
Src.main.java
• These are the java files that you will using to write codes.
Src.main.res
• Res stands for resources.
• All the layouts (xml files) , drawables , values.

AndroidManifest.xml

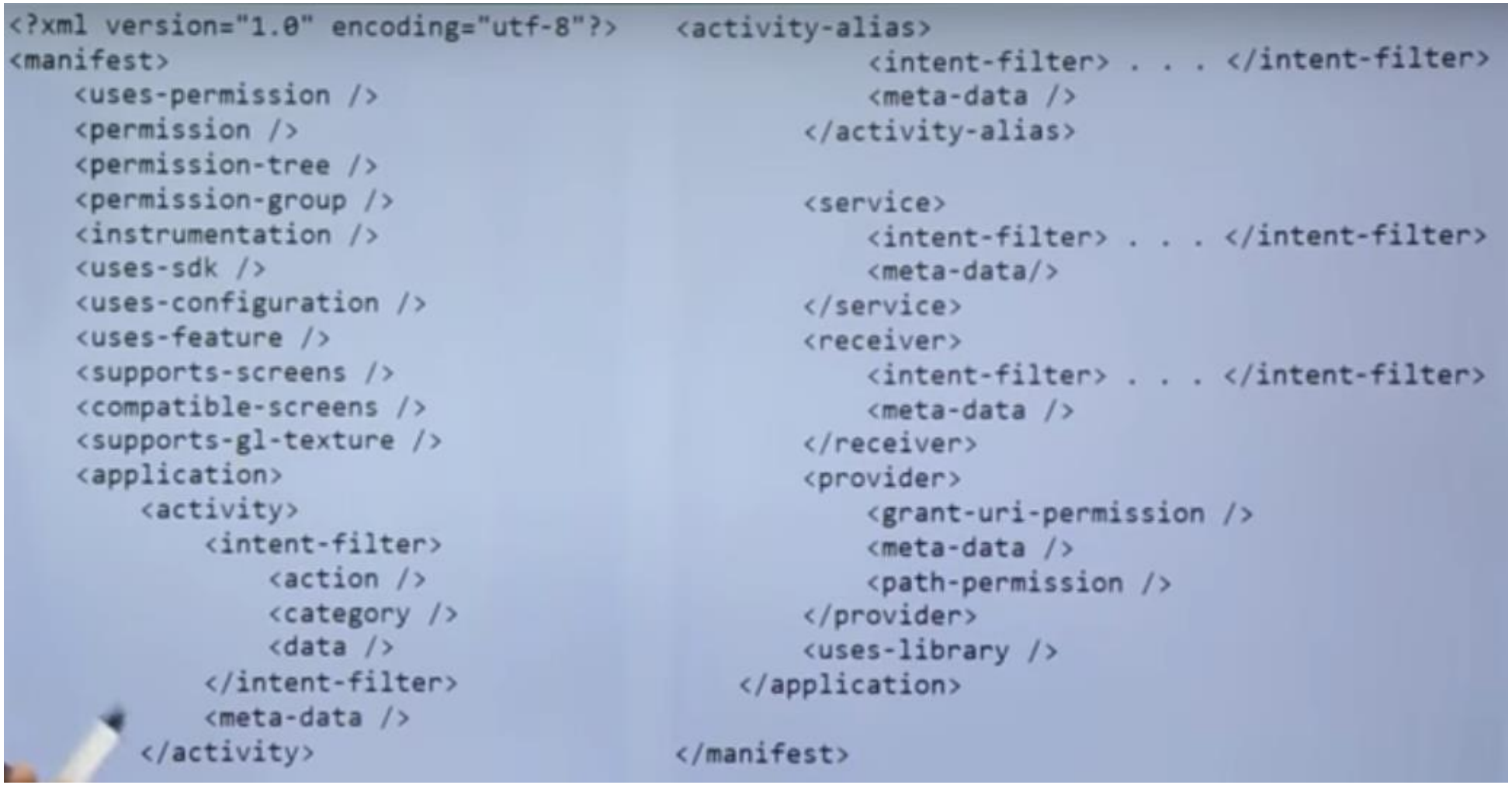
```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.javatpoint.hello"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="15" />
```

```
<application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/title_activity_main" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Few important points about AndroidManifest.xml

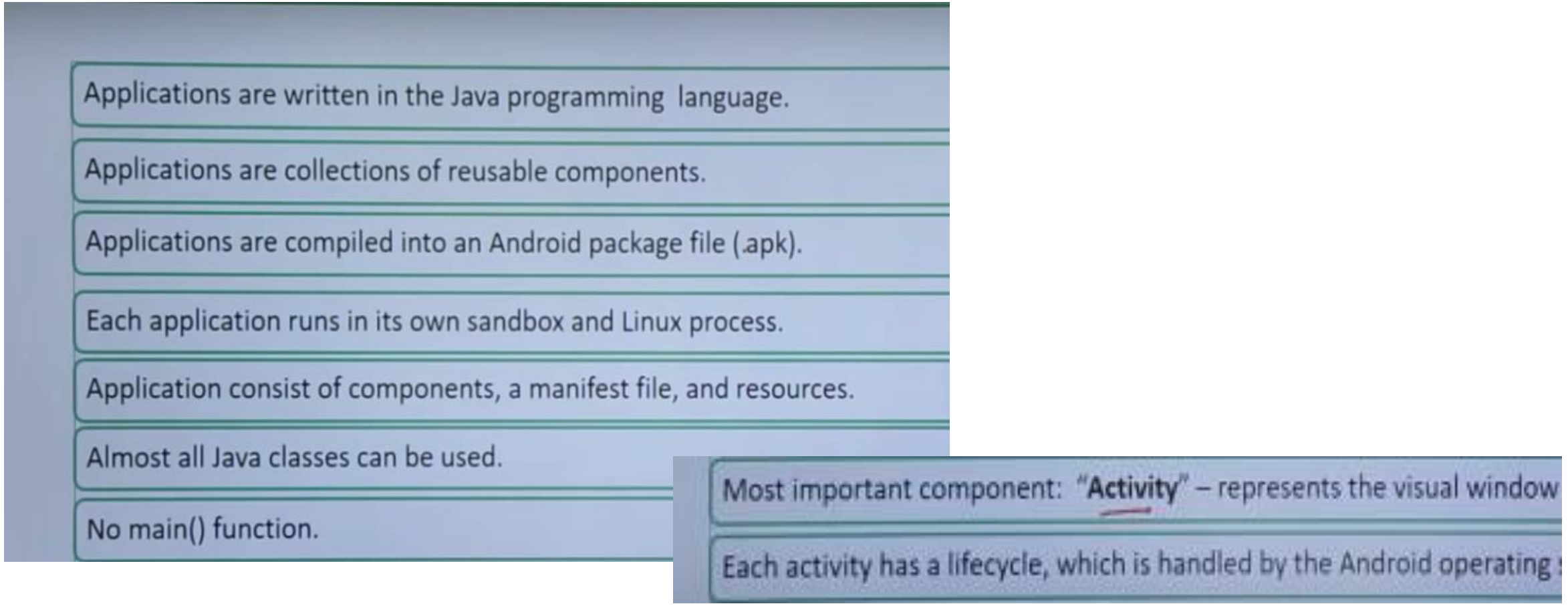
- Package is the Unique Identifier for any Android Application.
- xlmns stands for XML Namespace
- <application> :It’s the starting tag for any application.
- There is only one launcher in manifest file under activity tag and that launcher looks for the main activity and starts the Android Application with the main activity. Main activity in turns calls it’s UI.
- Mipmap folder is a resource and it’s under a res folder. ic_launcher is the icon.

Tags in a manifest file

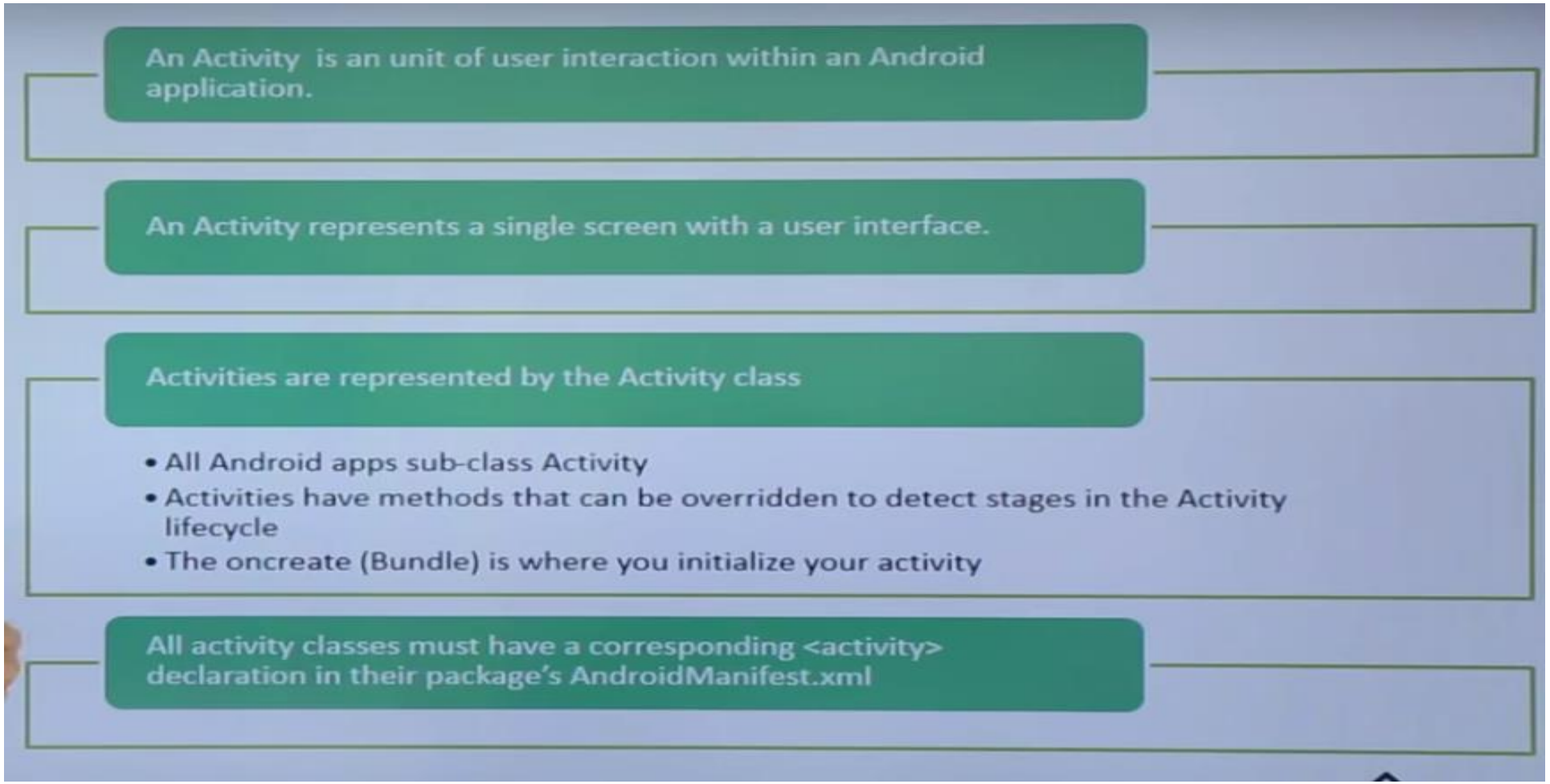


```
<?xml version="1.0" encoding="utf-8"?>
<manifest>
    <uses-permission />
    <permission />
    <permission-tree />
    <permission-group />
    <instrumentation />
    <uses-sdk />
    <uses-configuration />
    <uses-feature />
    <supports-screens />
    <compatible-screens />
    <supports-gl-texture />
    <application>
        <activity>
            <intent-filter>
                <action />
                <category />
                <data />
            </intent-filter>
            <meta-data />
        </activity>
        <activity-alias>
            <intent-filter> . . . </intent-filter>
            <meta-data />
        </activity-alias>

        <service>
            <intent-filter> . . . </intent-filter>
            <meta-data/>
        </service>
        <receiver>
            <intent-filter> . . . </intent-filter>
            <meta-data />
        </receiver>
        <provider>
            <grant-uri-permission />
            <meta-data />
            <path-permission />
        </provider>
        <uses-library />
    </application>

</manifest>
```

Few Important points about Android Studio Project Structure



Activity Life Cycle



Method	Description
onCreate	called when activity is first created.
onStart	called when activity is becoming visible to the user.
onResume	called when activity will start interacting with the user.
onPause	called when activity is not visible to the user.
onStop	called when activity is no longer visible to the user.
onRestart	called after your activity is stopped, prior to start.
onDestroy	called before the activity is destroyed.

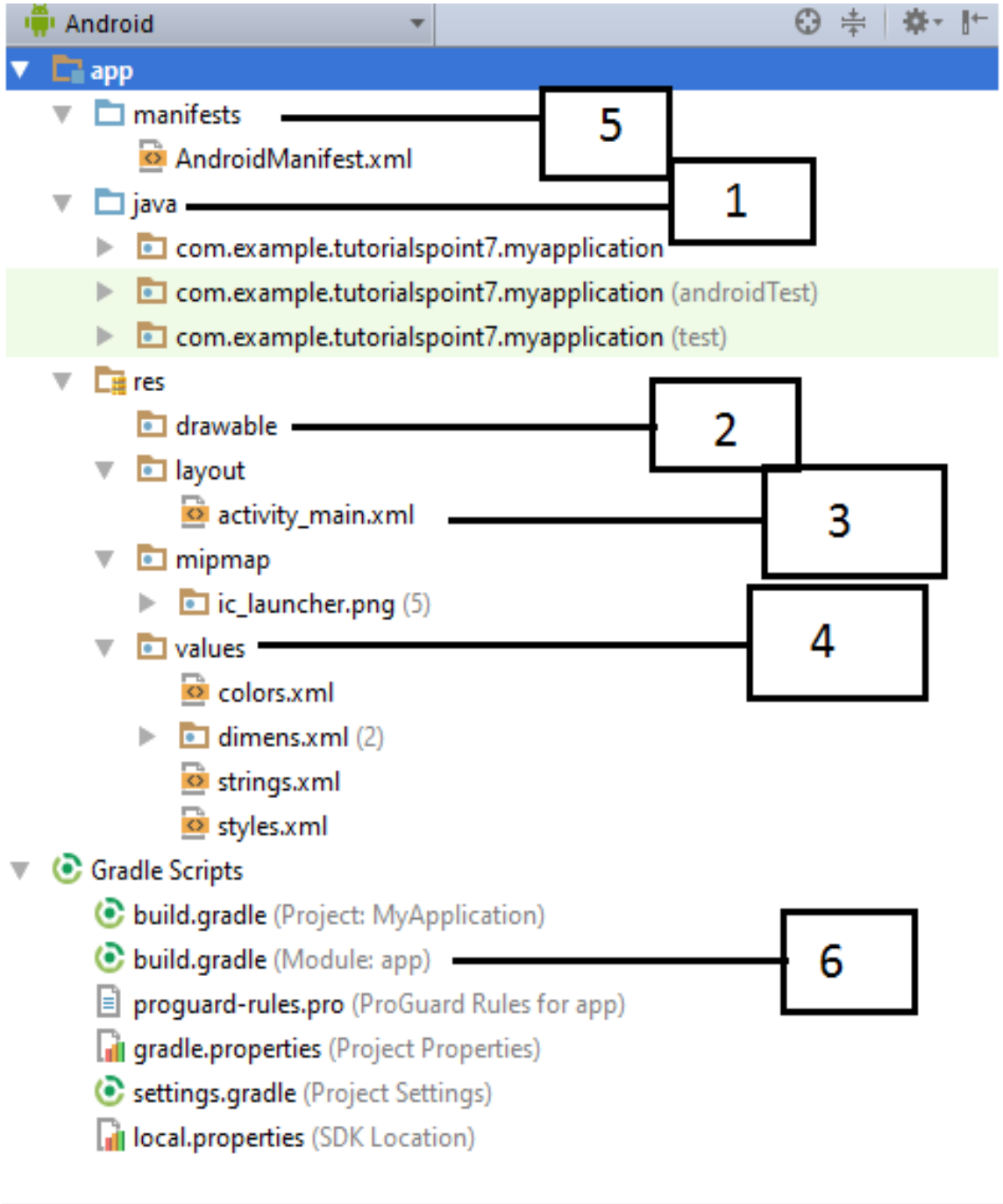
Hello World Example

The Main Activity File

The main activity code is a Java file **MainActivity.java**. This is the actual application file which ultimately gets converted to a Dalvik executable and runs your application. Following is the default code generated by the application wizard for *Hello World!* application –

```
package com.example.helloworld;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
   }
}
```

Here, *R.layout.activity_main* refers to the *activity_main.xml* file located in the *res/layout* folder. The *onCreate()* method is one of many methods that are figured when an activity is loaded.


Android
app
manifests
AndroidManifest.xml
java
com.example.tutorialspoint7.myapplication
com.example.tutorialspoint7.myapplication (androidTest)
com.example.tutorialspoint7.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

Sr.No.	Folder, File & Description
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

Manifest File

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.tutorialspoint7.myapplication">

   <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      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>
   </application>
</manifest>
```

The Strings File

The **strings.xml** file is located in the *res/values* folder and it contains all the text that your application uses. For example, the names of buttons, labels, default text, and similar types of strings go into this file. This file is responsible for their textual content. For example, a default strings file will look like as following file –

```
<resources>
   <string name="app_name">HelloWorld</string>
   <string name="hello_world">Hello world!</string>
   <string name="menu_settings">Settings</string>
   <string name="title_activity_main">MainActivity</string>
</resources>
```

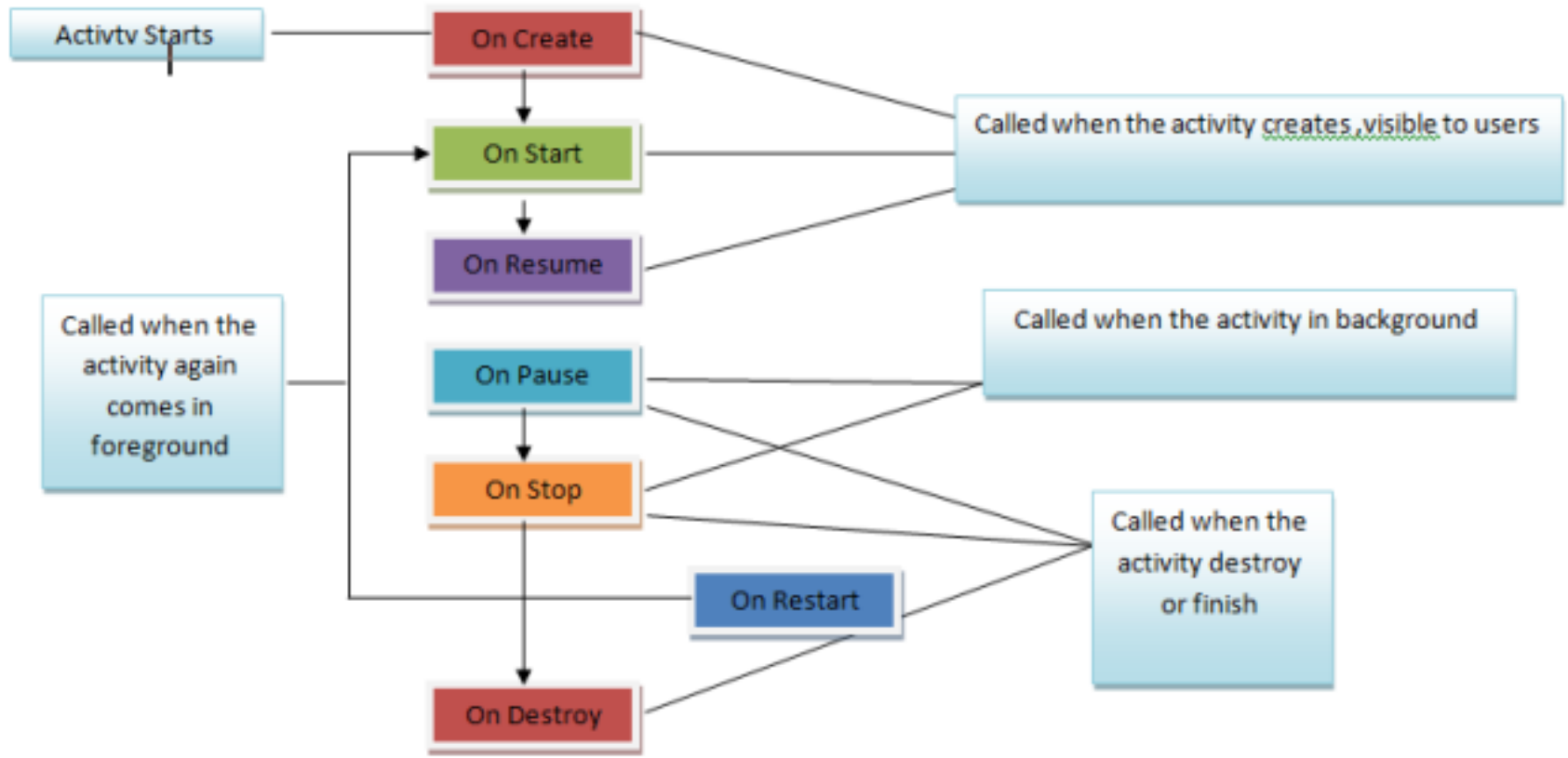
The Layout File

The **activity_main.xml** is a layout file available in *res/layout* directory, that is referenced by your application when building its interface. You will modify this file very frequently to change the layout of your application. For your "Hello World!" application, this file will have following content related to default layout −

```
<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" >

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_centerHorizontal="true"
      android:layout_centerVertical="true"
      android:padding="@dimen/padding_medium"
      android:text="@string/hello_world"
      tools:context=".MainActivity" />

</RelativeLayout>
```

`Activity` base class contains events that govern the life cycle of an activity.

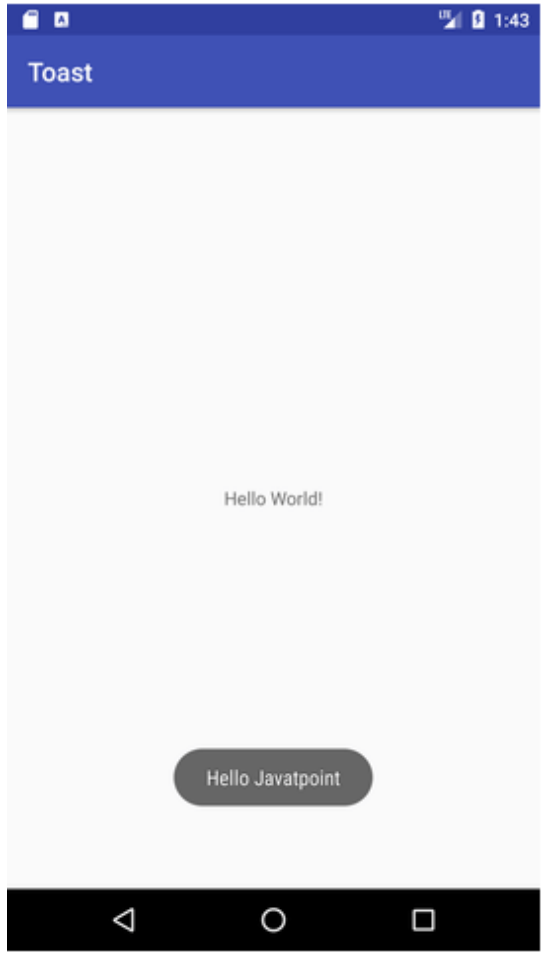
- `onCreate()`: Called when the activity is first created
- `onStart()`: Called when the activity becomes visible to the user
- `onResume()`: Called when the activity starts interacting with the user
- `onPause()`: Called when the current activity is being paused and the previous activity is being resumed
- `onStop()`: Called when the activity is no longer visible to the user
- `onDestroy()`: Called before the activity is destroyed by the system
- `onRestart()`: Called when the activity has been stopped and is restarting again



Android Toast

- Android Toast can be used to display information for the short period of time.
- A toast contains message to be displayed quickly and disappears after sometime.
- The android.widget.Toast class is the subclass of java.lang.Object class.

Example of Toast




```
//Displaying Toast with Hello Javatpoint message

    Toast.makeText(getApplicationContext(),"Hello Javatpoint",Toast.LENGTH_SHORT).show();
```

ANOTHER CODE:

```
Toast toast=Toast.makeText(getApplicationContext(),"Hello Javatpoint",Toast.LENGTH_SHORT);
toast.setMargin(50,50);
toast.show();
```

Here, getApplicationContext() method returns the instance of Context.

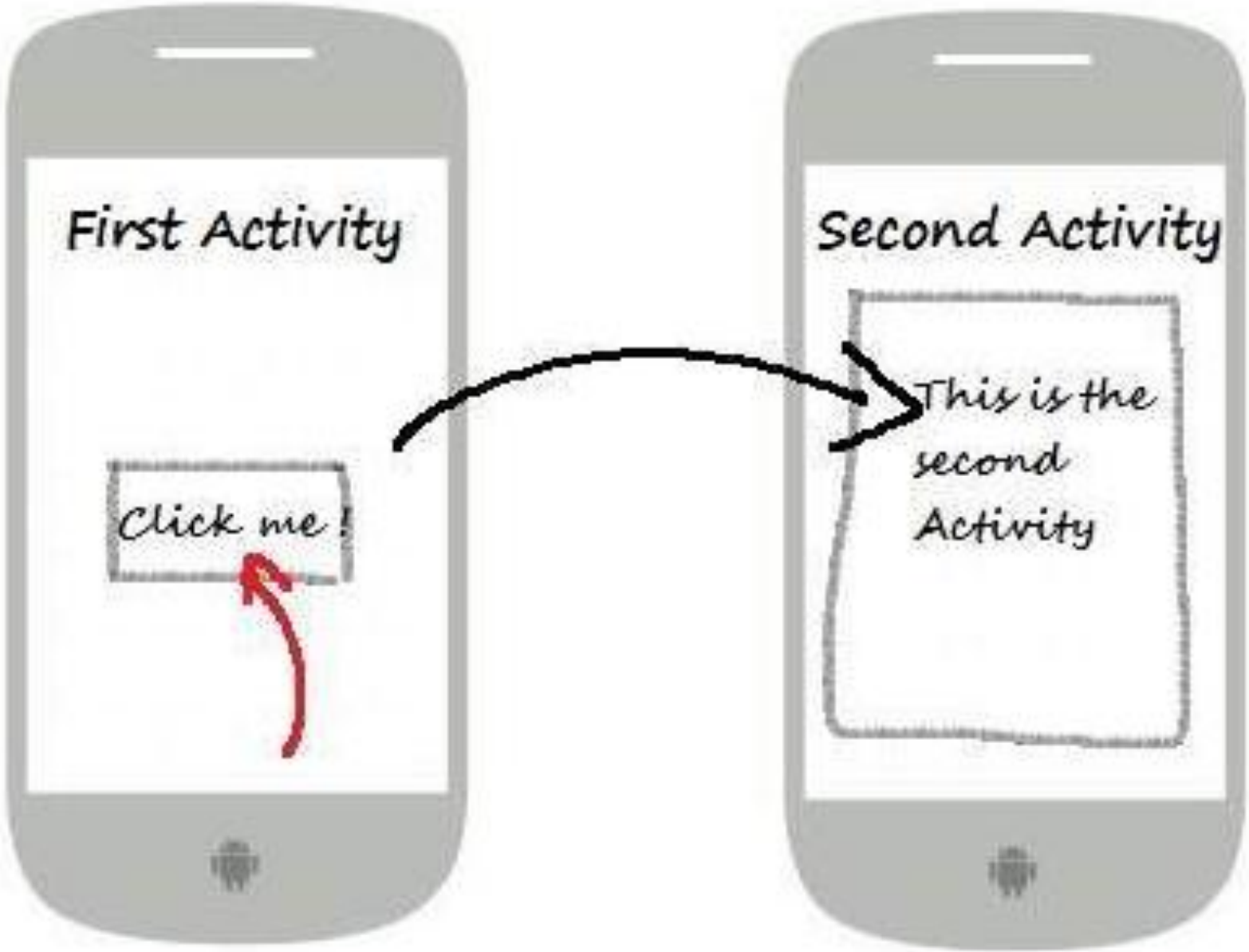
```java
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.widget.Toast;
public class LifeCycleActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Toast.makeText(LifeCycleActivity.this,"ON CREATE", Toast.LENGTH_SHORT).show();
    }
    @Override
    protected void onStart() {
       // TODO Auto-generated method stub
       super.onStart();
        Toast.makeText(LifeCycleActivity.this,"ON START", Toast.LENGTH_SHORT).show();
    }
    @Override
    protected void onResume() {
       // TODO Auto-generated method stub
       super.onResume();
        Toast.makeText(LifeCycleActivity.this,"ON RESUME", Toast.LENGTH_SHORT).show();
    }
    @Override
    protected void onPause() {
       // TODO Auto-generated method stub
       super.onPause();
        Toast.makeText(LifeCycleActivity.this,"ON PAUSE", Toast.LENGTH_SHORT).show();
    }
    @Override
    protected void onRestart() {
       // TODO Auto-generated method stub
       super.onRestart();
        Toast.makeText(LifeCycleActivity.this,"ON RESTART", Toast.LENGTH_SHORT).show();
    }
```

```
    @Override
    protected void onStop() {
        // TODO Auto-generated method stub
        super.onStop();
         Toast.makeText(LifeCycleActivity.this,"ON STOP", Toast.LENGTH_SHORT).show();
    }
    @Override
    protected void onDestroy() {
        // TODO Auto-generated method stub
        super.onDestroy();
         Toast.makeText(LifeCycleActivity.this,"ON DESTROY", Toast.LENGTH_SHORT).show();
    }
}
```

1. When you run the following code, you see when the application starts there are three toast messages that come after one another. First is **ON CREATE** , second is **ON START** and third is **ON RESUME**.
2. When you press the home button from the emulator, **ON PAUSE** and **ON STOP** methods call.
3. And when you open the application again **ON RESTART**, **ON START** and **ON RESUME** methods call.
4. And at last, when you press the back button from emulator, **ON PAUSE** , **ON STOP** and **ON DESTROY** method calls.

Explicit Intent

- **Android Explicit intent** specifies the component to be invoked from activity. In other words, we can call another activity in android by explicit intent.
- We can also pass the information from one activity to another using explicit intent.
- https://www.tutlane.com/tutorial/android/android-explicit-intents-with-examples


First Activity
Click me !
Second Activity
This is the second Activity

Implicit Intent

- In android, **Implicit** Intents won’t specify any name of the component to start, instead it declare an action to perform and it allow a component from other app to handle it.
- For example, by using implicit intents we can request another app to show the location details of user or etc.

```
<EditText
    android:id="@+id/editText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="60dp"
    android:ems="10"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.575"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginRight="8dp"
    android:layout_marginLeft="156dp"
    android:layout_marginTop="172dp"
    android:text="Visit"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/editText" />
```

```
public class MainActivity extends AppCompatActivity {

    Button button;
    EditText editText;

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

        button = findViewById(R.id.button);
        editText =  findViewById(R.id.editText);

        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String url=editText.getText().toString();
                Intent intent=new Intent(Intent.ACTION_VIEW, Uri.parse(url));
                startActivity(intent);
            }
        });
    }
```

URI(Uniform Resource Identifier)

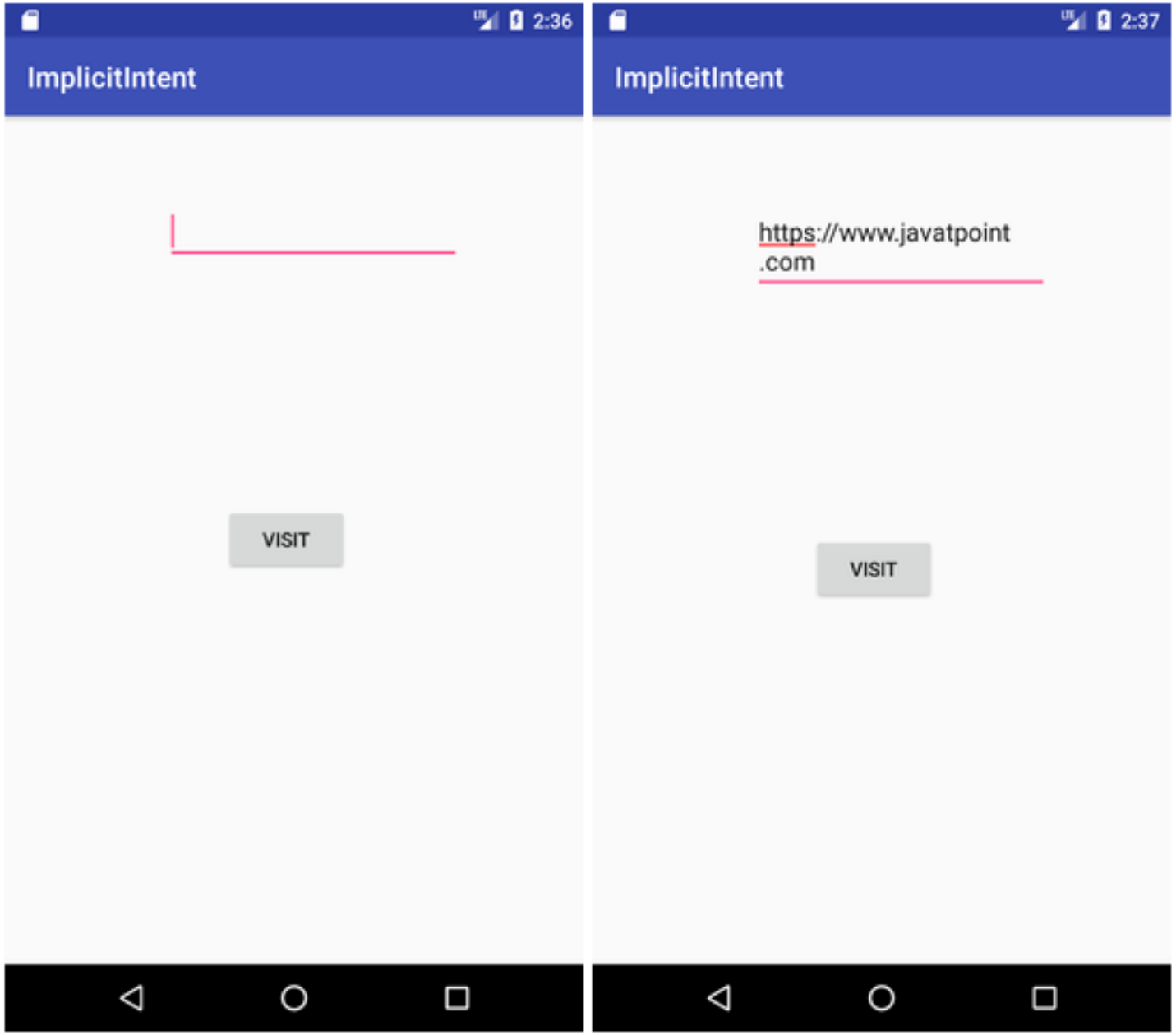
A Uri object is usually used to tell a Content Provider(helps data of other Applications to interact with our own application's data) what we want to access by reference.

The primary pieces of information in an intent are:

action -- The general action to be performed, such as ACTION_VIEW, ACTION_EDIT etc.

data -- The data to operate on.

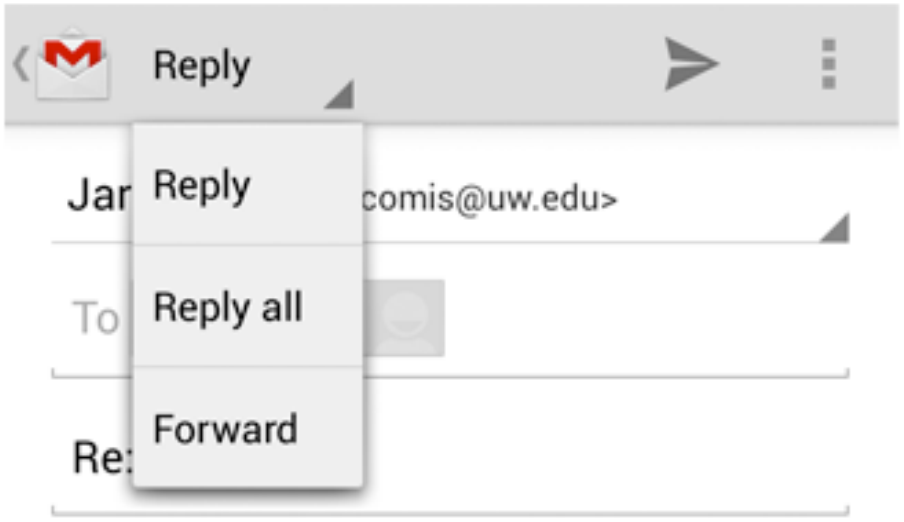
Output:

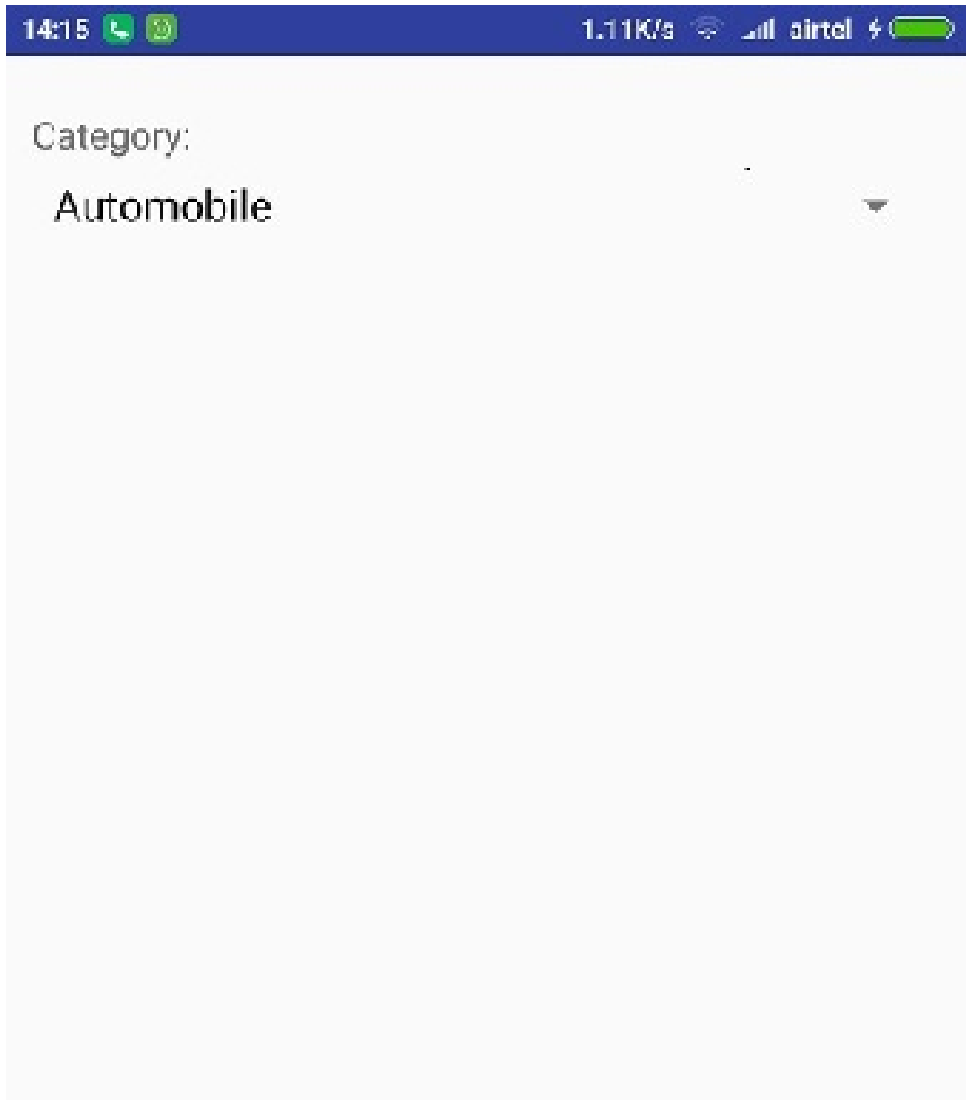

2:36
ImplicitIntent
VISIT
2:37
ImplicitIntent
https://www.javatpoint
.com
VISIT

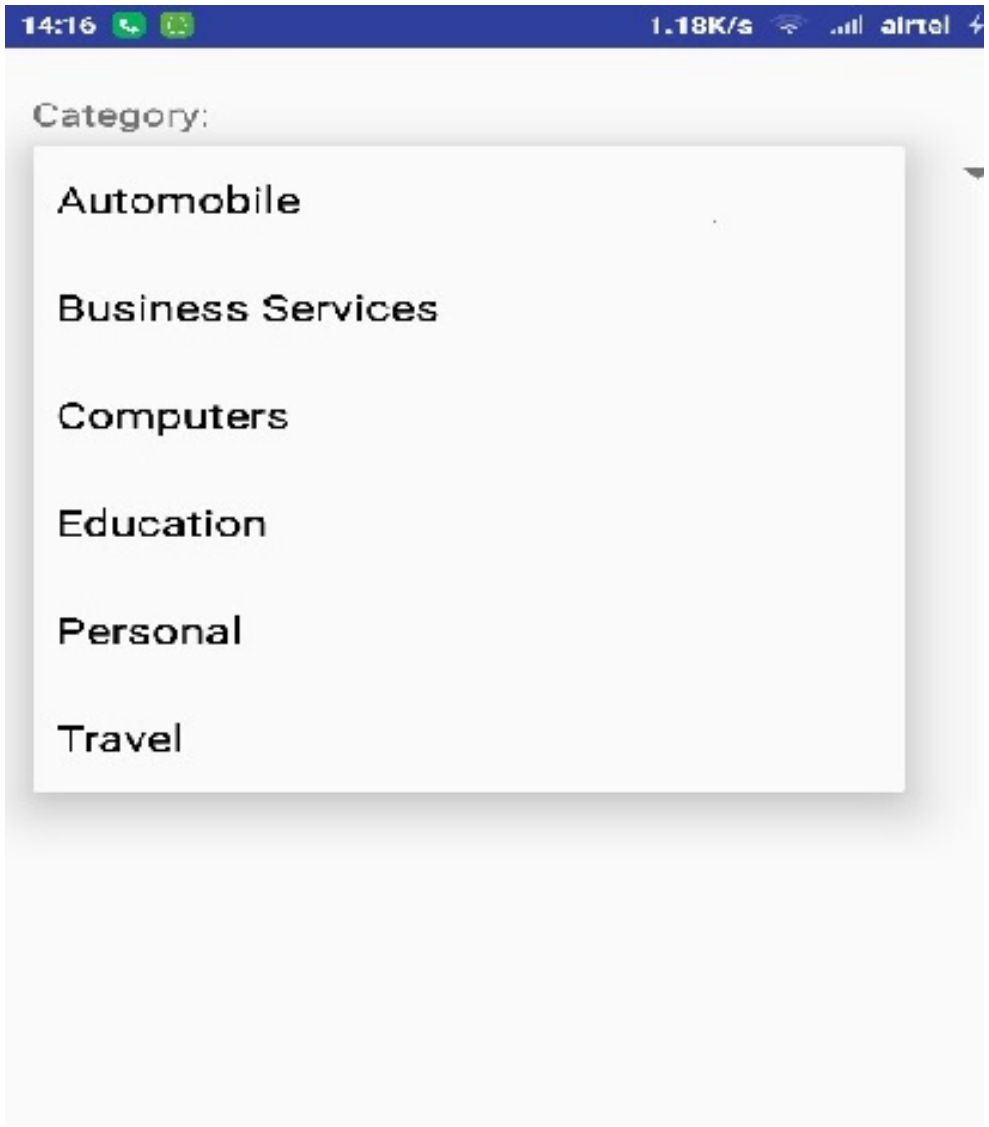
Android Spinner

> Spinner allows you to select an item from a drop down menu

For example. When you are using Gmail application you would get drop down menu as shown below, you need to select an item from a drop down menu.




14:15
1.11K/s
airtel
Category:
Automobile


14:16
1.18K/s
airtel
Category:
Automobile
Business Services
Computers
Education
Personal
Travel

ADAPTER

- An AdapterView is a group of widgets (aka view) components in Android that include the ListView, Spinner, and GridView.
- In general, these are the widgets that provide the selecting capability in the user interface

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<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:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin">

    <TextView

        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="14dp"
        android:text="Welcome to my Gallery!"
        android:textAppearance="?android:attr/textAppearanceLarge" />
```

```
<Spinner

    android:id="@+id/spinner1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textView1"
    android:entries="@array/imgarray"
    android:layout_centerHorizontal="true" />

  <ImageView

    android:id="@+id/imageView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/spinner1"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="28dp"
    android:src="@drawable/ap" />

</RelativeLayout>
```

strings.xml

```
<resources>

    <string name="app_name">Spinner</string>

    <string-array name="imgarray">

        <item> image1 </item>
        <item> image2 </item>
        <item> image3 </item>
      </string-array>
</resources>
```

```
public class MainActivity extends Activity {
    Spinner sp1;
    ImageView iv1;
    @Override
    protected void onCreate(Bundle savedInstanceState)
{
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        sp1 = (Spinner) findViewById(R.id.spinner1);
        iv1 = (ImageView) findViewById(R.id.imageView1);
        sp1.setOnItemSelectedListener(new OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> arg0, View arg1,
                                    int arg2, long arg3) {
                if(arg2==0)
```

```
                {
                  iv1.setImageResource(R.drawable.ap);
                }

                else if(arg2==1)

                {

                   iv1.setImageResource(R.drawable.aq);
                }

                else
               {
                  iv1.setImageResource(R.drawable.as);
               }
     }

    });
      }}
```

onItemSelected

added in API level 1

```
public abstract void onItemSelected (AdapterView<?> parent,
                View view,
                int position,
                long id)
```

Callback method to be invoked when an item in this view has been selected. This callback is invoked only when the newly selected position is different from the previously selected position or if there was no selected item.

Implementers can call getItemAtPosition(position) if they need to access the data associated with the selected item.

Parameters	
`parent`	**`AdapterView`**: The AdapterView where the selection happened
`view`	**`View`**: The view within the AdapterView that was clicked
`position`	**`int`**: The position of the view in the adapter
`id`	**`long`**: The row id of the item that is selected

OUTPUT

42% 9:48 PM

Welcome to my Gallery!

image1

42% 9:48 PM

Welcome to my Gallery!

image2

42% 9:49 PM

Welcome to my Gallery!

image3