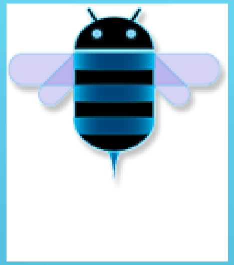
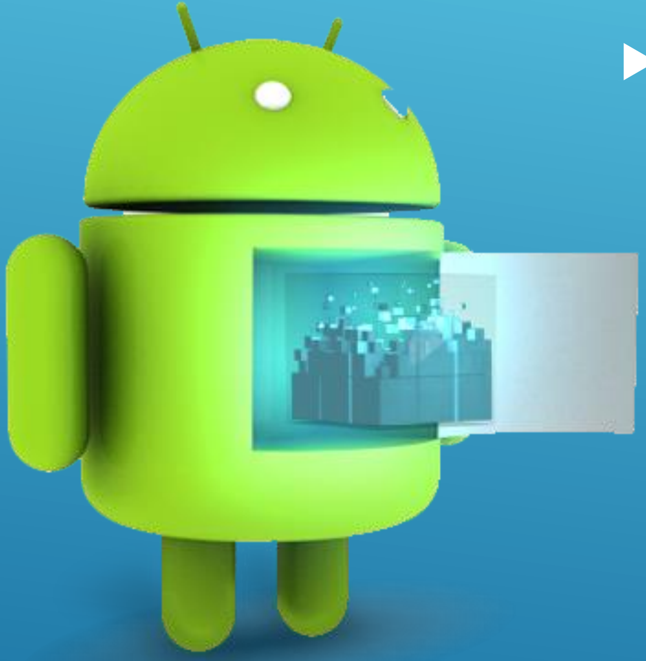


# CHAPTER 3: SYSTEM ARCHITECTURE

## OUTLINE

- Android Architecture: An **Overview**
- Android **Dalvik** Java **Virtual Machine**
- Android Components: **Activities**
- Android Components: **Intents**
- Android Components: **Services**
- Android Components: **Content Providers**
- Android Application **Distribution** and **Markets**



- **Android** is a *Linux-based* <u>platform </u>for *mobile devices …*
  - *Operating System*
  - *Middleware*
  - *Applications*
  - *Software Development Kit* **(SDK)**

## ANDROID WHAT?

Whic l of **de** . (exc )

SMARTPHONES

TABLETS

EREADERS

ANDROID TV

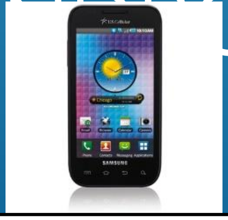
GOOGLE GLASSES

?

## ANDROID — WHAT?

SMART FRIDGE

ANDROID MICROWAVE



SMARTPHONES

TABLETS

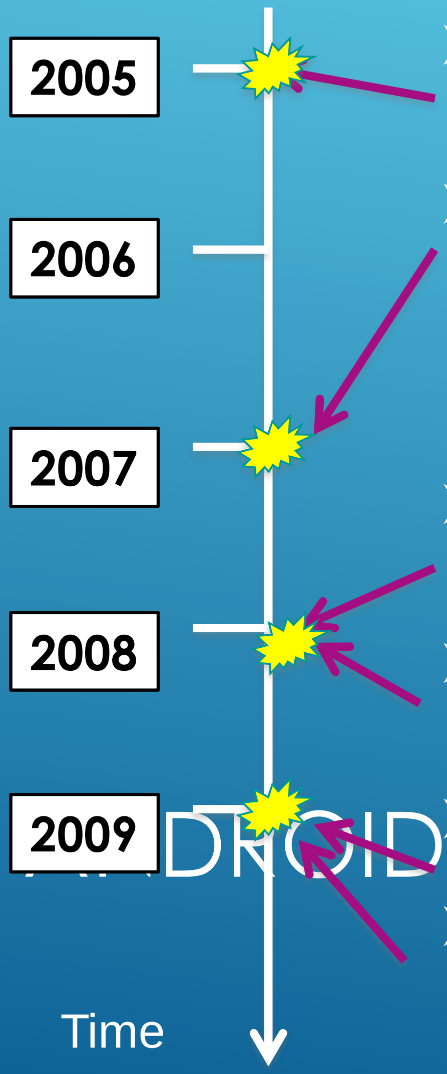
EREADERS

ANDROID TV

GOOGLE GLASSES

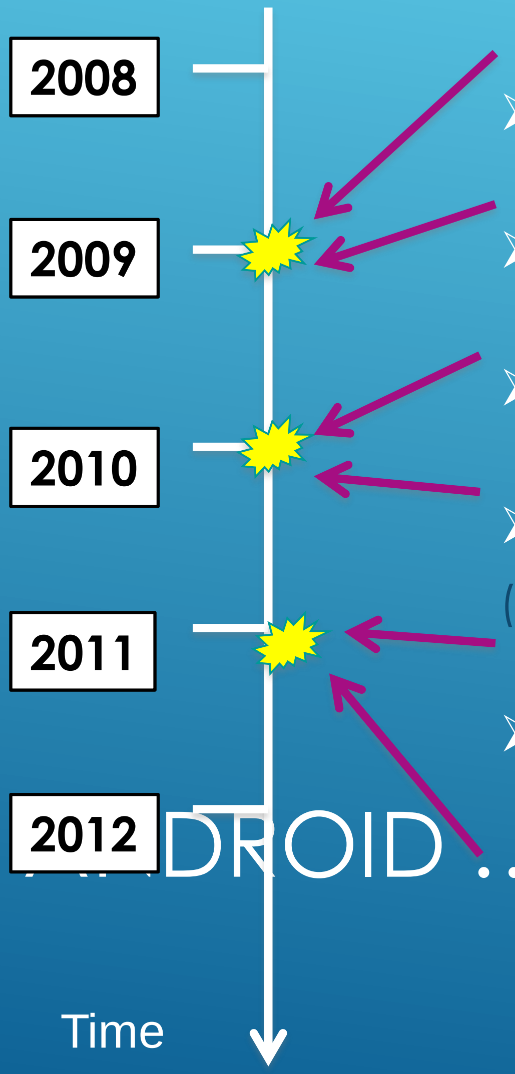
?

# ANDROID … WHEN?



- **Google** buys Android from the **Android Inch**
- Open Handset Alliance (**OHA**) created for open standards for mobile devices. Partners of OHA: Google, Motorola, Samsung, Vodafone, T-Mobile, etc
- Android 1.0 Released
- The first Android smartphone: G1 HTC-Dream
- Android **1.1** Released
- Android **1.5 (CupCake)** Released

# ANDROID … WHEN?

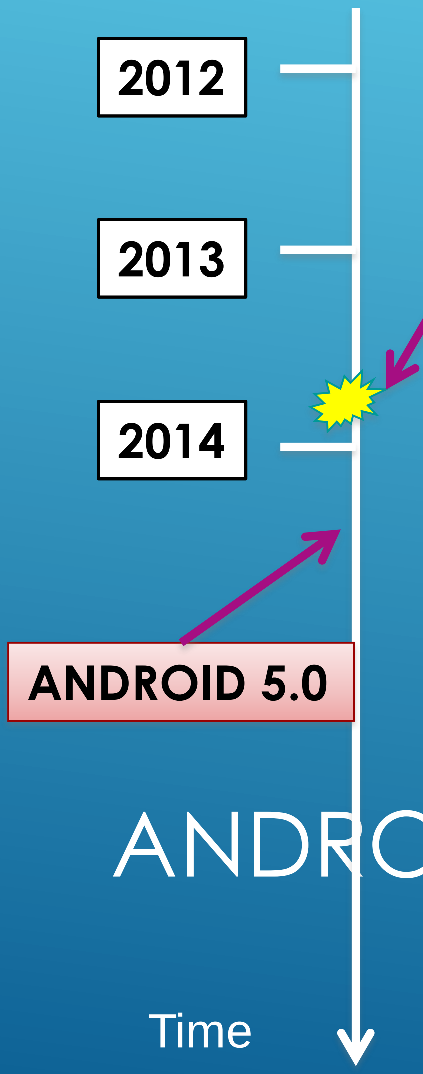


- Android **1.6** (**Donut**) Released
- Android **2.0** (**Eclair**) Released
- Android **2.2** (**Froyo**) Released
- Android **2.3** (**Gingerbread**) Released
- Android **3.0** (**Honeycomb**) Released

(First version for devices with larger screens such as tablets)

- Android **4.0** (**Ice-Cream Sandwich**) Released. ges the 3.x tab centric design and the v2.x phone based design into a single version.)

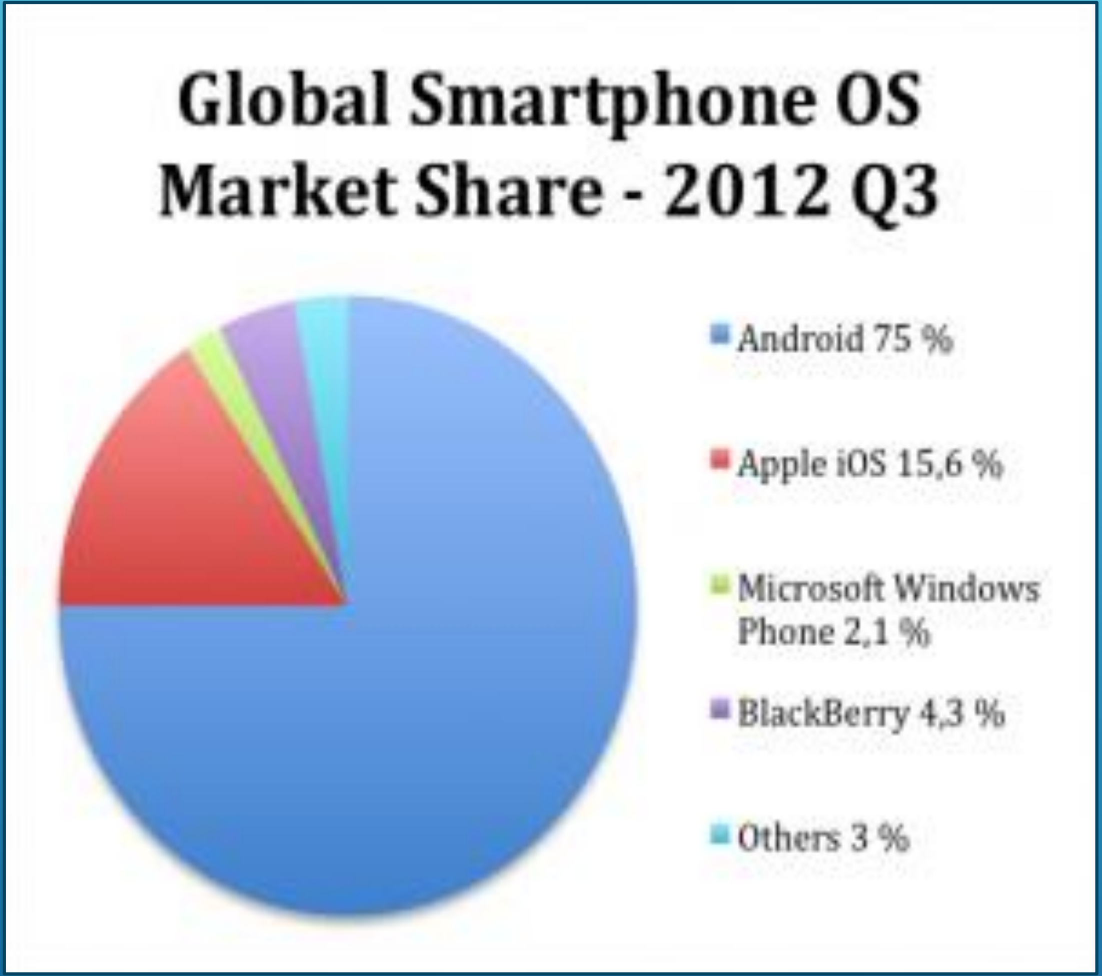
## ANDROID ... WHEN?

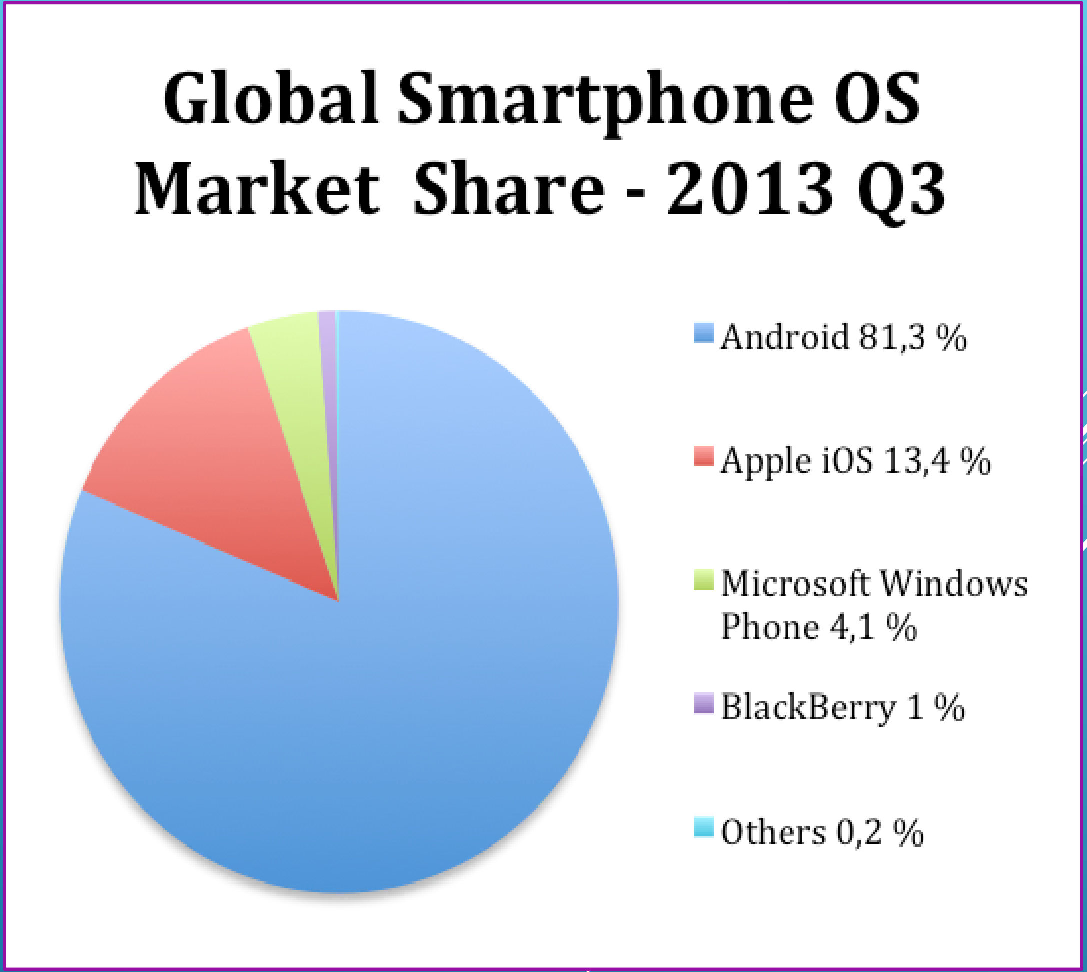


- Android **4.4** (**Kitkat**) Released
  - Wireless printing capability
  - Ability for applications to use "immersive mode"
  - Performance optimization
  - New experimental runtime virtual machine, ART…

**API Level 19 (Android 4.4):**

- Support to new embedded sensors (e.g. STEP_DETECTOR)
- Adaptive video playback functionalities
- Read and write SMS and MMS messages (managing default text messaging client)





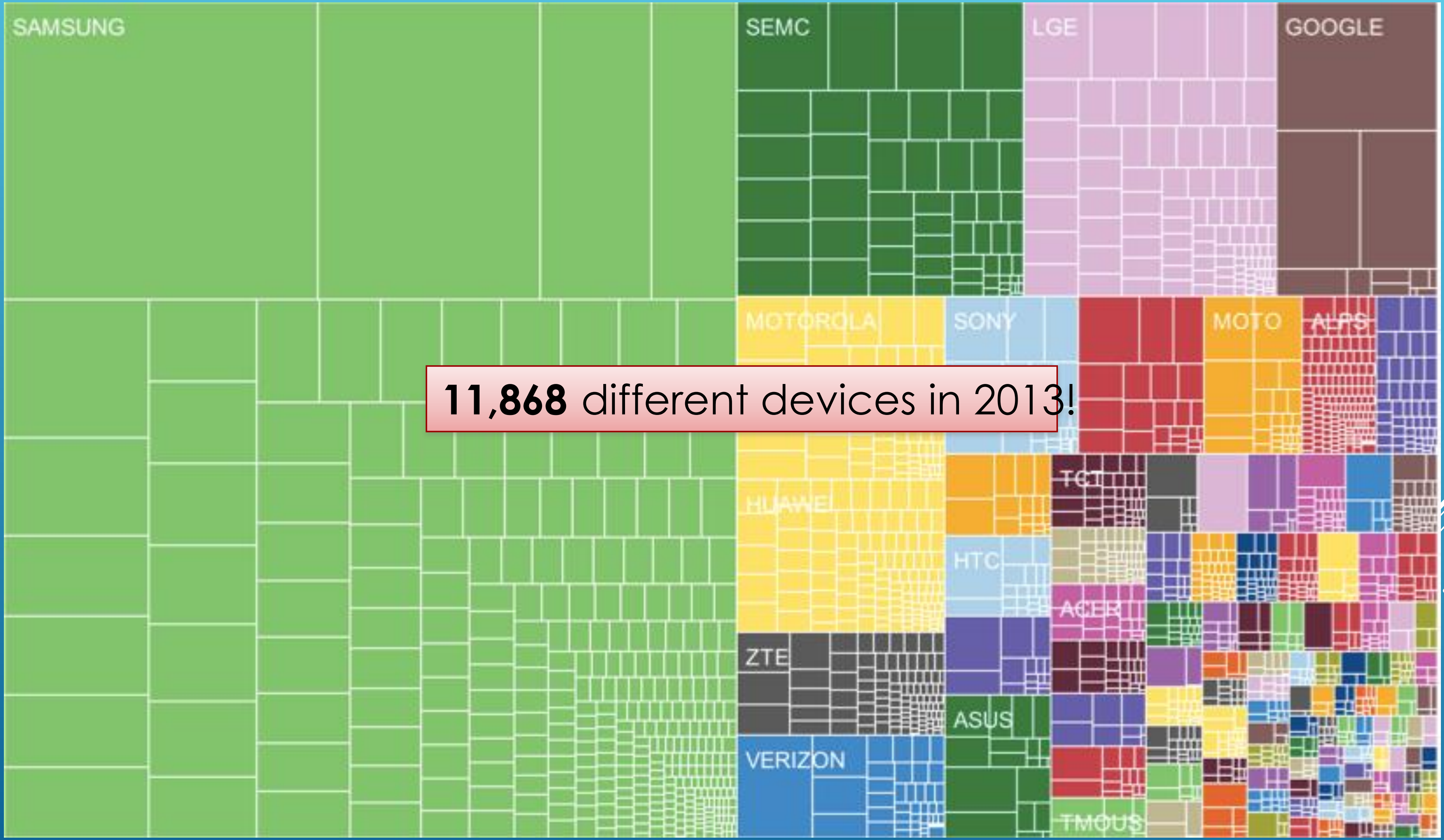
**2012** Market Share

www.gartner.com

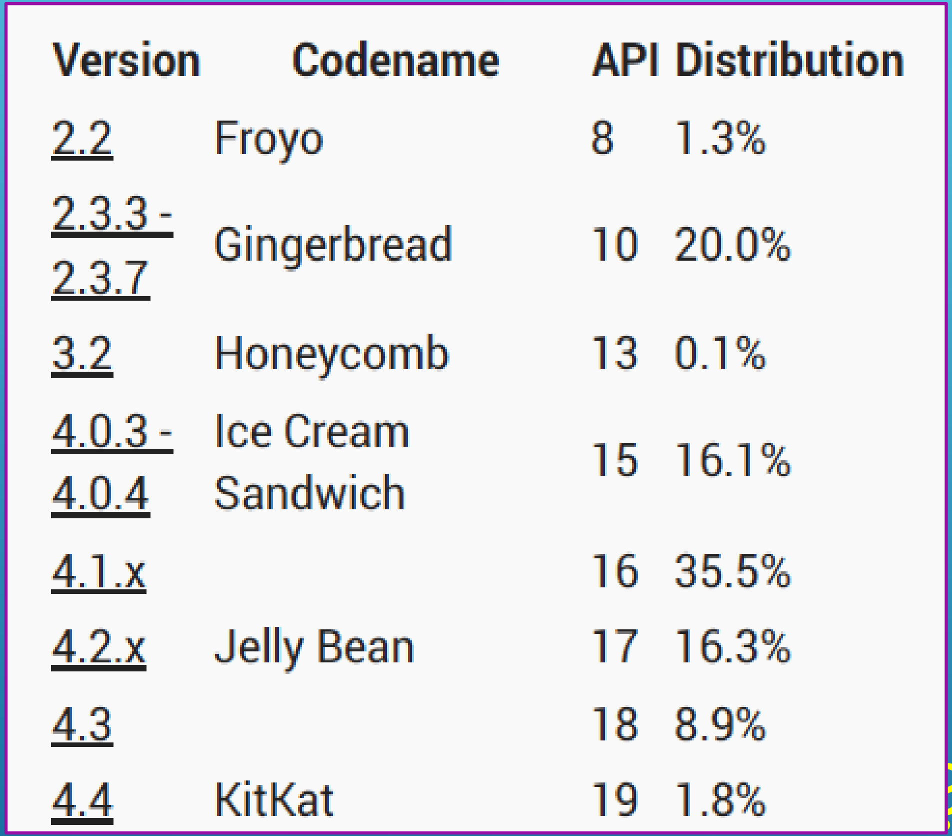
**2013** Market Share

AN

http://opensignal.com/reports/fragmentation-2013/

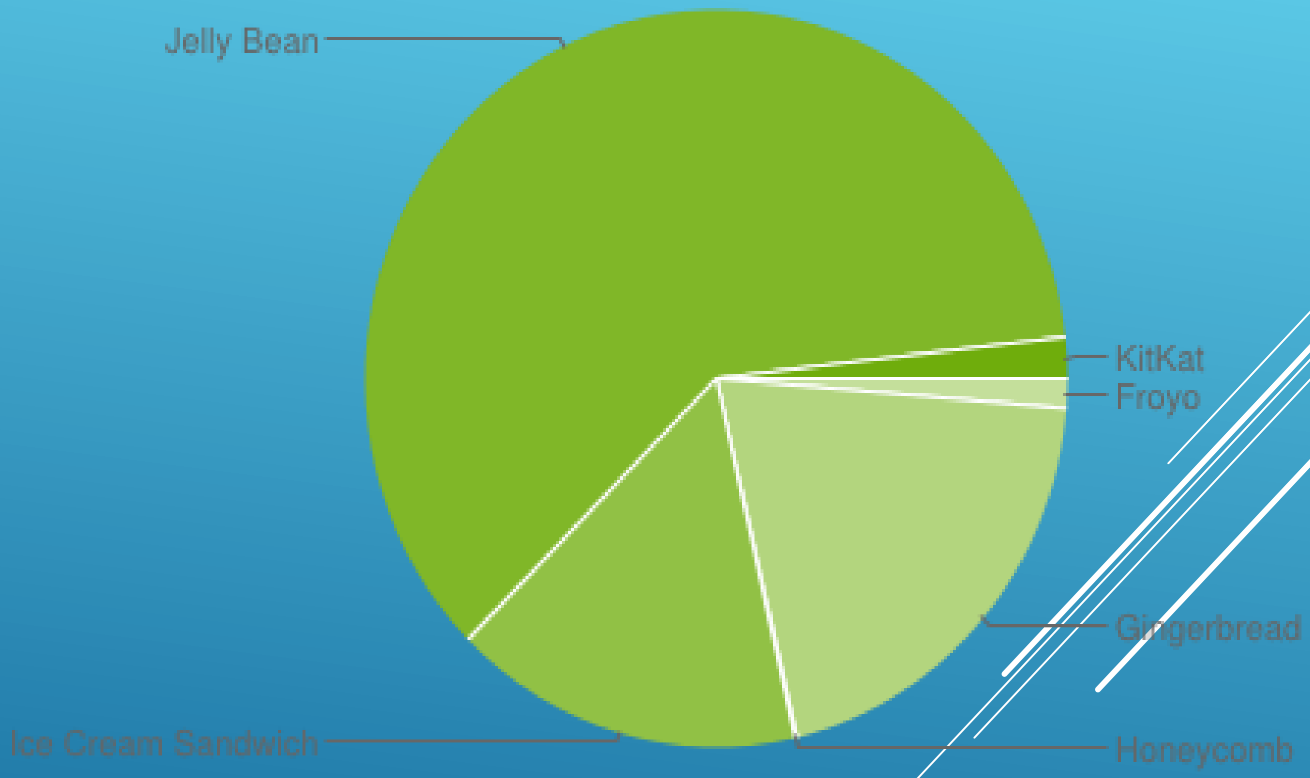


**11,868** different devices in 2013!

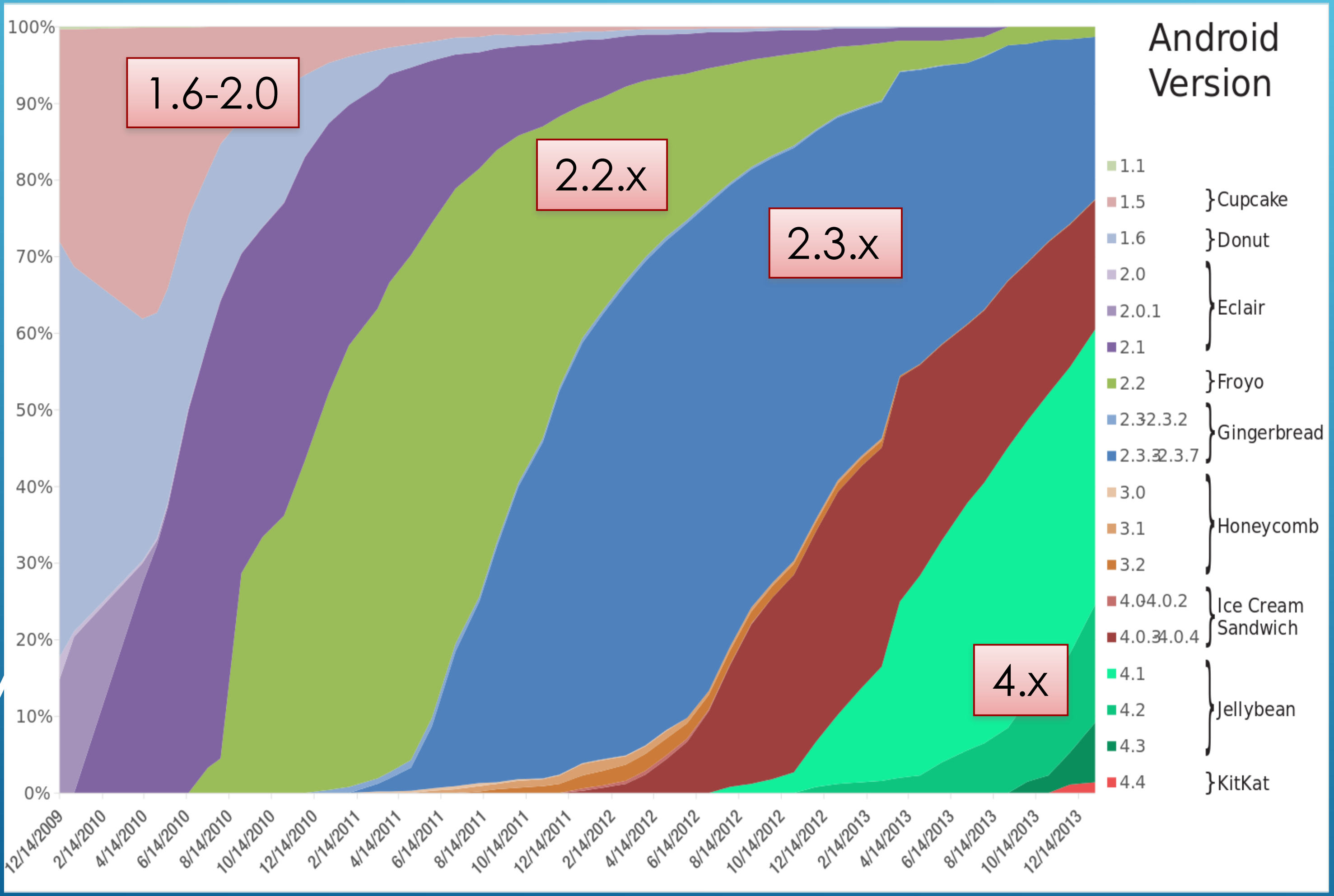


| Version | Codename | API | Distribution |
|---|---|---|---|
| 2.2 | Froyo | 8 | 1.3% |
| 2.3.3 - 2.3.7 | Gingerbread | 10 | 20.0% |
| 3.2 | Honeycomb | 13 | 0.1% |
| 4.0.3 - 4.0.4 | Ice Cream Sandwich | 15 | 16.1% |
| 4.1.x | | 16 | 35.5% |
| 4.2.x | Jelly Bean | 17 | 16.3% |
| 4.3 | | 18 | 8.9% |
| 4.4 | KitKat | 19 | 1.8% |

Updated at February 2014



http://developer.android.com/about/dashboards/index.html http://en.wikipedia.org/wiki/Android_version_history

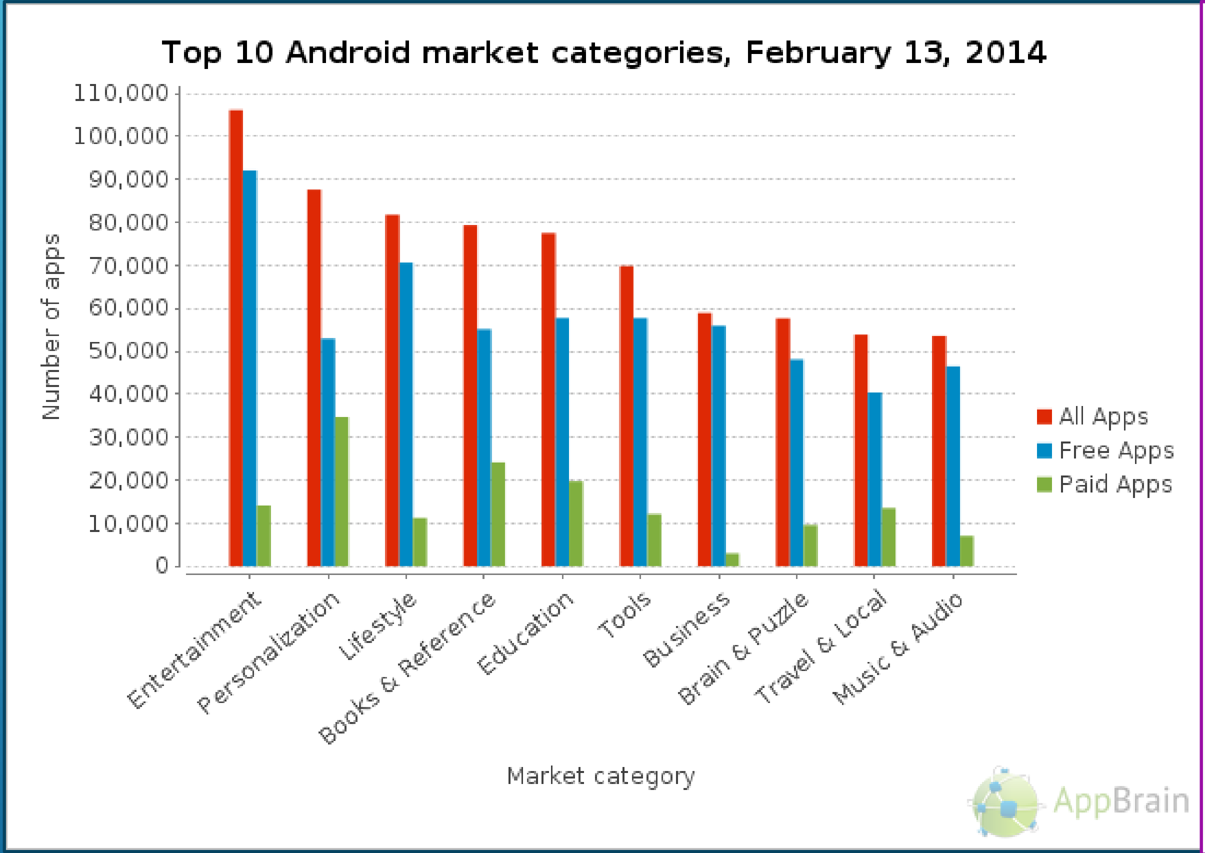


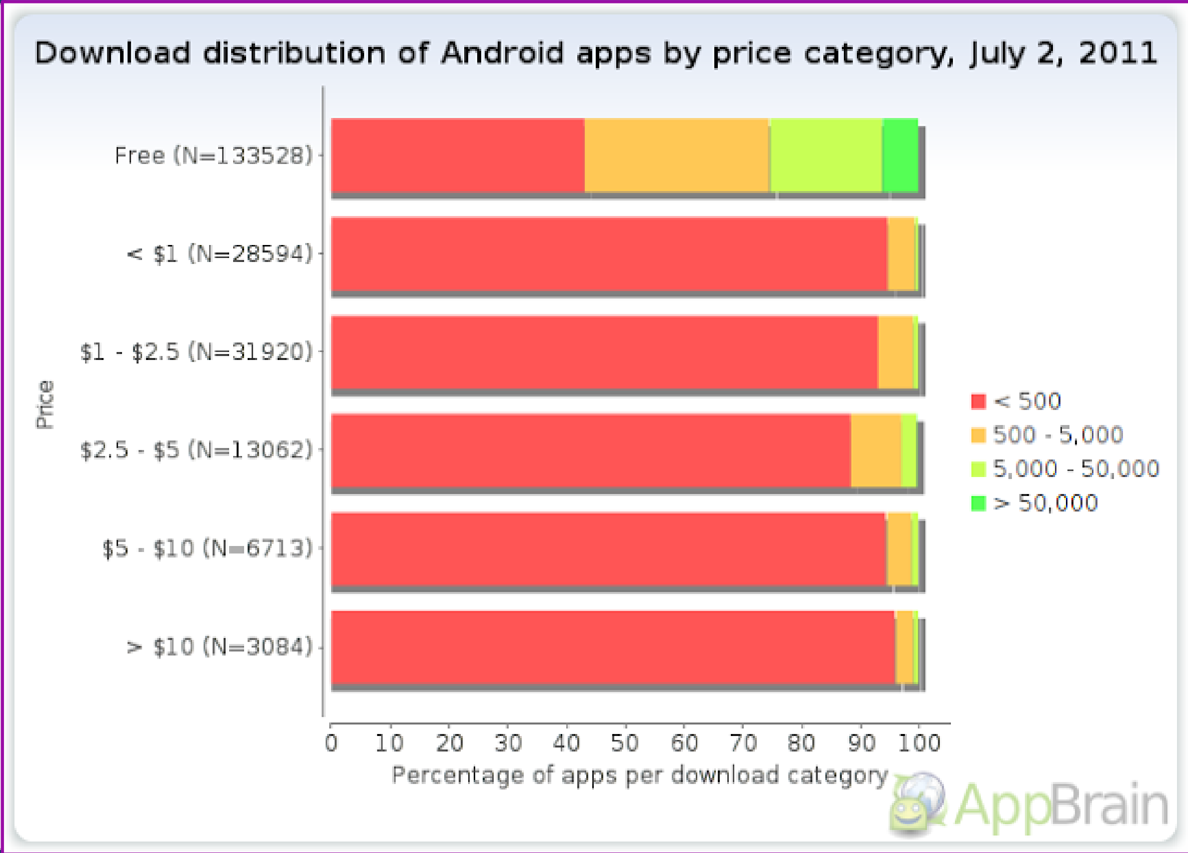
## ANDROID VERSION HISTORY AND POPULARITY

## (2009-2013)

## ANDROID APP CATEGORIES

## ANDROID APP PRICE

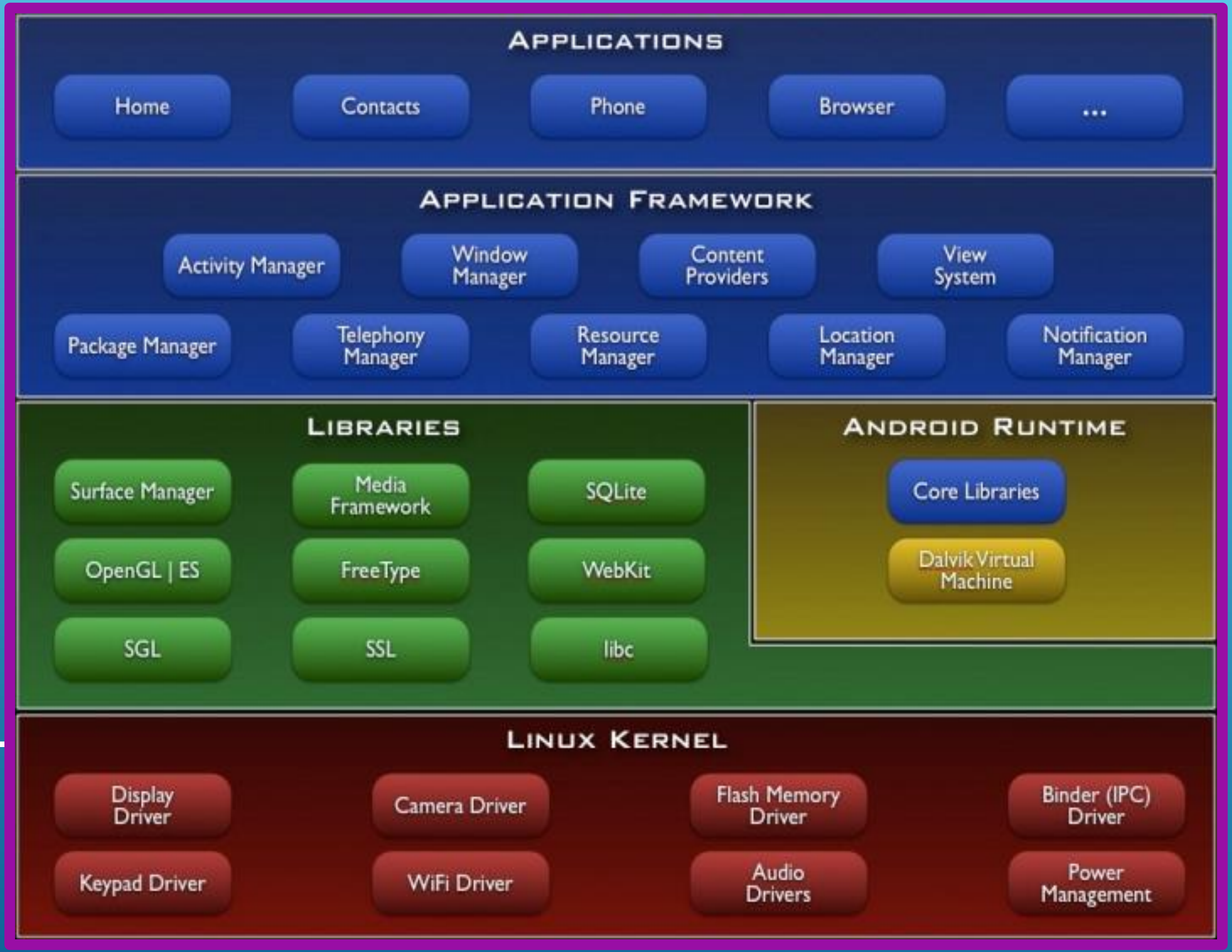




http://www.appbrain.com/stats/android-market-app-categories

http://www.onlinemarketing-trends.com/2011/07/android-marketplace-top-5-statistics.html

TH



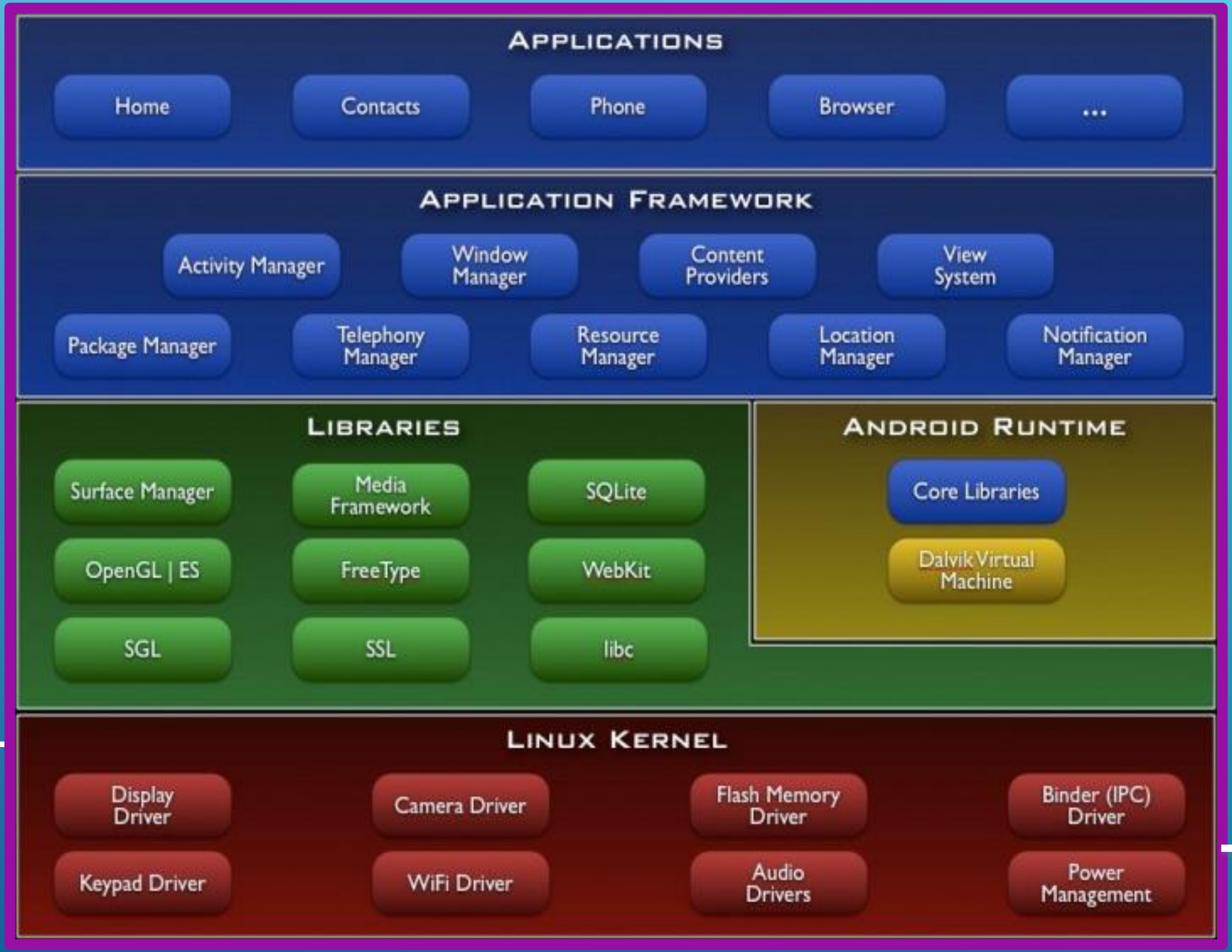
}

**Stack**
*Architecture*

**Open Source Architecture**
(Apache/MIT License v. 2.0)

*Business-friendly License*

TH

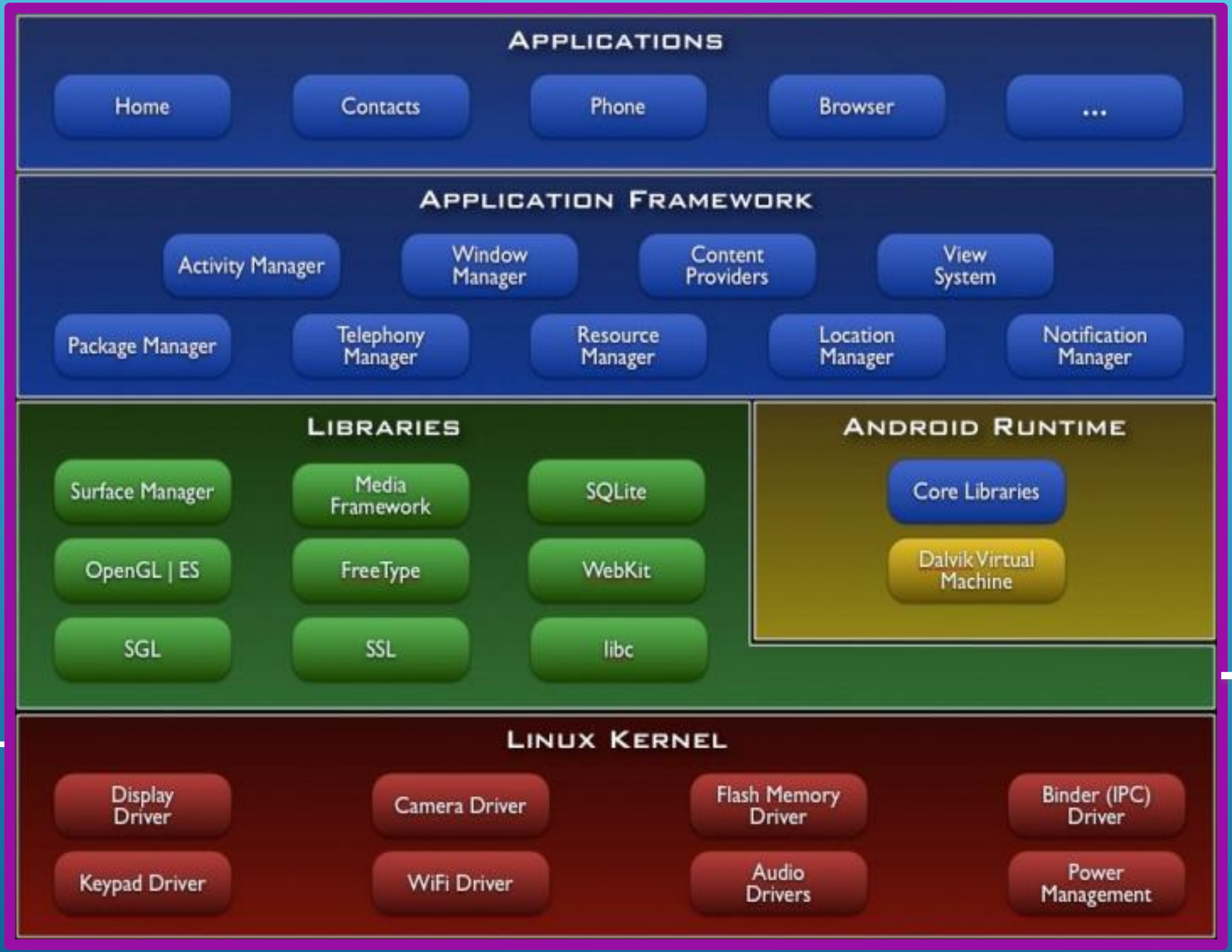


Built on top of **Linux kernel** (v. 2.6-3.4)

Advantages:

- **Portability** (i.e. easy to compile on different hardware architectures)
- **Security** (e.g. secure multi-process environment)
- **Power** Management

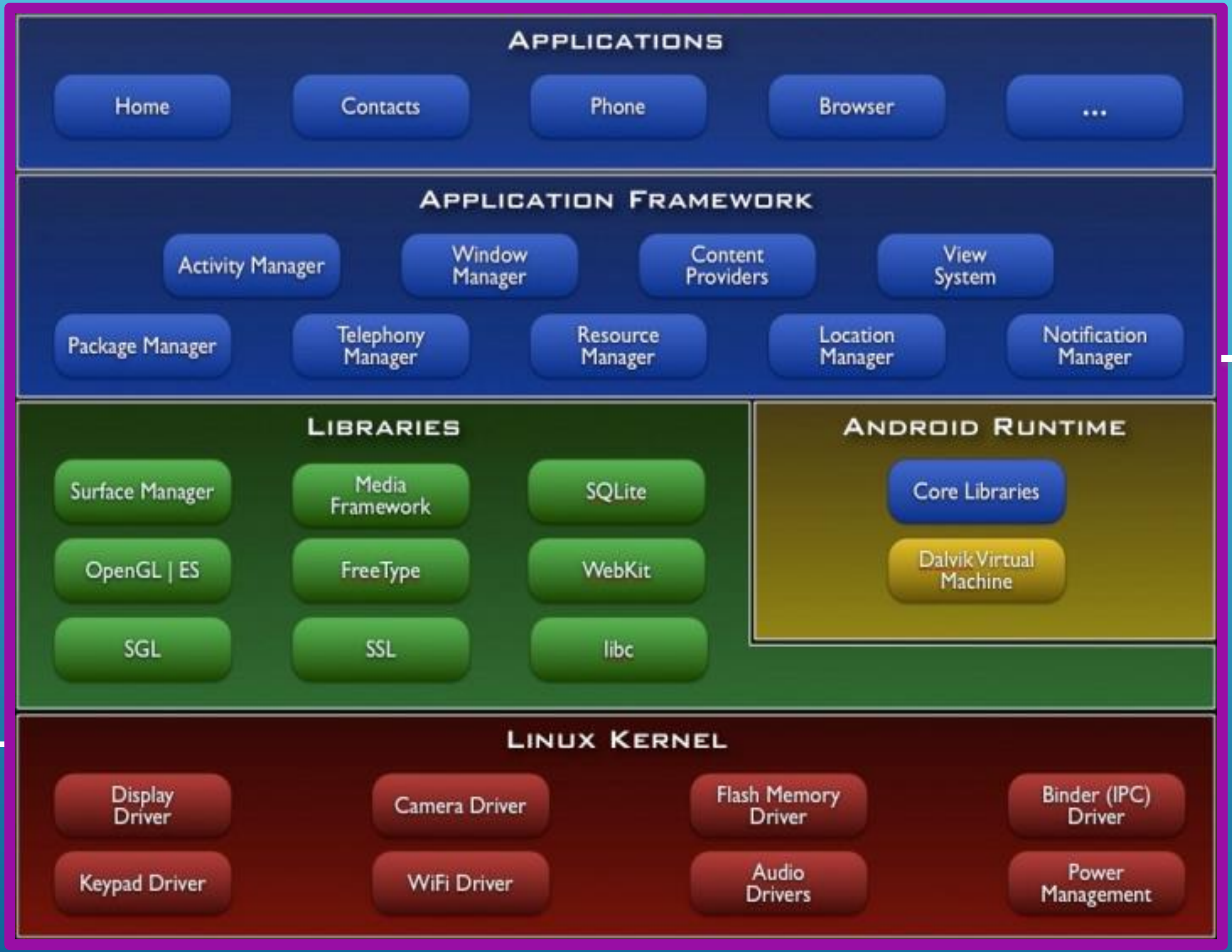
TH



## Native **Libraries**
(C/C++ code)

- **Graphics** (Surface Manager)
- **Multimedia** (Media Framework)
- **Database DBMS** (SQLite)
- **Font Management** (FreeType)
- **WebKit**
- **C libraries** (Bionic)
- **….**

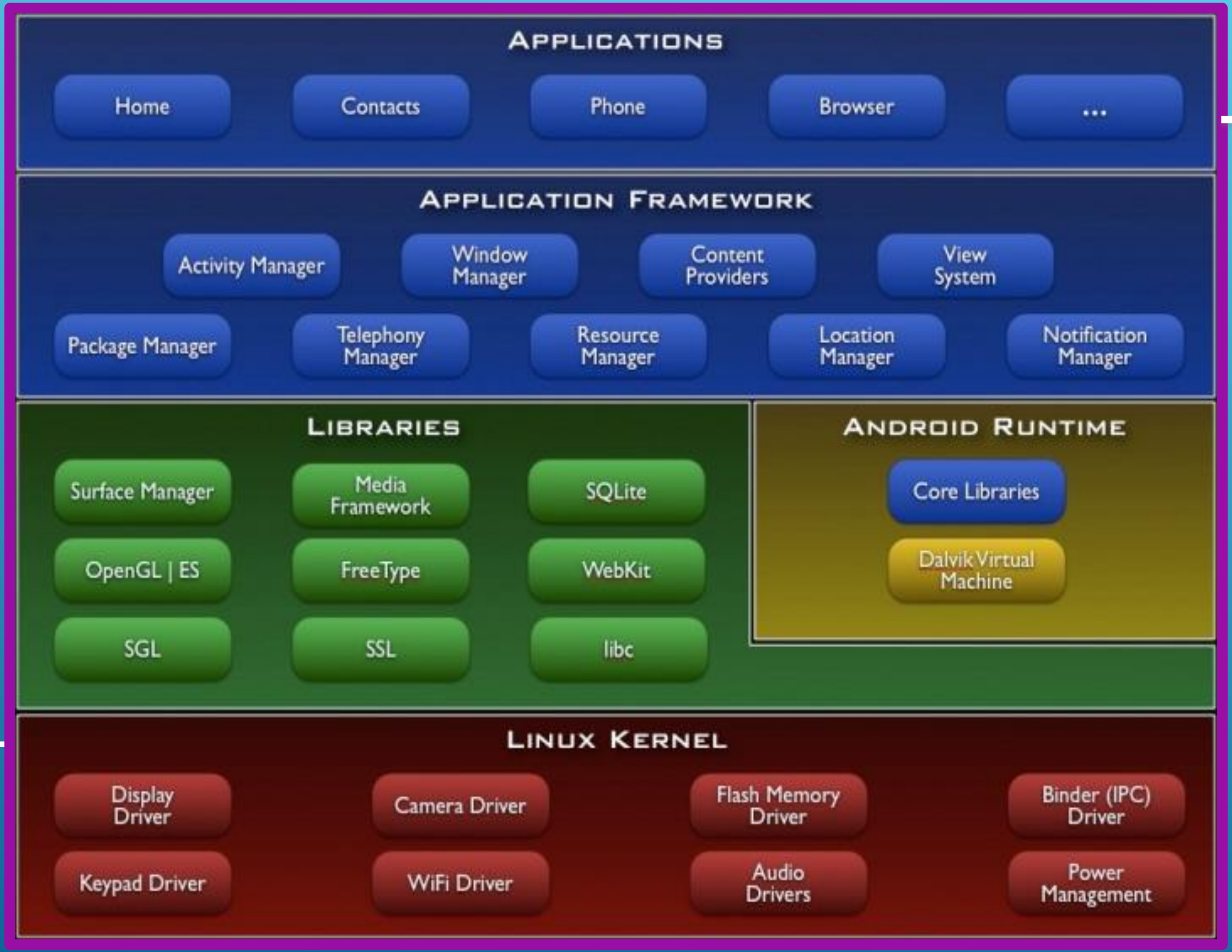
TH



## Application Libraries
(Core Components of Android)

- **Activity Manager**
- **Packet Manager**
- **Telephony Manager**
- **Location Manager**
- **Contents Provider**
- **Notification Manager**
- **….**

TH

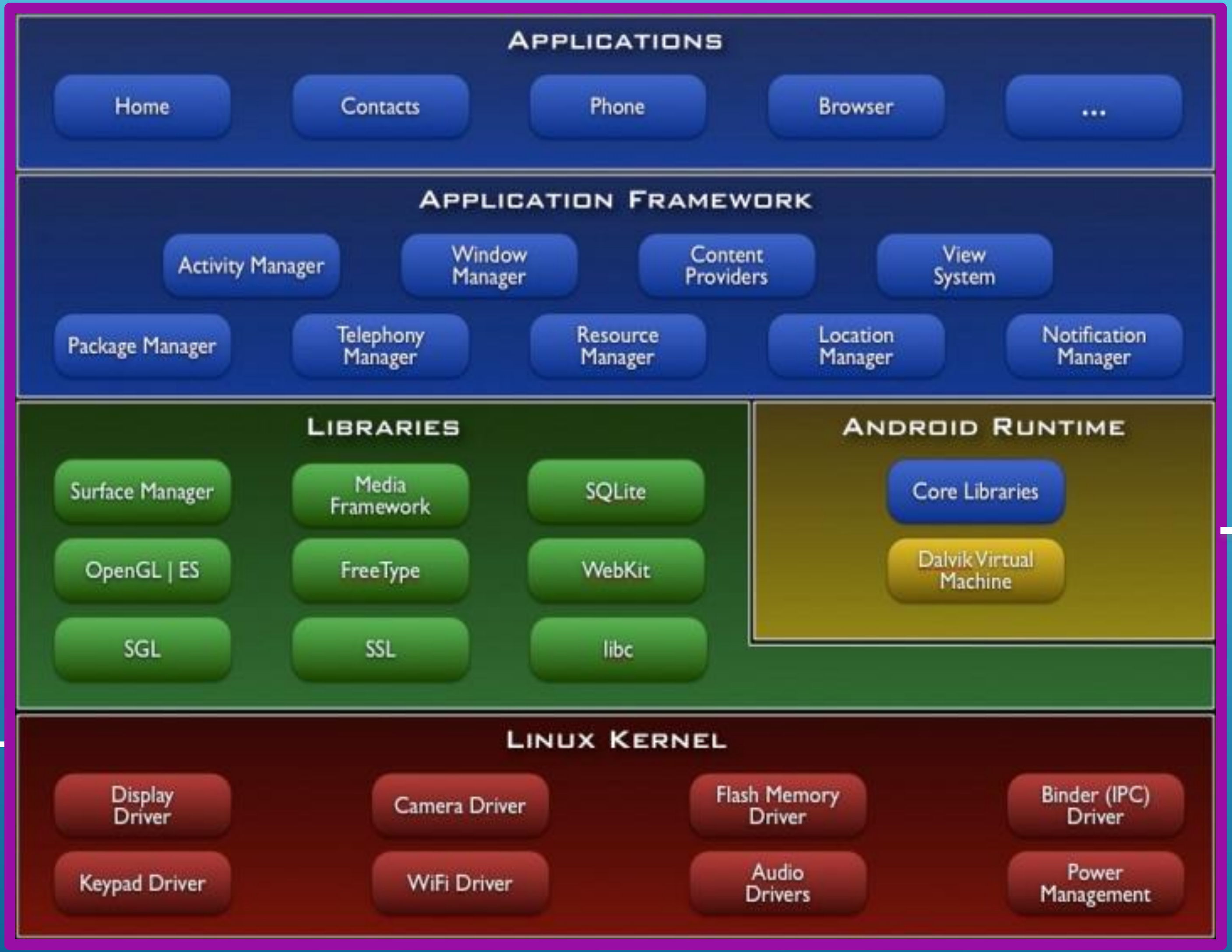


## Applications

(Written in **Java** code)

- **Android Play Store**
- **Entertainment**
- **Productivity**
- **Personalization**
- **Education**
- **Geo-communication**
- **….**

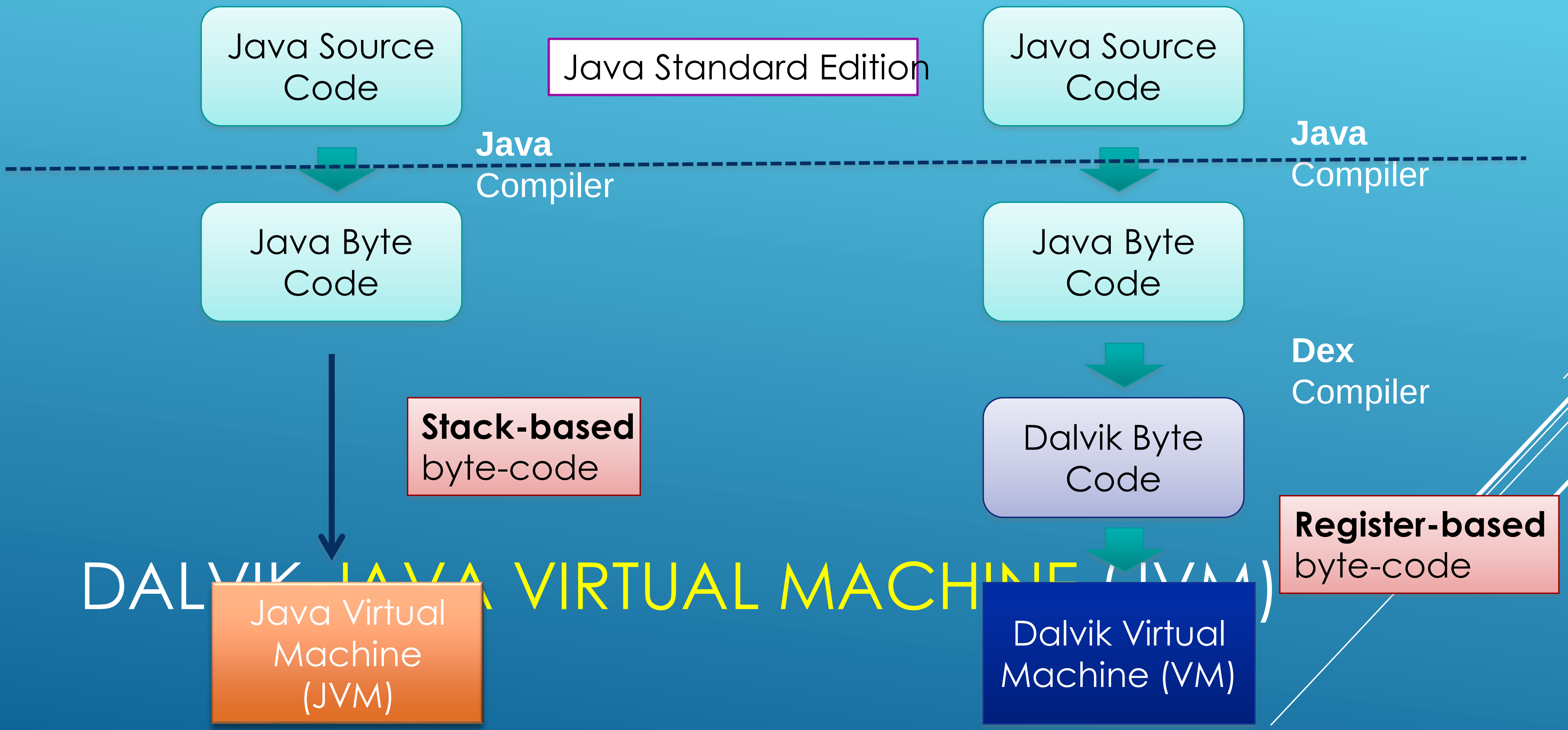
TH



## Dalvik Virtual Machine (VM)

- ➢**Novel** Java Virtual Machine implementation (not using the Oracle JVM)
- ➢Open **License** (Oracle JVM is not open!)
- ➢**Optimized** for memory-constrained devices
- ➢**Faster** than Oracle JVM
- ➢….

# DALVIK JAVA VIRTUAL MACHINE (JVM)



## APPLICATION **DESIGN**:

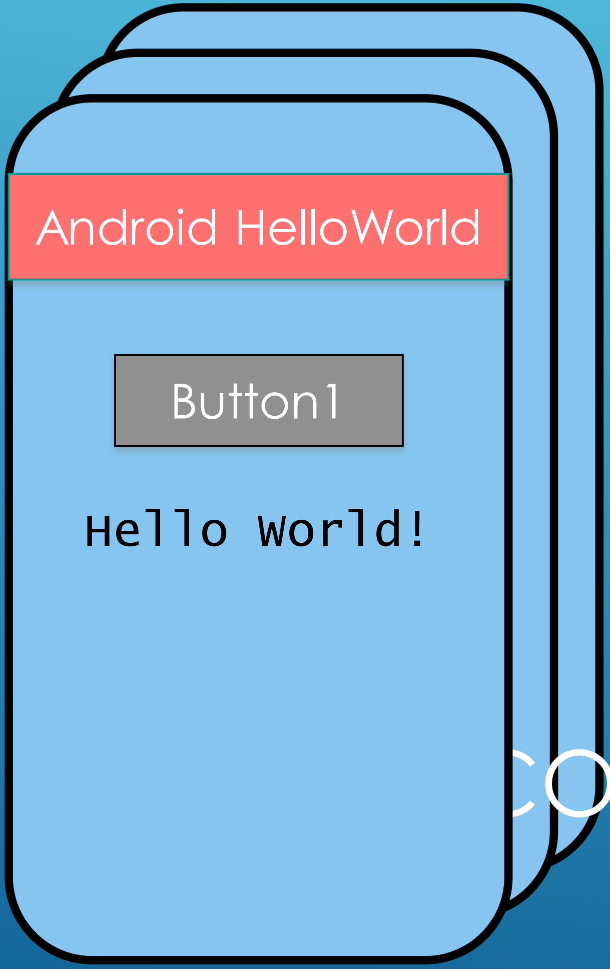
- **GUI** Definition
- **Events** Management
- Application **Data** Management
- **Background** Operations
- **User** Notifications

ATIONS DESIGN

## APPLICATION COMPONENTS

- Activities & Fragments
- Intents
- Services
- Content Providers
- Broadcast Receivers

ATIONS DESIGN



- An **Activity** corresponds to a **single screen** of the **Application**.
- An Application can be composed of *multiples screens* (Activities).
- The **Home Activity** is shown when the user launches an application.

## COMPONENTS: ACTIVITIES

- Different activities can exhange information one with each other.

- Each activity is composed by a list of *graphics components*.
- Some of these components (also called **Views**) can interact with the user by handling **events** (e.g. Buttons).
- Two ways to build the graphic interface:

**PROGRAMMATIC** APPROACH

Example:

```
Button button=new Button (this);
TextView text= new TextView();
text.setText("Hello world");
```

## ANDRO

- Each activity is composed by a list of *graphics components*.
- Some of these components (also called **Views**) can interact with the user by handling **events** (e.g. Buttons).
- Two ways to build the graphic interface:

**DECLARATIVE** APPROACH

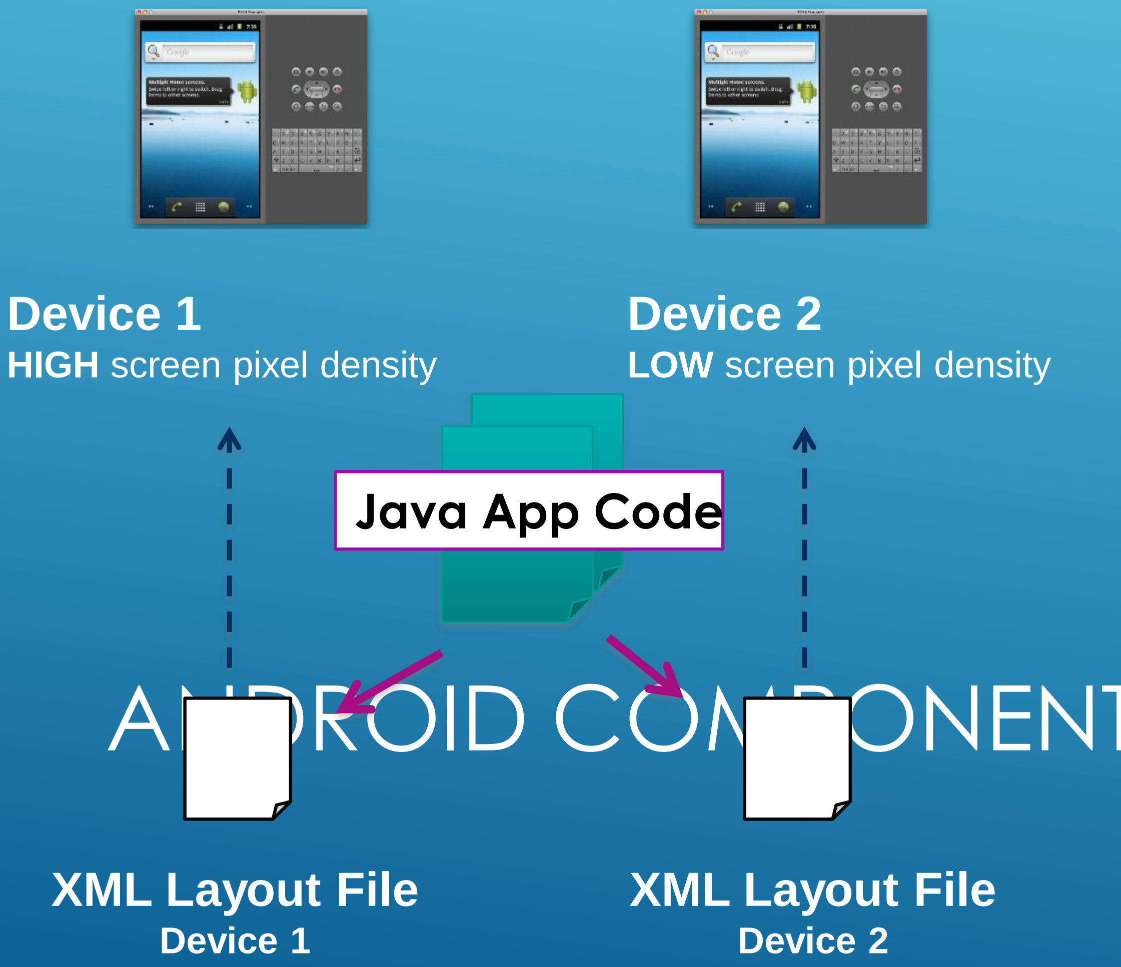
Example:

```
< TextView android.text=@string/hello” android:textcolor=@color/blue
android:layout_width=“fill_parent” android:layout_height=“wrap_content” />
< Button android.id=“@+id/Button01” android:textcolor=“@color/blue”
android:layout_width=“fill_parent” android:layout_height=“wrap_content” />
```

## AND

## ANDROID COMPONENT

EXAMPLE



- Build the **application layout** through XML files (like HTML)
- Define **two** different XML **layouts** for two different devices
- At **runtime**, Android detects the current device configuration and loads the appropriate resources for the application
- **No need to recompile**!
- Just add a new XML file if you need to support a new device

**EXAMPLE**

**Device 1**
**HIGH** screen pixel density

**Device 2**
**LOW** screen pixel density

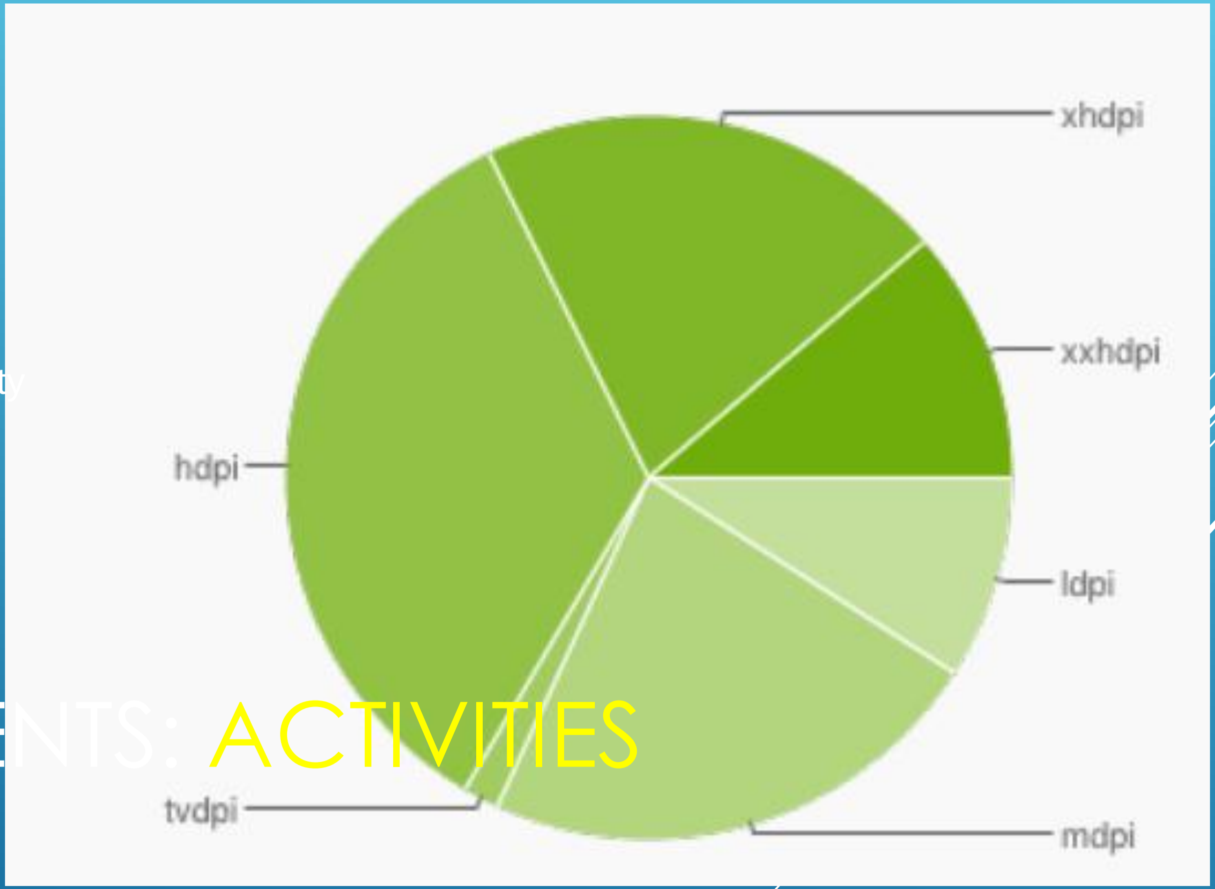
**Java App Code**

# ANDROID COMPONENTS: ACTIVITIES

**XML Layout File**
**Device 1**

**XML Layout File**
**Device 2**

**SCREEN CONFIGURATION DISTRIBUTION**



http://developer.android.com/about/dashboards/index.html

## ANDROID COMPONENTS: ACTIVITIES

➢ *Android applications typically use both the approaches!*

**DECLARATIVE** APPROACH

⇩

XML Code ⇨ Define the Application **layouts** and **resources** used by the Application (e.g. labels).

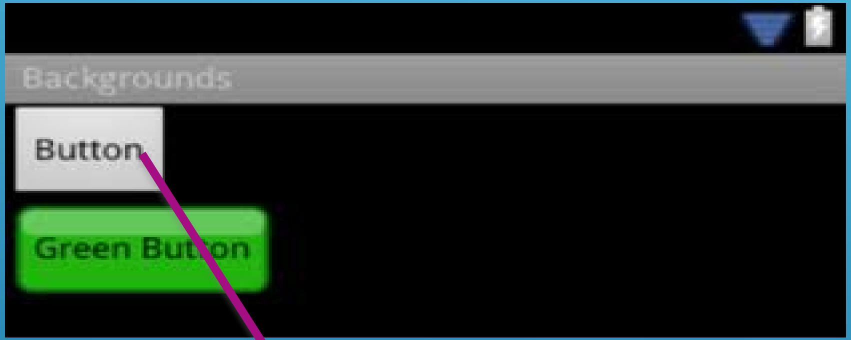
**PROGRAMMATIC** APPROACH

⇩

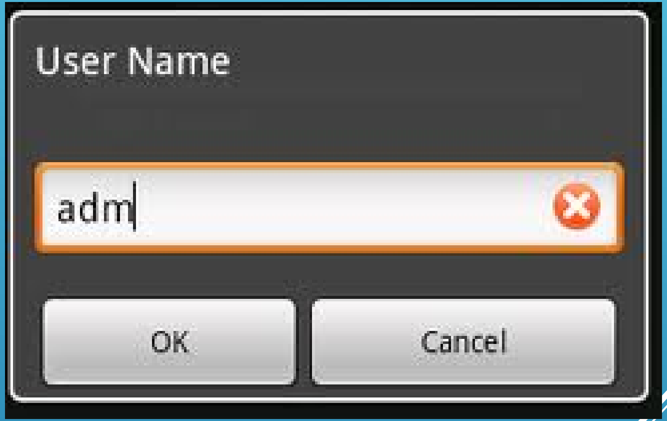
Java Code ⇨ Manages the **events**, and handles the **interaction** with the user.

- **Views** can generate **events** (caused by human interactions) that must be managed by the Android-developer.

Button



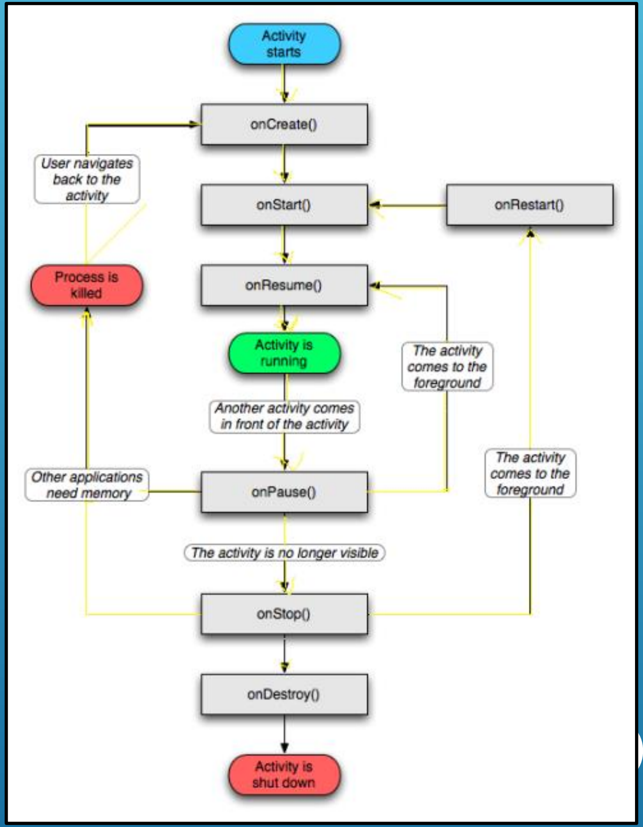
TextEdit



**ESEMPIO**

AND

```
public void onClick(View arg0) {
        if (arg0 == Button) {
                // Manage Button events
        }
}
```


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
Other applications need memory
onPause()
The activity comes to the foreground
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down

## MPONENTS: ACTIVITIES

- The **Activity Manager** is responsible for creating, destroying, managing activities.
- Activities can be on different **states**: *starting*, *running*, *stopped*, *destroyed*, *paused*.
- Only one activity can be on the **running** state at a time.
- Activities are organized on a **stack**, and have an event-driven life cycle (details later …)

- Main difference between Android-programming and Java (Oracle) -programming:
  - **Mobile devices have constrained resource capabilities!**
- Activity lifetime depends on **users' choice** (i.e. change of visibility) as well as on **system contraints** (i.e. memory shortage).

## ANDROID COMPONENTS: ACTIVITIES

- Developer must implement **lifecycle methods** to account for state changes of each Activity …

```
public class MyApp extends Activity {


   public void onCreate() { ... }
   public void onPause()  { ... }
   public void onStop()   { ... }
   public void onDestroy(){ ... }
   ….
}
```

Called when the Activity is **created** the first time.

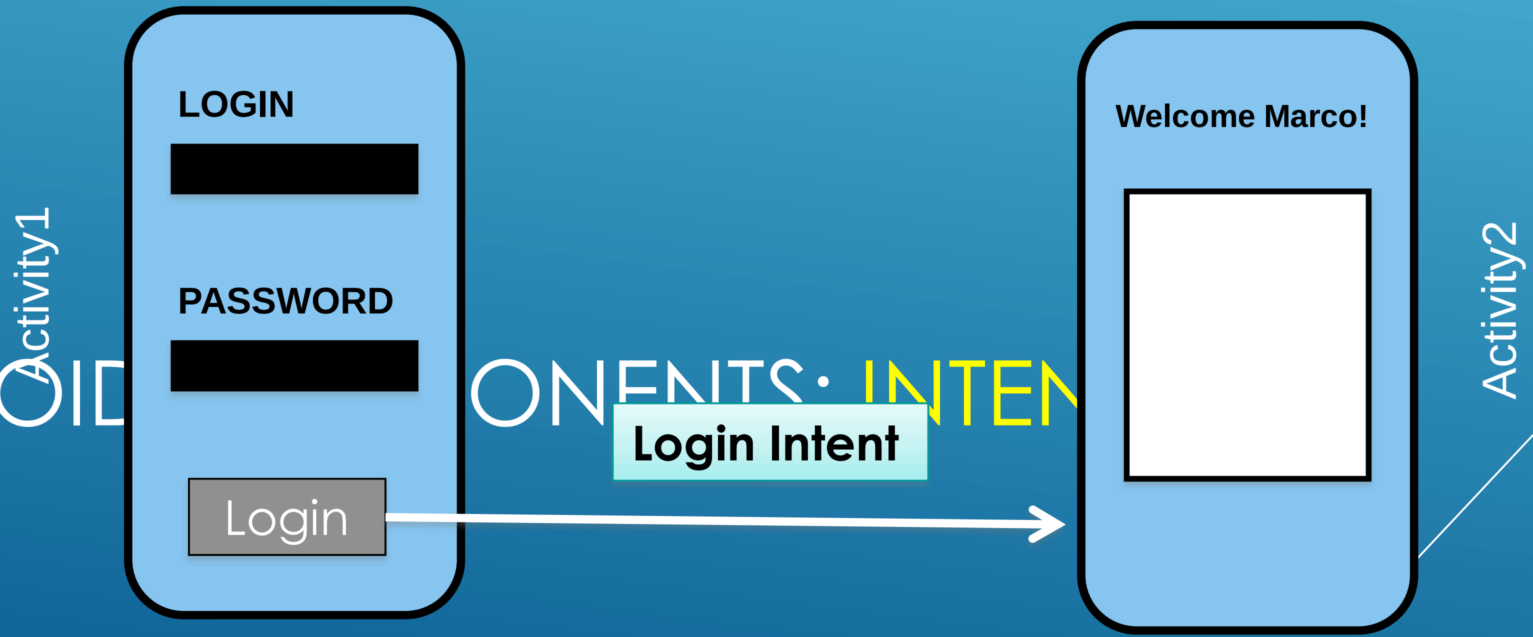
Called when the Activity is **partially visible**.

Called when the Activity is **no longer visible**.

Called when the Activity is **dismissed**.

A IES

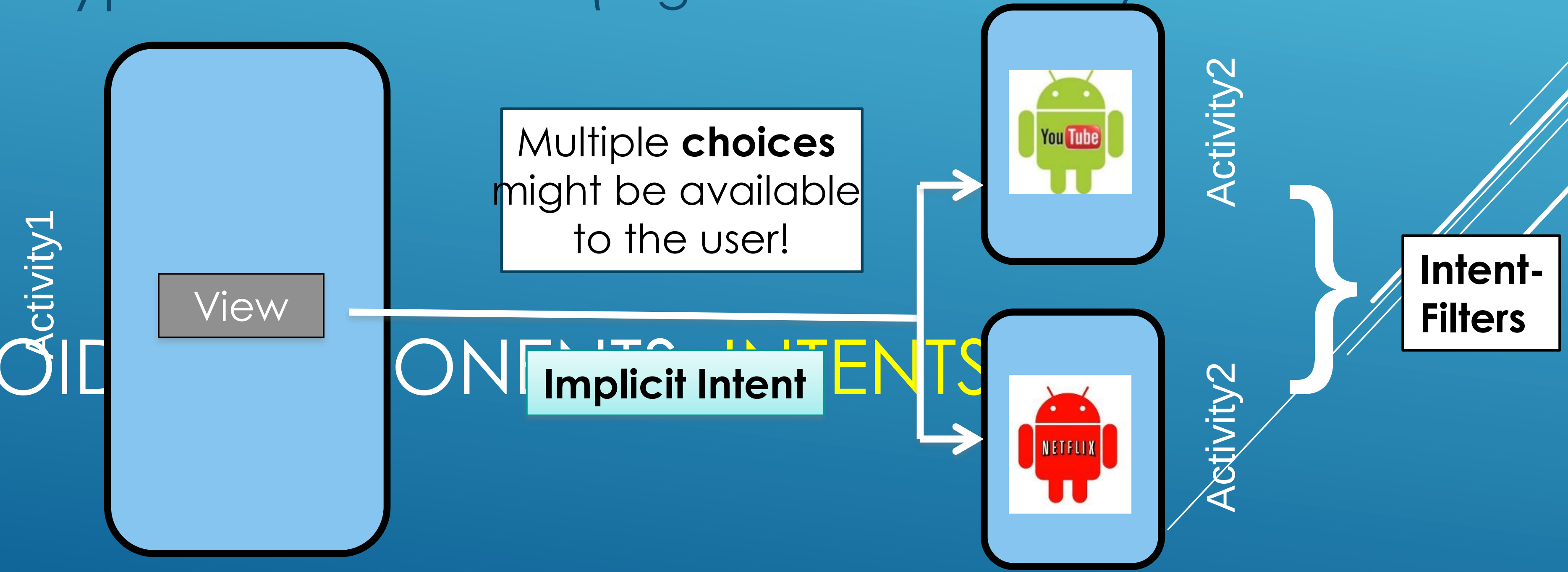
- **Intents**: asynchronous **messages** to activate core Android components (e.g. Activities).
- **Explicit** Intent → The component *(e.g. Activity1)* specifies the destination of the intent *(e.g. Activity 2)*.



## ANDROID ONENTS: INTEN

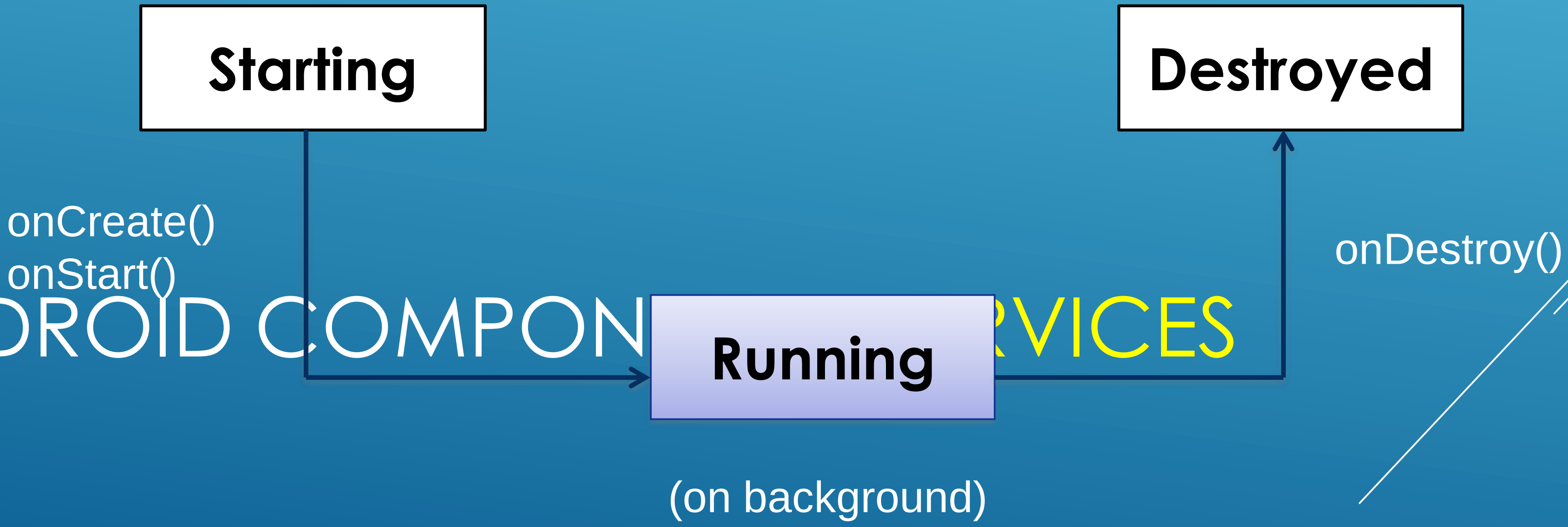
## ANDROID COMPONENTS: INTENTS

- **Intents**: asynchronous **messages** to activate core Android components (e.g. Activities).
- **Implicit** Intent → The component *(e.g. Activity1)* specifies the type of the intent *(e.g. "View a video")*.



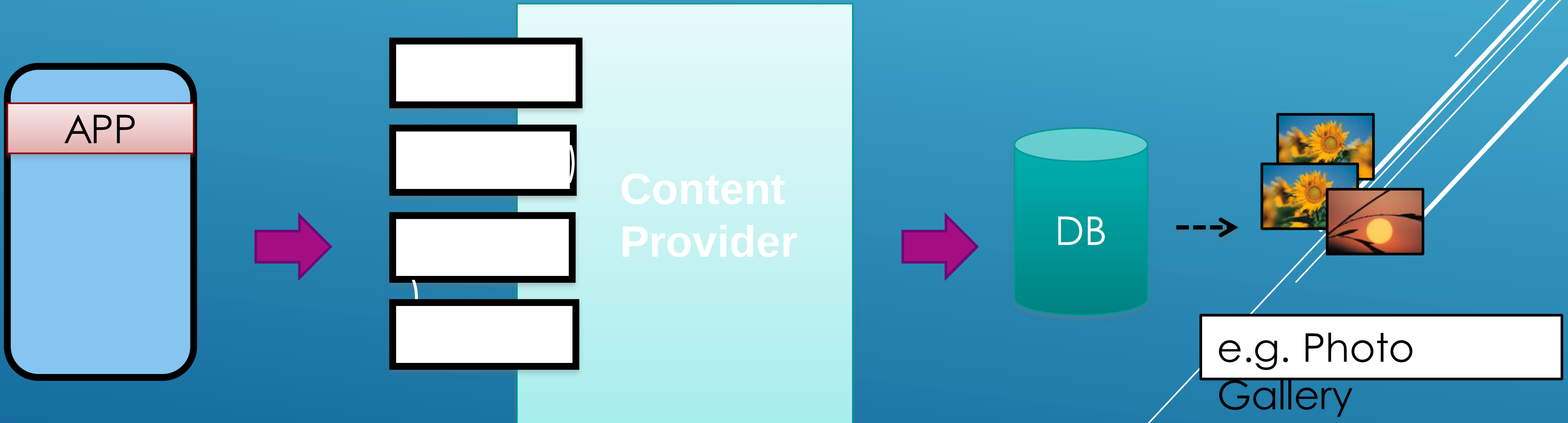
# ANDROID COMPON RVICES

- **Services**: like Activities, but run in **background** and do not provide an user interface.
- Used for **non-interactive** tasks (e.g. networking).
- Service life-time composed of 3 states:

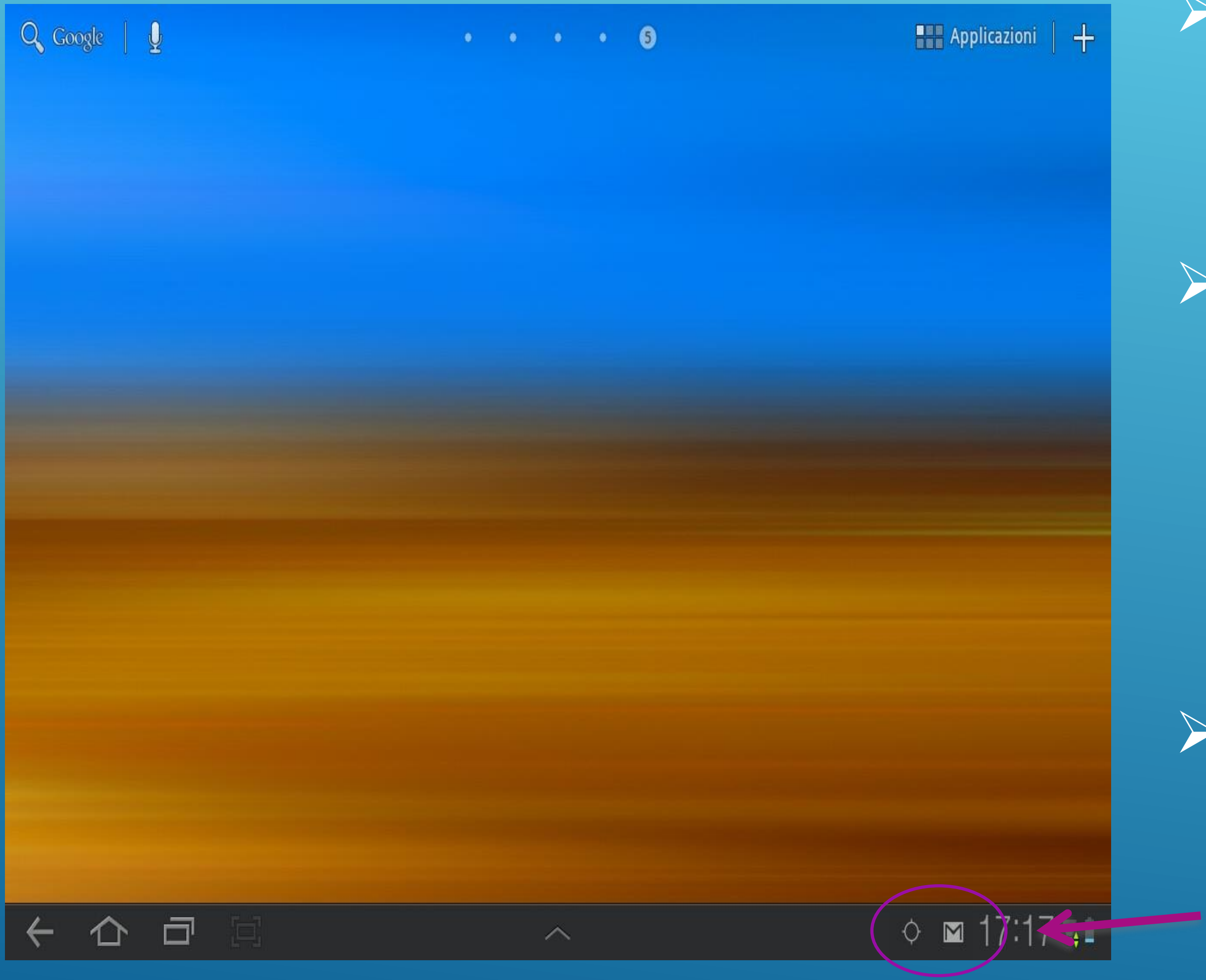


# ANDROID COMPONENTS: CONTENT PROVIDERS

- Each Android **application** has its own **private** set of data (managed through *files* or through *SQLite* database).
- **Content Providers**: Standard **interface** to *access and share data among different applications*.



# ANDROID COMPONENTS: BROADCAST RECEIVERS



- *Publish/Subscribe* paradigm
- **Broadcast Receivers**: An application can be signaled of **external events.**
- **Notification** types: Call incoming, SMS delivery, Wifi network detected, etc

# ANDROID COMPONENTS: BROADCAST RECEIVERS

**BROADCAST RECEIVER** example

```
class WifiReceiver extends BroadcastReceiver {
        public void onReceive(Context c, Intent intent) {
            String s = new StringBuilder();
            wifiList = mainWifi.getScanResults();
            for(int i = 0; i < wifiList.size(); i++){
                s.append(new Integer(i+1).toString() + ".");
                s.append((wifiList.get(i)).toString());
                s.append("\\n");
            }
            mainText.setText(sb);
        }
    }
```

## ANDROID COMPONENTS: SYSTEM API

- Using the **components** described so far, Android applications can then leverage the system API …

**SOME EXAMPLEs …**

- *Telephony Manager* data access (call, SMS, etc)
- *Sensor* management (GPS, accelerometer, etc)
- Network *connectivity* (Wifi, bluetooth, NFC, etc)
- *Web* surfing (HTTP client, WebView, etc)
- *Storage* management (files, SQLite db, etc)
- ….

## ANDROID COMPONENTS: GOOGLE API

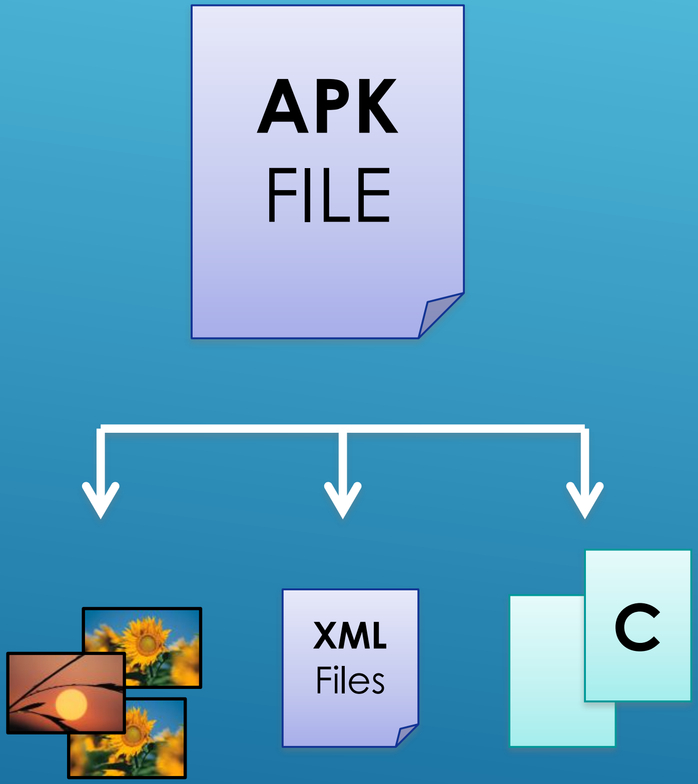
- … or easily interface with other **Google services**:

Google maps

Google places

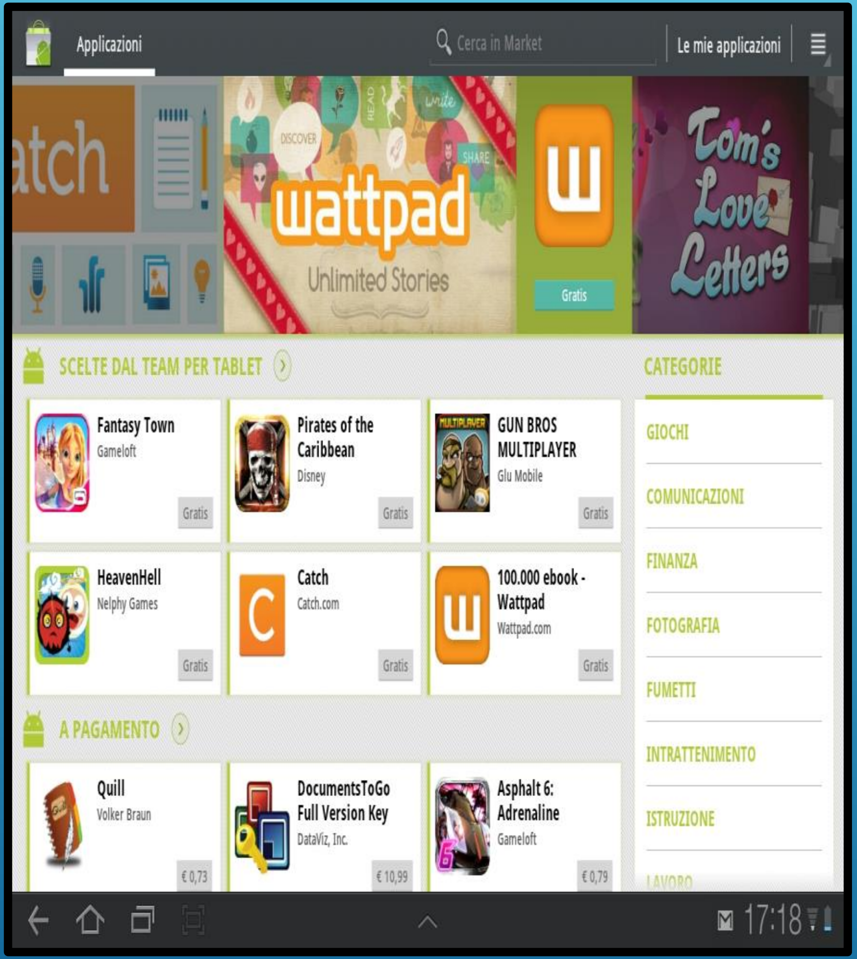
g+

# Android Application Distribution



- Each Android **application** is contained on a single **APK** file.
  - Java **Byte-code** (*compiled for Dalvik JVM*)
  - **Resources** (e.g. images. videos, XML layout files)
  - **Libraries** (optimal native C/C++ code)

# ANDROID APPLICATION DISTRIBUTION



- Each application must be signed through a **key** before being distributed.
- Applications can be **distributed** via *Web* or via *Stores*.
- **Android Play Store:** application store run by Google … but several other application stores are available (they are just normal applications).

## ANDROID APPLICATION SECURITY

- Android applications run with a distinct system identity (Linux user ID and group ID), in an **isolated** way.
- Applications must explicitly share resources and data. They do this by declaring the ***permissions*** they need for additional capabilities.
  - Applications statically **declare** the permissions they require.
  - User must **give his/her consensus** during the installation.

**ANDROIDMANIFEST.XML**

```
<uses-permission android:name="android.permission.IACCESS_FINE_LOCATION" />

<uses-permission android:name="android.permission.INTERNET" />
```