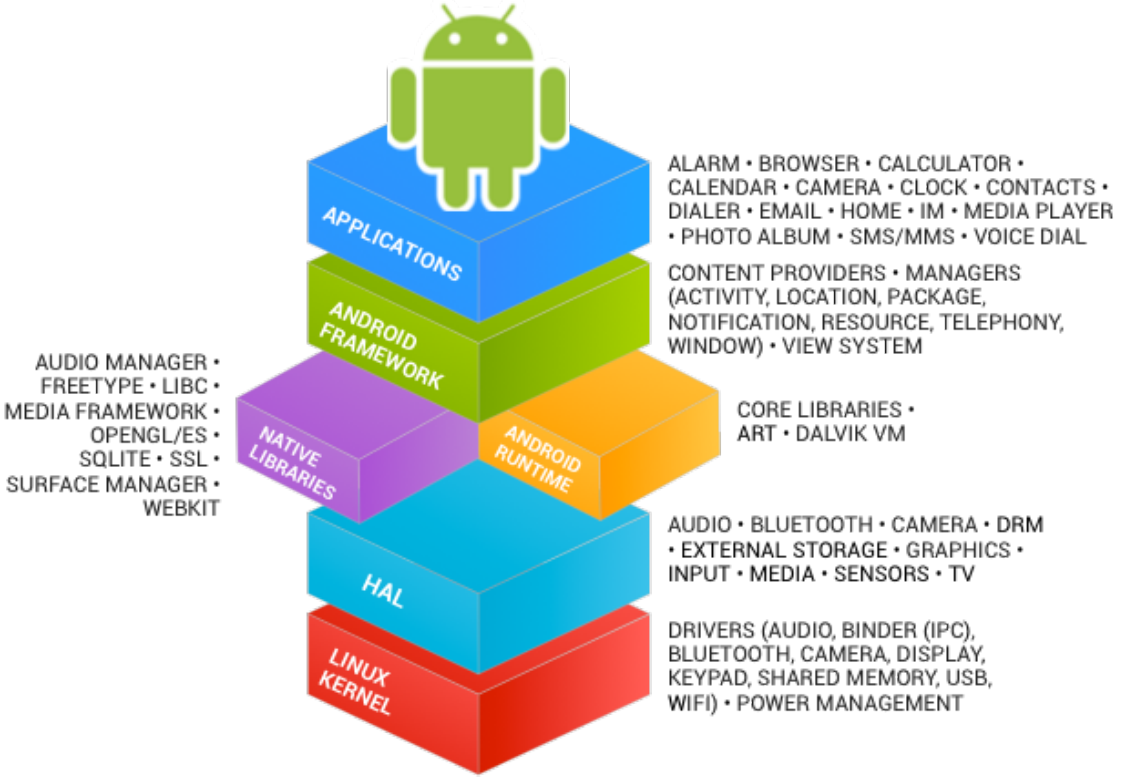
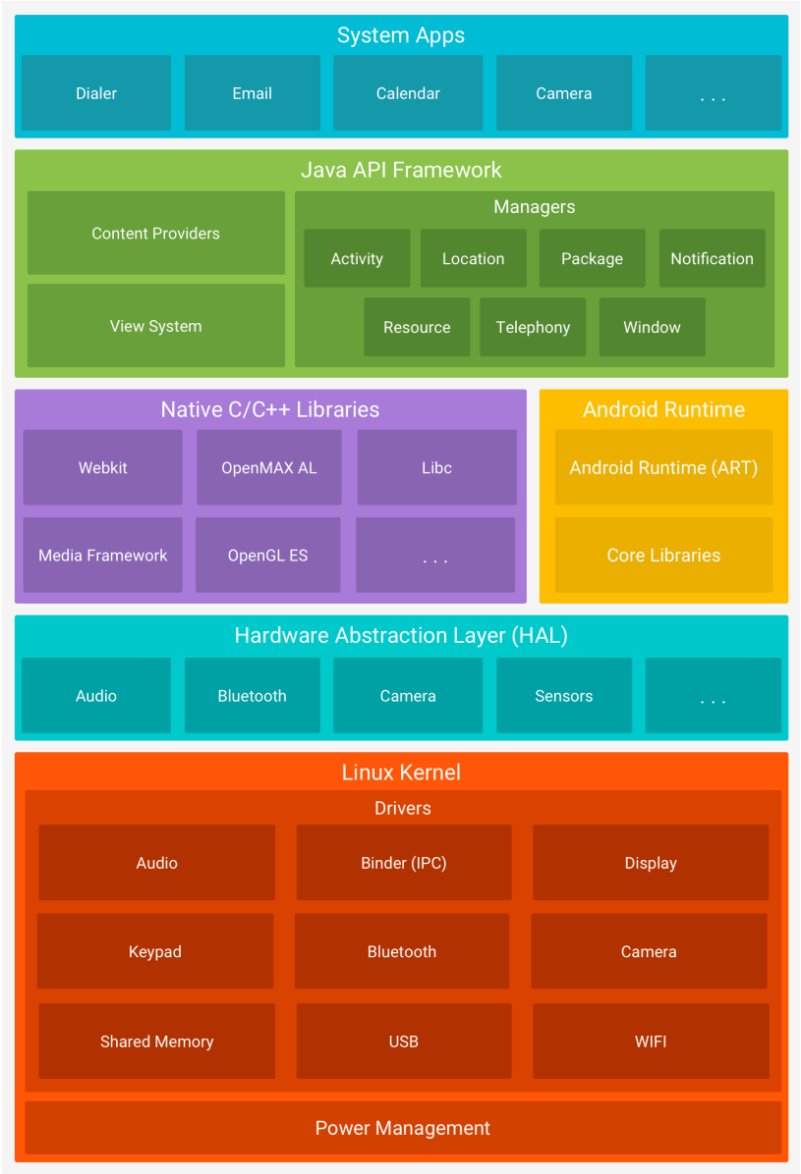
App Development

Android

Android Platform Architecture





[Android Open Source project]

Android Development

- Android Software Development Kit (Android SDK)
  - CLI tools to create, compile and package Android applications
  - Java primary programming language for applications
  - Gradle for build system
  - Java compiler converts source files to class files
  - dx tool: converts Java class files into a .dex (Dalvik Executable) file
  - aapt (Android Asset Packaging Tool) packs
    .dex file and Android project resources (e.g., XML, images) into
    .apk (android package) file containing all data to run the Android application
  - adb (Android debug bridge) tool deploys .apk file to an Android device (virtual or real) and enables managing and debugging it
- Android Studio is the official IDE for Android platform development
  - Based on JetBrains' IntelliJ IDEA, replaced Eclipse Android Development Tools (ADT)

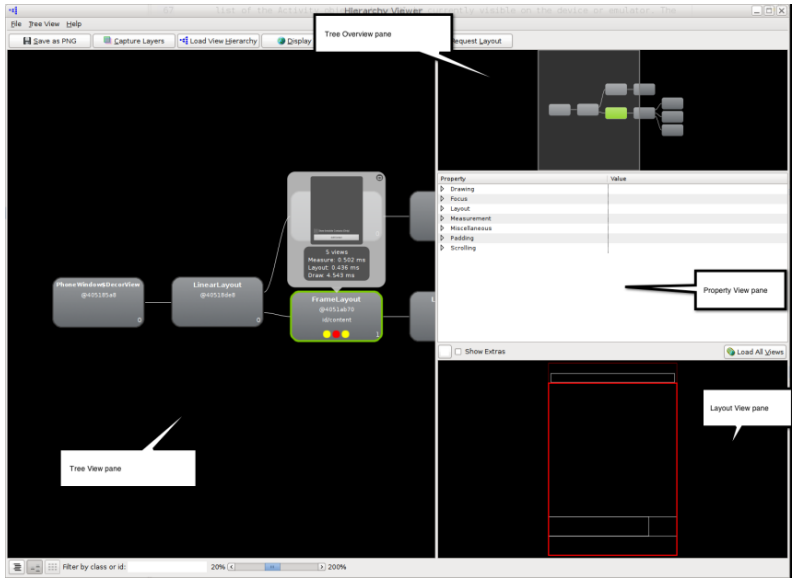
[Android Open Source project]

Android's Java History

- Sun controlled Java, bought by Oracle. Google sued by Oracle.
- Android initially based on Apache Harmony (open source, free Java implementation), retired late 2011.
- Apache Harmony
- Sun released OpenJDK
- Dalvik virtual machine up to version 4.4 (and Android Runtime (ART), that followed), uses a subset of Harmony for the core of its Class Library
- Switching to OpenJDK since late 2015
- Support for Java 8 language features
  - https://developer.android.com/preview/j8-jack.html
  - Default and static interface methods
  - Lambda expressions (also available on API level 23 and lower)
  - Repeatable annotations
  - Method References (also available on API level 23 and lower)
  - Type Annotations (also available on API level 23 and lower)

SDK Tools

- Layoutopt – static analyzer for XML layout files
- Monkey – test automation tool that runs in device or emulator
  - Sends 500 random events to specified application
- sqlite3 – tool to perform operations on sqlite
- keytool – generates encryption keys, e.g., signing certificate for release
- android – CLI for SDK and AVD manager
- Hierarchy Viewer - visual representation of the layout's View hierarchy (the Layout View) and a magnified inspector of the display (the Pixel Perfect View).



[Android Open Source project]

Android App Packaging

- Application
  - From a component perspective, an Android application consists of one or more activities, services, listeners, and intent receivers.
  - From a source file perspective, an Android application consists of code, resources, assets, and a single manifest.
- .apk file (Android application package)
  - Each Android application packaged in a single zipped file that includes all of the application's compiled code (.dex files), resources, assets, and manifest file.
- .dex file (Dalvik Executable)
  - Compiled Android application code file from Java, which are in turn zipped into a single .apk file on the device.

[https://developer.android.com/guide/appendix/glossary.html]

App Manifest (AndroidManifest.xml)

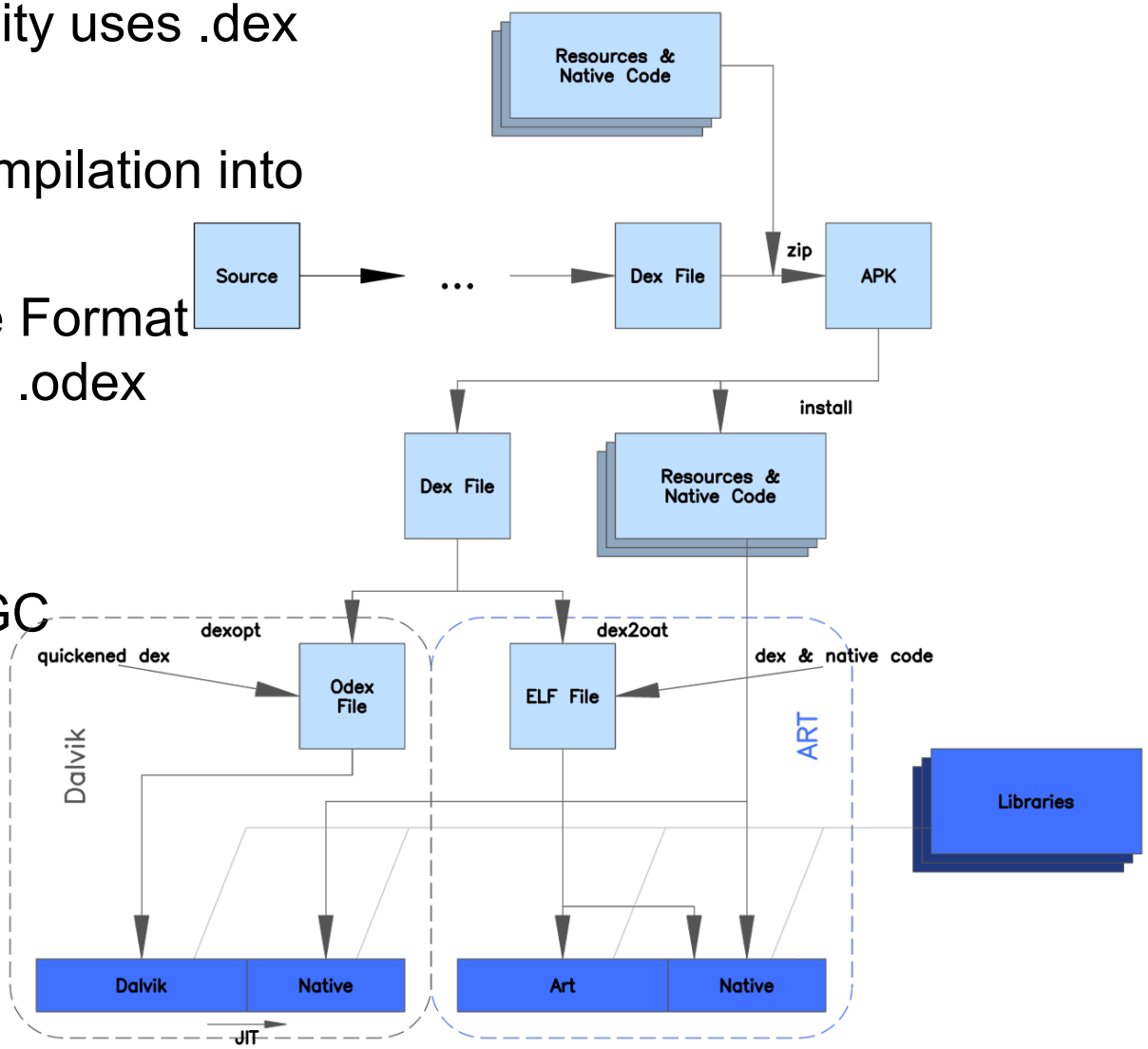
- Names Java package for app. The package name serves as UID for app.
- Describes app components: activities, services, broadcast receivers, content providers. Names classes that implement these and publishes their capabilities, such as the Intent messages that they can handle. These declarations inform the Android system of the components and the conditions in which they can be launched.
- Determines processes that host the application components.
- Declares the permissions app needs and declares the permissions that others need to interact with the app components.
- During development: lists Instrumentation classes for profiling.
- Declares minimum Android API level required.
- Lists libraries app must be linked against.
- Example XML structure: https://developer.android.com/guide/topics/manifest/manifest-intro.html

Dalvik

- Process virtual machine for Android
- Used up to Android 4.4 (KitKat)
- Java bytecode interpreted and trace-based JIT compilation of frequently executed segments into native code

Android Runtine (ART)

- For backward compatibility uses .dex input bytecode
- Ahead-of-time (AOT) compilation into native machine code
- Executable and Linkable Format (ELF) instead of Dalvik's .odex transformed by dex2oat
- Faster execution
- Improved memory and GC
- Better debugging and profiling



Resources

- Animation Resources: Define pre-determined animations.
  - in res/anim/ tween animations accessed from the R.anim class.
  - in res/drawable/ frame animations accessed from the R.drawable class.
- Color State List Resource: Define color resources that change based on the View state.
  - in res/color/ and accessed from the R.color class.
- Drawable Resources: Define various graphics with bitmaps or XML.
  - in res/drawable/ and accessed from the R.drawable class.
- Layout Resource: Define the layout for your application UI.
  - in res/layout/ and accessed from the R.layout class.
- Menu Resource: Define the contents of your application menus.
  - in res/menu/ and accessed from the R.menu class.
- String Resources: Define strings, string arrays, and plurals (and include string formatting and styling).
  - in res/values/ and accessed from the R.string, R.array, and R.plurals classes.
- Style Resource: Define the look and format for UI elements.
  - in res/values/ and accessed from the R.style class.
- More Resource Types: Define values such as booleans, integers, dimensions, colors, and other arrays.
  - in res/values/ but each accessed from unique R sub-classes (such as R.bool, R.integer, R.dimen, etc.).
- https://developer.android.com/guide/topics/resources/available-resources.html

[Android Open Source project]

build/generated/source/r/debug/.../R.java

- Resources. Do not edit this automatically generated file.
  - Referenced in code via R.string. or R.layout.
- Contains needed constants in programmatic form.

```
public static final class attr {
    public static final int actionBarDivider = 0x7f01003b;
    public static final int actionBarItemBackground = 0x7f01003c;
    public static final int actionBarPopupTheme = 0x7f010035;
    public static final int actionBarSize = 0x7f01003a;
    public static final int actionBarSplitStyle = 0x7f010037;
    public static final int actionBarStyle = 0x7f010036;
    public static final int actionBarTabBarStyle = 0x7f010031;
```

```
public static final class styleable {
    public static final int[] ActionBar = { 0x7f010001, 0x7f010003, 0x7f010004, 0x7f010005, 0x7f010006, 0x7f010007, 0x7f010008,
    public static final int[] ActionBarLayout = { 0x010100b3 };
    public static final int ActionBarLayout_android_layout_gravity = 0;
    public static final int ActionBar_background = 10;
    public static final int ActionBar_backgroundSplit = 12;
    public static final int ActionBar_backgroundStacked = 11;
    public static final int ActionBar_contentInsetEnd = 21;
    public static final int ActionBar_contentInsetLeft = 22;
    public static final int ActionBar_contentInsetRight = 23;
    public static final int ActionBar_contentInsetStart = 20;
    public static final int ActionBar_customNavigationLayout = 13;
    public static final int ActionBar_displayOptions = 3;
    public static final int ActionBar_divider = 9;
    public static final int ActionBar_elevation = 24;
    public static final int ActionBar_height = 0;
    public static final int ActionBar_hideOnContentScroll = 19;
```

Android Components

Activities

A single screen in an application, with supporting Java code, derived from the Activity class. Usually, an activity is visibly represented by a full screen window that can receive and handle UI events and perform complex tasks, because of the Window it uses to render its window. Though typically full screen, it can also be floating or transparent.

Services

An object of class Service that runs in the background (without any UI presence) to perform various persistent actions, such as playing music or monitoring network activity.

Intents

A message object used to launch or communicate with other applications/activities asynchronously. App sends Intent to Android system, rather than directly to another application/activity. The application can send the Intent to a single target application or it can send it as a broadcast, which can in turn be handled by multiple applications sequentially. Android system resolves the best-available receiver for each Intent, based on the criteria supplied in the Intent and the Intent Filters defined by other applications.
Explicit intent when the target component name is explicitly specified and passed on directly, otherwise Implicit.

Intent Filters

A filter object that an application declares in its manifest file, to tell the system what types of Intents each of its components is willing to accept and with what criteria. Best match receives the Intent.

Broadcast receivers

An application class that listens for Intents that are broadcast, rather than being sent to a single target application/activity. The system delivers a broadcast Intent to all interested broadcast receivers, which handle the Intent sequentially.

Content providers

A data-abstraction layer that you can use to safely expose your application's data to other applications. Builds on ContentProvider class, which handles content query strings of a specific format to return data in a specific format.

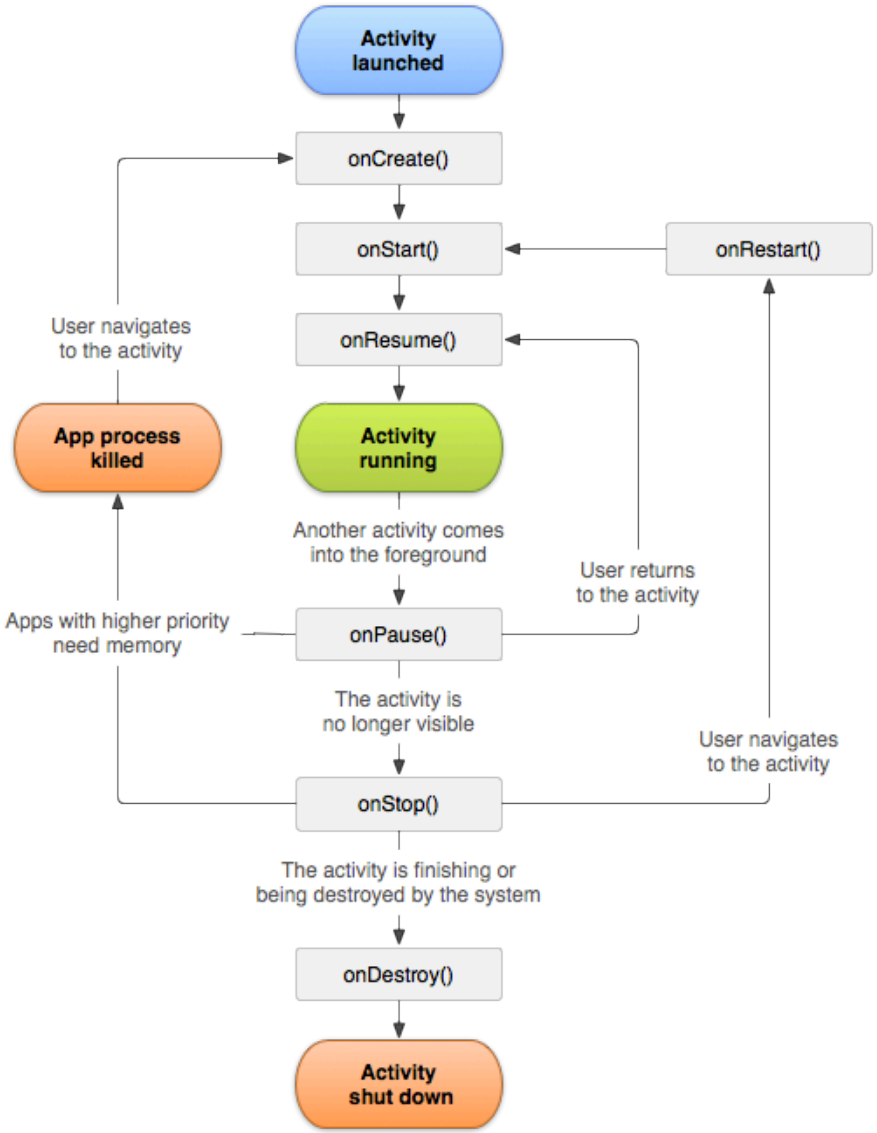
App Widgets

One of a set of fully implemented View subclasses that render form elements and other UI components, such as a text box or popup menu. Handles measuring and drawing itself and responding to screen events.

Processes and Threads

By default, all components of the same application run in the same process, but this can be configured. 2 rules to Android's single thread model: don’t block the UI thread; don’t access the Android UI toolkit from outside the UI thread.

Activity Lifecycle



Method	Description
onCreate()	Called when the activity first created. Perform normal static set up — create views, bind data to lists, etc. This method is passed a Bundle object containing the activity's previous state, if that state was captured.
onRestart()	Called after activity stopped, just prior to it being started again.
onStart()	Called just before activity is visible to user.
onResume()	Called just before the activity starts interacting with the user. At this point the activity is at the top of the activity stack, with user input going to it.
onPause()	Called when the system is about to start resuming another activity. Typically used to commit unsaved changes to persistent data, stop animations and other things consuming CPU, etc. Should quickly finish, as next activity will not be resumed until it returns.
onStop()	Called when the activity is no longer visible to the user. This may happen because it is being destroyed, or because another activity (either an existing one or a new one) has been resumed and is covering it.
onDestroy()	Called before activity destroyed. This is the final call that the activity will receive. It could be called either because the activity is finishing (someone called finish() on it), or because the system is temporarily destroying this instance of the activity to save space. You can distinguish between these two scenarios with the isFinishing() method.

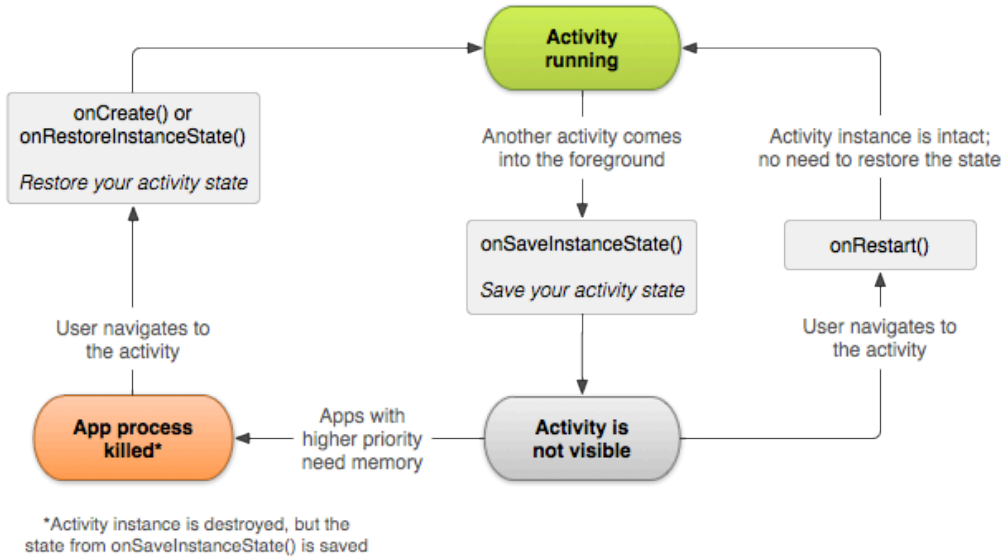
[Android Open Source project]

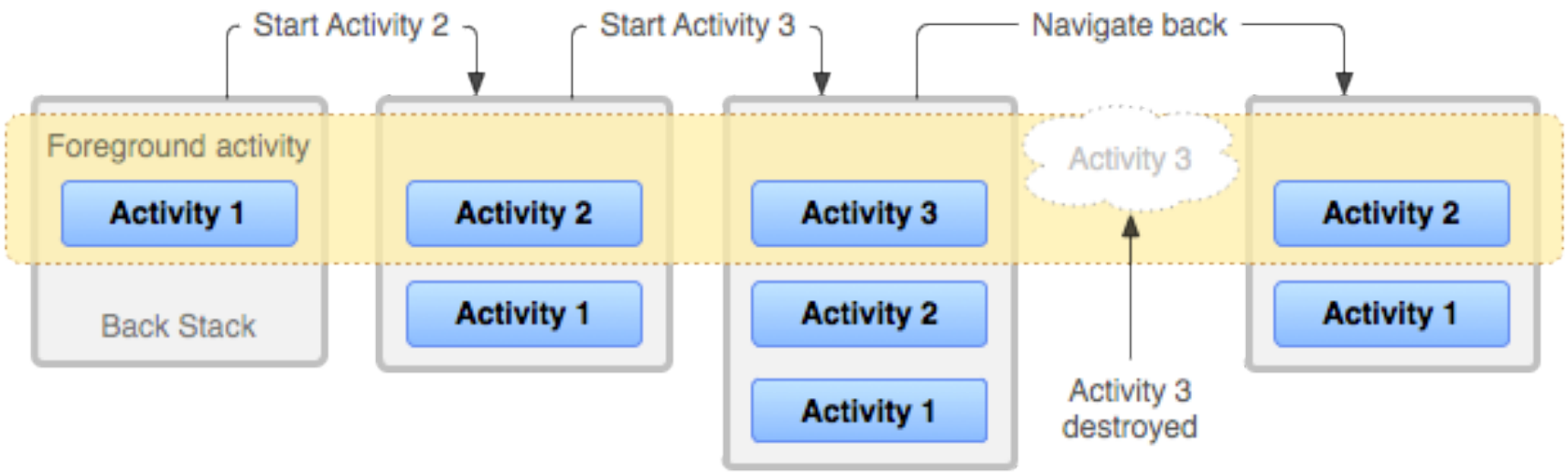
Activity Lifecycle Callback Methods

Method	Description	Killable after?	Next
onCreate()	Called when the activity first created. Perform normal static set up — create views, bind data to lists, etc. This method is passed a Bundle object containing the activity's previous state, if that state was captured.	No	onStart()
onRestart()	Called after activity stopped, just prior to it being started again.	No	onStart()
onStart()	Called just before activity is visible to user.	No	onResume() - if becomes foreground or onStop() - becomes invisible
onResume()	Called just before the activity starts interacting with the user. At this point the activity is at the top of the activity stack, with user input going to it.	No	onPause()
onPause()	Called when the system is about to start resuming another activity. Typically used to commit unsaved changes to persistent data, stop animations and other things consuming CPU, etc. Should quickly finish, as next activity will not be resumed until it returns.	Yes	onResume() - if becomes foreground or onStop() - becomes invisible
onStop()	Called when the activity is no longer visible to the user. This may happen because it is being destroyed, or because another activity (either an existing one or a new one) has been resumed and is covering it.	Yes	onRestart()- if becomes foreground or onDestroy()
onDestroy()	Called before activity destroyed. This is the final call that the activity will receive. It could be called either because the activity is finishing (someone called finish() on it), or because the system is temporarily destroying this instance of the activity to save space. You can distinguish between these two scenarios with the isFinishing() method.	Yes	nothing

[Android Open Source project]

Back Stack





[Android Open Source project]

Loaders

- Available to every Activity and Fragment.
- Provide asynchronous loading of data in an activity or fragment.
- Monitor the source of their data and deliver new results when the content changes.
- Automatically reconnect to the last loader's cursor when being recreated after a configuration change. Thus, they don't need to re-query their data.

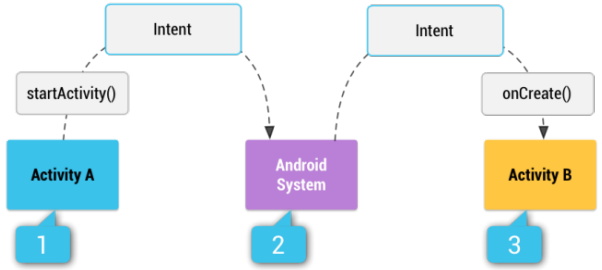
[Android Open Source project]

Service vs. Thread

- A service is simply a component that can run in the background even when the user is not interacting with your application.
- If you need to perform work outside your main thread, but only while the user is interacting with your application, then you should probably instead create a new thread and not a service.
  - E.g., play music, but only while your activity is running, you might
    - create a thread in onCreate(),
    - start running it in onStart(),
    - then stop it in onStop().
    - Also consider using AsyncTask or HandlerThread, instead of the traditional Thread class.

[Android Open Source project]

Intents and Intent Filters

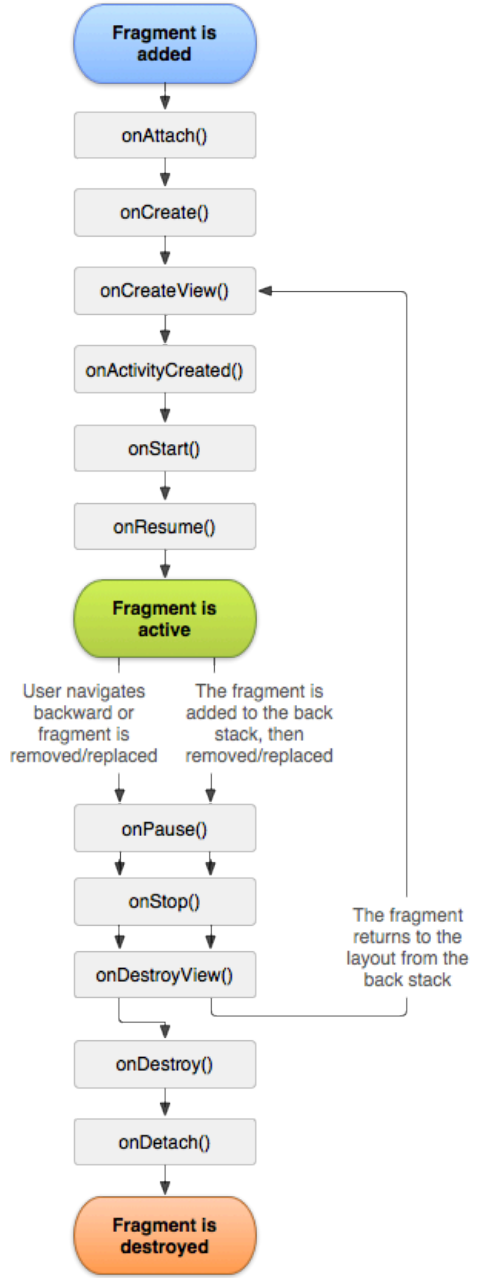


- Intents
  - Implicit - do not name a specific component, but instead declare a general action to perform, which allows a component from another app to handle it.
  - Implicit intent delivery:
    1. *Activity A* creates an Intent with an action description and passes it to startActivity().
    2. The Android System searches all apps for an intent filter that matches the intent. When a match is found,
    3. the system starts the matching activity (*Activity B*) by invoking its onCreate() method and passing it the Intent.
  - Explicit - the target component name is explicitly specified and passed on directly. E.g. to explicitly start the DownloadService class in the app :
    - // Executed in an Activity, so 'this' is the Context
    - // The fileUrl is a string URL, such as "http://www.example.com/image.png"
    - Intent downloadIntent = new Intent(this, DownloadService.class);
    - downloadIntent.setData(Uri.parse(fileUrl));
    - startService(downloadIntent);
- Intent Filter – specify <action> <data> and <category>
  - E.g., an intent filter to receive an ACTION_SEND intent when the data type is text:
  - <activity android:name="ShareActivity">
    <intent-filter>
    <action android:name="android.intent.action.SEND"/>
    <category android:name="android.intent.category.DEFAULT"/>
    <data android:mimeType="text/plain"/>
    </intent-filter>
    </activity>

[Android Open Source project]

Fragments

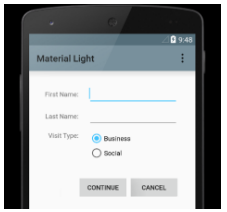
- Fragment
  - Represents a behavior or a portion of UI in an Activity.
  - Multiple fragments can be combined in a single activity to build a multi-pane UI and reuse a fragment in multiple activities.
  - Like a modular section of an activity, having its own lifecycle, its own input events, and can be added/removed while activity running (like subactivity).
- Initially created to better support tablets with large screens
- https://developer.android.com/guide/components/fragments.html



Coding Hints

- In XML use lowercase and '_' to connect strings.
  - layout_width
- In Java use CamelCase mixed upper and lowercase

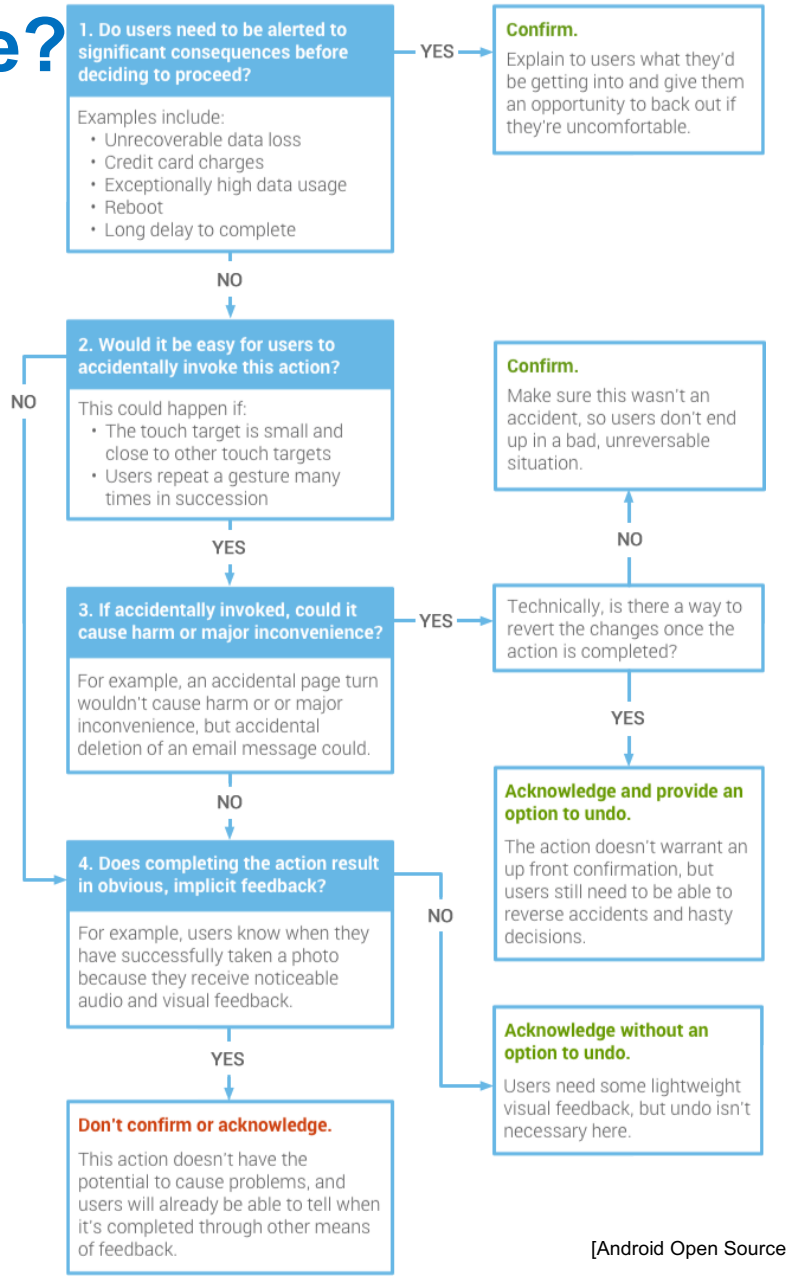
Android UI: Material Design Principles



- Material is the metaphor
  - A material metaphor is the unifying theory of a rationalized space and a system of motion. The material is grounded in tactile reality, inspired by the study of paper and ink, yet technologically advanced and open to imagination and magic.
  - Surfaces and edges of the material provide visual cues that are grounded in reality. The use of familiar tactile attributes helps users quickly understand affordances (Affordanz, Aufforderungscharakter, Angebotscharakter). Yet the flexibility of the material creates new affordances that supercede those in the physical world, without breaking the rules of physics.
  - The fundamentals of light, surface, and movement are key to conveying how objects move, interact, and exist in space and in relation to each other. Realistic lighting shows seams, divides space, and indicates moving parts.
- Bold, graphic, intentional
  - The foundational elements of print-based design – typography, grids, space, scale, color, and use of imagery – guide visual treatments. These elements do far more than please the eye. They create hierarchy, meaning, and focus. Deliberate color choices, edge-to-edge imagery, large-scale typography, and intentional white space create a bold and graphic interface that immerse the user in the experience.
  - An emphasis on user actions makes core functionality immediately apparent and provides waypoints for the user.
- Motion provides meaning
  - Motion respects and reinforces the user as the prime mover. Primary user actions are inflection points that initiate motion, transforming the whole design.
  - All action takes place in a single environment. Objects are presented to the user without breaking the continuity of experience even as they transform and reorganize.
  - Motion is meaningful and appropriate, serving to focus attention and maintain continuity. Feedback is subtle yet clear. Transitions are efficient yet coherent.
- https://material.google.com/
- Material Design for Android

When Confirm/Acknowledge?

- Confirm
  - Ask user to verify they truly want to proceed with an action
- Acknowledge
  - Display text to let user know the action they just invoked is completed, removing uncertainty about implicit operations.
- https://developer.android.com/design/patterns/confirming-acknowledging.html



[Android Open Source project]

Custom Android ROM

- CyanogenMod
  - Discontinued. Forked to LineageOS.
- Hive ROM
- MIUI by Xiaomi

Reference

- Android API reference
  - https://developer.android.com/reference/packages.html

Android

- *android*
  - A robot or synthetic organism designed to look/act like human
- Name of an OS based on the Linux kernel
- Android Open Source Project (AOSP) https://source.android.com/
  - Originated by Open Handset Alliance, group of companies led by Google
  - Android Compatibility Program to avoid uncontrolled customization
    - Defines “Android compatible”
  - Device builders “port” Android to a device, not implementing a specification
- Device types
  - Phones and tablets
  - Wear
  - TV
  - Auto
- Proprietary software is required for accessing Google services

History of Android

- Android OS initially developed by Android, Inc., founded 2003
  - Acquired by Google in 2005
- Open Handset Alliance founded 2007
  - 84 firms to develop standards for mobile devices
- Unveiled late 2007, Initial release Sep. 2008
- First handset T-Mobile G1 (HTC Dream) shipped Oct. 2008

- 2005-2011 Apache Harmony Project for Java libraries
- 2016 Switch to OpenJDK for Java libraries

Android Version History

Code name	Version number	Initial release date	API level
	1.0	September 23, 2008	1
	1.1	February 9, 2009	2
Cupcake	1.5	April 27, 2009	3
Donut	1.6	September 15, 2009	4
Eclair	2.0–2.1	October 26, 2009	5–7
Froyo	2.2–2.2.3	May 20, 2010	8
Gingerbread	2.3–2.3.7	December 6, 2010	9–10
Honeycomb	3.0–3.2.6	February 22, 2011	11–13
Ice Cream Sandwich	4.0–4.0.4	October 18, 2011	14–15
Jelly Bean	4.1–4.3.1	July 9, 2012	16–18
KitKat	4.4–4.4.4, 4.4W–4.4W.2	October 31, 2013	19–20
Lollipop	5.0–5.1.1	November 12, 2014	21–22
Marshmallow	6.0–6.0.1	October 5, 2015	23

For pics and comparisons between versions:
http://arstechnica.com/gadgets/2014/06/building-android-a-40000-word-history-of-googles-mobile-os/1/

[wikipedia]

Android Compatibility

- Goals
  - Provide a consistent application and hardware environment to application developers.
  - Enable a consistent application experience for consumers.
  - Enable device manufacturers to differentiate while being compatible. The Android compatibility program focuses on the aspects of Android relevant to running third-party applications, which allows device manufacturers the flexibility to create unique devices that are nonetheless compatible.
  - Minimize costs and overhead associated with compatibility.
- Android Compatibility Definition Document (CDD)
- Compatibility Test Suite (CTS)
- Licensing Google Mobile Services (GMS)
  - Producers of an Android compatible device can consider licensing Google Mobile Services (GMS), Google's proprietary suite of apps (Google Play, YouTube, Google Maps, Gmail, and more) on Android.
  - GMS is not part of the Android Open Source Project and is available only through a license with Google
- https://source.android.com/compatibility/index.html

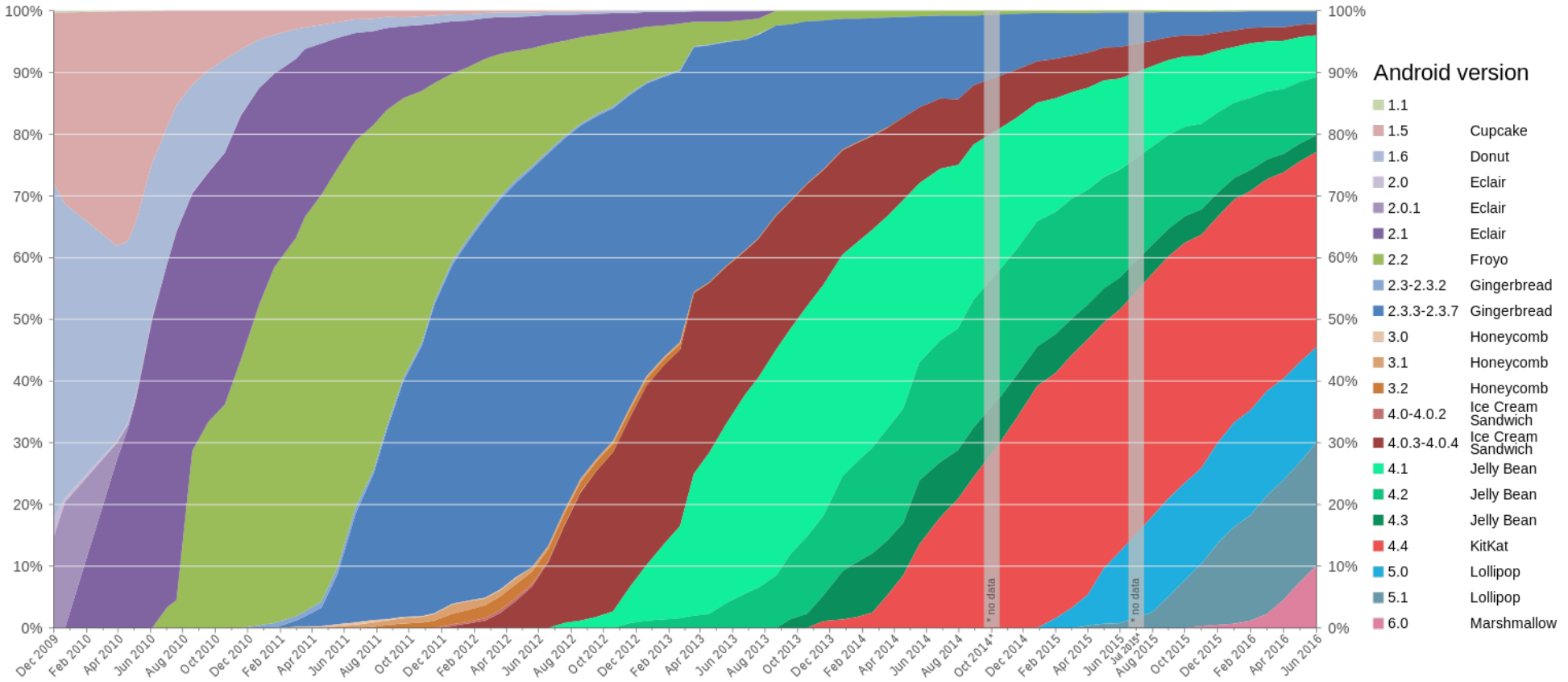
Excerpts from the CDD

- Device Configurations

Category	Feature	Section	Handheld	Television	Watch	Automotive	Other
Input	D-pad	7.2.2. Non-touch Navigation		MUST			
	Touchscreen	7.2.4. Touchscreen input	MUST		MUST		SHOULD
	Microphone	7.8.1. Microphone	MUST	SHOULD	MUST	MUST	SHOULD
Sensors	Accelerometer	7.3.1 Accelerometer	SHOULD		SHOULD		SHOULD
	GPS	7.3.3. GPS	SHOULD			SHOULD	
Connectivity	Wi-Fi	7.4.2. IEEE 802.11	SHOULD	MUST		SHOULD	SHOULD
	Wi-Fi Direct	7.4.2.1. Wi-Fi Direct	SHOULD	SHOULD			SHOULD
	Bluetooth	7.4.3. Bluetooth	SHOULD	MUST	MUST	MUST	SHOULD
	Bluetooth Low Energy	7.4.3. Bluetooth	SHOULD	MUST	SHOULD	SHOULD	SHOULD
	USB peripheral/host mode	7.7. USB	SHOULD			SHOULD	SHOULD
Output	Speaker and/or Audio output ports	7.8.2. Audio Output	MUST	MUST		MUST	MUST

- Default App Settings
- Android includes settings that provide users an easy way to select their default applications, for example for Home screen or SMS. Where it makes sense, device implementations MUST provide a similar settings menu and be compatible with the intent filter pattern and API methods described in the SDK documentation as below.
- Device implementations:
- MUST honor the android.settings.HOME_SETTINGS intent

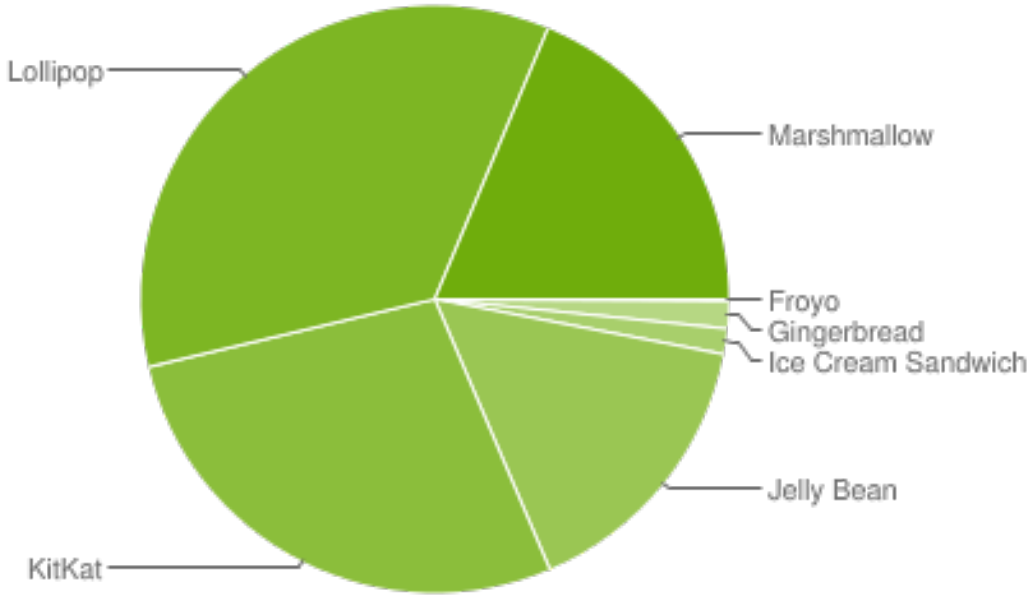
Android Version Distribution History



[Erikrespo CC BY-SA 3.0 https://en.wikipedia.org/wiki/Android_version_history]

Current Android Version Distribution

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	1.5%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	1.4%
4.1.x	Jelly Bean	16	5.6%
4.2.x		17	7.7%
4.3		18	2.3%
4.4	KitKat	19	27.7%
5.0	Lollipop	21	13.1%
5.1		22	21.9%
6.0	Marshmallow	23	18.7%



[https://developer.android.com/about/dashboards/index.html
Data collected during a 7-day period ending on September 5, 2016]