Mobile Application Development

By,

Prof. Hiten M Sadani

U. V. Patel College of Engineering, Ganpat University

History of Mobile Software Development

- Key Players in Mobile Development
  - Mobile Operators
  - Mobile developers
  - Handset Manufacturers

Different Platforms

- Palm OS /Garnet OS
- RIM BlackBerry
- J2ME
- BREW (Binary Runtime Environment for Wireless) (Qualcomm-CDMA, Brew is described as a pseudo operating system, but not a true mobile operating system)
- Symbian OS
- iPhone OS (The Apple iPhone OS)
- Android OS

Different Platforms

Different Platforms world wide

- https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/

Android Was Not A Google's Idea

- the company Android Inc. is established in October 2003. The founder of this company is Andy Rubin, Rich Miner, Chris White and Nick Sears. The main aim of this company is to create a smart mobile device that aware of the user wants.

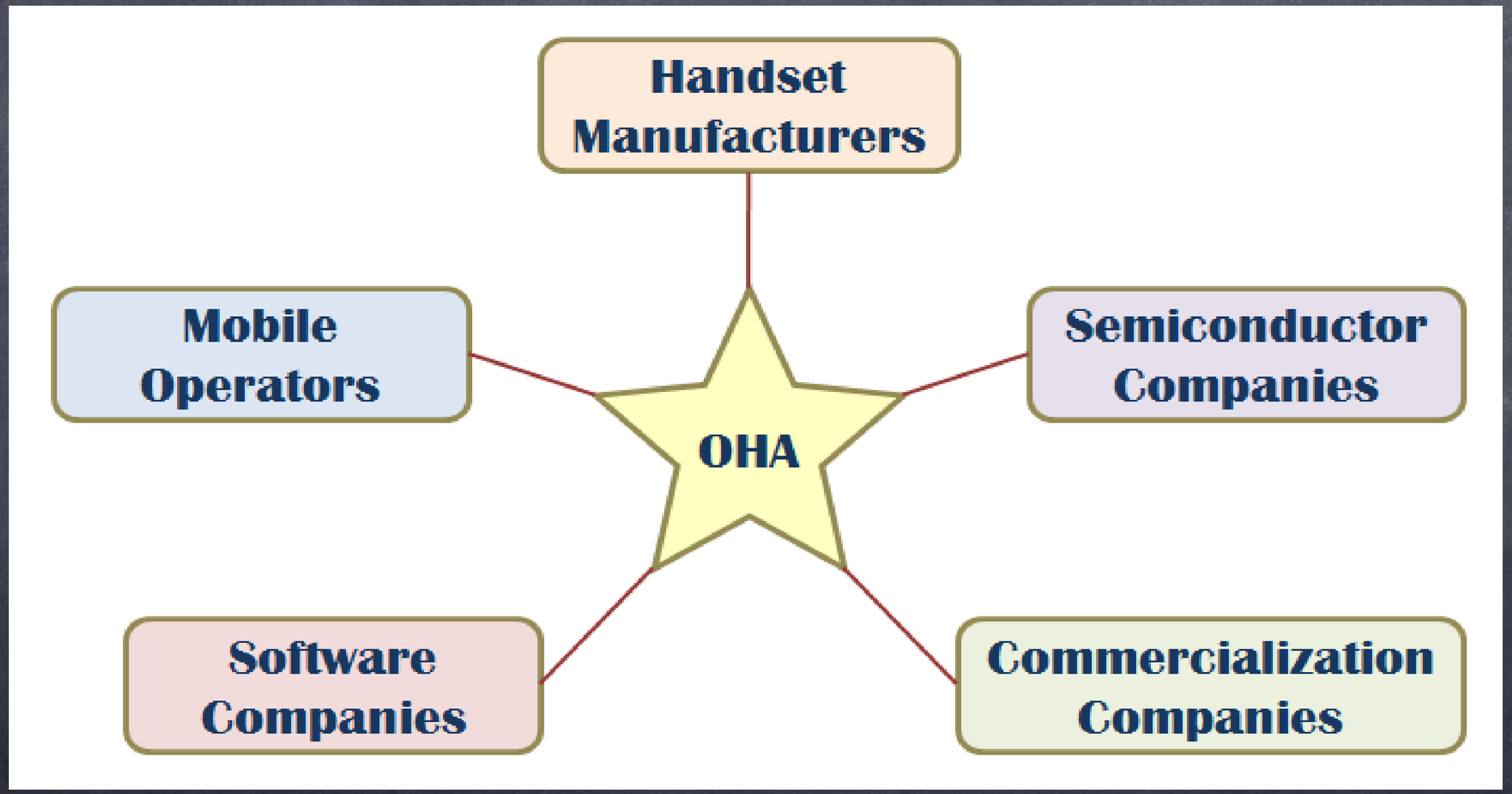
Android Was Not A Google's Idea

- In 2004 the first android operating system developed by Android with the help of Google. In 2005 Google purchase them for a $50 million and hired Andy Rubin to head the Android

Open Handset Alliance

- It’s consortium of several companies.
- This group of companies are allowed to use source code of Android and develop applications.
- Reason for Nokia not to develop Android Mobiles is Nokia is not part of OHA

Open Handset Alliance



What is an Android?

- Android is a mobile operating system based on a modified version of the Linux kernel and other open source software, designed primarily for touchscreen mobile devices such as smartphones and tablets

- A software stack for mobile devices that includes
  - An operating system
  - Middleware
    Key Applications
- Uses Linux to provide core system services
  - Security
  - Memory management
  - Process management
  - Power management
  - Hardware drivers

What is Android?

- Mobile operating system based on Linux kernel
- User Interface for touch screens
- Used on over 80% of all smartphones
- Powers devices such as watches, TVs, and cars
- Over 2 Million Android apps in Google Play store
- Highly customizable for devices / by vendors
- Open source

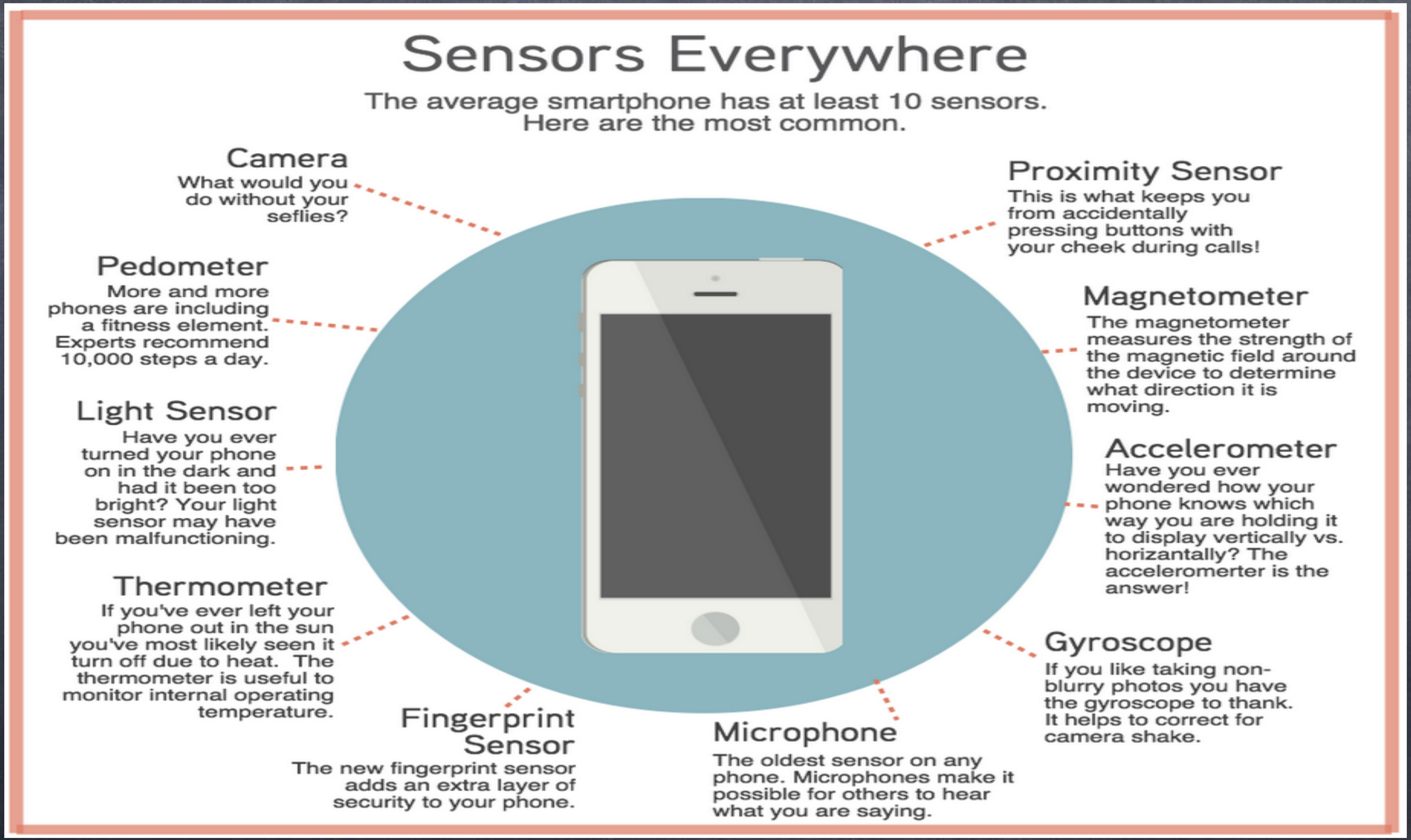
Android user interaction

- Touch gestures: swiping, tapping, pinching
- Virtual keyboard for characters, numbers, and emoji
- Support for Bluetooth, USB controllers and peripherals

Android and sensors

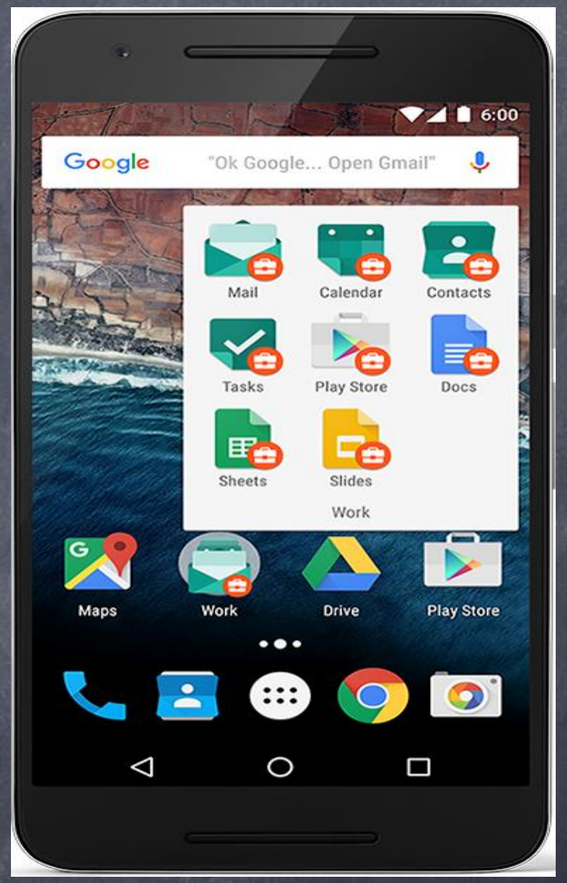
- Sensors can discover user action and respond
- Device contents rotate as needed
- Walking adjusts position on map
- Tilting steers a virtual car or controls a physical toy
- Moving too fast disables game interactions

Android and sensors

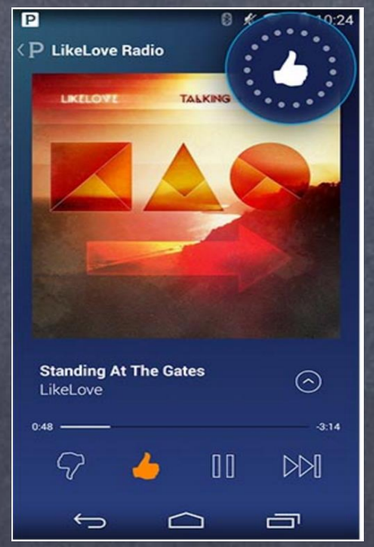


Android home screen

- Launcher icons for apps
- Self-updating widgets for live content
- Can be multiple pages
- Folders to organize apps
- "OK Google"

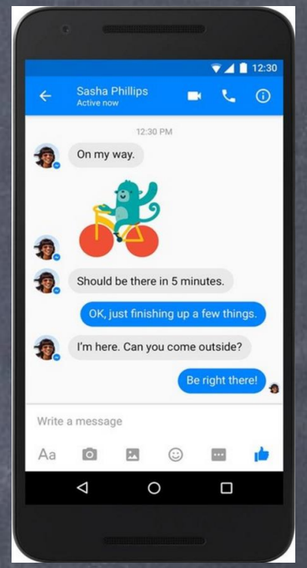


Android app examples



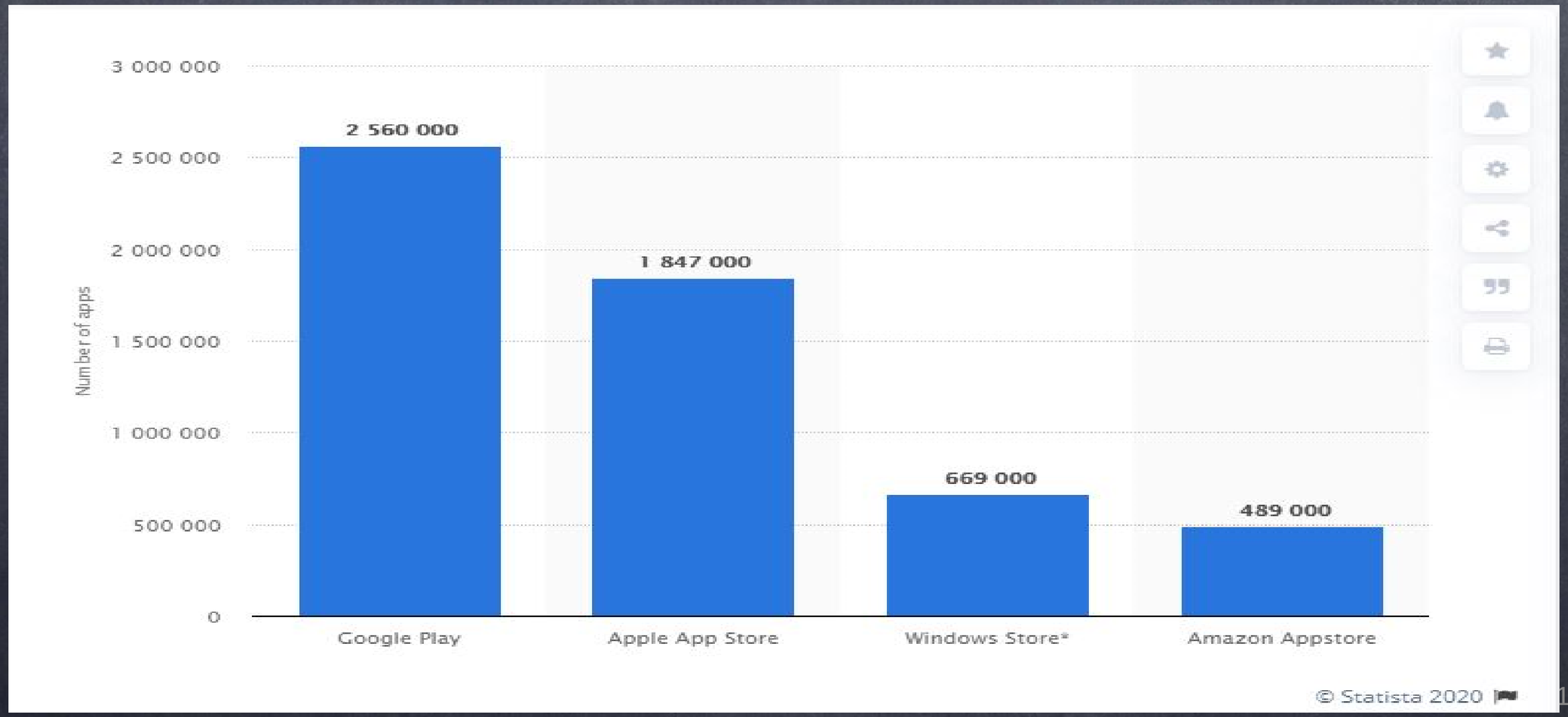
Pandora

Pokemon GO



Facebook Messenger

Number of App available in leading AppStore as 1st Quarter 2020.



History of Android

- **In October 2003, well before the term “smartphone” was used by most of the public, and several years before Apple announced its first iPhone and its iOS operating system, the company Android Inc was founded in Palo Alto, California**
- Its four founders were Rich Miner, Nick Sears, Chris White and Andy Rubin. At the time of its public founding, Rubin was quoted as saying that Android Inc was going to develop “smarter mobile devices that are more aware of its owner’s location and preferences.”

History of Android

2005	Google buys Android, Inc. Work on Dalvik starts
2007	Open Handset Alliance announced Early Software Development Kit
2008-2010	Android becomes the dominant mobile platform
2011	Games, Tablets, TVs
Future?	Beyond phones

History of Android

VERSIONS OF ANDROID

Name	Version number(s)	Initial release date	Play supported	API level
No codename	1.0	September 23, 2008	No	1
	1.1	February 9, 2009	No	2
Cupcake	1.5	April 27, 2009	No	3
Donut	1.6	September 15, 2009	No	4
Eclair	2.0 – 2.1	October 26, 2009	No	5 – 7
Froyo	2.2 – 2.2.3	May 20, 2010	No	8
Gingerbread	2.3 – 2.3.7	December 6, 2010	No	9 – 10
Honeycomb	3.0 – 3.2.6	February 22, 2011	No	11 – 13
Ice Cream Sandwich	4.0 – 4.0.4	October 18, 2011	No	14 – 15
Jelly Bean	4.1 – 4.3.1	July 9, 2012	No	16 – 18
KitKat	4.4 – 4.4.4	October 31, 2013	No	19 – 20
Lollipop	5.0 – 5.1.1	November 12, 2014	No	21 – 22
Marshmallow	6.0 – 6.0.1	October 5, 2015	No	23
Nougat	7.0 – 7.1.2	August 22, 2016	No	24 – 25
Oreo	8.0 – 8.1	August 21, 2017	Yes	26 – 27
Pie	9	August 6, 2018	Yes	28
Android 10	10	September 3, 2019	Yes	29
Android 11	Red Velvet Cake[21]	11	30	September 8, 2020	July 2022
Android 12	Snow Cone	**12**	31	October 4, 2021	
Android 12L	Snow Cone v2	**12.1**[a]	32	March 7, 2022	
Android 13	Tiramisu[23]	13[b]	33	Q3 2022	

Legend: Old version | Older version, still maintained | **Latest version** | Latest preview version

VERSIONS OF ANDROID

- **Android Astro/apple pi 1.0**
  - ✓ First full version of android.
  - ✓ Released on September 23, 2008.
  - ✓ Wi-Fi and Bluetooth support.
  - ✓ Quite slow in operating.
  - ✓ copy and paste feature in the web browser is not present.

VERSIONS OF ANDROID

- **Android Astro/apple pi 1.0**
  - ✓ First full version of android.
  - ✓ Released on September 23, 2008.
  - ✓ Wi-Fi and Bluetooth support.
  - ✓ Quite slow in operating.
  - ✓ copy and paste feature in the web browser is not present.

The First Android Smartphone Was HTC Dream or T-Mobile G1

VERSIONS OF ANDROID

VERSIONS OF ANDROID

- **Android Cupcake 1.5**
  - ✓ Released on April 30, 2009.
  - ✓ Added auto-rotation option.
  - ✓ Copy and Paste feature added in the web browser.
  - ✓ Increased speed and performance but not upto required level.

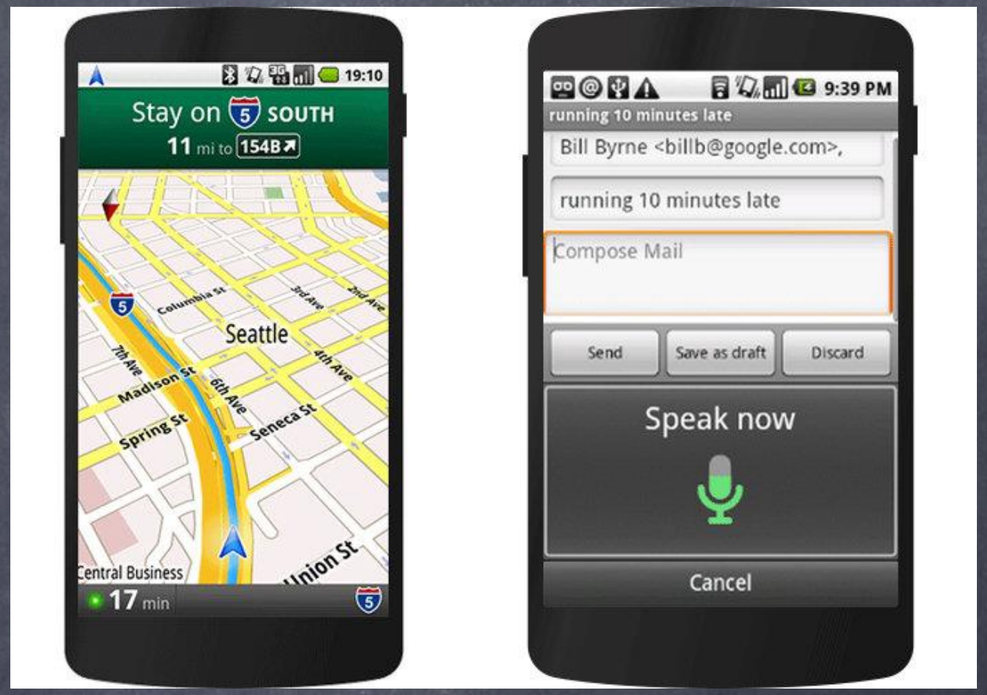
VERSIONS OF ANDROID

Android Donut 1.6

- ✓Released on September 15, 2009.
- ✓ Voice search and Search box were added.
- ✓ Faster OS boot times and fast web browsing experience.
- ✓ Typing is quite slower.
- ✓It also added support for CDMA networks like Verizon

VERSIONS OF ANDROID

- **Android Éclair 2.0/2.1**
  - ✓Released on October 26, 2009.
  - ✓Improved typing speed on virtual keyboard, with smarter dictionary.
  - ✓ no Adobe flash media support.
  - ✓First version of turn by turn naigation and speech to Text in Eclair.



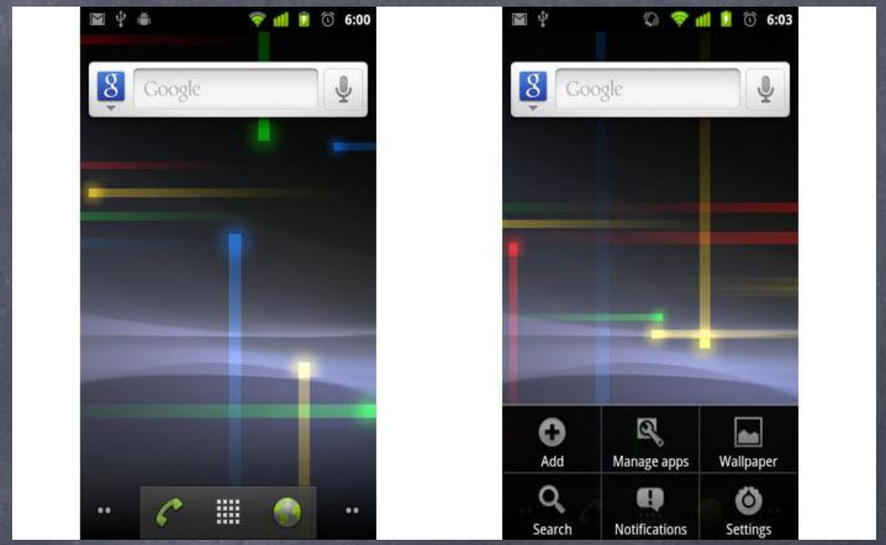
VERSIONS OF ANDROID

- **Android Froyo 2.2**
  - ✓ Released on May 20, 2010.
  - ✓ Support for Adobe Flash 10.1
  - ✓ Improved Application launcher with better browser
  - ✓ No internet calling.

VERSIONS OF ANDROID

Android Gingerbread 2.3

- ✓ Released on December 6, 2010.
- ✓ Updated User Interface with high efficiency and speed
- ✓ Internet calling
- ✓ One touch word selection and copy/paste.
- ✓ New keyboard for faster word input.
- ✓ More successful version of Android than previous versions.
- ✓ not supports multi-core processors.



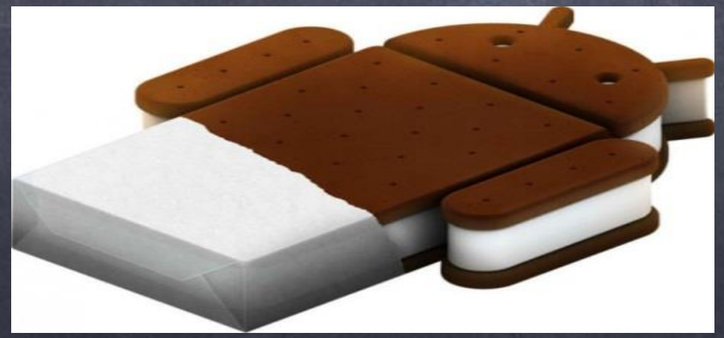
VERSIONS OF ANDROID

Android Honeycomb 3.0

- ✓Released on February 22, 2011.
- ✓Ability to encrypt all user data.
- ✓ This version of android is only available for tablets.

VERSIONS OF ANDROID

- **Android IceCreamSandwich(ICS) 4.0**
  - ✓ Released on November 14, 2011.
  - ✓ Virtual button in the UI.
  - ✓ A new typeface family for the UI, Roboto.
  - ✓ Ability to shut down apps that are using data in the background.



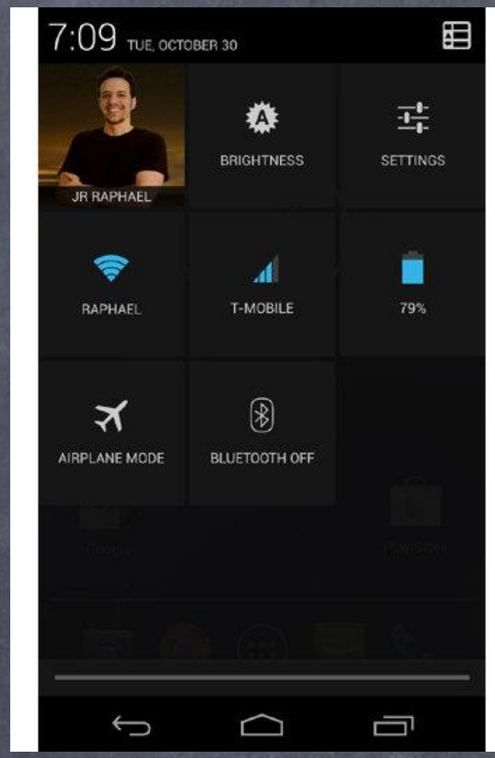
VERSIONS OF ANDROID

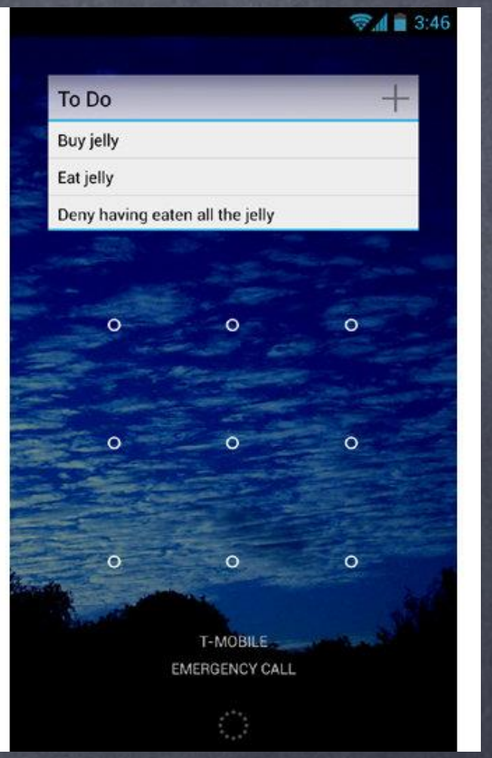
- **Android JellyBean 4.1 - 4.3**

✓Released on June 27, 2012.

- Enable Home screen rotation
- Fix bugs and enhance performances

✓ Smoother user interface.



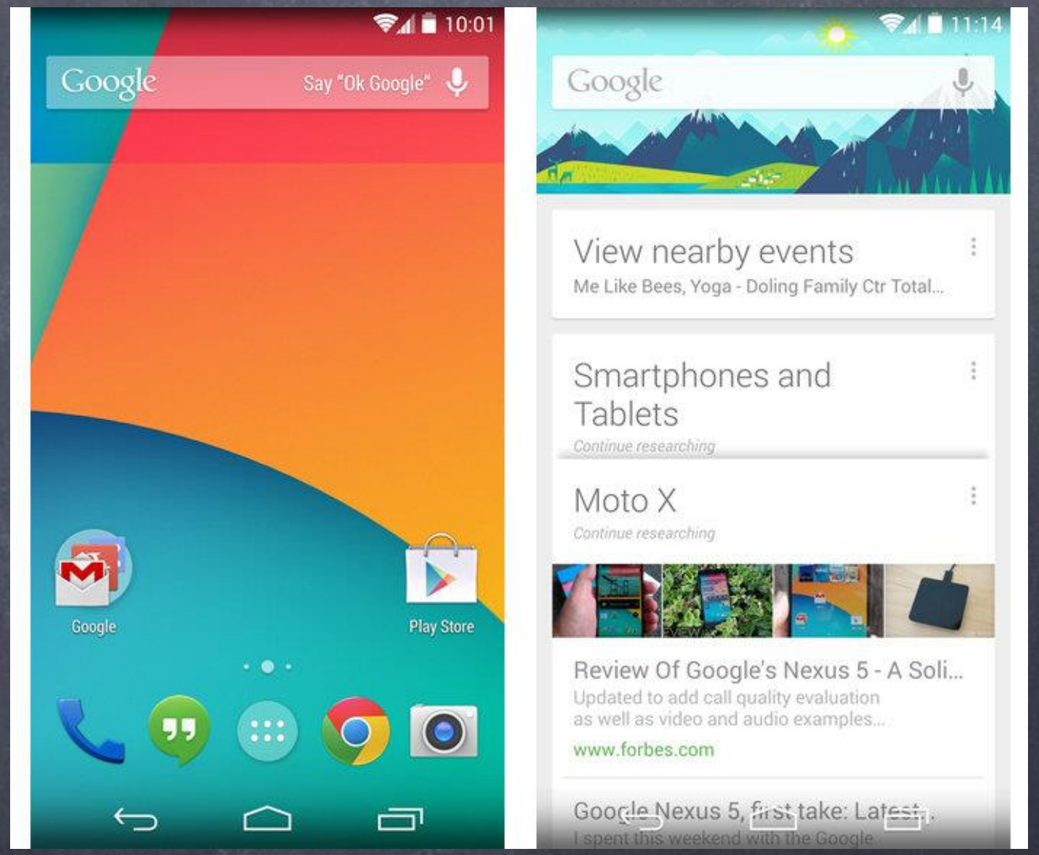


VERSIONS OF ANDROID

- **Android KitKet 4.4 -4.4.4**

✓Released on October 31, 2013.

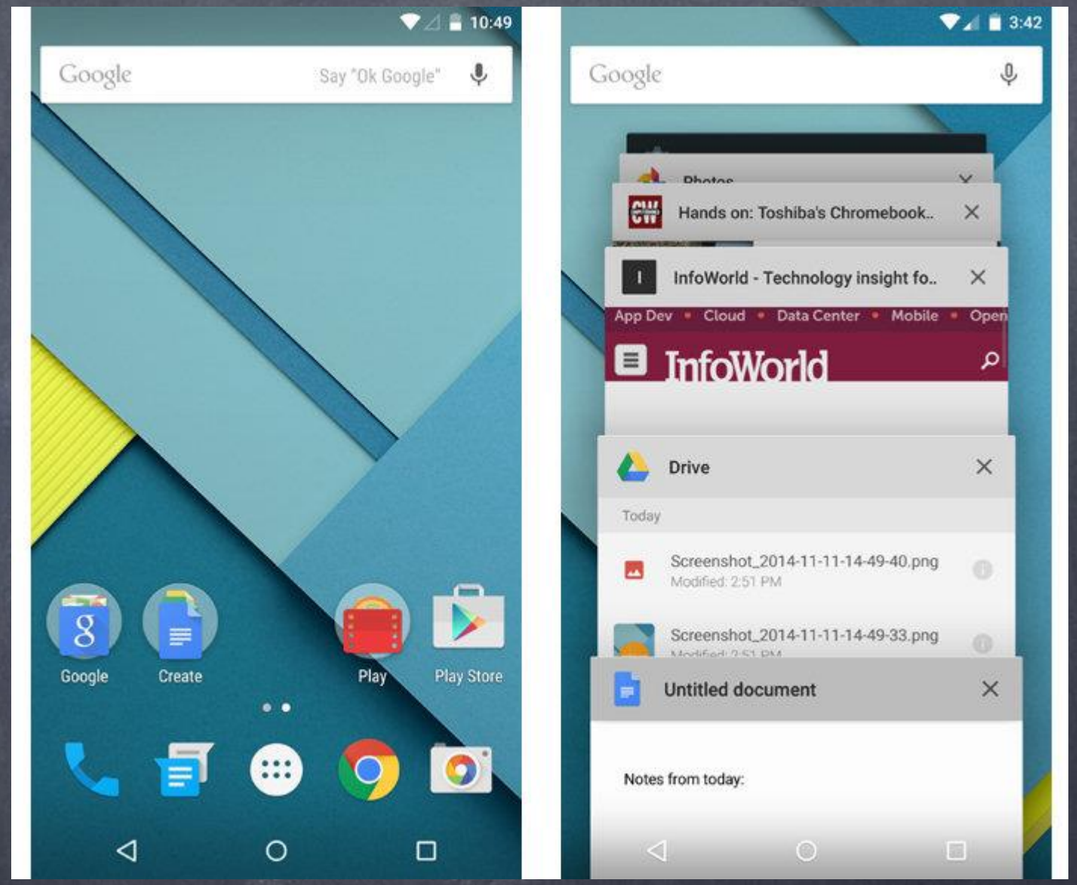
- Screen recording
- Enhanced notification access
- Performance improvements
- Public API for SMS management.
- Improved memory usage



VERSIONS OF ANDROID

VERSIONS OF ANDROID

- Android 5.0 and 5.1: Lollipop
- fall of 2014
- Lollipop launched the still-present-today Material Design standard
- The card-based concept that had been scattered throughout Android became a core UI pattern



VERSIONS OF ANDROID

- **Android 5.1**
- **9 March 2015**

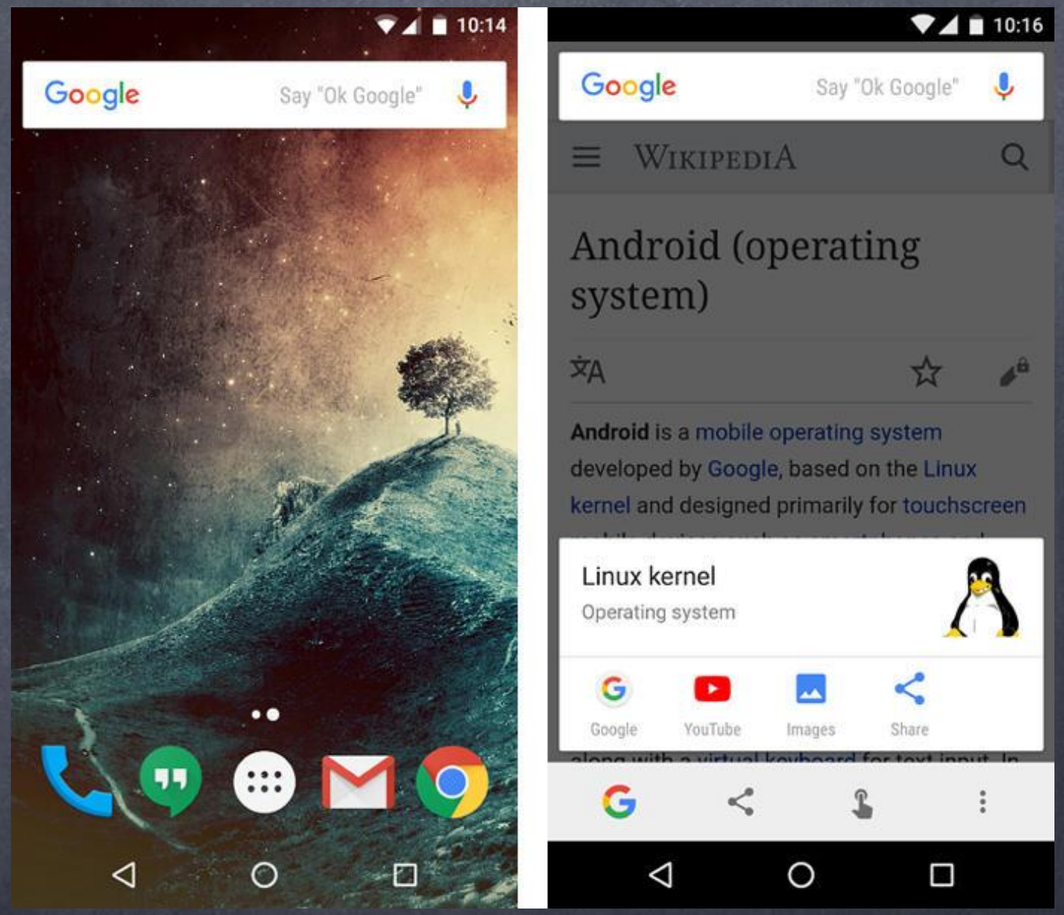
- Ability to join Wi-Fi networks and control paired Bluetooth devices from quick settings
- Official support for multiple SIM cards
- Device protection: if a device is lost or stolen it will remain locked until the owner signs into their Google account, even if the device is reset to factory settings.
- High-definition voice calls, available between compatible devices running Android 5.1

- **Android 5.1.1**
- **21 April 2015**

- **Fixes a memory leak issue introduced with the Android 5.0 "Lollipop" release**
- **Various other bug fixes**

VERSIONS OF ANDROID

- Android version 6.0: Marshmallow
  - Marshmallow's most attention-grabbing element was a screen-search feature called Now On Tap

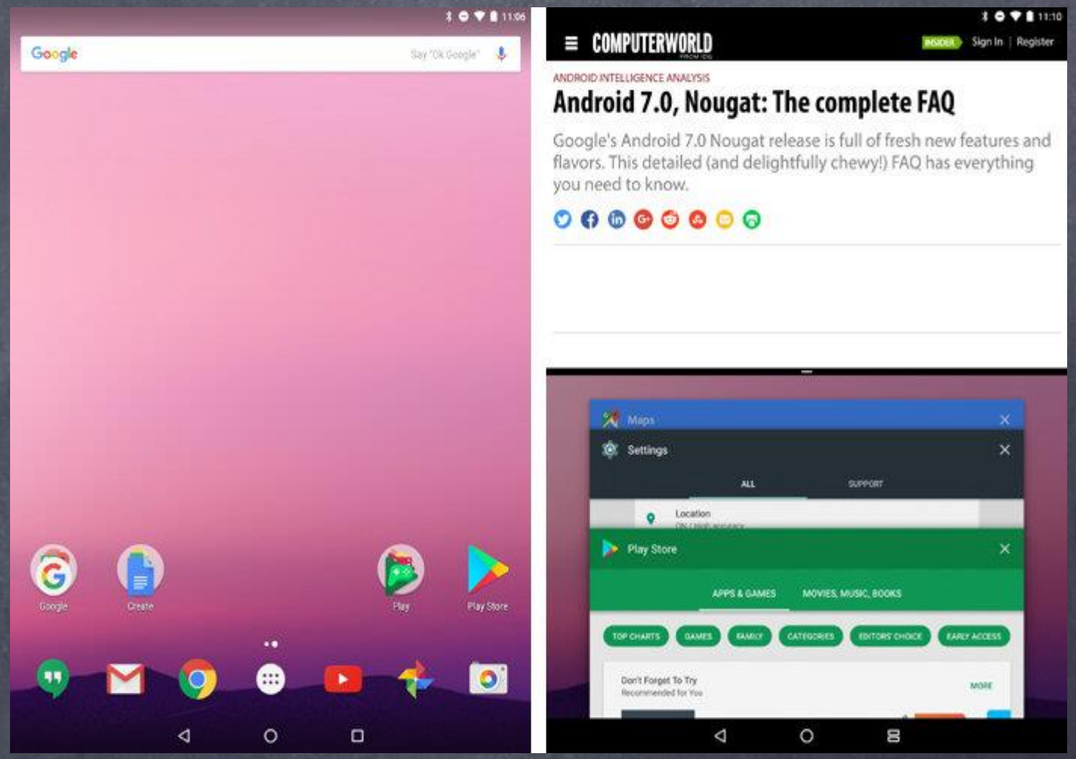


VERSIONS OF ANDROID

- **Android 6.0 Marshmallow**
  Released in the fall of 2015
  - It included features such a new vertically scrolling app drawer, along with Google Now on Tap, native support for fingerprint biometric unlocking of a smartphone, USB Type-C support, the introduction of Android Pay, and much more.
  - The first devices that shipped with Marshmallow pre-installed were Google's Nexus 6P and Nexus 5X smartphones, along with its Pixel C tablet. The current stats on Android platform use shows that Marshmallow has just marginally overtaken Lollipop as the most installed OS version, accounting for 32.2 percent of all Android-based devices.

VERSIONS OF ANDROID

- Android version 7.0 and 7.1: Nougat
  - provided Android with a native split-screen mode, a new bundled-by-app system for organizing notifications, and a Data Saver feature.



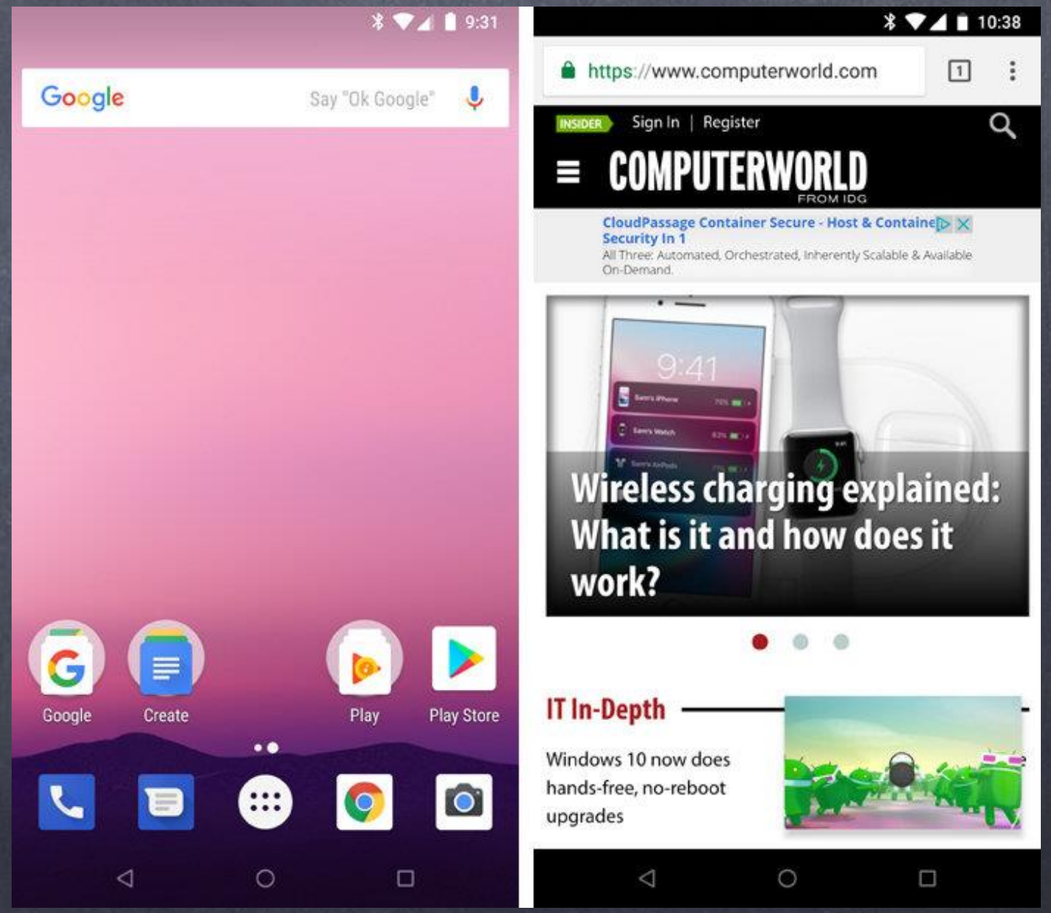
VERSIONS OF ANDROID

- **Android 7.0 NOUGAT**
  launched in the fall of 2016
  - better multi-tasking functions for the growing number of smartphones that have bigger displays, such as split-screen mode, along with quick switching between apps.
  - switching to a **new JIT compiler to speed up apps**, supported the Vulkan API for faster 3D rendering, and enabled OEMs to support its DayDream Virtual Reality platform.
  - Google also used the release to make a bold push into the premium smartphone market. The company's own branded smartphones, the Pixel and Pixel XL, along with the LG V20, were the first to be released with Nougat pre-installed. In early July 2016, Google's Android platform stats page showed that Nougat is now installed on 15.8 percent of all Android-based devices.

VERSIONS OF ANDROID

- Android version 8.0 and 8.1: Oreo

  - Android Oreo added a variety of niceties to the platform, including a native picture-in-picture mode, a notification snoozing option.



VERSIONS OF ANDROID

- **Android 8.0 OREO**
  - launched in the August 2017
  - As far as its features, Android Oreo packs in lots of visual changes to the Settings menu, along with native support for picture-in-picture mode, notification channels, new autofill APIs for better management of passwords and fill data, and much more.
  - Android Oreo is available today as a download via Google's Android Open Source Project, and is also available as an over-the-air update for Google's older (and supported) Nexus and Pixel devices.

VERSIONS OF ANDROID

- Android version 9: Pie

  - Pie's most transformative change was its hybrid gesture/button navigation system.

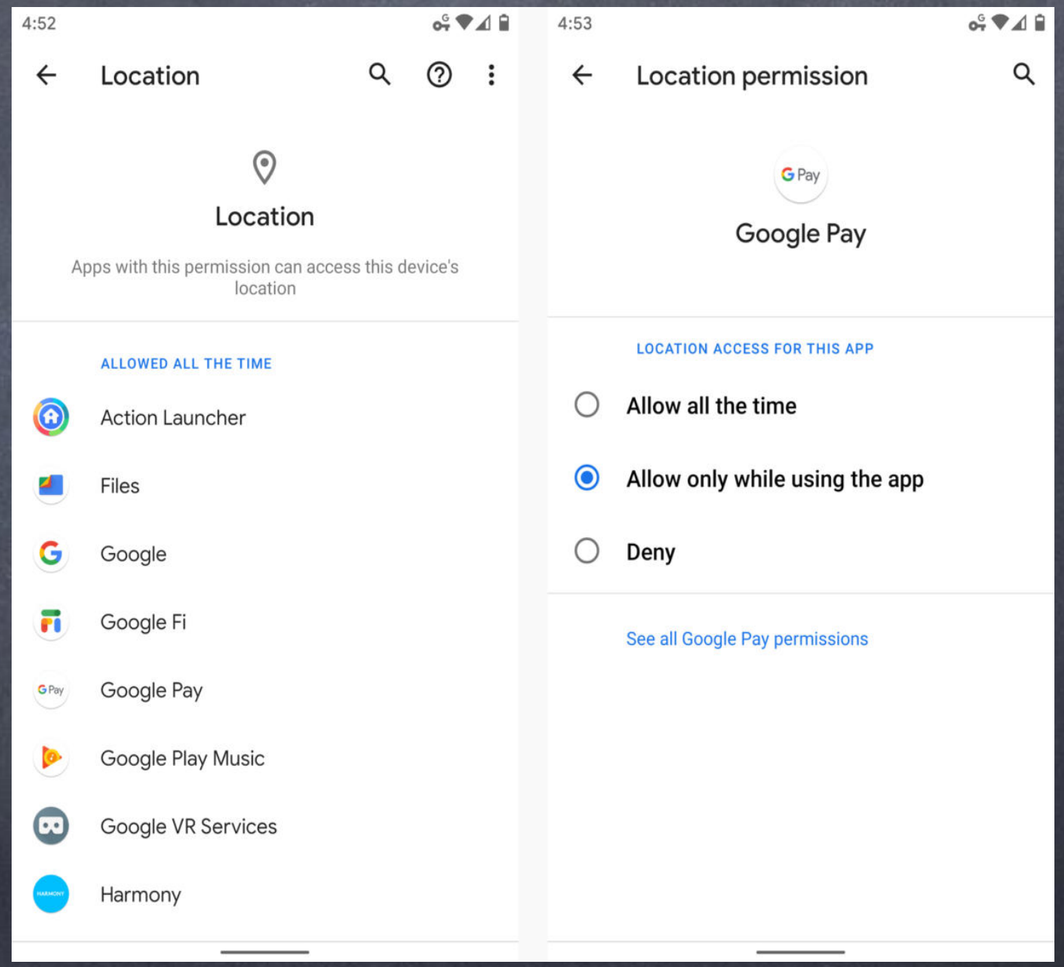
VERSIONS OF ANDROID

- **Android 9.0 Pie**
  - launched in the August 2018
  - came with a lot of new features and improvements
  - The new home-button was added in this version
  - When swiped up, it brings the apps used recently, a search bar and suggestions of five apps at the bottom. There was a new option added of swiping left to see the currently running apps
  - Improvements in battery life were also made in this version
  - It automatically puts the smartphone in Do not disturb mode. Many more features were also added

VERSIONS OF ANDROID

- Android version 10: Q

  - Android 10 includes a system-wide dark theme, a new Focus Mode that lets you limit distractions from specific apps with the tap of an on-screen button.



VERSIONS OF ANDROID

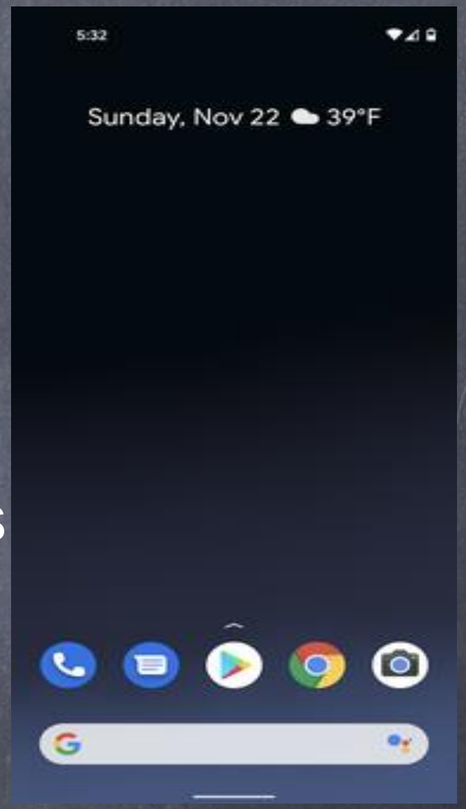
Android 10.0 Q

- launched in the September 3, 2019
- includes a new logo and. Finally, Google opted to drop the tradition of naming the Android version after sweets and desserts
- A number of features were added such as support for the upcoming foldable smartphones with flexible displays
- Android 10 also has a system-wide dark mode, along with the newly introduced navigation control using gestures, the feature for smart reply for all the messaging apps, and a sharing menu that is more effective. The control over app-based permissions is also more in it.

VERSIONS OF ANDROID

Android 11.0

- launched in the September 8, 2020
- Chat bubbles.
- Screen recorder.
- Notification history.
- New permissions.
- Increased number of updatable core OS components in Google Play from 12 to 21



VERSIONS OF ANDROID

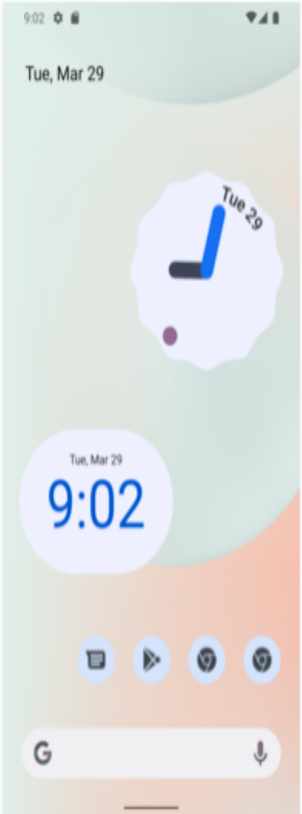
Android 12.0

- TBA (Feb 18-2021)
- Easier Wi-Fi sharing.
- AVIF image support.
- Material You, an updated design language
- based off of Material Design.
- Android Runtime (ART) module added to the
- updatable core OS components via Google Play

VERSIONS OF ANDROID

Android 12 (API 31) [hide]

Android 12 is the twelfth major version of the Android operating system. It was first announced by Google on February 18, 2021, and the first developer preview was released on the same day.[279][280]

Version	Release date	Features	Image(s)
12	October 4, 2021	• Easier Wi-Fi sharing. • AVIF image support. • Material You, an updated design language based on Material Design. • Scrolling Screenshot.[281] • One Handed Mode.[282] • Android Runtime (ART) module added to the updatable core OS components via Google Play, added functionality to existing modules.[283][284] • Area Magnification can zoom in any content on device.[285] • Extra Dim reduces brightness below minimum level.[286] • Bold Text. • Greyscale. • Mic and Camera indicator and toggle.[citation needed] • Option to choose precise or approximate location.[287] • Privacy Dashboard.[288] • Gestures can work in immersive mode.[289] • Performance improvements to system services to improve transitions, power efficiency, and reduce app startup times.[290] • Third party app stores now have the ability to update apps without constantly asking the user for permission.[291]	 Android 12 home screen with themed icons and widgets

VERSIONS OF ANDROID

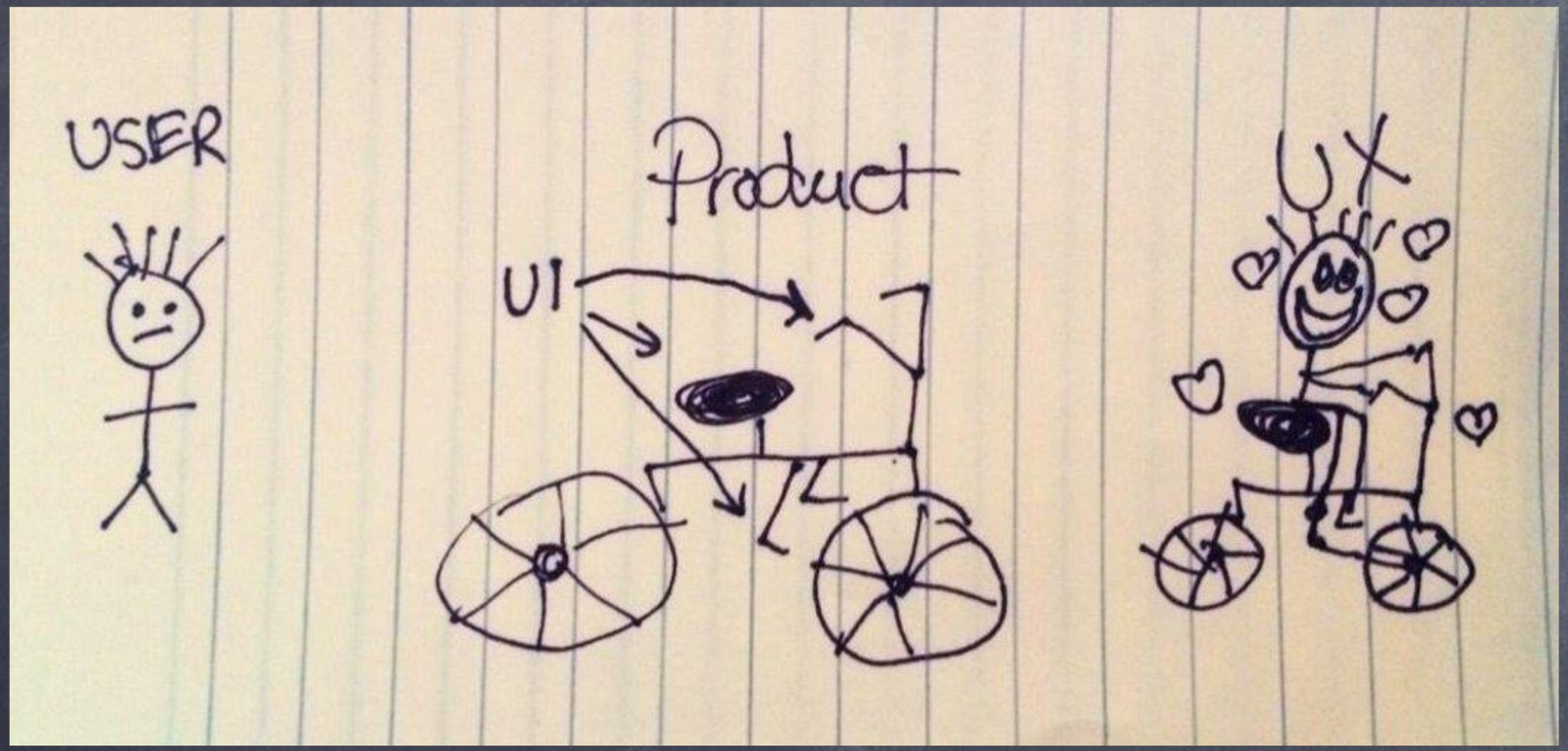
Add-ons

Add-Ons
htc
quietly brilliant
SenseUI
SAMSUNG
TouchWiz
MotoBlur

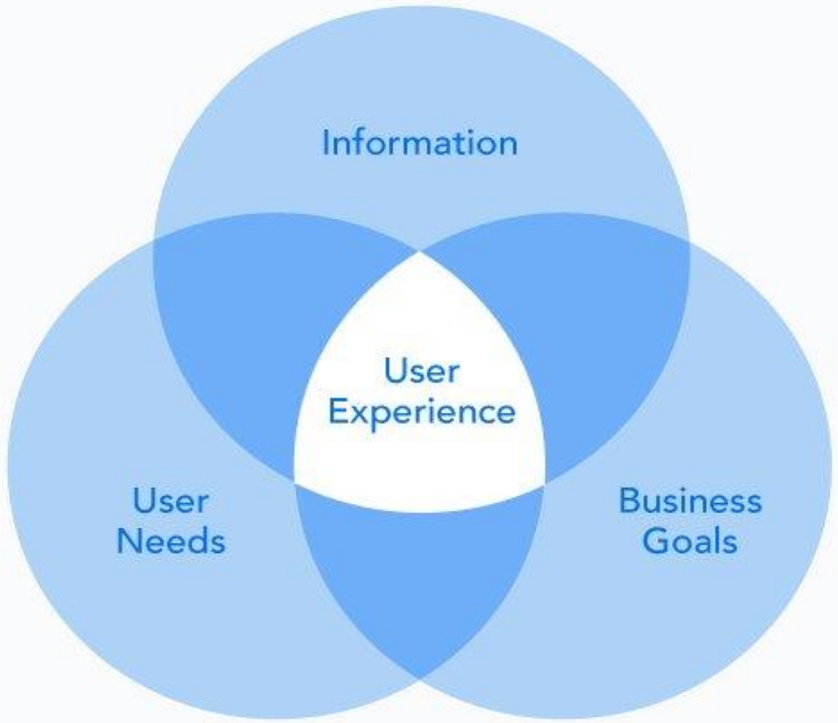
Platform Version

ANDROID PLATFORM VERSION	API LEVEL	CUMULATIVE DISTRIBUTION
4.1 Jelly Bean	16	
4.2 Jelly Bean	17	99.8%
4.3 Jelly Bean	18	99.5%
4.4 KitKat	19	99.4%
5.0 Lollipop	21	98.0%
5.1 Lollipop	22	97.3%
6.0 Marshmallow	23	94.1%
7.0 Nougat	24	89.0%
7.1 Nougat	25	85.6%
8.0 Oreo	26	82.7%
8.1 Oreo	27	78.7%
9.0 Pie	28	69.0%
10. Q	29	50.8%
11. R	30	24.3%

UI and UX



UI and UX



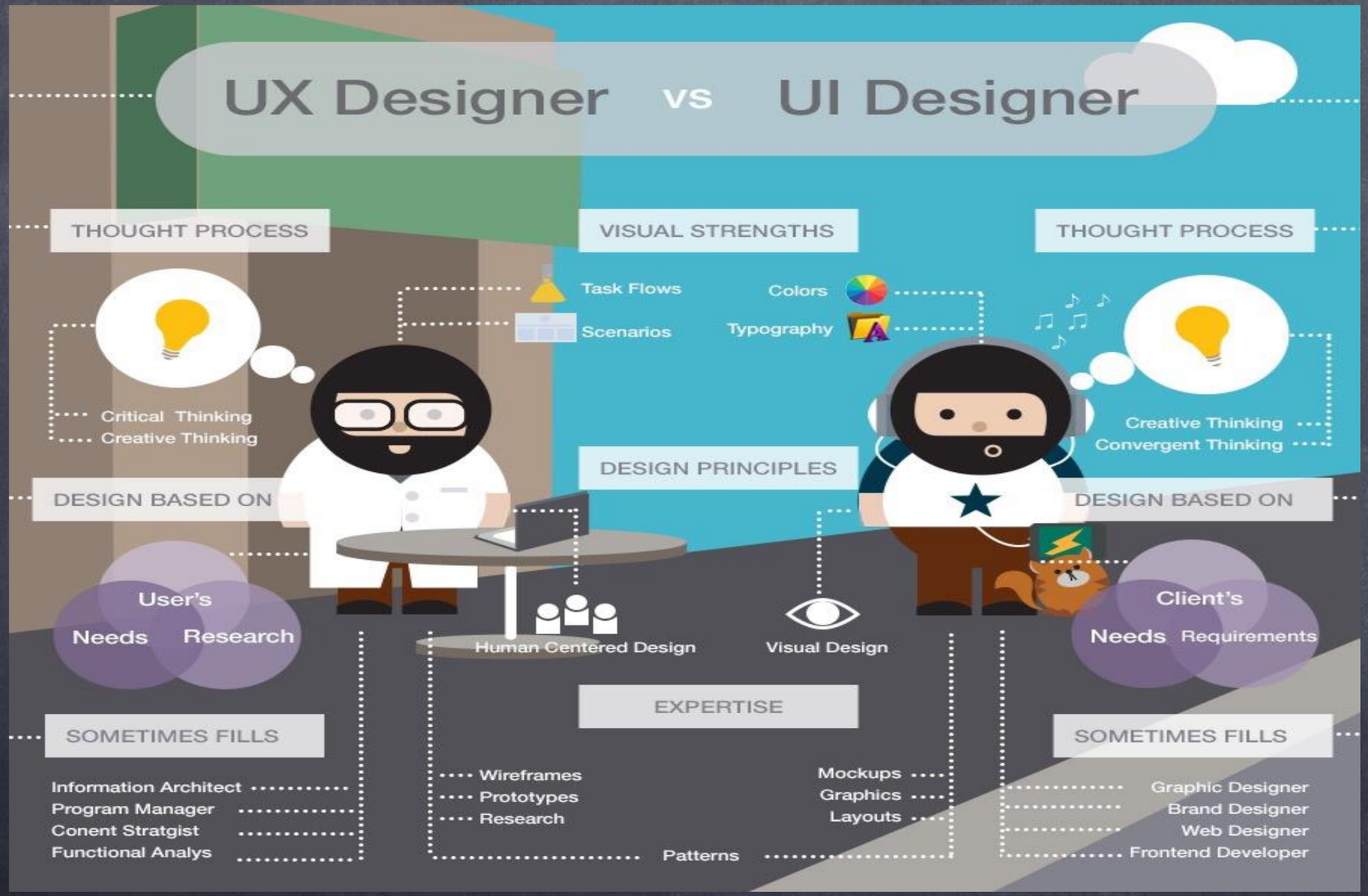
UI and UX

- *User Experience (UX) and User Interface (UI) are some of the most confused and misused terms in our field.*

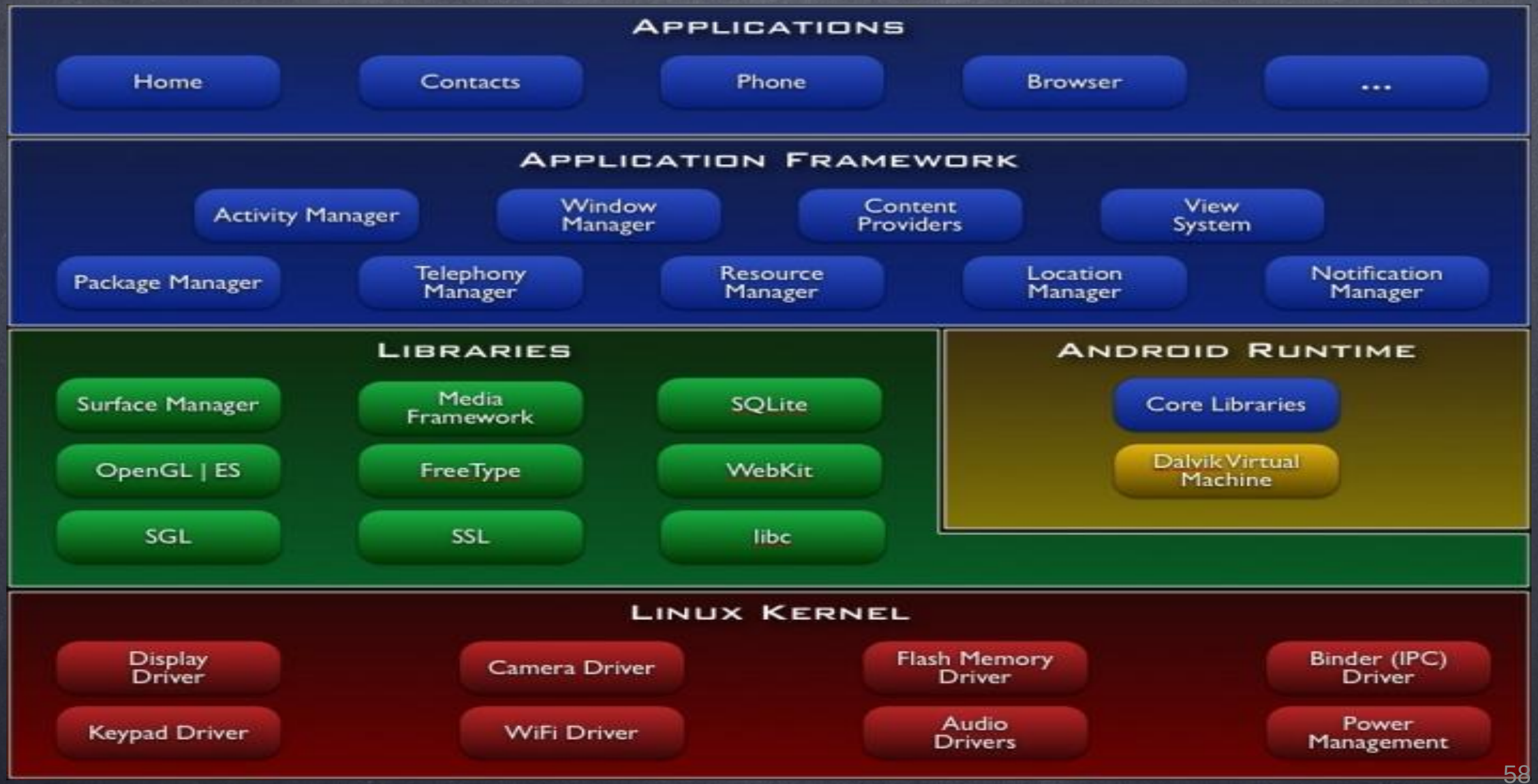
  A UI without UX is like a painter slapping paint onto canvas without thought; while UX without UI is like the frame of a sculpture with no paper match on it.

  A great product experience starts with UX followed by UI. Both are essential for the product's success

UI and UX


UX Designer vs UI Designer
THOUGHT PROCESS
VISUAL STRENGTHS
THOUGHT PROCESS
Task Flows
Scenarios
Colors
Typography
Critical Thinking
Creative Thinking
Creative Thinking
Convergent Thinking
DESIGN PRINCIPLES
DESIGN BASED ON
DESIGN BASED ON
User's
Needs
Research
Client's
Needs
Requirements
Human Centered Design
Visual Design
EXPERTISE
SOMETIMES FILLS
SOMETIMES FILLS
Information Architect
Program Manager
Conent Stratgist
Functional Analys
Wireframes
Prototypes
Research
Mockups
Graphics
Layouts
Patterns
Graphic Designer
Brand Designer
Web Designer
Frontend Developer

ANDROID Platform Architecture



Linux Kernel

- Works as a HAL
- Device drivers
- Memory management
- Process management
- Networking

Linux Kernel

- At the bottom of the Android stack is the Linux Kernel. It never really interacts with the users and developers, but is at the heart of the whole system. Its importance stems from the fact that it provides the following functions in the Android system:

- •Hardware Abstraction
•Memory Management Programs
•Security Settings
•Power Management Software
•Other Hardware Drivers (Drivers are programs that control hardware devices.)
•Support for Shared Libraries
•Network Stack

Linux Kernel

The Android system uses a binder framework for its Inter-Process Communication (IPC) mechanism. The binder framework was originally developed as OpenBinder and was used for IPC in BeOS.

Linux kernels and its Android version names

Android Version	Android version name	API Level	Linux kernel
1.0	Apple pie	1	2.6.25
1.1	Banana bread	2	2.6.25
1.5	Cupcake	3	2.6.27
1.6	Donut	4	2.6.29
2.0	Eclair	5	2.6.29
2.0.1	Eclair	6	2.6.29
2.1	Eclair	7	2.6.29
2.2	FroYo	8	2.6.32
2.3	Gingerbread	9	2.6.35
(2.3.3) - (2.3.7)	Gingerbread	10	2.6.35
3	Honeycomb	11	2.6.36
3.1	Honeycomb	12	2.6.36
3.2, 3.2.1, 3.2.2, 3.2.4, 3.2.6	Honeycomb	13	2.6.36
4.0, 4.0.1, 4.0.2	Ice Cream Sandwich	14	3.0.1
4.0.3, 4.0.4	Ice Cream Sandwich	15	3.0.1
4.1, 4.1.1, 4.1.2	Jelly Bean	16	3.0.31
4.2, 4.2.1, 4.2.2	Jelly Bean	17	3.4.0
4.3	Jelly Bean	18	3.4.0
4.4, (4.4.1) – (4.4.4)	KitKat	19	3.10
4.4	KitKat Wearable Extensions	20	3.10
5.0, 5.0.1, 5.0.2	Lollipop	21	3.16.1
5.1, 5.1.1	Lollipop	22	3.16.1
6.0, 6.0.1	Marshmallow	23	3.18.10

Native Libraries Layer

- Libraries carry a set of instructions to guide the device in handling different types of data. For instance, the playback and recording of various audio and video formats is guided by the Media Framework Library.

Native Libraries Layer

- Open Source Libraries:
•Surface Manager: composing windows on the screen
•SGL: 2D Graphics
•Open GL|ES: 3D Library
•Media Framework: Supports playbacks and recording of various audio, video and picture formats.
•Free Type: Font Rendering
•WebKit: Browser Engine
•libc (System C libraries)
•SQLite
•Open SSL

Android Runtime

- Dalvik VM
  - Register-based VM
  - Optimized for low memory requirements
  - Designed to allow multiple VM instances to run at once
  - Every program runs in its own process / own VM instance
  - Relies on the underlying OS for process isolation,
  - memory management and threading support
  - Operates on DEX files

Core Libraries

– Java 5 Std edition

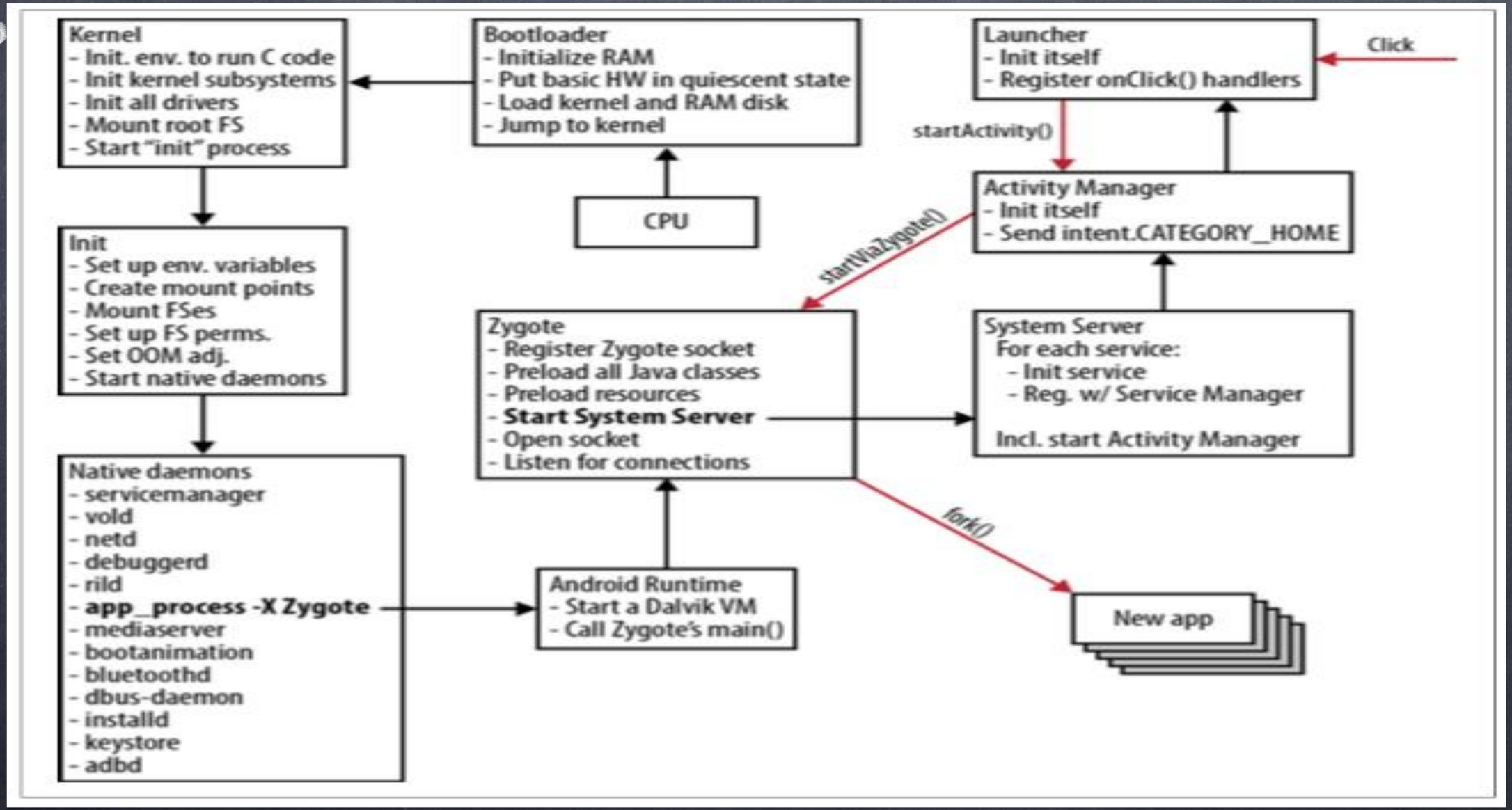
– Collections, I/O etc…

Android Runtime

- Zygote is a special process in Android which handles the forking of each new application process.

  It is launched by the Android runtime, which also starts the first Virtual Machine (VM). The VM then calls Zygote's main() method which causes Zygote to preload all shared Java classes and resource into memory

Android Runtime



Android Runtime

- **Bootrom** is a small piece of write-protected flash rom memory embedded inside the processor chip. It contains the very first code that is executed by the processor when it powers up or resets.

- **BootLoader** is started by the bootrom, it's job is to execute any specific setup before starting the Kernel, which literary means to copy os files to working memory.

- **Kernel** will start the setup cache, protected memory, scheduling and loads drivers. When the kernel finishes the system setup, the first thing it does is look for "**init**" in the system files and launch the root process or the first process of the system.

Android Runtime

- **init** is a root process. It has two responsibilities, mount directories like /sys, /dev, /proc and runs the init.rc script that starts, among other things, native daemons like Service Manager, Media Server etc..

- **The Android runtime** is started by init root process with the app_process command, which tells it to start Art or the Dalvik process virtual machine and to call Zygote's main() function.

- **Art/Dalvik** are process virtual machines. Dalvik is used on devices below Lollipop where it is replaced by Art. The biggest difference between them is that Dalvik uses JIT (just in time) and ART uses AOT (ahead of time) compilation.

Android Runtime

- **Zygote** is a special Android OS process that enables shared code across Dalvik/Art VM in contrast with Java VM where each instance has its own copy of core library class files and heap objects.

- Efficient and fast app launch is achieved thanks to the fact that **Zygote** starts by preloading all classes and resources which an app may potentially need at runtime into the system's memory. It then listens for connections on its socket for requests to start new apps. When it gets a request to start an app, it forks itself and launches the new app. It serves as a parent to all Android apps.

Android Runtime- Zygote

- This forking feature comes from the Linux kernel implementation of copy-on-write resource management technique.
- Forking involves creating a new process that is an exact copy of the parent process. It doesn't actually copy anything, instead it maps pages of the new process over to those of the parent process and makes copies only when the new process writes to a page.
- All processes started from **Zygote** use their own copy and only one copy of the system classes and the resources that are already loaded in the system memory.
- **Zygote** is designed not only to respond to fork requests by new apps, but there is one process that Zygote actually starts explicitly and that is the System Server process.

Android Runtime- Zygote

- **System Server** is the first process started by Zygote. After it starts, it lives on as an entirely separate process from its parent. It starts initializing each system service it houses and registering them with the previously started Service Manager. It also starts the Activity Manager.

- **Activity Manager** is responsible for a new Activity thread process creation, maintenance of the Activity lifecycle and managing the Activity stack. By the end of its startup it also executes an Intent to start a Home Launcher over which it receives onClick events to start new apps from icons on the home screen.

Application Framework Layer

- Our applications directly interact with these blocks of the Android architecture. These programs manage the basic functions of phone like resource management, voice call management etc.

Application Framework Layer

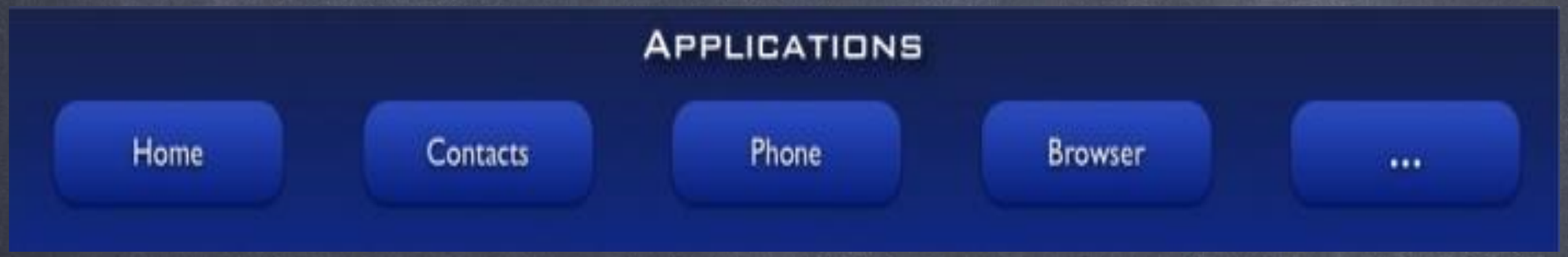
- Enabling and simplifying the reuse of components
  - ✓Developers have full access to the same framework APIs used by the core applications.
  - ✓Users are allowed to replace components

Application Framework Layer

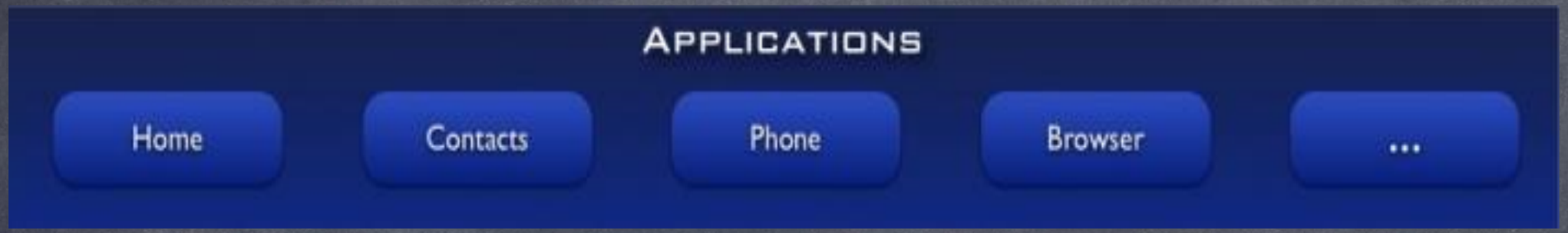
Feature	Role
View System	Used to build an application, including lists, grids, text boxes, buttons, and embedded web browser
Content Provider	Enabling applications to access data from other applications or to share their own data
Resource Manager	Providing access to non-code resources (localized strings, graphics, and layout files)
Notification Manager	Enabling all applications to display customer alerts in the status bar
Activity Manager	Managing the lifecycle of applications and providing a common navigation backstack

Application Layer

- **The applications are at the topmost layer of the Android stack**. An average user of the Android device would mostly interact with this layer (for **basic functions, such as making phone calls, accessing the Web browser** etc.). **The layers further down are accessed mostly by developers, programmers** and the likes.

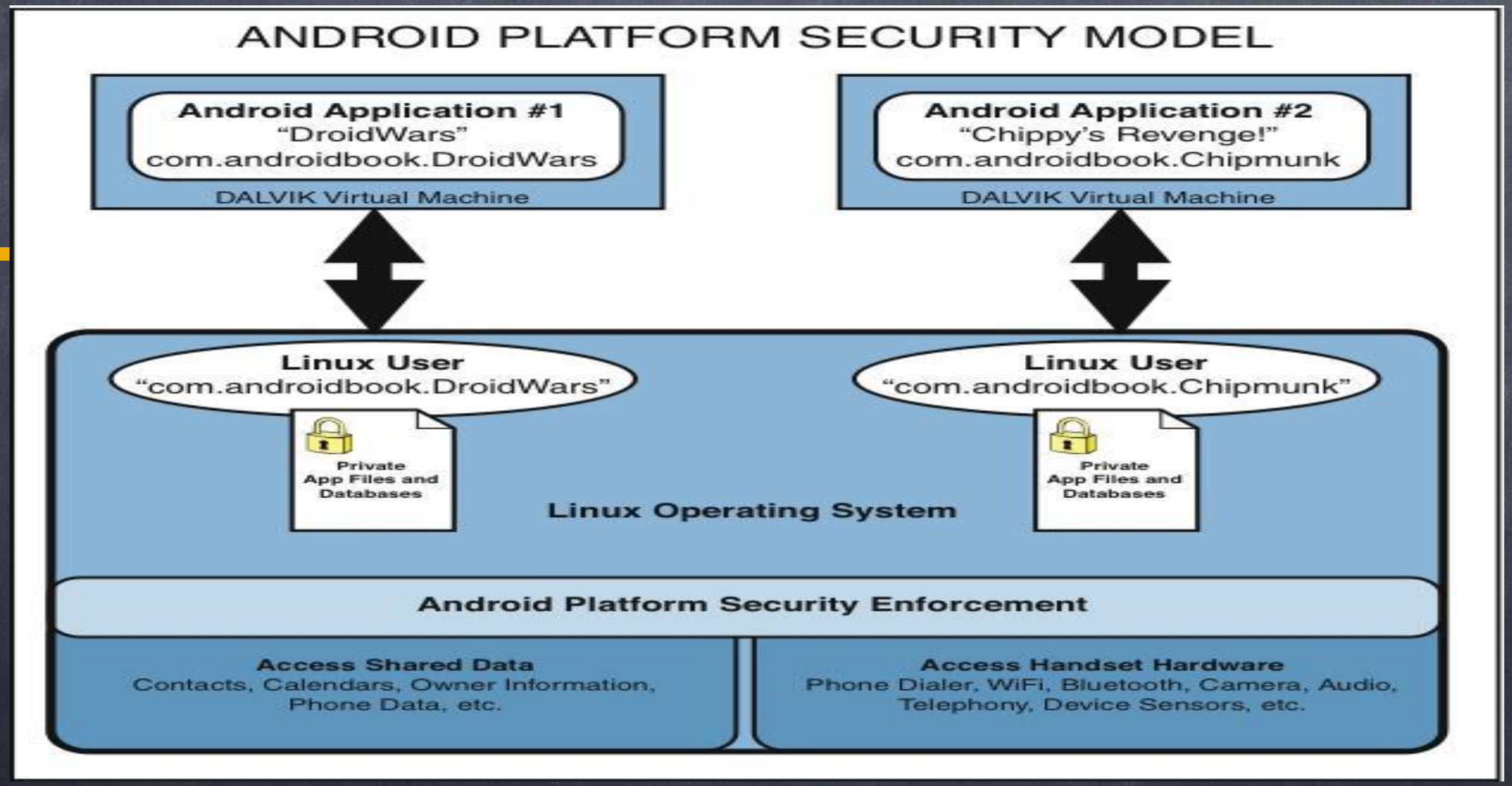


Application Layer



- Android provides a set of core applications:
  - ✓ Email Client
  - ✓ SMS Program
  - ✓ Calendar
  - ✓ Maps
  - ✓ Browser
  - ✓ Contacts
  - ✓ Etc
- All applications are written using the Java/kotlin language.

The Android Platform

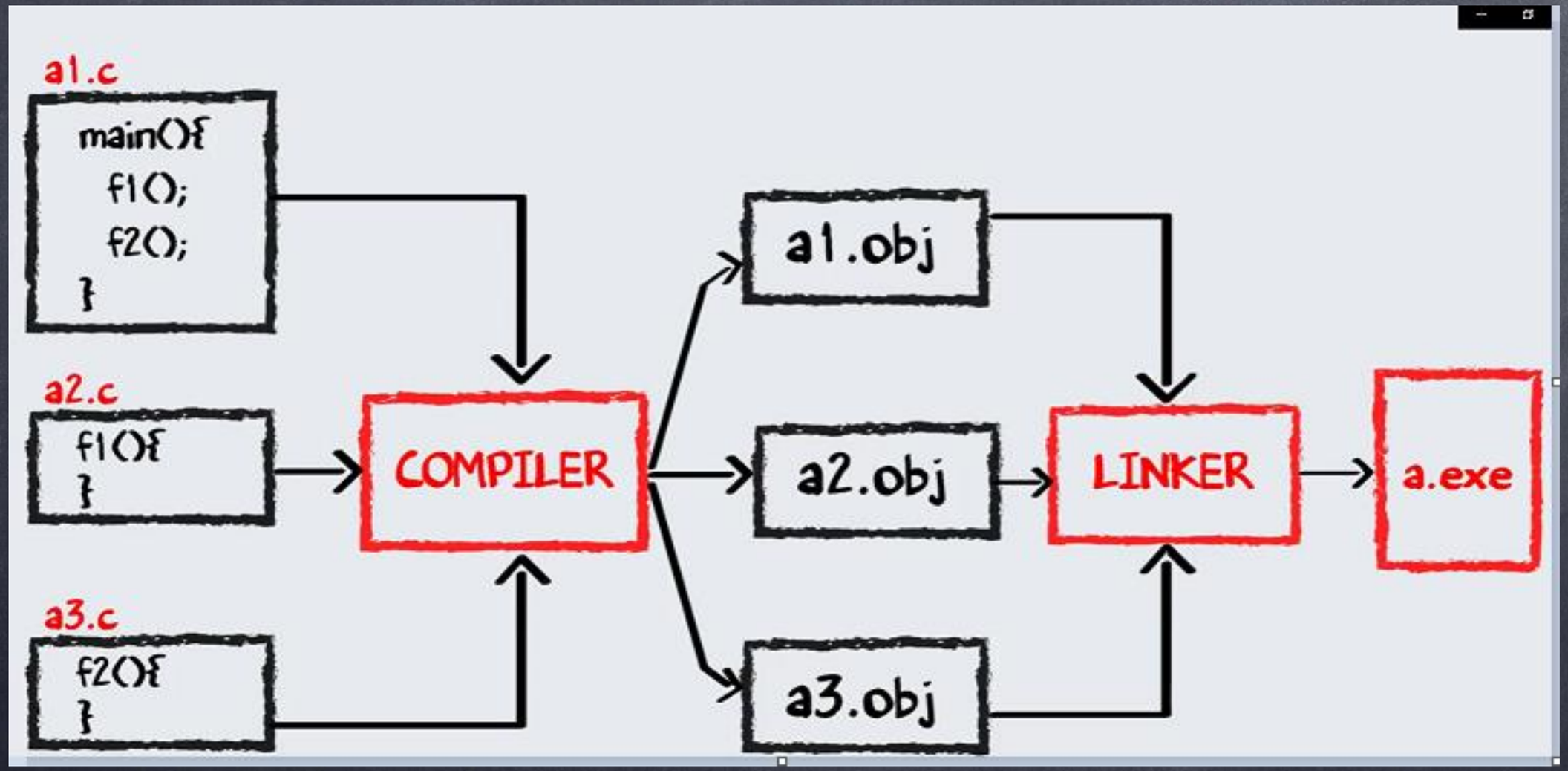


Process of compiling Android app with Java/Kotlin code

- Android apps can be written in Java and Kotlin programming languages. So the process of Android app compilation is based on the compilation process of Java and Kotlin code

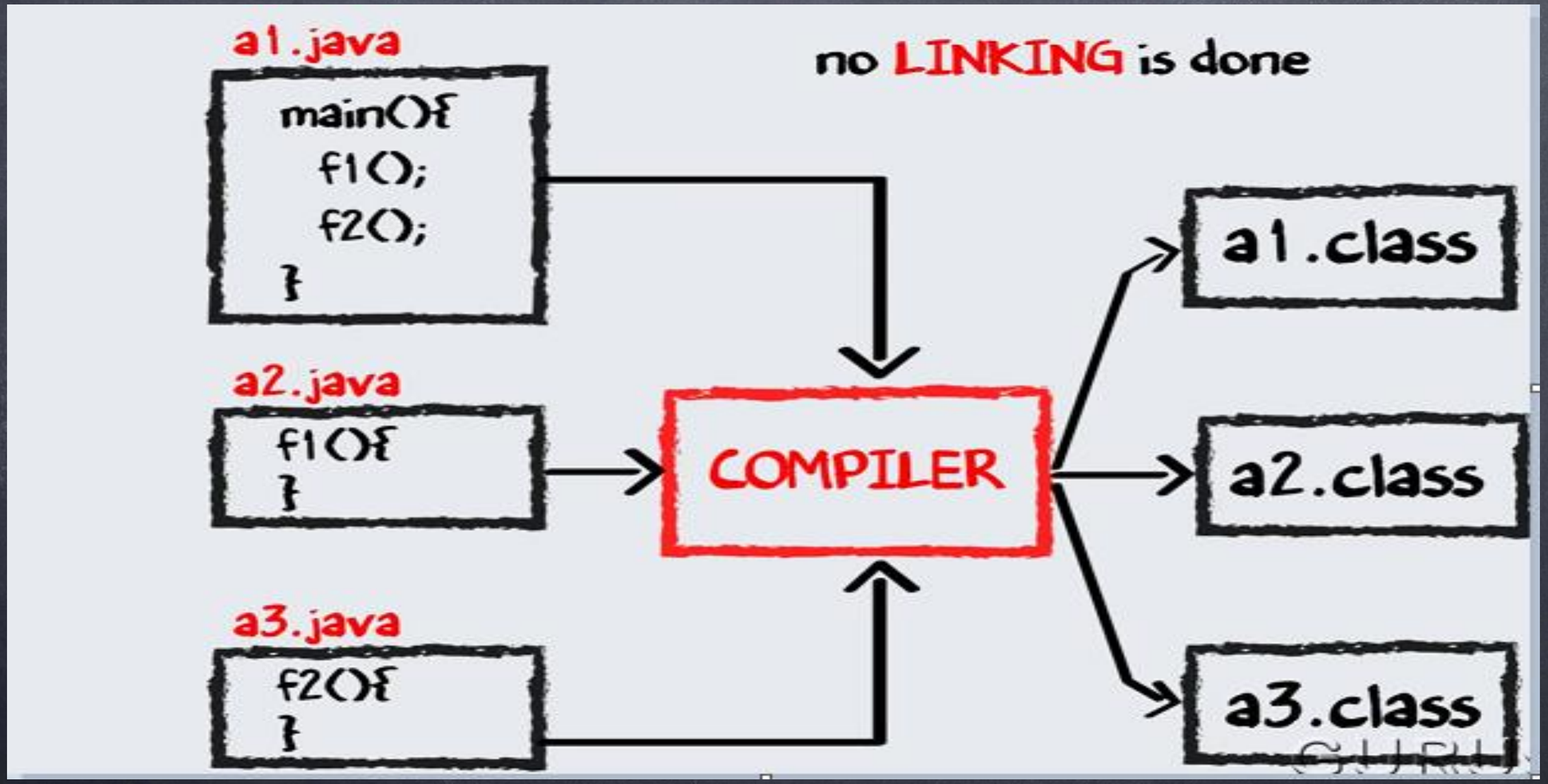
Process of compiling Android app with Java/Kotlin code

- C code compilation



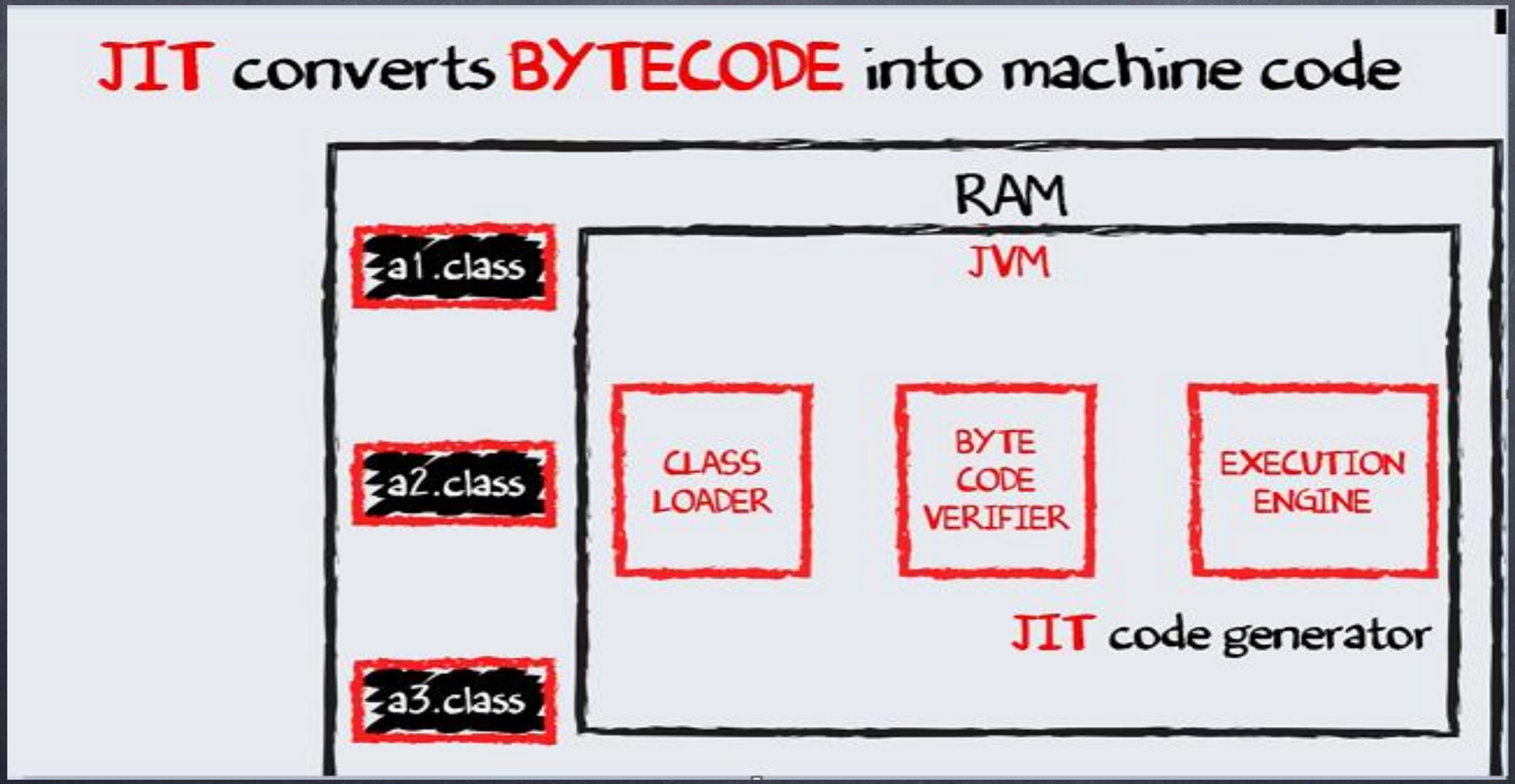
Process of compiling Android app with Java/Kotlin code

- Java code compilation



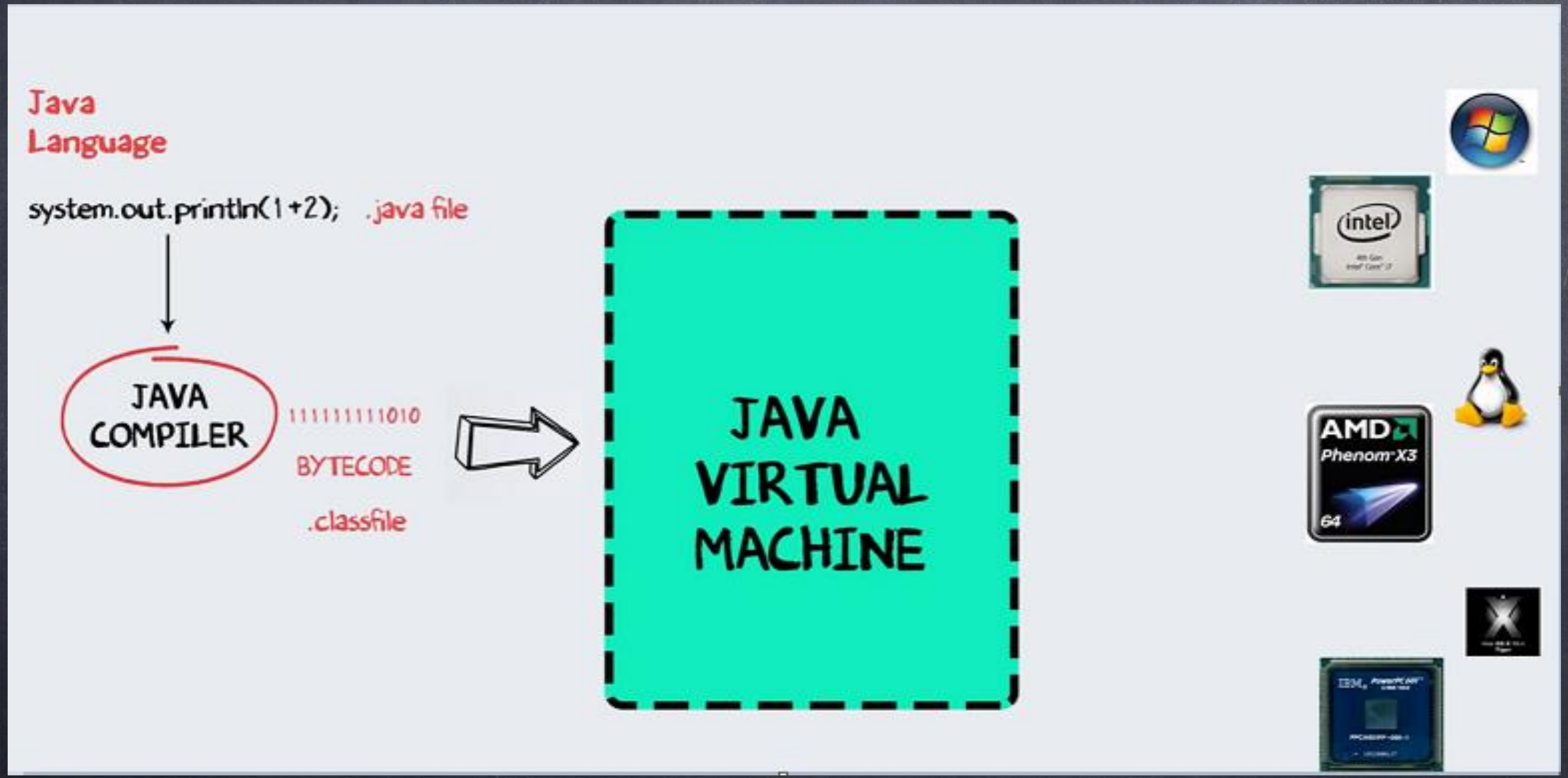
Process of compiling Android app with Java/Kotlin code

- Java code compilation



Process of compiling Android app with Java/Kotlin code

- Java code compilation



Process of compiling Android app with Java/Kotlin code

- **Java code compilation**

1. ***TestClass.java*** is compiled by **javac (Java compiler).**
2. Javac compiles Java source file into **Java byte-code** file as ***TestClass.class.***
3. Java byte-code file (*TestClass.class*) ends up in **JVM (Java Virtual Machine).**
4. JVM understands byte-code and converts it into machine code, using **JIT (Just-In-Time) compiler**.
5. The machine code is then fed to the memory and executed by computer's central processing unit.

Process of compiling Android app with Java/Kotlin code

Java code compilation

Java byte-code — representation of Java assembly.

```
// Java regular code
public int addTwoNumbers(int a, int b) {
    return a + b;
}
// Java byte-code equivalent
public int addTwoNumbers(int, int);
    Code:
       0: iload_1
       1: iload_2
       2: iadd
       3: ireturn
```

Process of compiling Android app with Java/Kotlin code

- **Java code compilation**

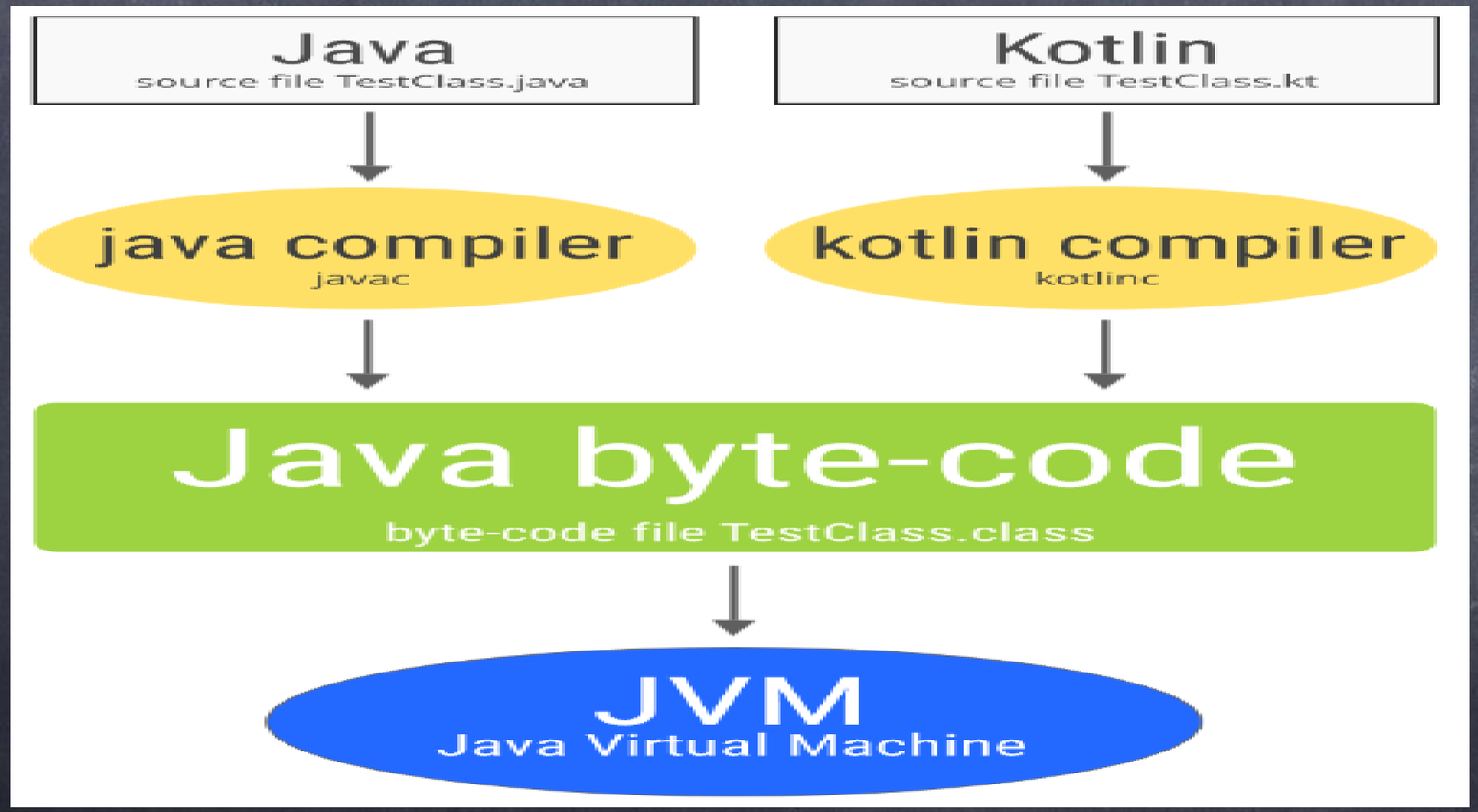
•**JVM (Java Virtual Machine)**— engine which provides runtime environment for execution of Java code.

•**JIT (Just-In-Time) compiler**— type of compiler which does the compilation during the execution of a program (compiles the app when user opens it).

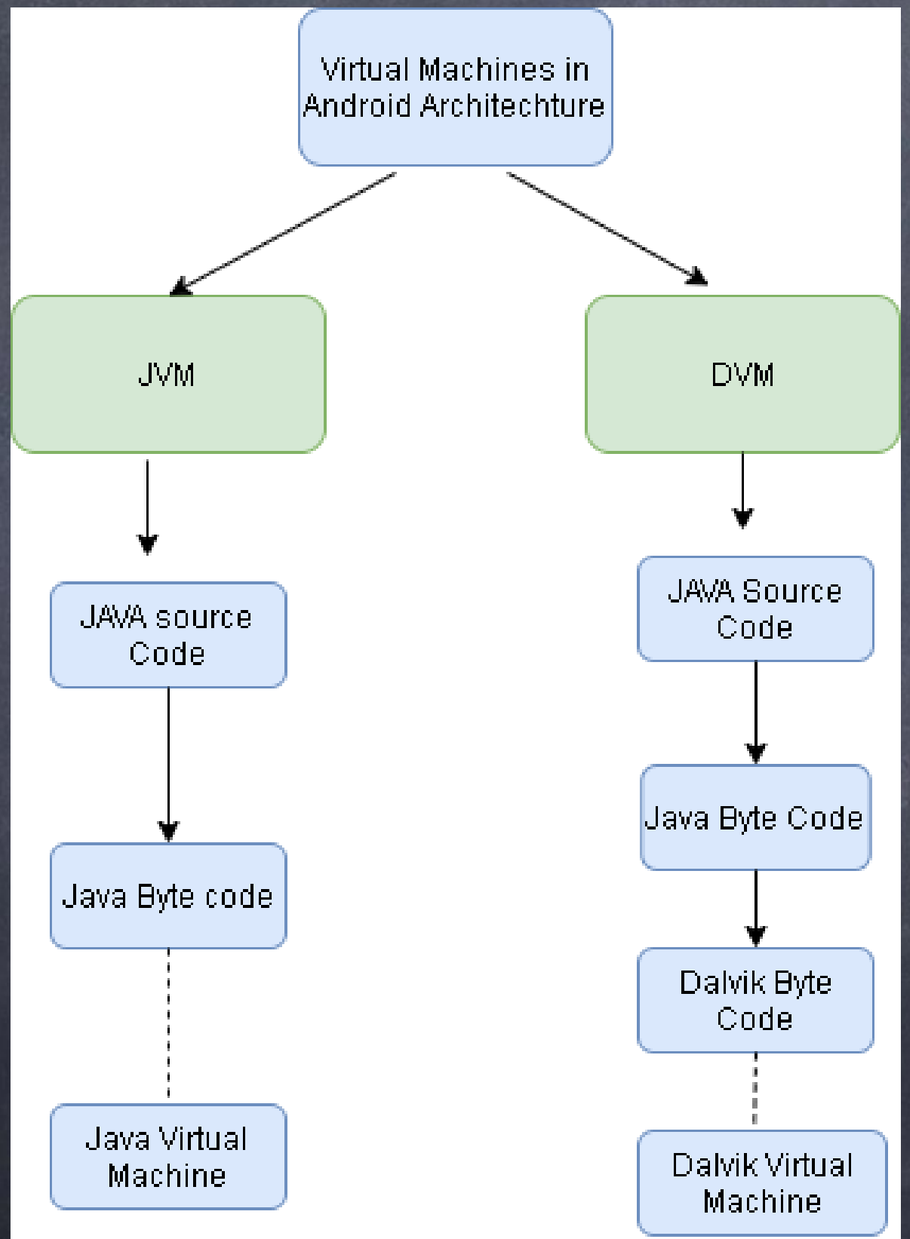
Process of compiling Android app with Java/Kotlin code

- **Kotlin code compilation**

Kotlin is a statically-typed open-source programming language generating code that can run on the JVM.



Android Runtime



Android Runtime

DVM (Dalvik Virtual Machine)	JVM (Java Virtual Machine)
It is Register based which is designed to run on low memory.	It is Stack based.
DVM uses its own byte code and runs ".Dex" file. From Android 2.2 SDK Dalvik has got a Just in Time compiler	JVM uses java byte code and runs ".class" file having JIT (Just In Time).
DVM has been designed so that a device can run multiple instances of the VM efficiently. Applications are given their own instance.	Single instance of JVM is shared with multiple applications.
DVM supports Android operating system only.	JVM supports multiple operating systems.
For DVM very few Re-tools are available.	For JVM many Re-tools are available.
There is constant pool for every application.	It has constant pool for every class.
Here the executable is APK.	Here the executable is JAR.

Process of compiling Android app with Java/Kotlin code

- **Android compilation process:**

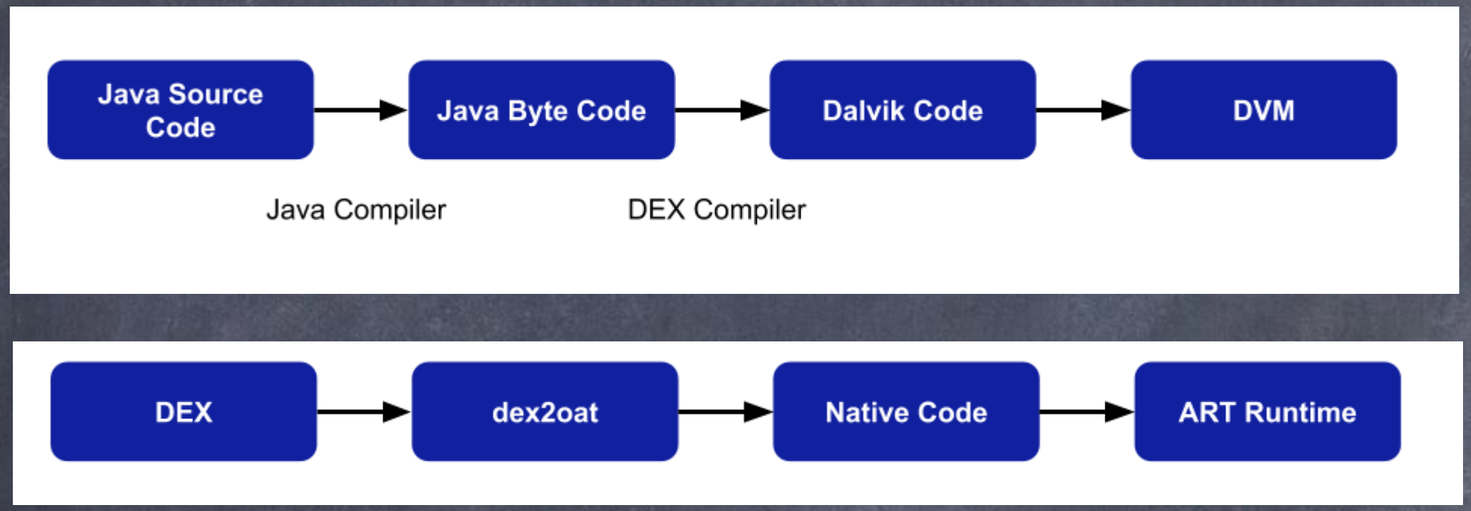
The main difference between regular Java/Kotlin code compilation and Android compilation process is that Android doesn't work with JVM

1.DVM (Dalvik Virtual Machine)
2.ART (Android Runtime) — introduced with the release of Android 4.4 (Kitkat), and before it the runtime environment for Android apps was DVM.

Regarding their functionalities, ART and Dalvik are compatible runtimes running **Dex byte-code** which can be found in **.dex** file.

Android Runtime

- DVM vs ART



https://source.android.com/devices/tech/dalvik

Android Runtime-ART

More of Android updated through Google Play -

- They are continuing to expand our investment in Google Play system updates (Project Mainline) to give apps a more consistent, secure environment across devices.
- In Android 12 They have added the Android Runtime (ART) module that lets us push updates to the core runtime and libraries on devices running Android 12.
- Developer can improve runtime performance and correctness, manage memory more efficiently, and make Kotlin operations faster - all without requiring a full system update.
- They have also expanded the functionality of existing modules - for example, we're delivering our compatible media transcoding feature inside an updatable module

Process of compiling Android app with Java/Kotlin code

- Android compilation process:

where do we get .dex file?

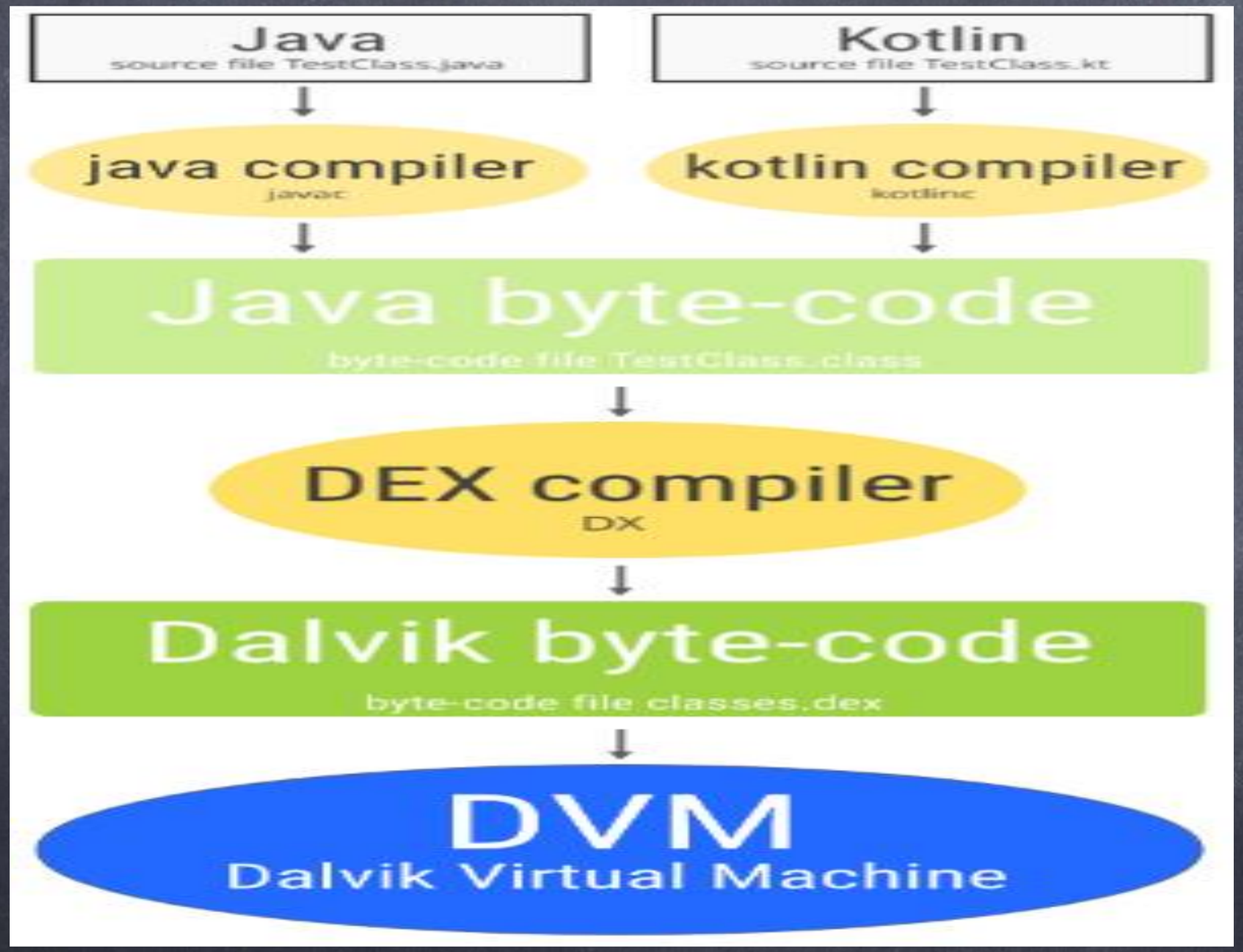
Process of compiling Android app with Java/Kotlin code

- **Android compilation process:**

  there is one more compiler placed between Java byte-code (.class file) and DVM. Its name is **DX (DEX compiler) and its purpose is to convert Java byte-code into Dalvik byte-code.**

Process of compiling Android app with Java/Kotlin code

- **Android compilation process:**



Process of compiling Android app with Java/Kotlin code

Android compilation process:

```
/ Java regular code
public int addTwoNumbers(int a, int b) {
    return a+ b;
}
// Java byte-code equivalent
public int addTwoNumbers(int, int);
    Code:
       0: iload_1
       1: iload_2
       2: iadd
       3: ireturn
// Dalvik byte-code equivalent
.method public addTwoNumbers(II)I
    .registers 4
    .param p1, "a"    # I
    .param p2, "b"    # I
    .line 6
    add-int v0, p1, p2
    return v0
.end method
```

Process of compiling Android app with Java/Kotlin code

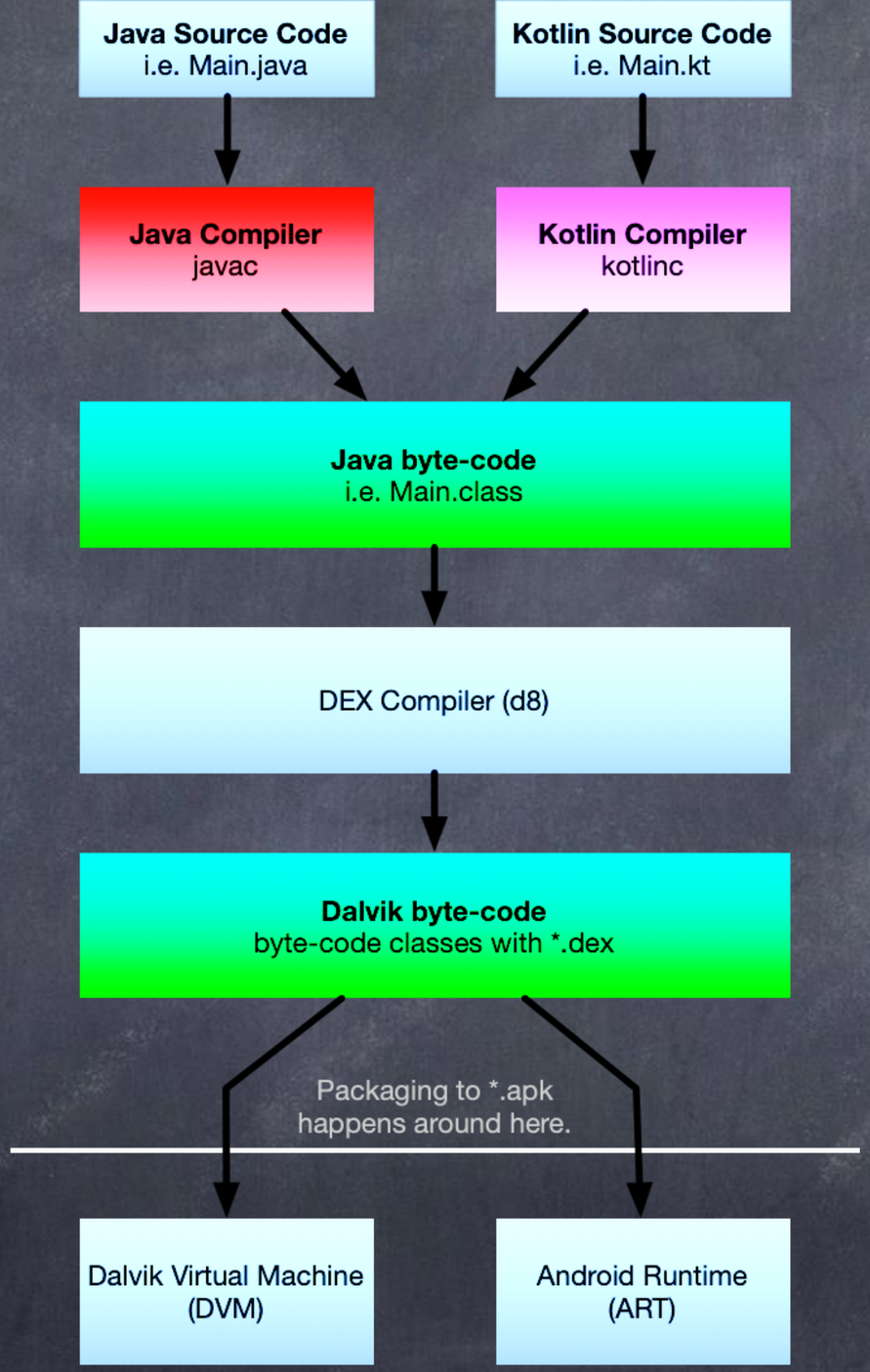
- **Android compilation process:**
- **DVM vs ART:**
- ART was built as a replacement for DVM.
  It was introduced in Android 4.4, Kitkat.
- **ART** uses **AOT (Ahead-Of-Time) compilation,**
  while **DVM** uses **JIT (Just-In-Time)**.
- JIT compilation does the compilation during the execution of a program
  (compiles the app on its startup process).
- This leads to slower startup time of apps,
  because it needs to compile app every time user opens it.

Process of compiling Android app with Java/Kotlin code

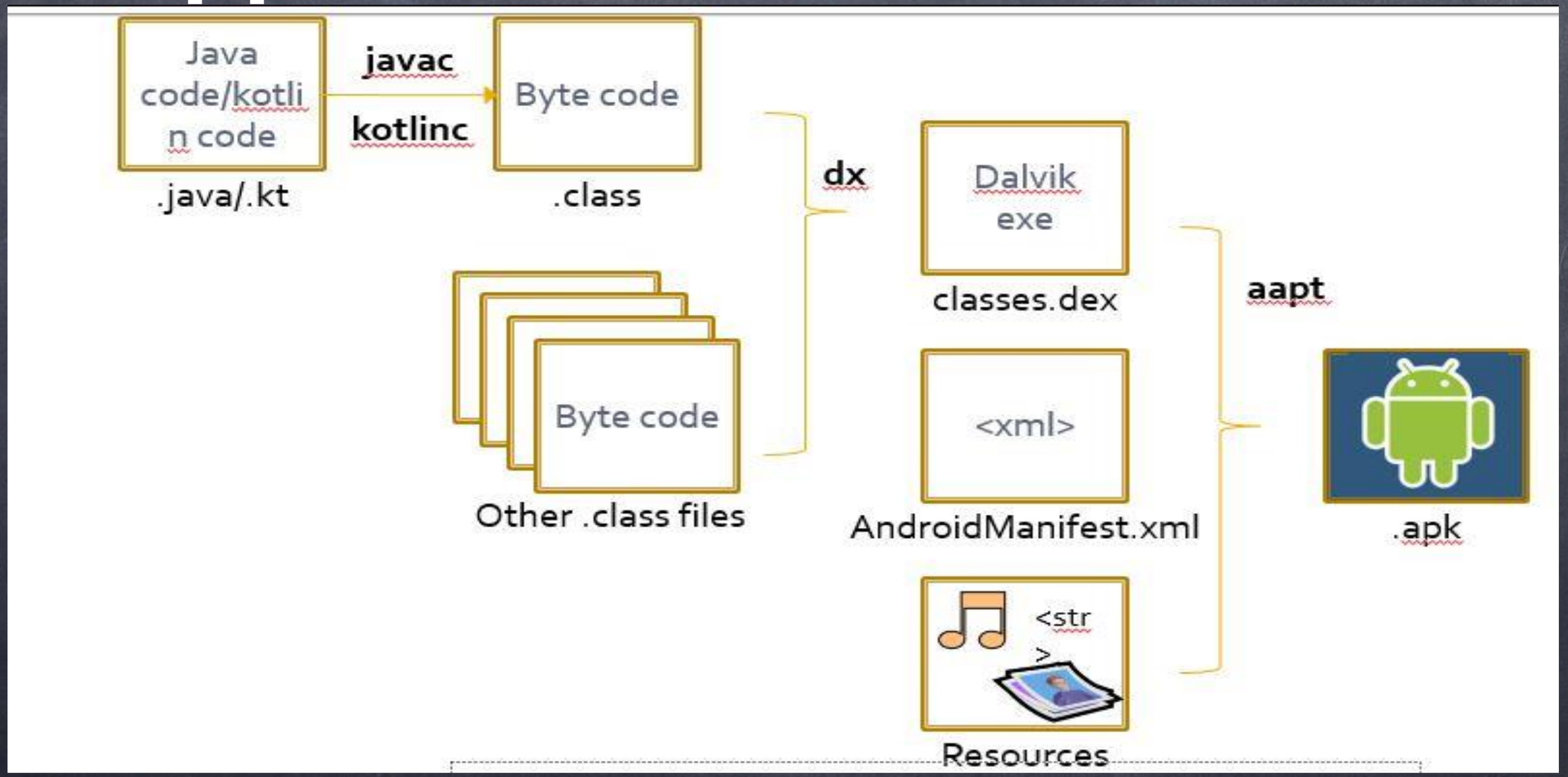
- **Android compilation process:**
- AOT compilation does the compilation when the app is installed.
- By using on-device **dex2oat** tool, AOT accepts .dex files as input and generates a compiled app executable for the target device.
- This can lead to much faster and better app performance.
- As ART compiles app machine code at its installation, there is no need for translation to machine code every time user opens the app
- AOT doesn’t hit the CPU as hard as Just-In-Time code compiling on Dalvik
- Less CPU usage results in less battery drain, which is also important benefit for any Android device

Process of compiling Android app with Java/Kotlin code

- **Android compilation process:**
- AOT compilation does the compilation when the app is installed.
- By using on-device **dex2oat** tool, AOT accepts .dex files as input and generates a compiled app executable for the target device.
- This can lead to much faster and better app performance.
- As ART compiles app machine code at its installation, there is no need for translation to machine code every time user opens the app
- AOT doesn’t hit the CPU as hard as Just-In-Time code compiling on Dalvik
- Less CPU usage results in less battery drain, which is also important benefit for any Android device


Java Source Code
i.e. Main.java
Kotlin Source Code
i.e. Main.kt
Java Compiler
javac
Kotlin Compiler
kotlinc
Java byte-code
i.e. Main.class
DEX Compiler (d8)
Dalvik byte-code
byte-code classes with *.dex
Packaging to *.apk
happens around here.
Dalvik Virtual Machine
(DVM)
Android Runtime
(ART)
Pre Android KitKat

Producing Android Application



Android vs iOS

Edit	Android User Rating (929):	iOS User Rating (864):
Widgets	Yes	No, except in NotificationCenter
Company/Developer	Google	Apple Inc.
OS family	Linux	OS X, UNIX
Customizability	A lot. Can change almost anything.	Limited unless jailbroken
Initial release	September 23, 2008	July 29, 2007
Programmed in	C, C++, java	C, C++, Objective-C
Dependent on a PC or a Mac	No	No
Easy media transfer	depends on model	with desktop application
Source model	Open source	Closed, with open source components.
Open source	Kernel, UI, and some standard apps	The iOS kernel is not open source but is based on the open-source Darwin OS.
Call features supported	Auto-respond	Auto-respond, call-back reminder, do not disturb mode

Android vs iOS

Internet browsing	Google Chrome (or Android Browser on older versions; other browsers are available)	Mobile Safari (Other browsers are available)
Available on	Many phones and tablets, including Kindle Fire(modified android), LG, HTC, Samsung, Sony, Motorola, Nexus, and others.	iPod Touch, iPhone, iPad, Apple TV (2nd and 3rd generation)
Interface	Touch screen, Smartwatch	Touch screen
Messaging	Google Hangouts	iMessage
Voice commands	Google Now (on newer versions)	Siri
Maps	Google Maps	Apple Maps
Video chat	Google Hangouts	Facetime
App store	Google Play – 1,000,000+ Apps. Other app stores like Amazon and Getjar also distribute Android apps. (unconfirmed ".APK's")	Apple app store – 1,000,000+ Apps
Market share	81% of smartphones, 3.7% of tablets in North America (as of Jan'13) and 44.4% of tablets in Japan (as of Jan'13). In the United States in Q1 2013 - 52.3% phones, 47.7% tablets.	12.9% of smartphones, 87% of tablets in North America (as of Jan'13) and 40.1% of tablets in Japan (as of Jan'13)
Available language(s)	32 Languages	34 Languages