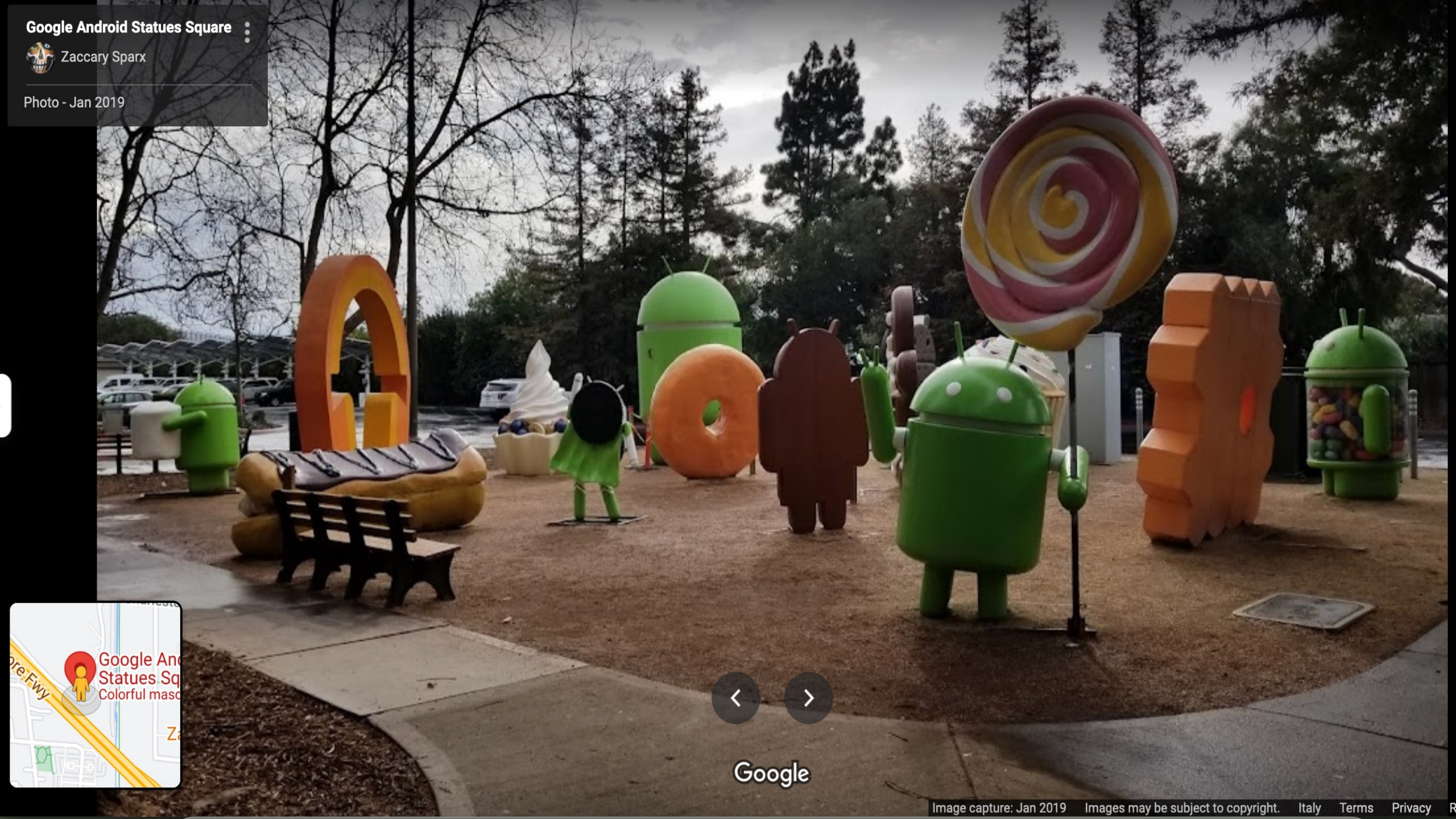
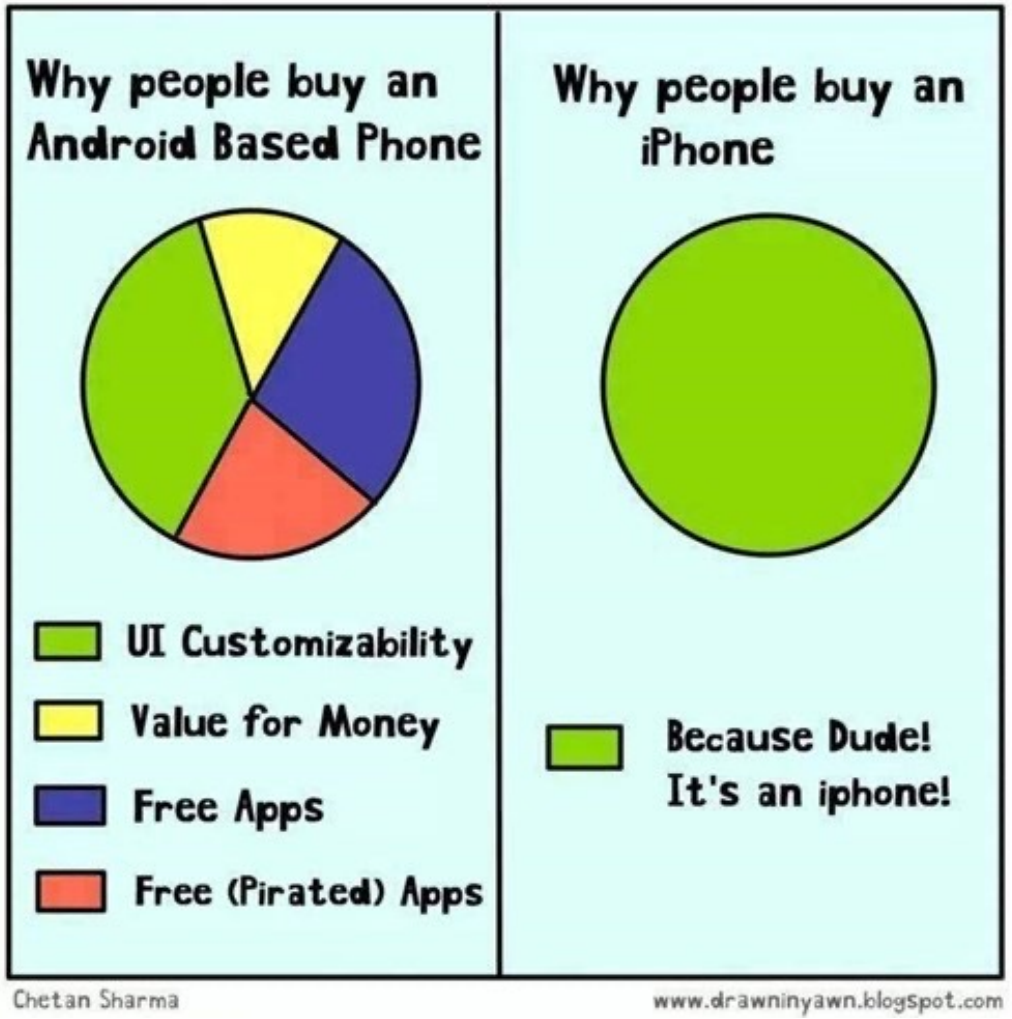
Android: an introduction

Catia Prandi,
Università di Bologna


Google Android Statues Square
Zaccary Sparx
Photo - Jan 2019
Google
Image capture: Jan 2019
Images may be subject to copyright.
Italy
Terms
Privacy


Google Android Statues Square
Velenson
Photo - Jan 2022
Google
Image capture: Jan 2022
Images may be subject to copyright.
Italy
Terms
Privacy

Why Android?

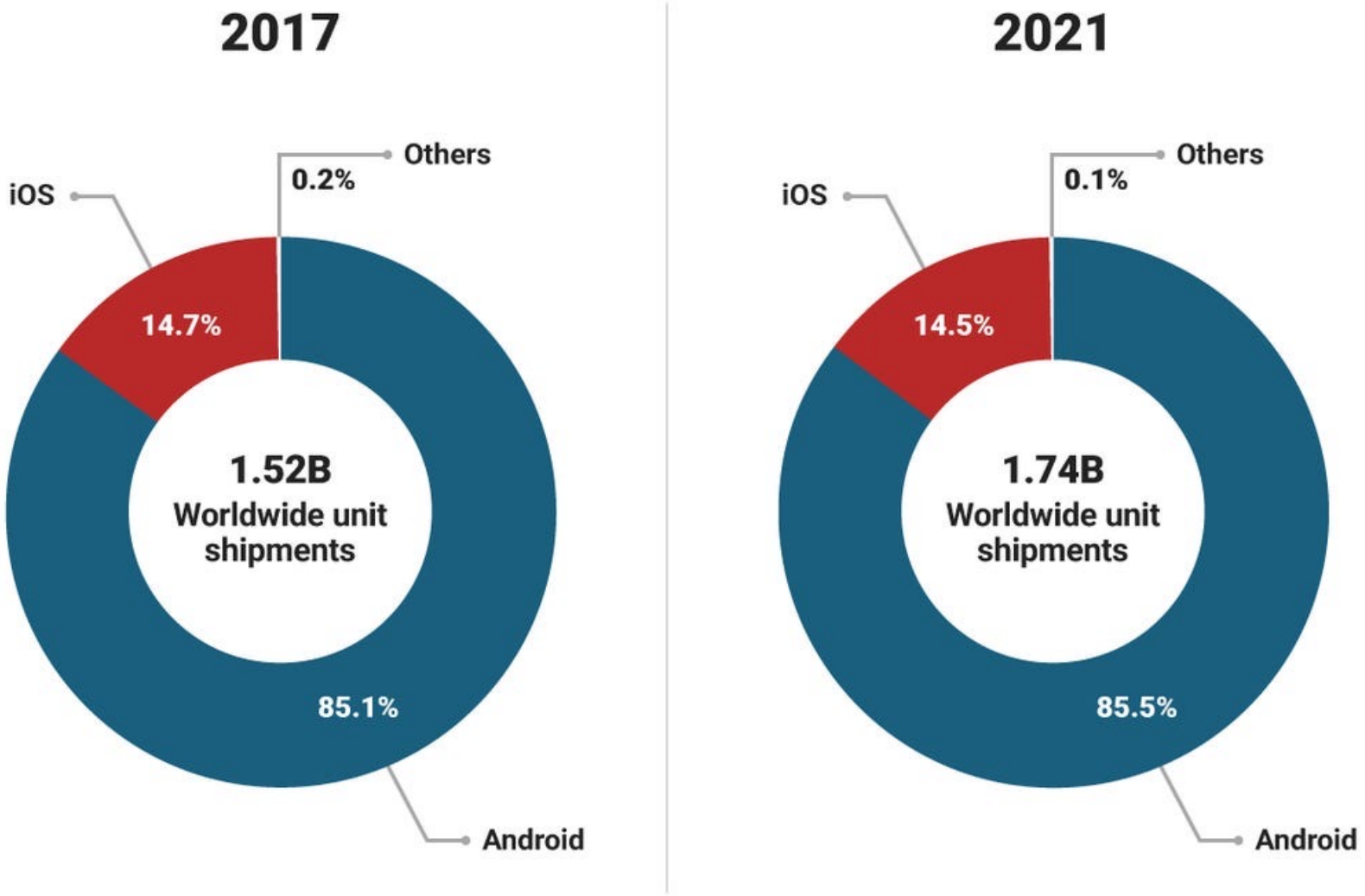

Why people buy an Android Based Phone
UI Customizability
Value for Money
Free Apps
Free (Pirated) Apps
Why people buy an iPhone
Because Dude! It's an iphone!
Chetan Sharma
www.drawninyawn.blogspot.com

Why Android?

- Android is the leading mobile operating system in the world
- According to statistics, it occupies about 80% of the mobile market
- It has not only imposed itself on smartphones and tablets but also on
  - Wear OS for wearables
  - Android TV for smart televisions
  - Android for Cars for the creation of apps integrated with cars
  - Android Things dedicated to the IoT (Internet of Things),
  - Up to the possibility of distributing Android apps on Chrome OS systems such as Chromebooks

TECH | CHART OF THE DAY

GLOBAL MOBILE OPERATING SYSTEM MARKET SHARE



SOURCE: IDC Based on unit shipments, sums may not add to 100% due to rounding

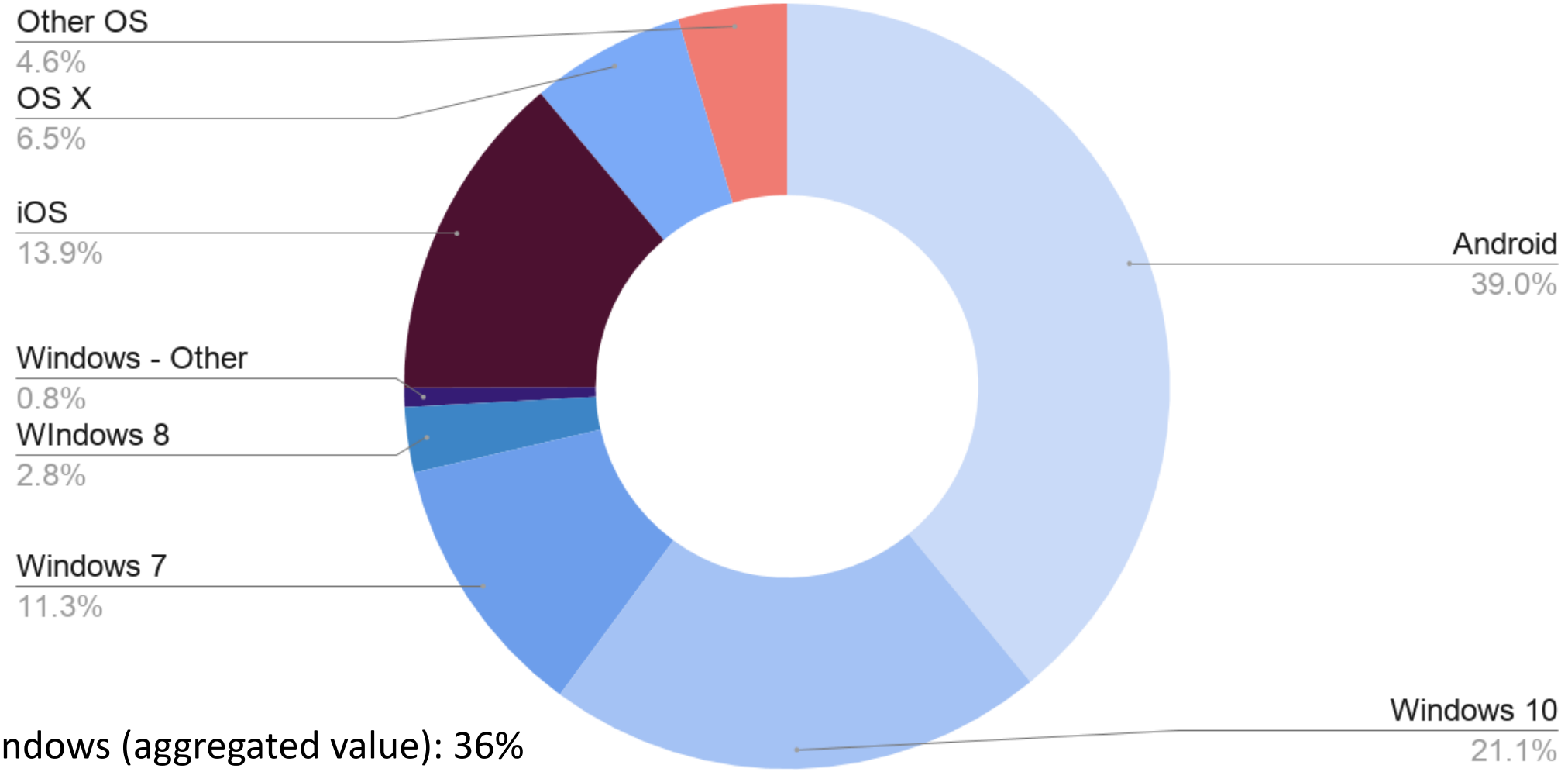
statista | BUSINESS INSIDER

Question

- Is it the most used OS only in the mobile context?

Operating System Market Share, Install Base, 2019

www.T4.ai



Windows (aggregated value): 36%

A bit of history..

- Do you recognize these faces?

Android History

- In **2003**, Andy Rubin, Rich Miner, Nick Sears and Chris White founded Android Inc. whose purpose was to develop “cellular devices more aware of their owner's location and preferences”
- In **2005**, with the acquisition by **Google** for an amount close to 50 million dollars, Android begins to become an operating system for mobile devices, with a Linux kernel
- From the acquisition by Google, it still took over two years of development, starting from a Linux kernel, by the Rubin’s team, who remained in the project together with Miner and White, before the official presentation on November 5, 2007 by the company
- On that occasion, the Open Handset Alliance (a consortium that now includes 84 companies to develop open standards for mobile devices), born. Partners: Google, Motorola, Samsung, Vodafone, T-Mobile, etc.

Android History

- On October 22, **2008 G1 HTC Drea**m (the first Android smartphone) was launched on the market
- Since then, a very long process of updates and improvements has begun that have led Android to be the most popular operating system in the world, with **constant updates** that in recent years have reached an annual frequency, with minor releases released between one version and another

G1 HTC Dream


T-Mobile
android
MENU


2:00 PM
myFaves
Email
Dialer
Contacts
Browser
Maps
T-Mobile
MENU

Android Versions

- Versions 1.0 and 1.1 were later referred to as Alpha and Beta, or Apple Pie and Banana Bread, but never actually had an official name
- Starting with version 1.5, Google has decided to match the name of a dessert to each version

Alpha

September 23, 2008

Petit Four

February 9, 2009

Cupcake

April 27, 2009

Donut

September 15, 2009

Eclair

October 26, 2009

Ice Cream Sandwich

October 18, 2011

Honeycomb

February 22, 2011

Gingerbread

December 6, 2010

Froyo

May 20, 2010

Jelly Bean

July 9, 2012

KitKat

October 31, 2013

Lollipop

November 12, 2014

Marshmallow

October 5, 2015

Nougat

August 22, 2016

Counterpoint
Technology Market Research

Infographic

- https://earthlymission.com/android-version-history-a-visual-timeline/

Apple Pie 1.0

Cupcake 1.5

Donut 1.6

Eclair 2.0/ 2.1

Froyo 2.2

Gingerbread 2.3.x

Honeycomb 3.x

Ice Cream Sandwich 4.0.x

Jelly Bean 4.1/4.2/4.3

KitKat 4.4

Lollipop 5.0

Marshmallow 6.0

Nougat 7.0

Oreo 8.0

Pie 9.0

And.. After??

Android versions

- After a decline in imagination with the 9.0 version of Android, called Android 9 Pie, the Californian giant decided, in August 2019, to drastically change course
- Enough names of desserts, which are difficult to understand for some countries and cultures
- In their place a much simpler solution, with only the version number -> Android 10 (initially called Q), and later versions will be marked exclusively by the number

Android 10


Apple Pie 1.0


Cupcake 1.5


Donut 1.6


Eclair 2.0/ 2.1


Froyo 2.2


Gingerbread 2.3.x


Honeycomb 3.x


Ice Cream Sandwich 4.0.x


Jelly Bean 4.1/4.2/4.3


KitKat 4.4


Lollipop 5.0


Marshmallow 6.0


Nougat 7.0


Oreo 8.0


Pie 9.0

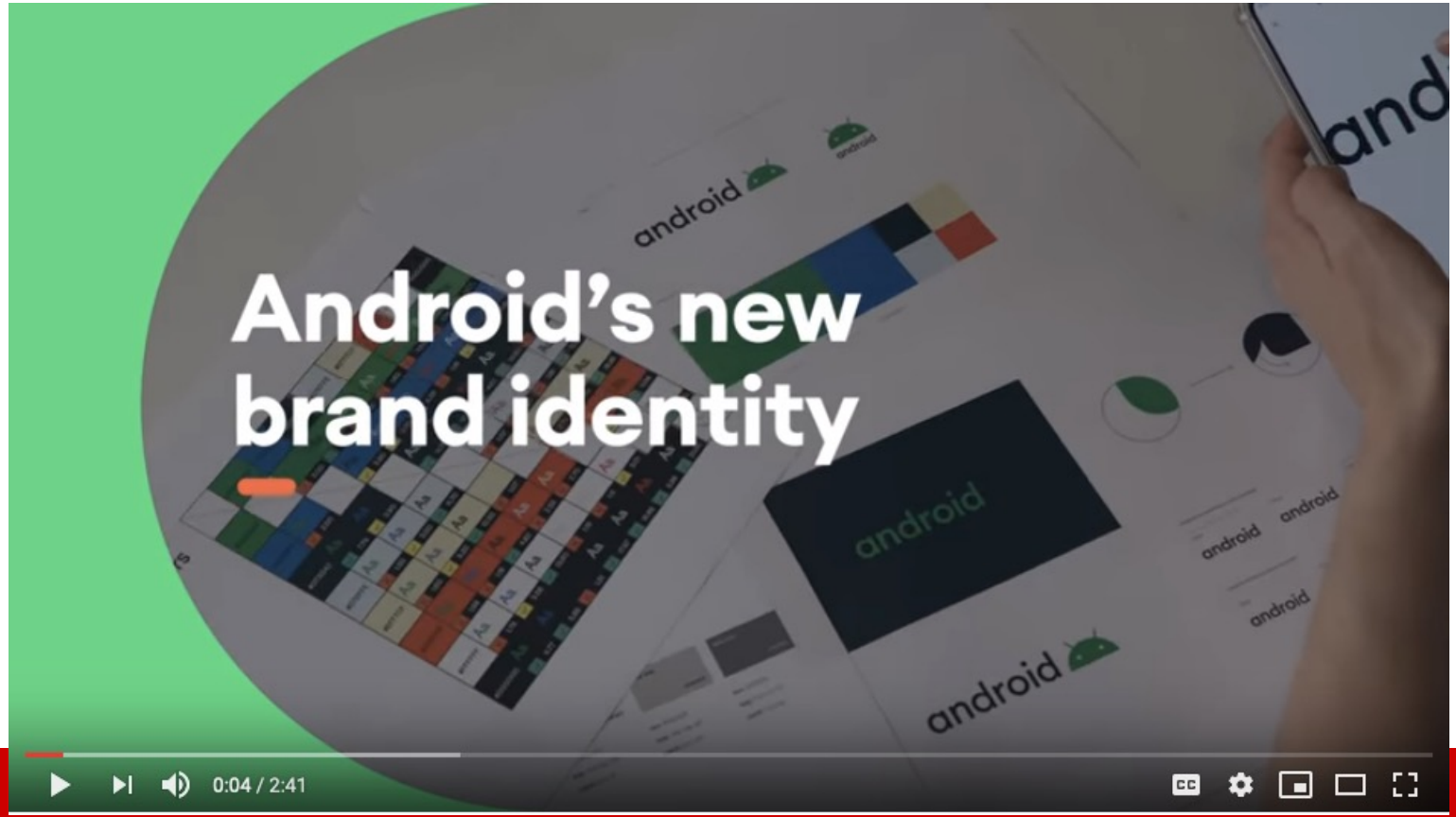
Android 10, September 3, 2019

android

Nuovo logo

Android 10

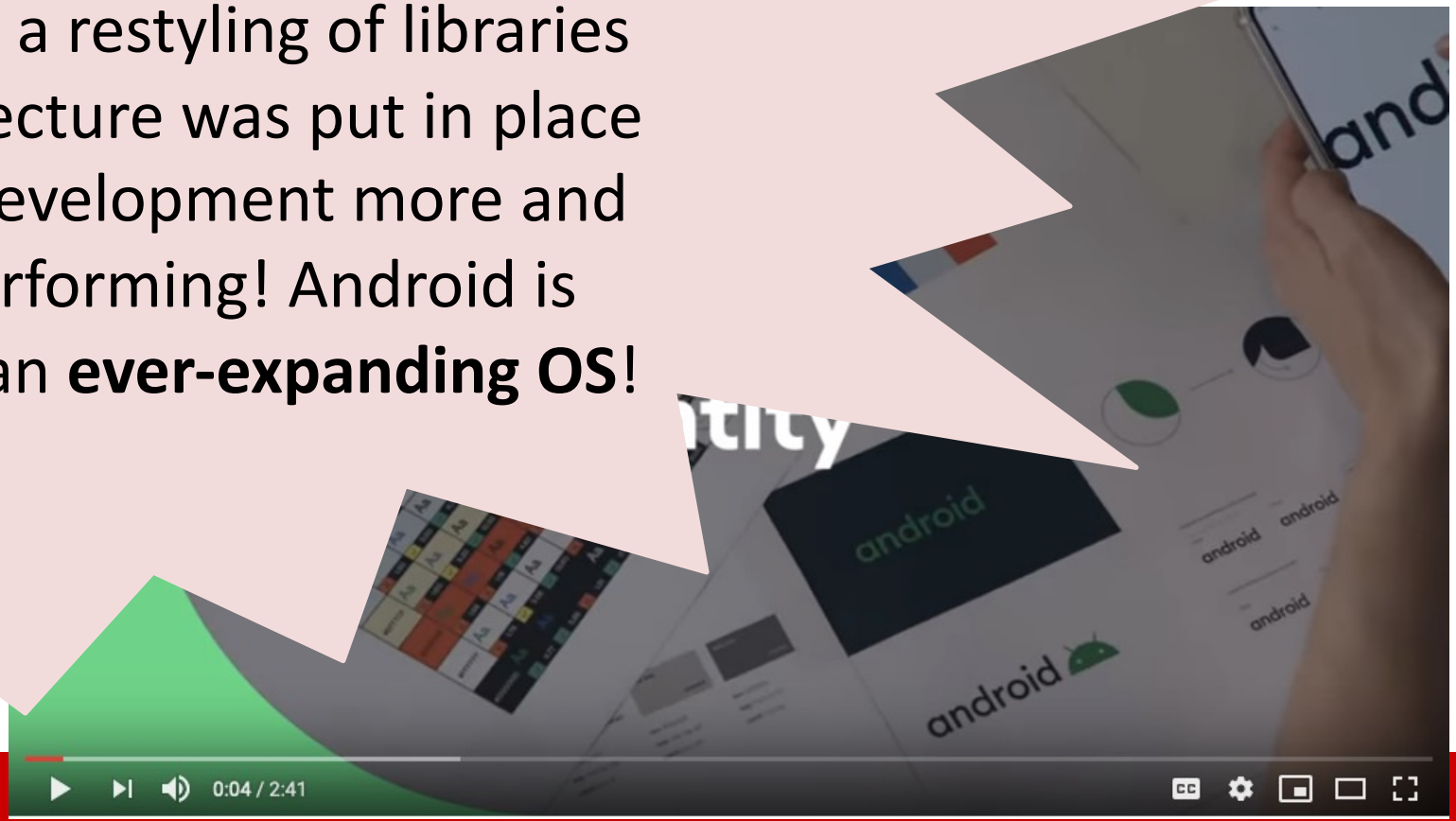
- https://www.youtube.com/watch?time_continue=28&v=-2w7RKAlKTs&feature=emb_logo



Android 10

- https://ww[illegible]tube.co[illegible]nti[illegible]8&v=-2w7RKAlKTs

Moreover, a restyling of libraries and architecture was put in place to make development more and more performing! Android is definitely an **ever-expanding OS**!



Android 10: Foldables

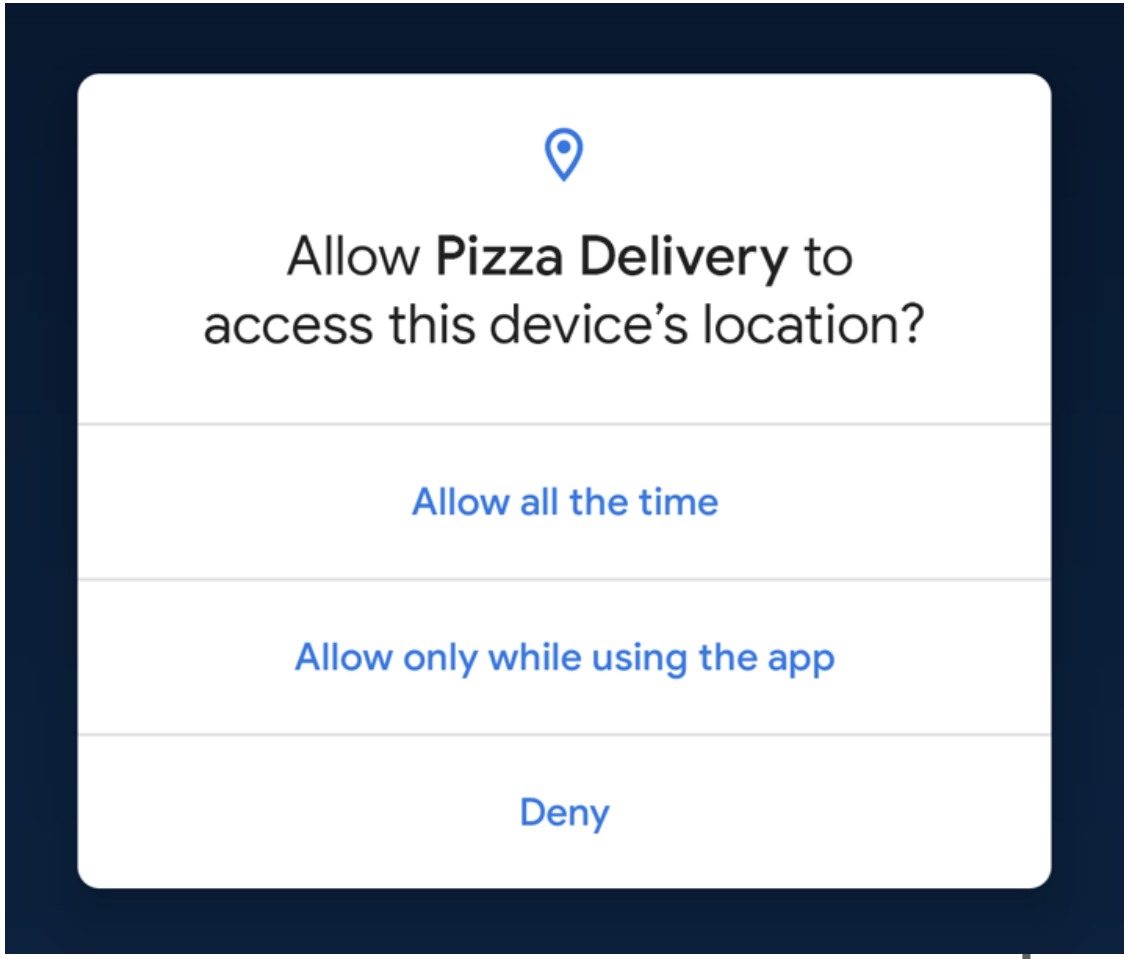
- Foldables
  - Building on solid **multi-window support**, Android 10 extends multitasking between app windows and offers screen continuity to maintain app state as the device folds or unfolds
  - Android 10 adds a number of improvements to onResume and onPause to support multiple reset and notify your app when it's active
  - It also modifies the behavior of the resizeableActivity manifest attribute, to help you manage how your app is displayed on foldable and large screens

Android 10: 5G

- 5G network
  - 5G promises to deliver consistently faster speeds and lower latency
  - Android 10 adds platform support for 5G and extends existing APIs to help you take advantage of these improvements. You can use the connectivity APIs to detect if the device has a high bandwidth connection

Android 10: Privacy

- Focus on **privacy** and security
  - Greater control over location data new permission option - can now allow an app to **access location only while the app is actually in use** (running in the foreground)
  - Device tracking protection **Randomized MAC addresses** User data protection in external memory



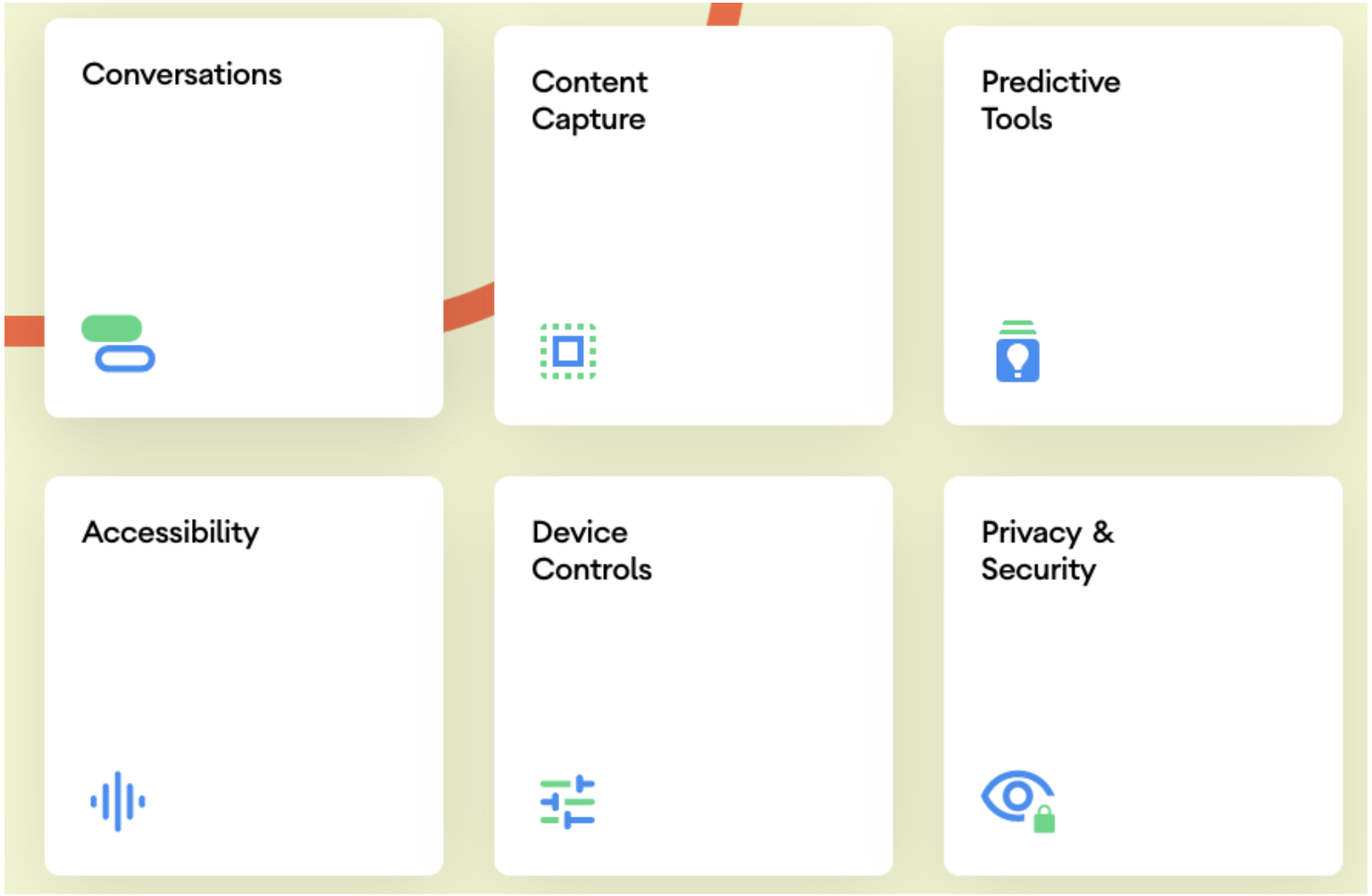
And.. Then?

Android 11

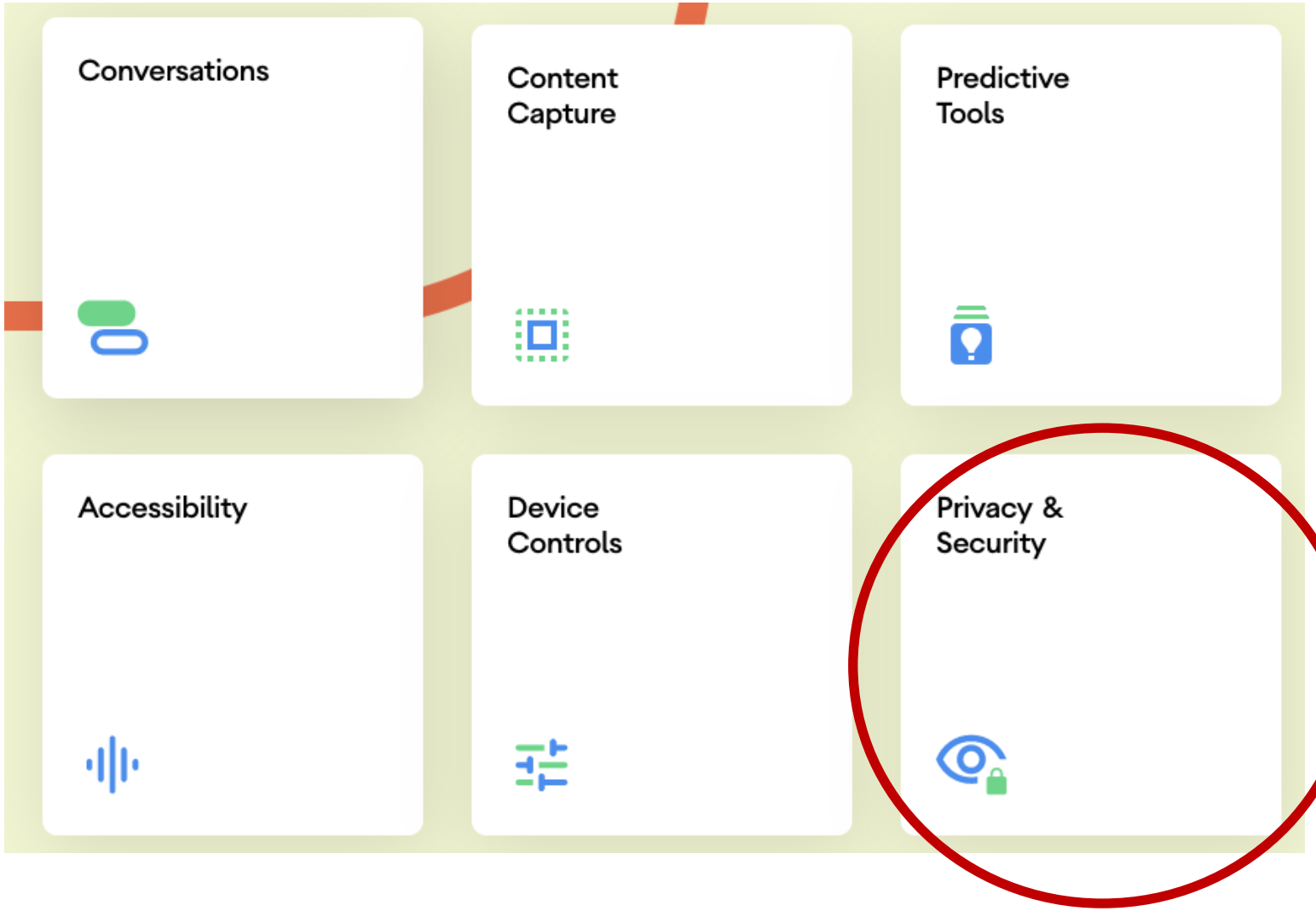
- **The OS that gets to what's important"**
  - "Go straight to the stuff that matters most. Because Android 11 is optimized for how you use your phone. Giving you powerful device controls. And easier ways to manage conversations, privacy settings and so much more."

- Released on 8 September 2020
- https://developer.android.com/about/versions/11

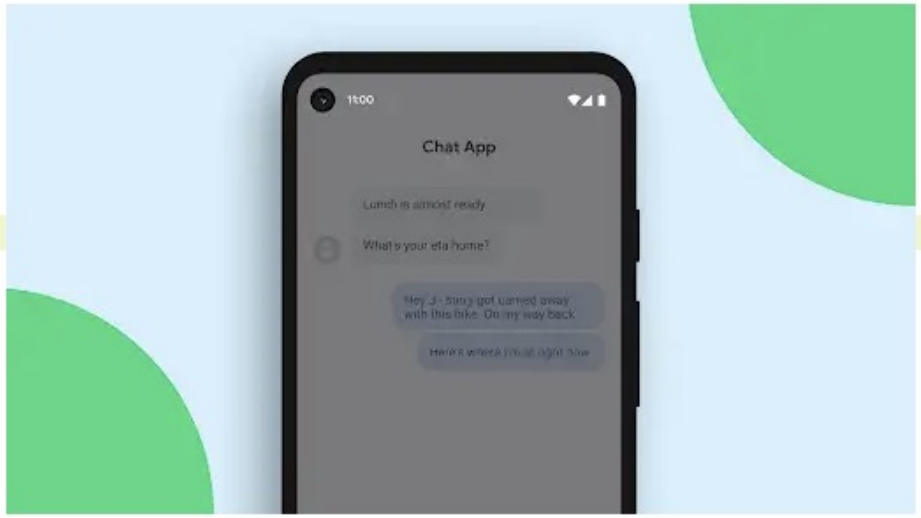
Most important news



Most important news



Privacy and Security



One-time permissions

Give one-time permissions to apps that need your mic, camera or location. The next time the app needs access, it must ask for permission again.

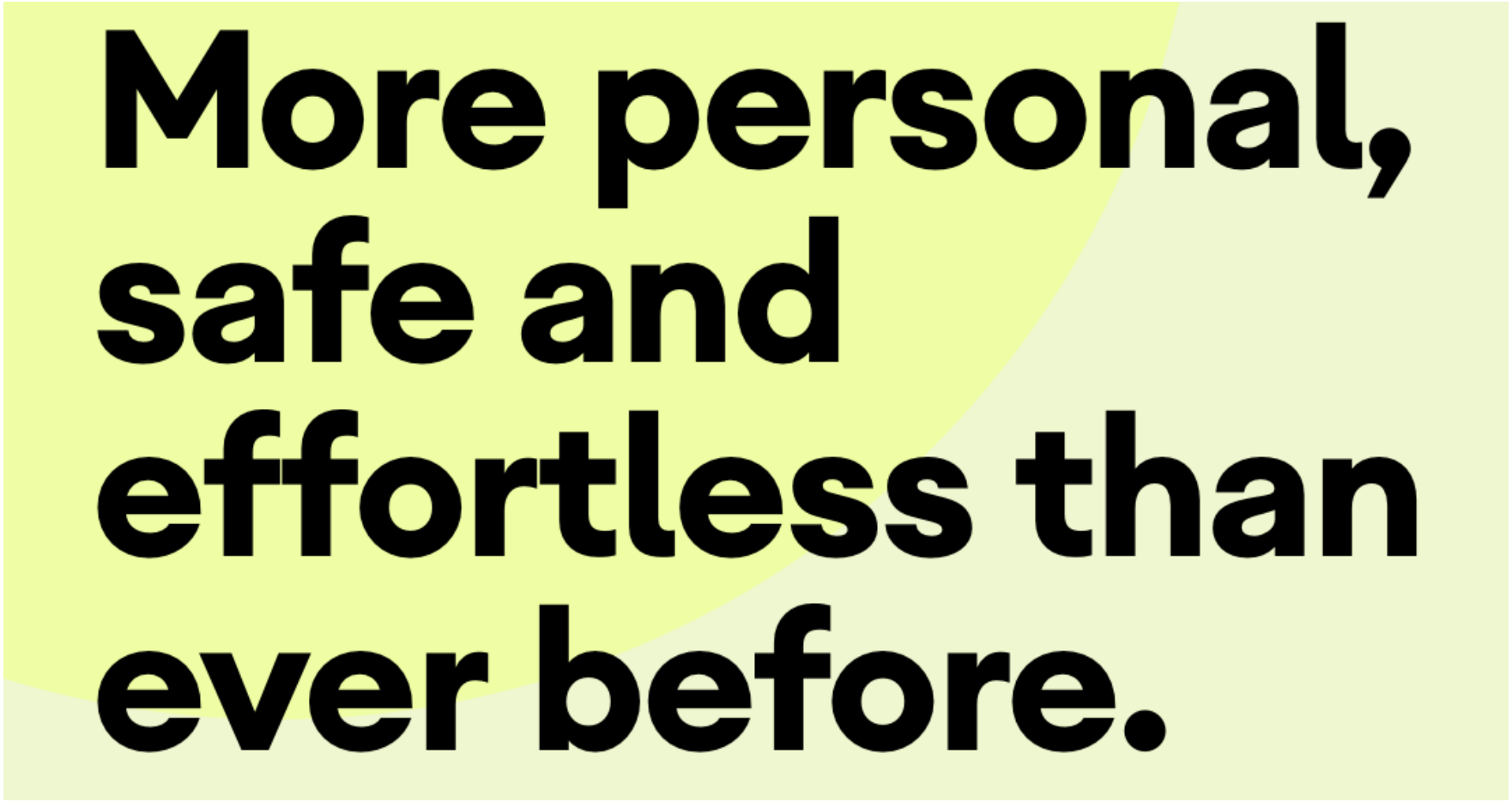
Permissions auto-reset

If you haven't used an app in a while, you may not want it to keep accessing your data. So Android will reset permissions for your unused apps. You can always turn permissions back on.

https://www.youtube.com/watch?v=vaD-DPI6sgU

And.. Then?

Android 12



Android 12

- Released on **October 4, 2021**
- Android 12 delivers even more **personal**, **safe** and **effortless** experiences on your device.
- Featuring a totally reimagined UI just for you, new privacy features that are designed for your safety and put you in control, and more seamless ways to get right to your gameplay or even switch to a new device.

Android 12

- https://www.android.com/android-12/

- Promotional video
  - https://youtu.be/UHQPdP8qgrk

Android 12: accessibility

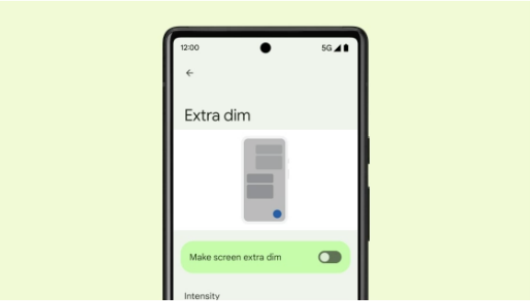
ACCESSIBILITY IMPROVEMENTS

Built for accessibility.

Android 12 is designed to be even more accessible with new visibility features, including:

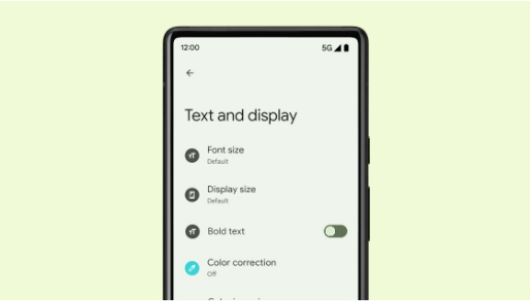
Area magnification

A new window magnifier lets you zoom in on a part of your screen without having to lose context on the rest of the screen content.



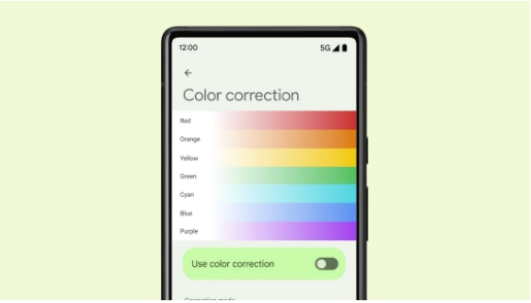
Extra dim

Make your display extra dim for night-time scrolling or situations when even the lowest brightness setting is too bright.



Bold text

See text more clearly with the ability to switch the font to bold across the whole phone.



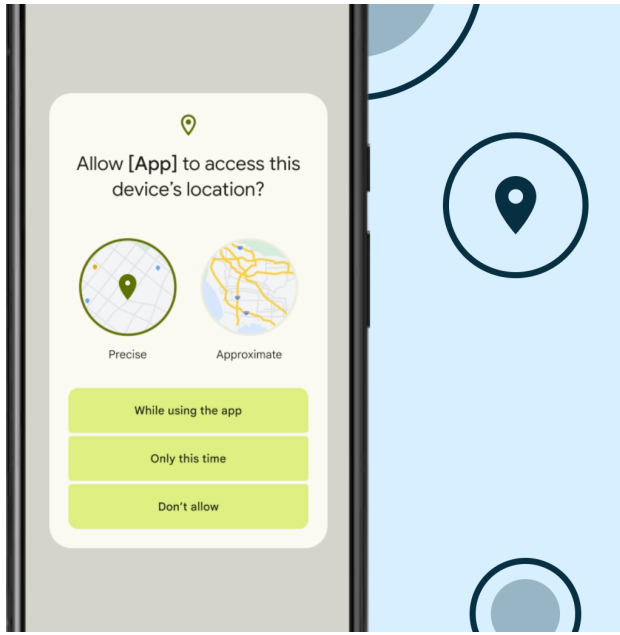
Grayscale

Adjust how colors display on your device to grayscale.

Android 12: privacy

SAFE

Private by design so you're in control.



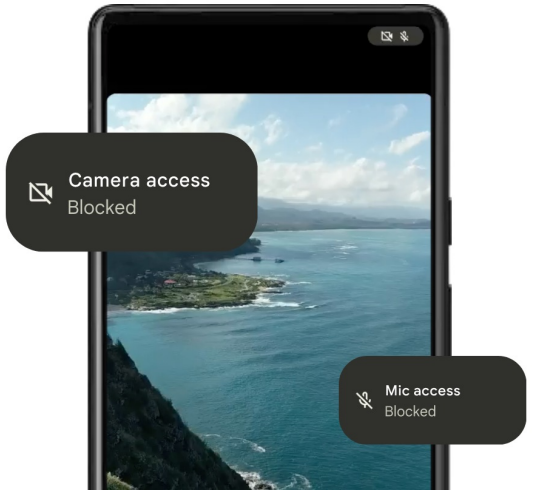
Keep your precise location private.

While some apps need precise location information for tasks like turn-by-turn navigation, many other apps only need your approximate location to be helpful. With Android 12, you can choose between giving apps access to your precise location or an

BACK TO TOP

MIC & CAMERA INDICATORS AND TOGGLES

Stronger mic and camera access controls.



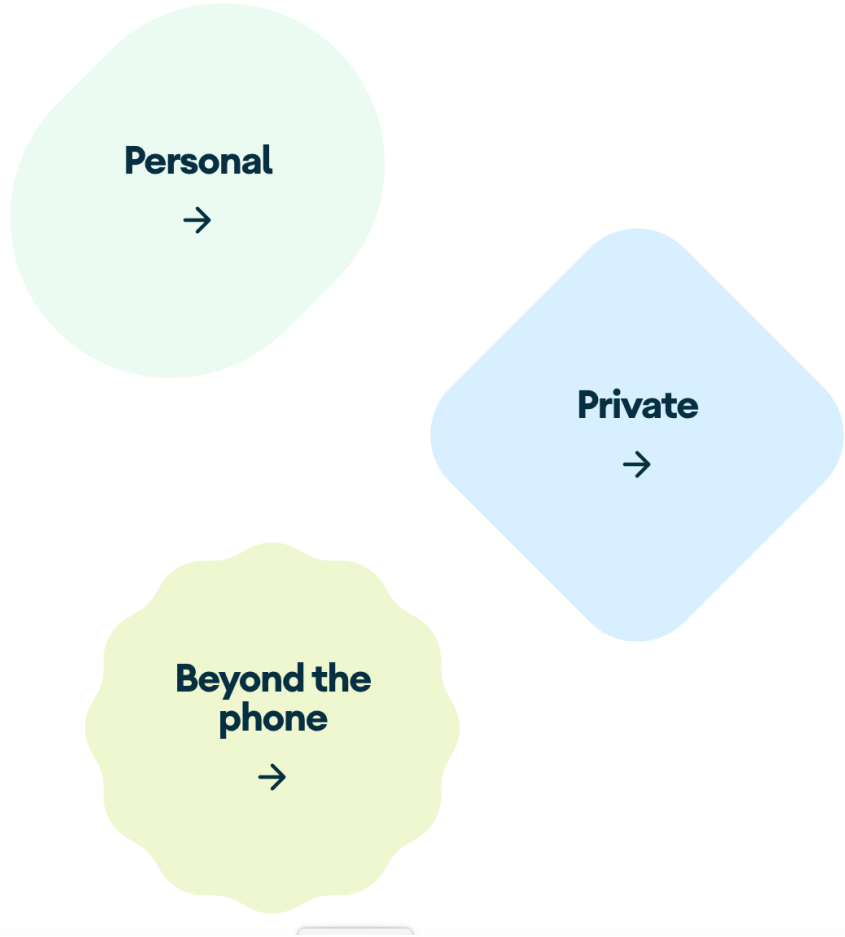
And.. Today?

- **Android 13** (released -> August 15, 2022)

- Your phone, tablet and more—all on your terms.
- «Build for user privacy with photo picker and notification permission. Improve productivity with themed app icons, per-app languages, and clipboard preview. Build for modern standards like Bluetooth LE Audio. Deliver a better experience on tablets and large screens”

- https://www.android.com/android-13/

Android 13 Highlights



Android 13: Security

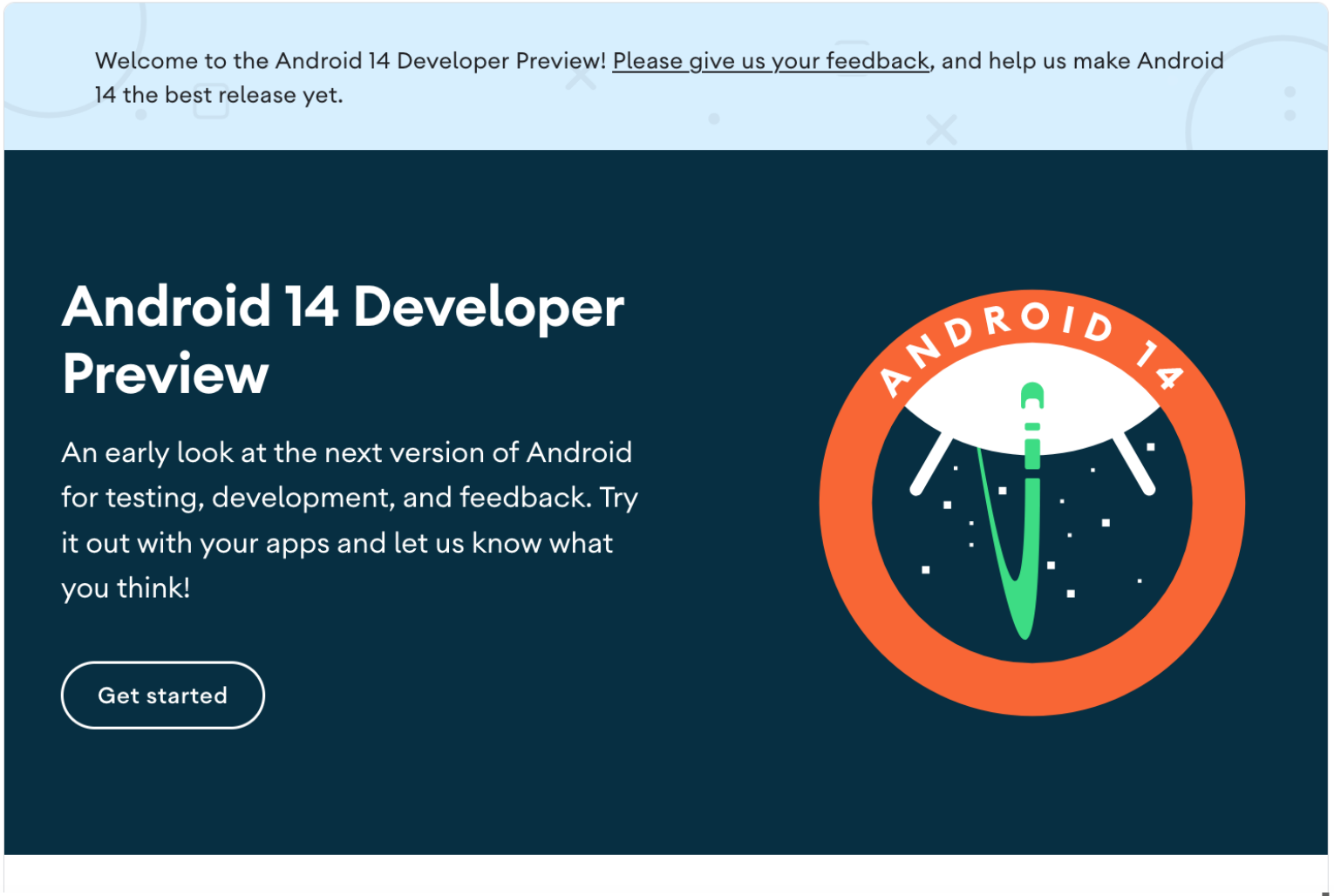
Security that keeps your data protected all day.

From the moment you turn on your device, Android works to keep your data safe and secure. With Android 13, you have more control over what information apps can and can't access—including specific photos, videos and clipboard history.

And today?

- … let’s try again…

- 08 February 2023



https://android-developers.googleblog.com/2023/02/first-developer-preview-android14.html

Android

Android is definitely an ever-expanding OS!

And this, combined with the fact that it is an Open Source system, certainly brings many advantages but also…. Problems!!

Survey time

- What about you?
- Do you have Android?
  - If yes, which version?

https://forms.office.com/e/aM8RMKYtXp

Android.. In details!

- Android is

Android.. In details!

- Android is
  - Open source
    - software for which the original source code is made freely available and may be redistributed and modified

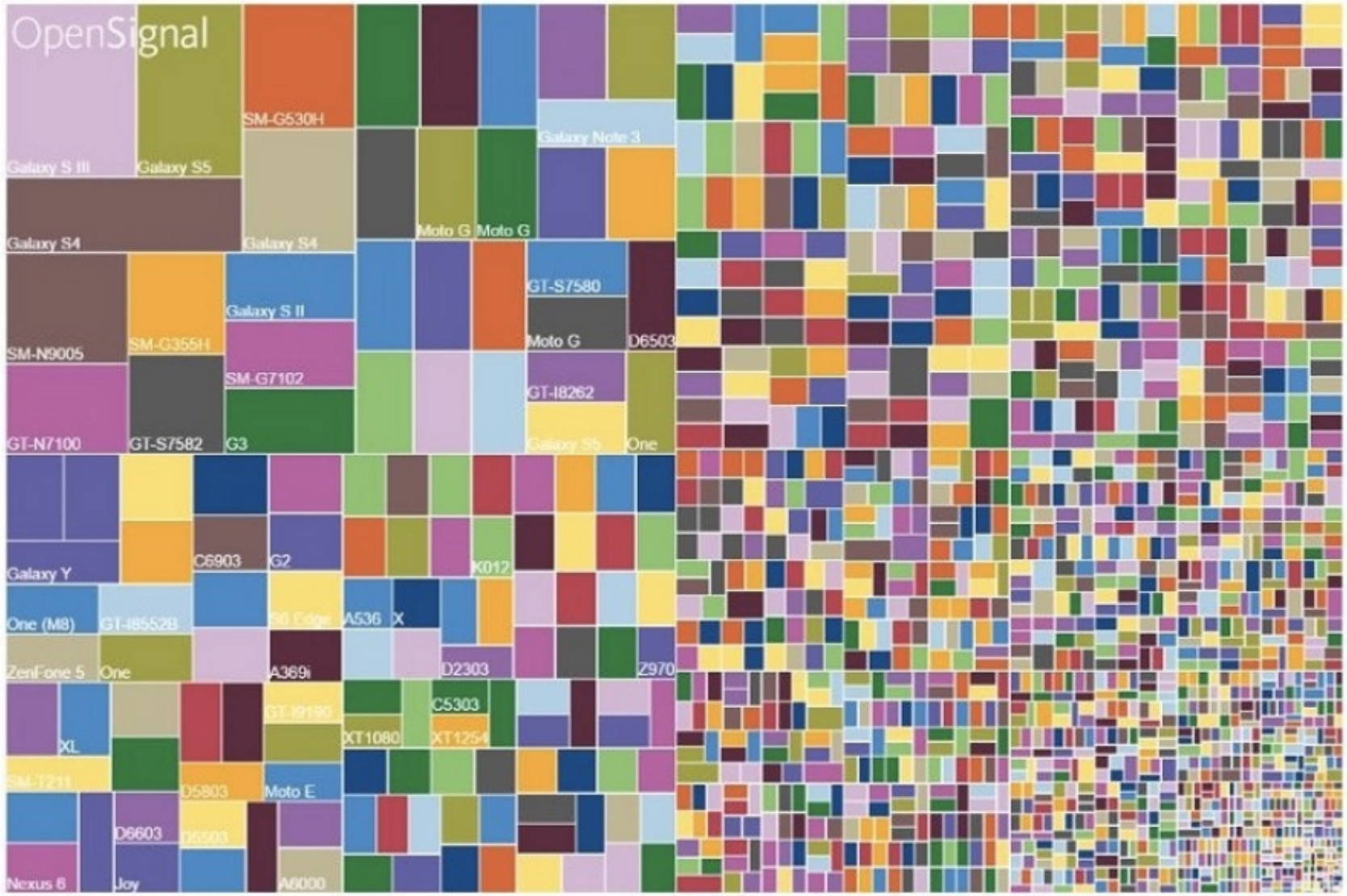
Android.. In details!

- Android is
  - Open source
    - software for which the original source code is made freely available and may be redistributed and modified
  - Linux-based software stack

Open source

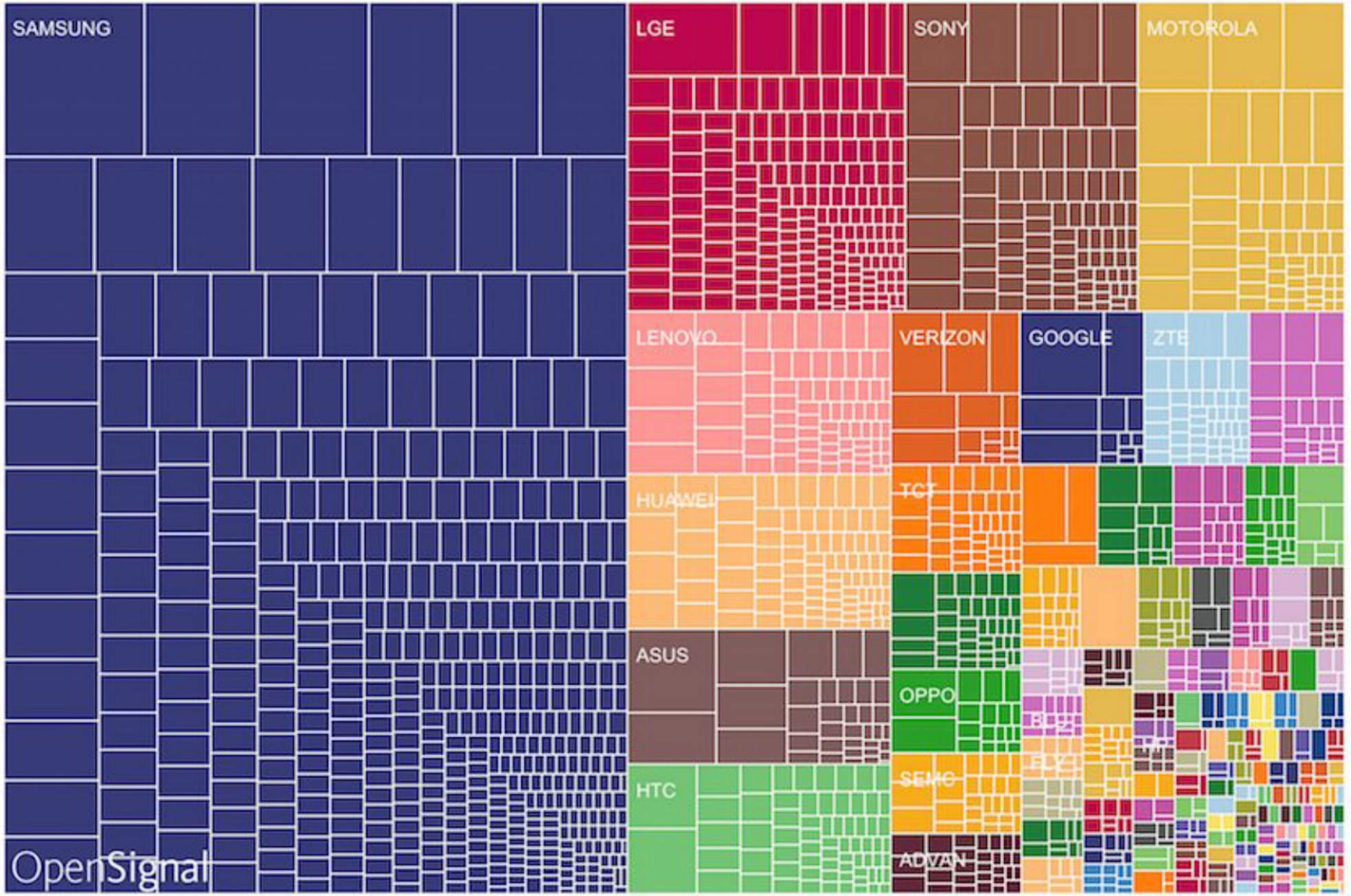
- Some issues…

Device Fragmentation (2015)



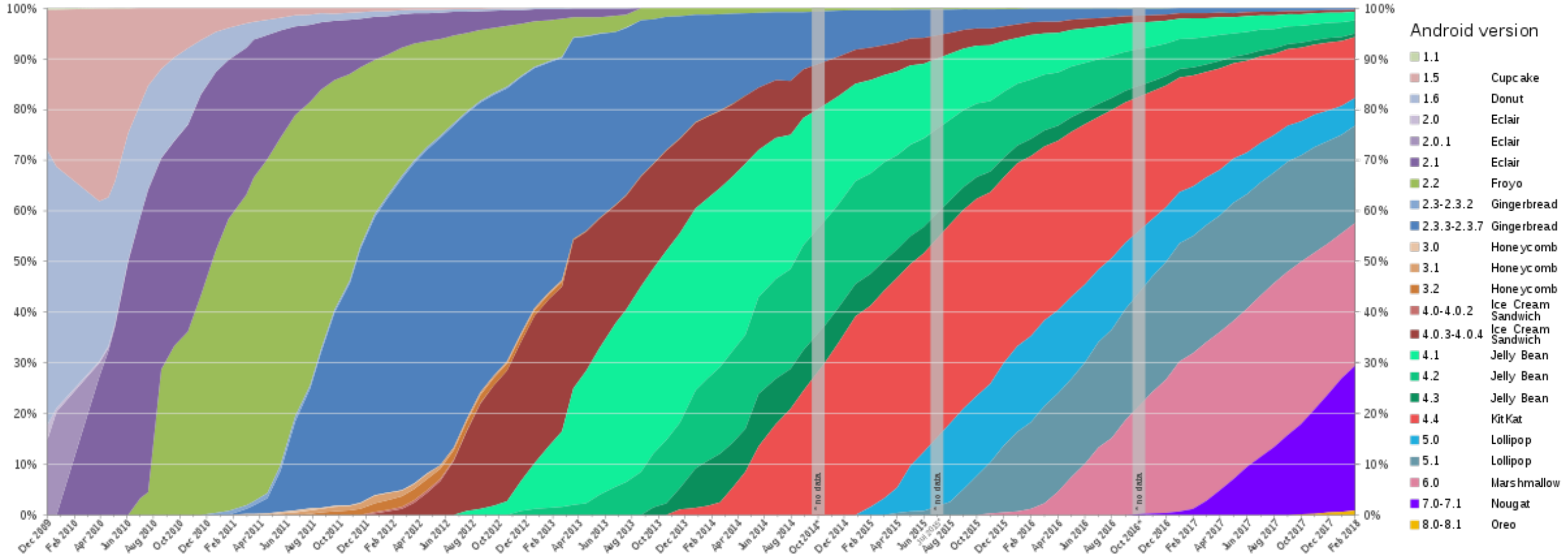
A visualization of the most common Android devices in use. Image: OpenSignal

BRAND FRAGMENTATION (2015)



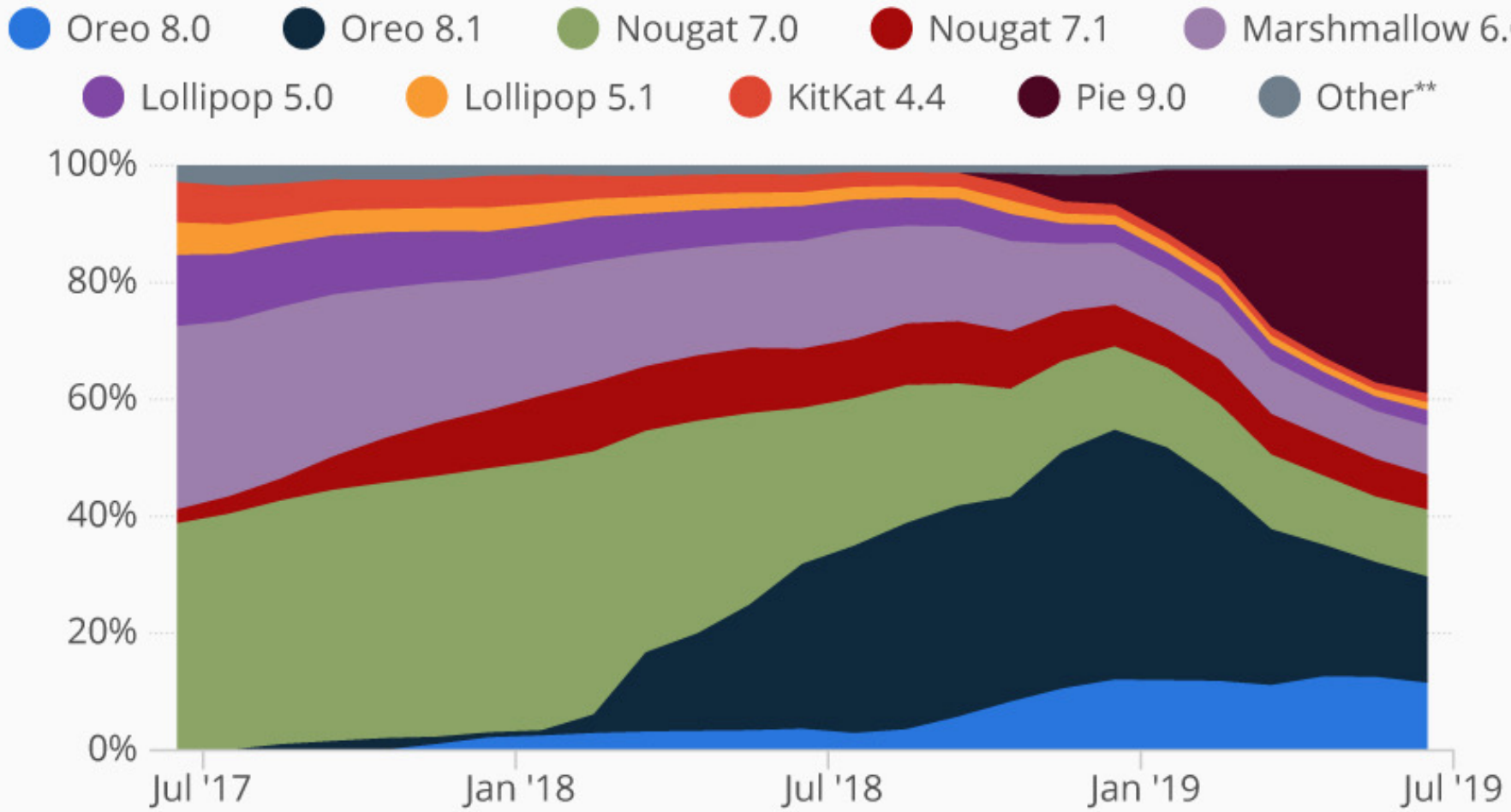
Android version fragmentation

From 2010 to 2018



Android OS Still a Fragmented Market

Mobile Android operating system market share by version in the United States 2017-2019*



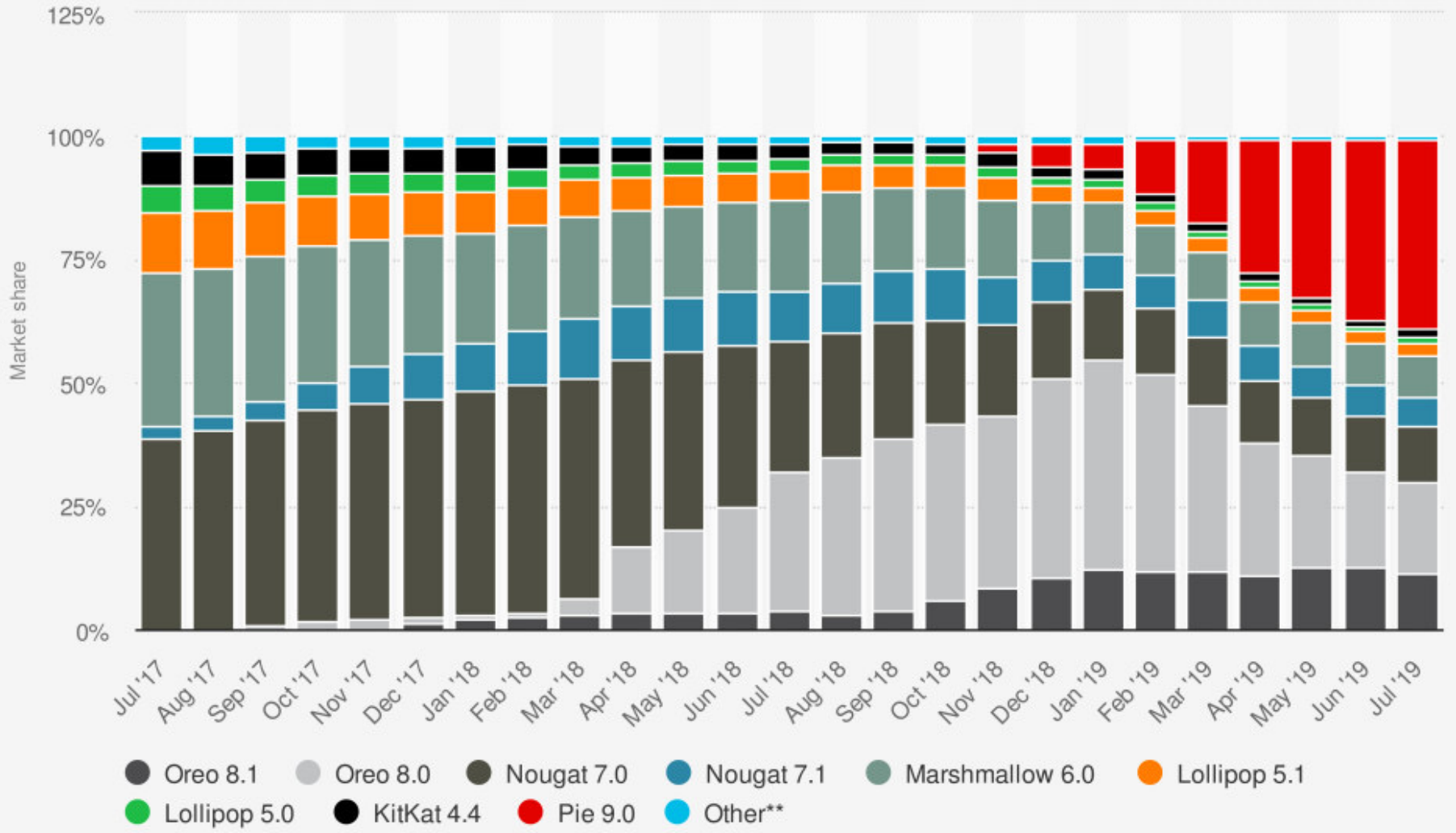
* Based on aggregate data collected by StatCounter on a sample exceeding 10 billion pageviews per month collected from across the StatCounter network of more than 2 million websites.

** Other includes all versions with a share of less than one percent

@StatistaCharts

Source: StatCounter

statista



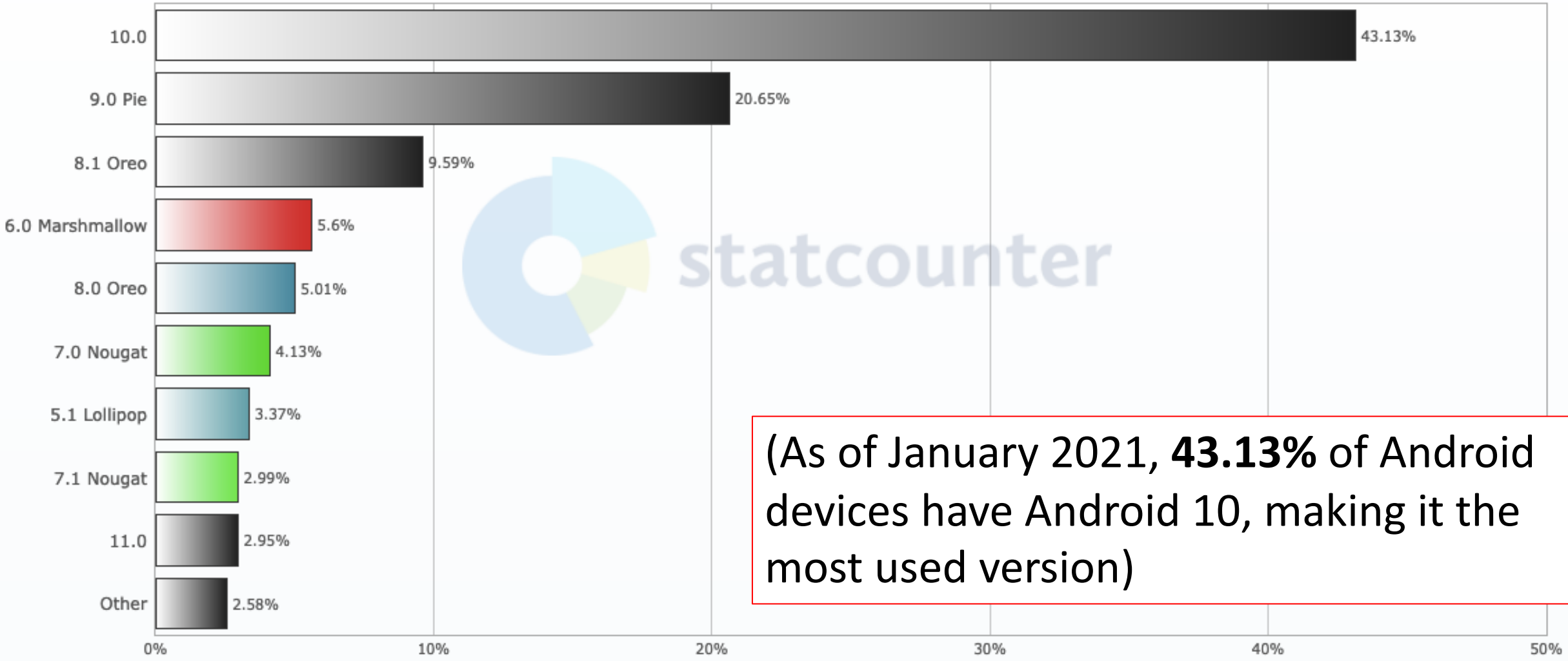
Source
StatCounter
© Statista 2019

Additional Information:
United States; StatCounter; 2017 to 201

statista PC PCMAG.COM

Mobile & Tablet Android Version Market Share Worldwide

Jan 2021



(As of January 2021, **43.13%** of Android devices have Android 10, making it the most used version)

Just a clarification..

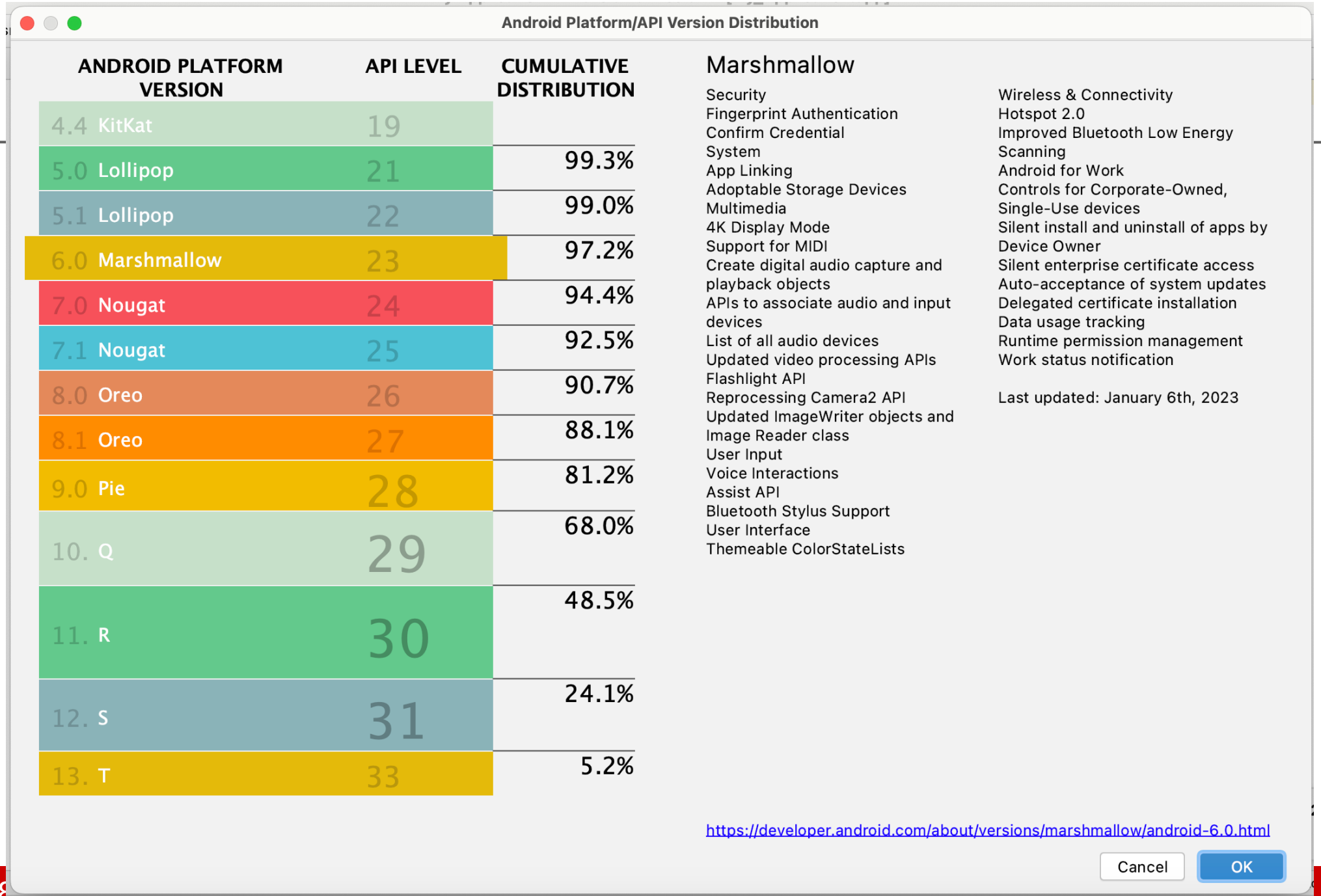
- A Generic System Image (GSI) is a pure Android implementation with unmodified AOSP (Android Open Source Project) code

Which Android version?

- For which Android version should we (as developers) design our app?

Which Android version?

- For which Android version should we (as developers) design our app?
  - We should ensure a certain degree of compatibility with the previous version
  - We have to select a target version (the primary one) and then a min version for compatibility


Android Platform/API Version Distribution
ANDROID PLATFORM VERSION
API LEVEL
CUMULATIVE DISTRIBUTION
4.4 KitKat 19
5.0 Lollipop 21 99.3%
5.1 Lollipop 22 99.0%
6.0 Marshmallow 23 97.2%
7.0 Nougat 24 94.4%
7.1 Nougat 25 92.5%
8.0 Oreo 26 90.7%
8.1 Oreo 27 88.1%
9.0 Pie 28 81.2%
10. Q 29 68.0%
11. R 30 48.5%
12. S 31 24.1%
13. T 33 5.2%
Marshmallow
Security
Fingerprint Authentication
Confirm Credential
System
App Linking
Adoptable Storage Devices
Multimedia
4K Display Mode
Support for MIDI
Create digital audio capture and playback objects
APIs to associate audio and input devices
List of all audio devices
Updated video processing APIs
Flashlight API
Reprocessing Camera2 API
Updated ImageWriter objects and Image Reader class
User Input
Voice Interactions
Assist API
Bluetooth Stylus Support
User Interface
Themeable ColorStateLists
Wireless & Connectivity
Hotspot 2.0
Improved Bluetooth Low Energy Scanning
Android for Work
Controls for Corporate-Owned, Single-Use devices
Silent install and uninstall of apps by Device Owner
Silent enterprise certificate access
Auto-acceptance of system updates
Delegated certificate installation
Data usage tracking
Runtime permission management
Work status notification
Last updated: January 6th, 2023
https://developer.android.com/about/versions/marshmallow/android-6.0.html
Cancel
OK

Android Architecture

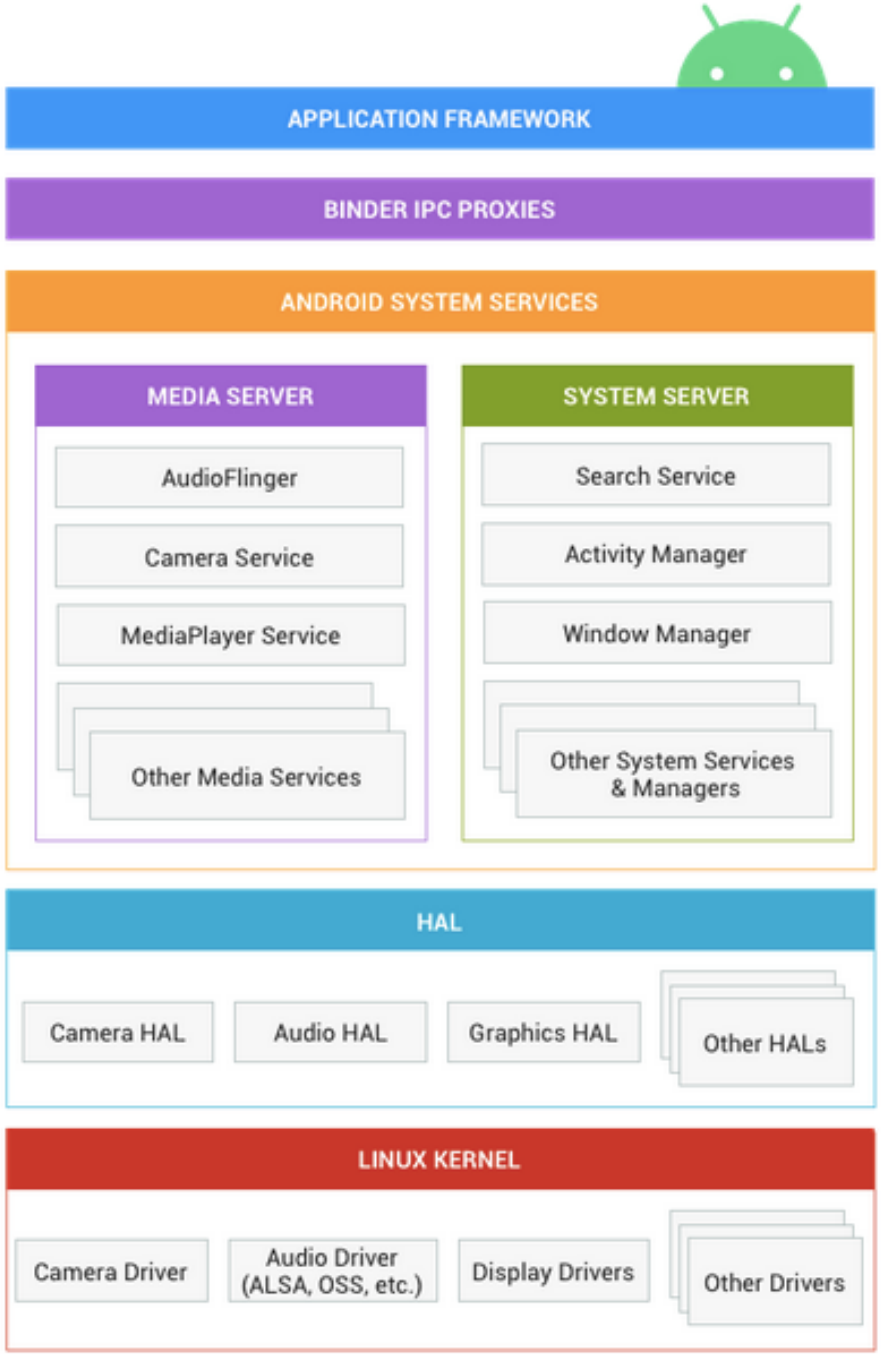


Figure 1. Android system architecture

Architecture

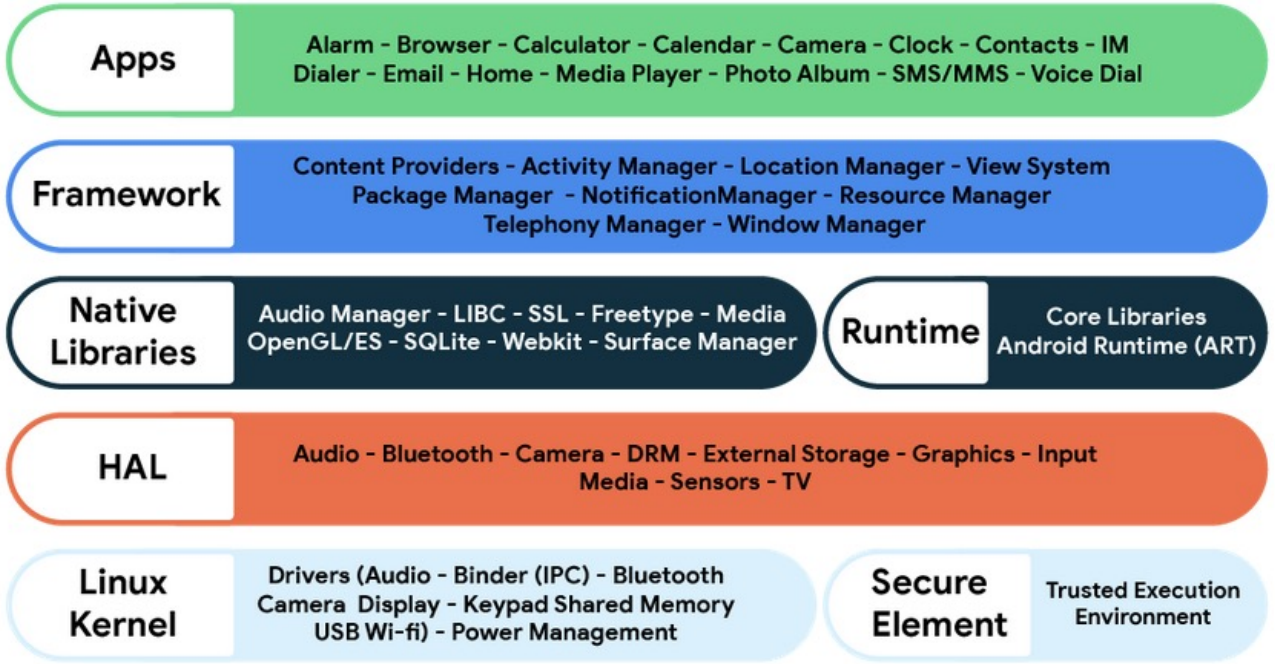
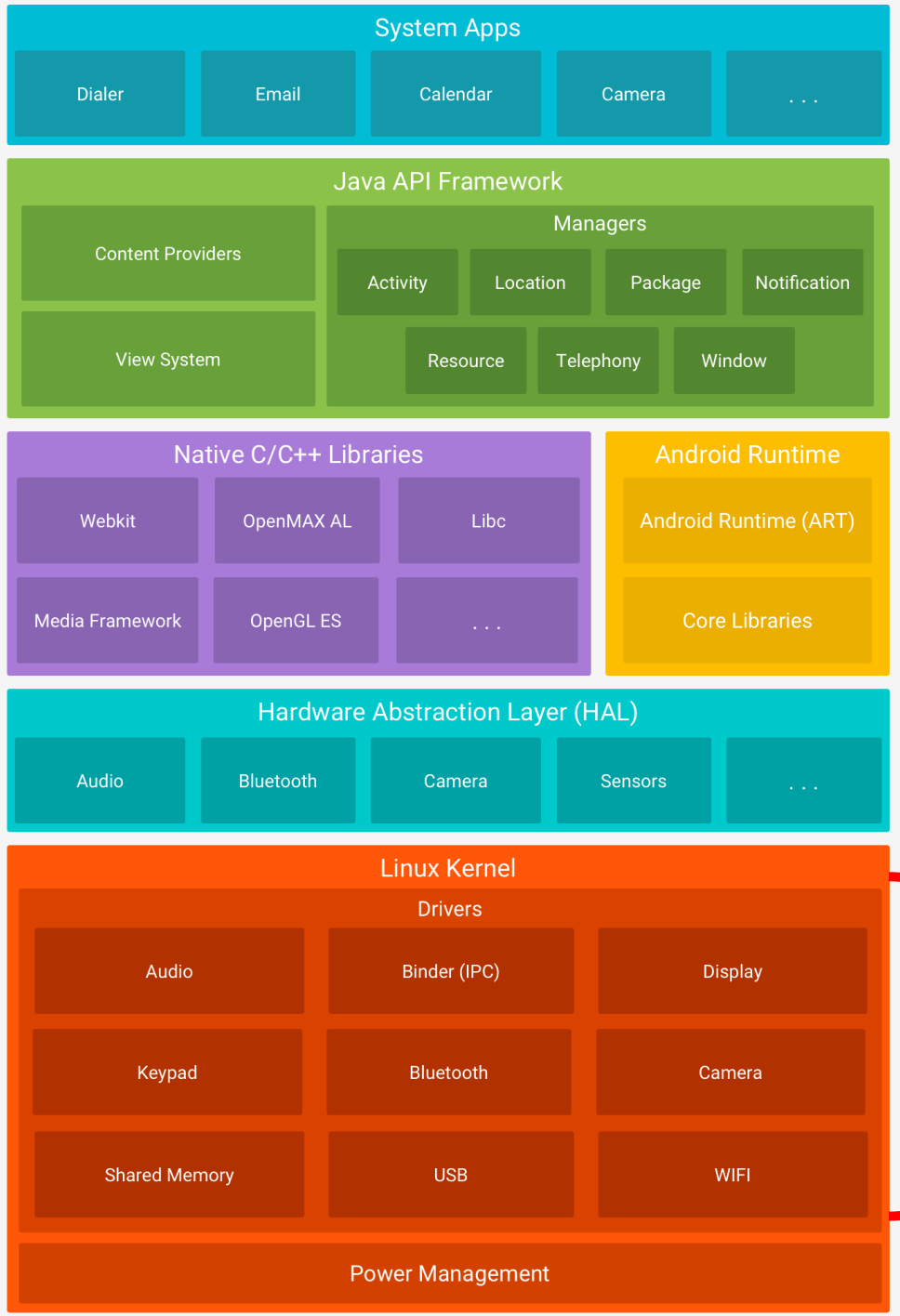


Figure 1. Android stack



Linux Kernel

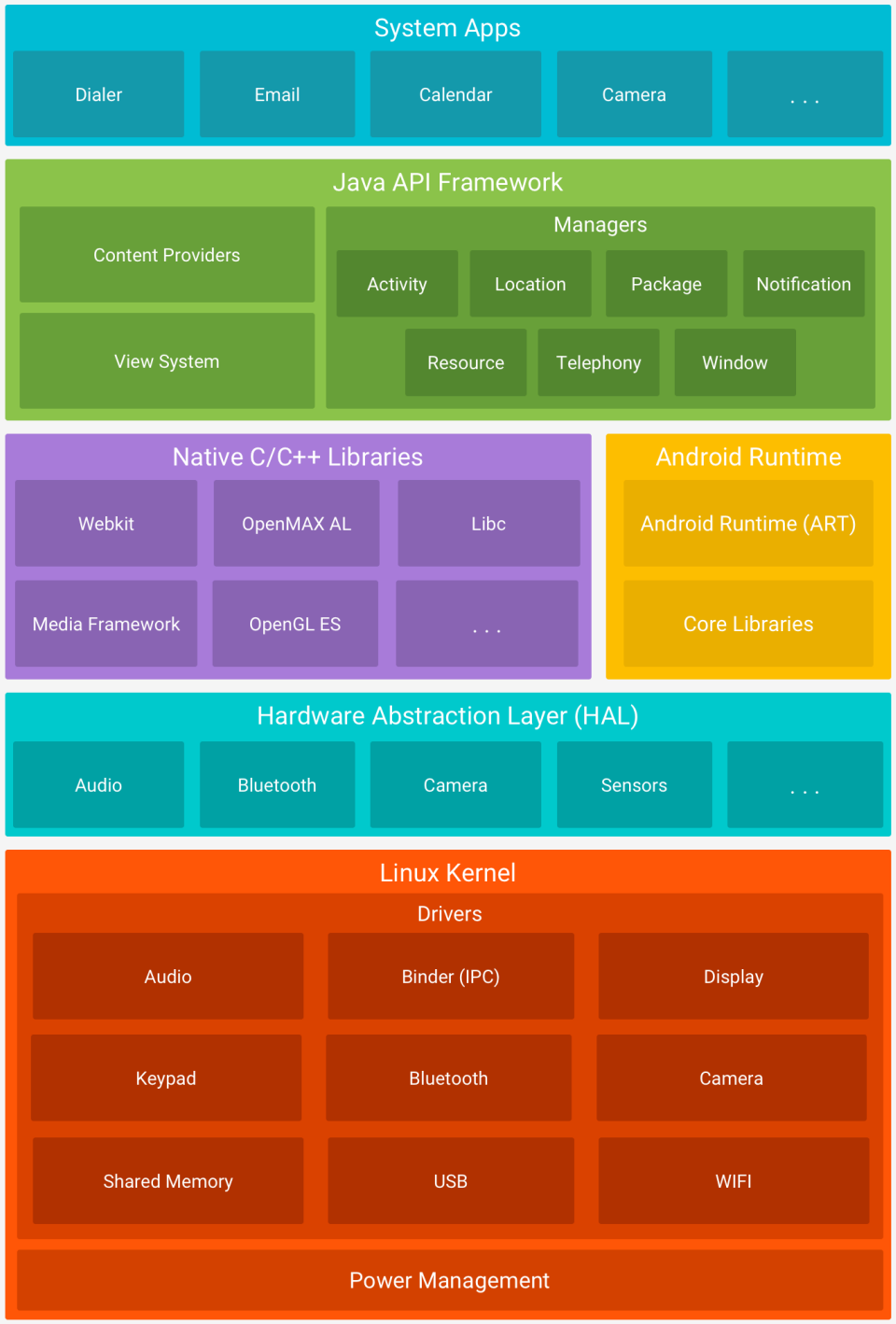
- The foundation of the Android platform is the Linux kernel
  - For example, the Android Runtime (ART) relies on the Linux kernel for underlying functionalities such as threading and low-level memory management
- Using a Linux kernel allows Android to take advantage of key security features and allows device manufacturers to develop hardware drivers for a well-known kernel

Il Linux Kernel

- Benefits
  - Portability (i.e. easy to compile on different hardware architectures)
  - Safety (e.g. Safe multi-process environment)
  - Power Management ART (Android Runtime) relies on the kernel for threading and memory management
  - Manufacturers built on a reliable kernel

Kernel security

- User-based permissions model
- The processes are isolated
- Inter-process communication (IPC)
- Resources are protected from other processes
- Each application has its own User ID (UID)
- Sandbox
- Verified Boot



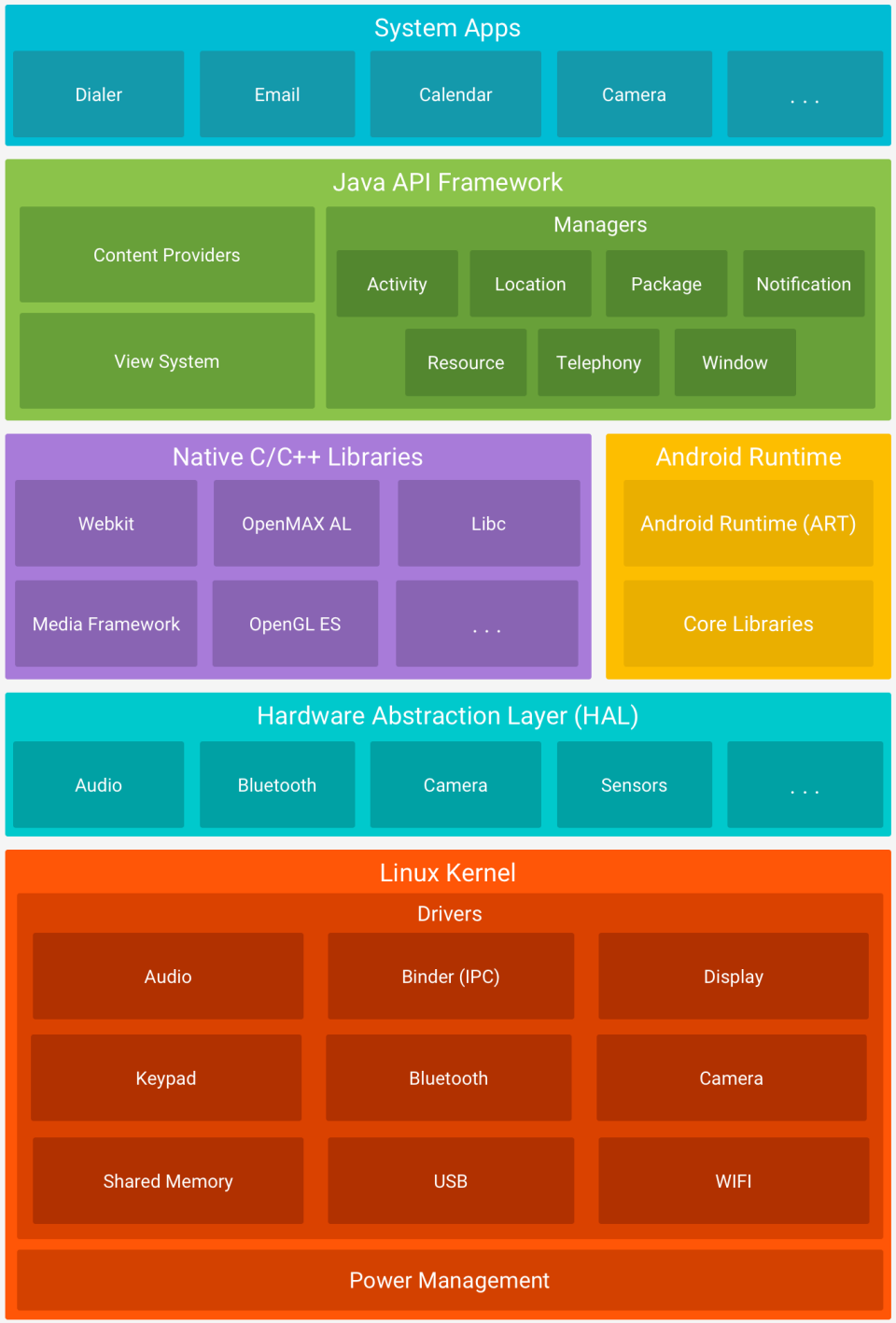
Hardware Abstraction Layer (HAL)

- The hardware abstraction layer (HAL) provides standard interfaces that expose the hardware capabilities of the device to the higher level Java API framework
- The HAL consists of multiple library modules, each of which implements an interface for a specific type of hardware component, such as the camera or Bluetooth module
- When a framework API makes a call to access the device hardware, the Android system loads the library module for that hardware component
  - HAL implementations are grouped into modules and loaded from the Android system at the appropriate time

Hardware Abstraction Layer (HAL)

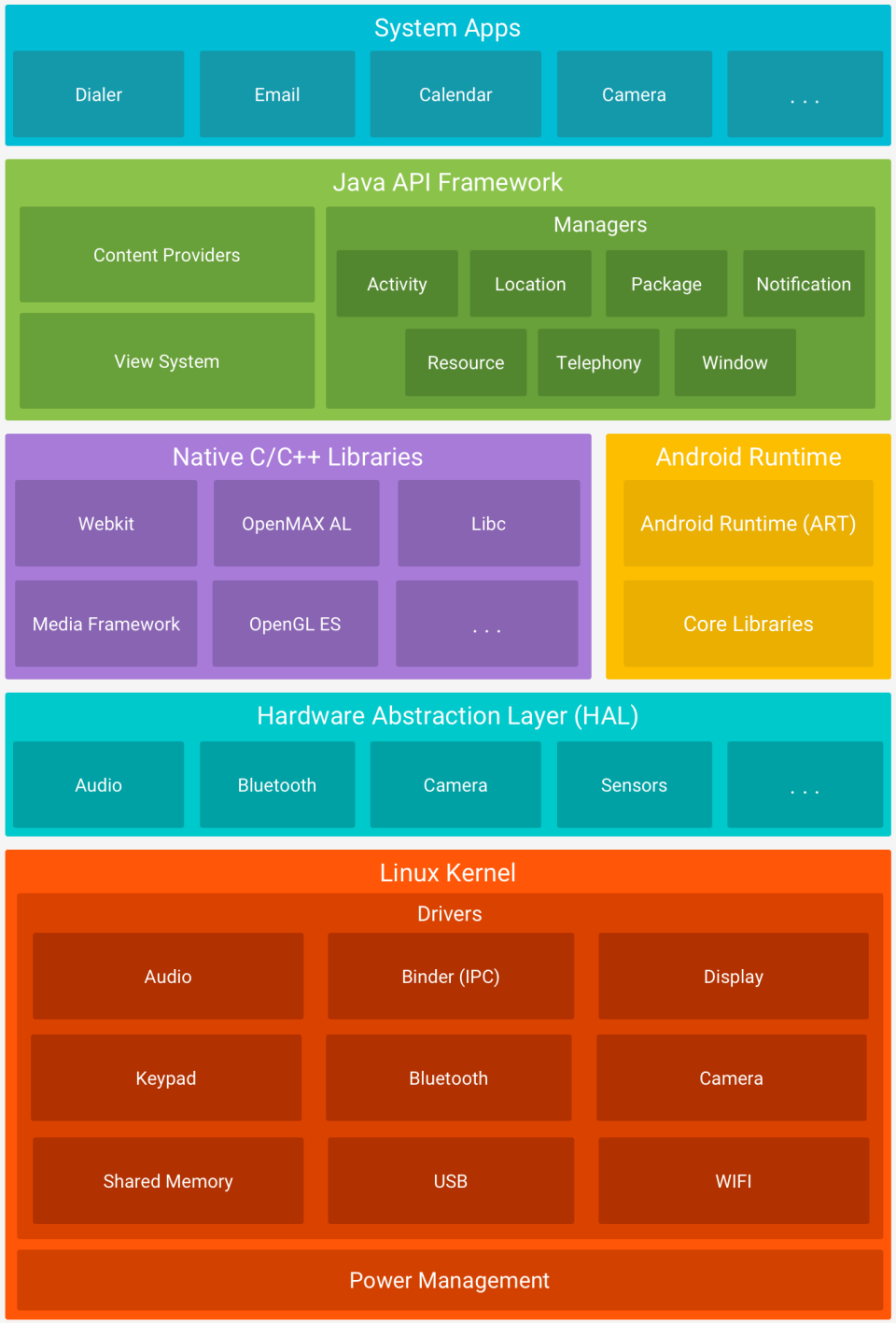
- Benefits
  - Hide the real device
  - Standard interfaces to expose lower-level functionality to higher-level APIs
    - There are standard interfaces that manufacturers need to implement (Android is independent of lower level driver implementations)
    - Application developers rely on common APIs
    - Depending on the hardware, the appropriate libraries are loaded
  - Using a HAL allows you to implement functionality without affecting or modifying the higher level system

Hardware Abstraction Layer (HAL)



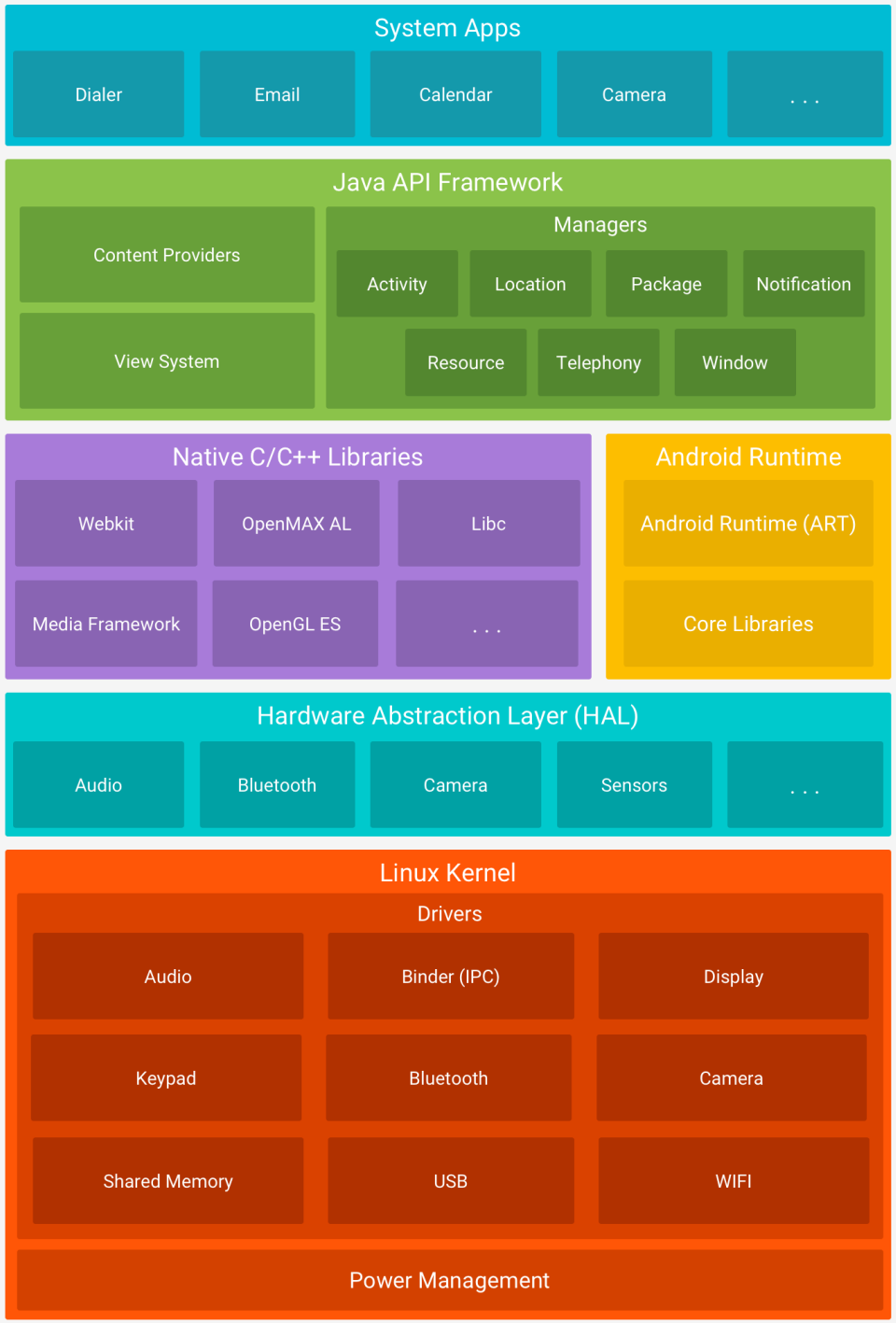
Android Runtime (1/2)

- For devices running Android version 5.0 (API level 21) or higher, **each app runs in its own process** and with its own instance of Android Runtime (ART)
- ART is written to run multiple virtual machines on low-memory devices by running DEX files, a bytecode format designed specifically for Android optimized for a minimal memory footprint
- Build toolchains, like Jack, compile Java sources in bytecode DEX, which can run on the Android platform



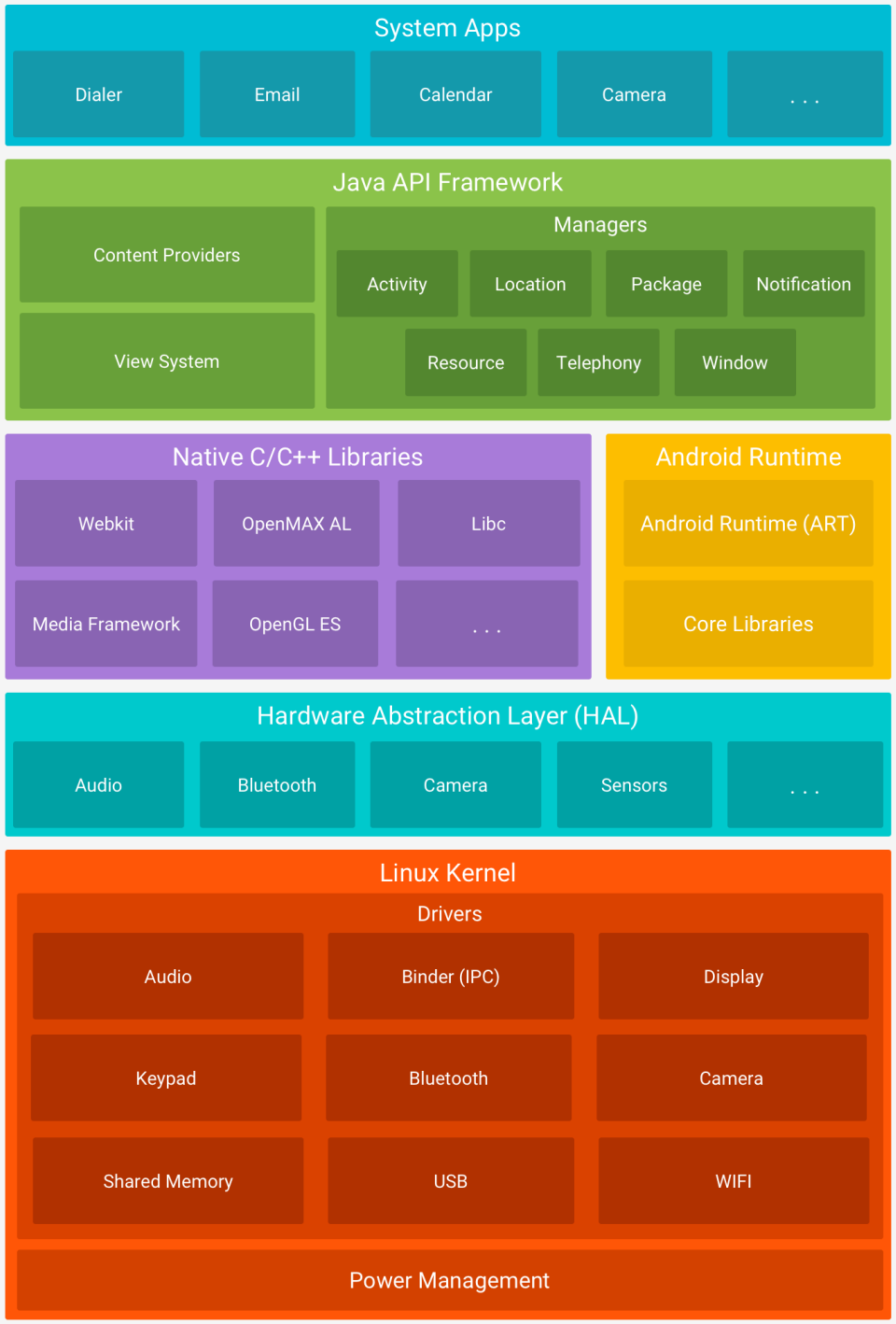
Android Runtime (2/2)

- Some of the main features of ART include:
  - Advance compilation (AOT) and just-in-time (JIT)
  - Optimized Garbage Collection (GC)
  - On Android 9 (API level 28) and later, converting Dalvik executable (DEX) files of an app package into a more compact machine code
  - Improved debugging support, including a dedicated sampling profiler, detailed diagnostic exceptions and crash reports, and the ability to set checkpoints to monitor specific fields
- Android also includes a set of core runtime libraries that provide most of the functionality of the Java programming language, including some features of the Java 8 language, which uses the Java API framework



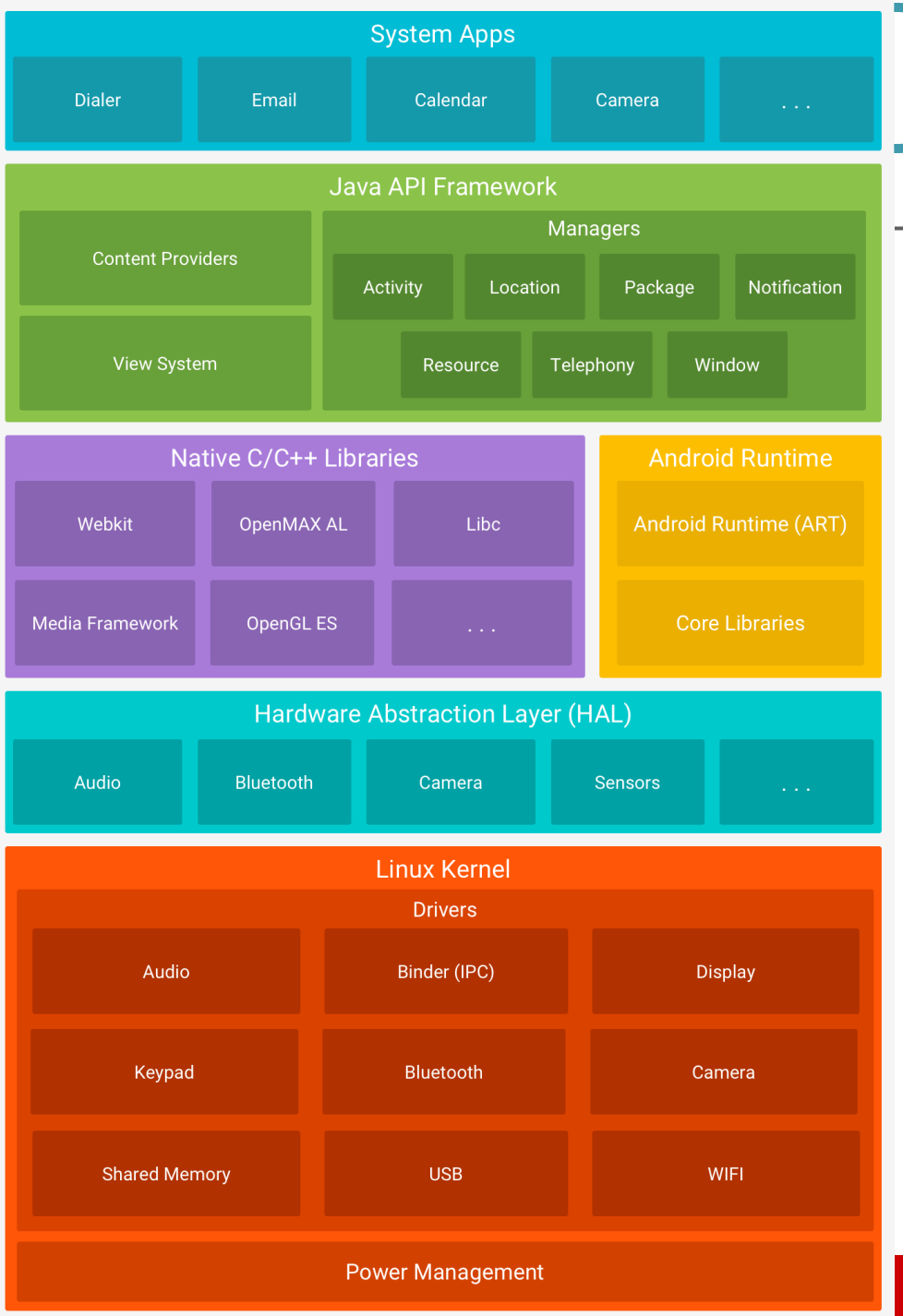
Native C/C++ Libraries

- Many core Android system components and services, such as ART and HAL, are built from native code that requires native libraries written in **C and C ++**
- The Android platform provides Java framework APIs to expose the functionality of some of these native libraries to apps
  - For example, OpenGL ES can be accessed via the Android framework's Java OpenGL API to add support for drawing and manipulating 2D and 3D graphics into your app
- If you need to develop an app that requires C or C ++ code, you can use Android NDK to access some of these native platform libraries directly from your native code.



Java API Framework

- The entire feature set of the Android operating system is available through APIs written in the **Java language**
- These APIs form the building blocks needed to build Android apps by simplifying the reuse of modular and core system components and services, which include the following
  - A rich and extensible **View System** that you can use to create an app's user interface, including lists, grids, text boxes, buttons, and even an embeddable web browser
  - A **Resource Manager** that provides access to non-code resources such as localized strings, graphics, and layout files
  - A **Notification Manager** that allows all apps to display custom alerts in the status bar
  - An **Activity Manager** that manages the app lifecycle and provides a common navigation back stack
  - **Content Providers** that allow apps to access data from other apps, such as the Contacts app or to share their data

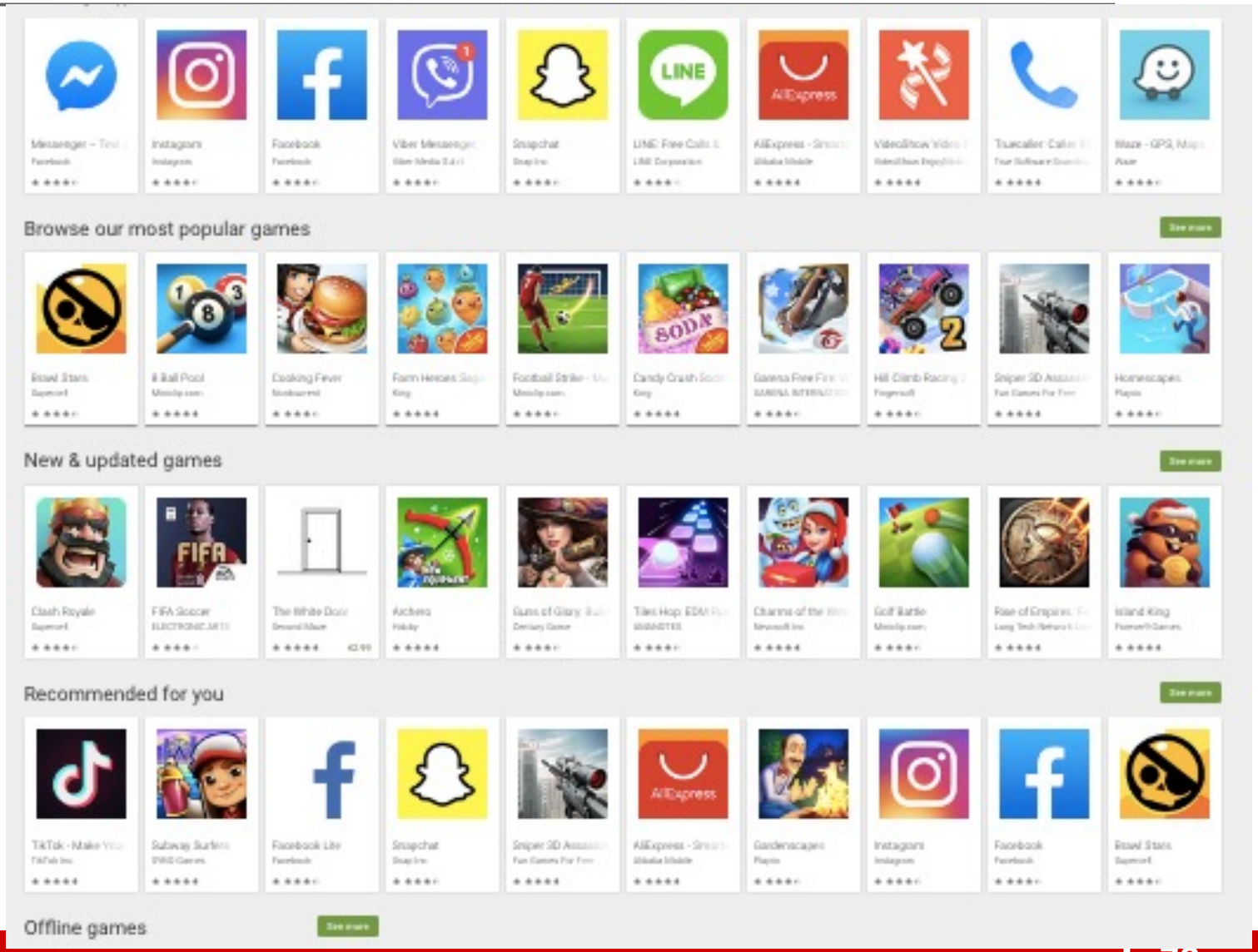


System Apps

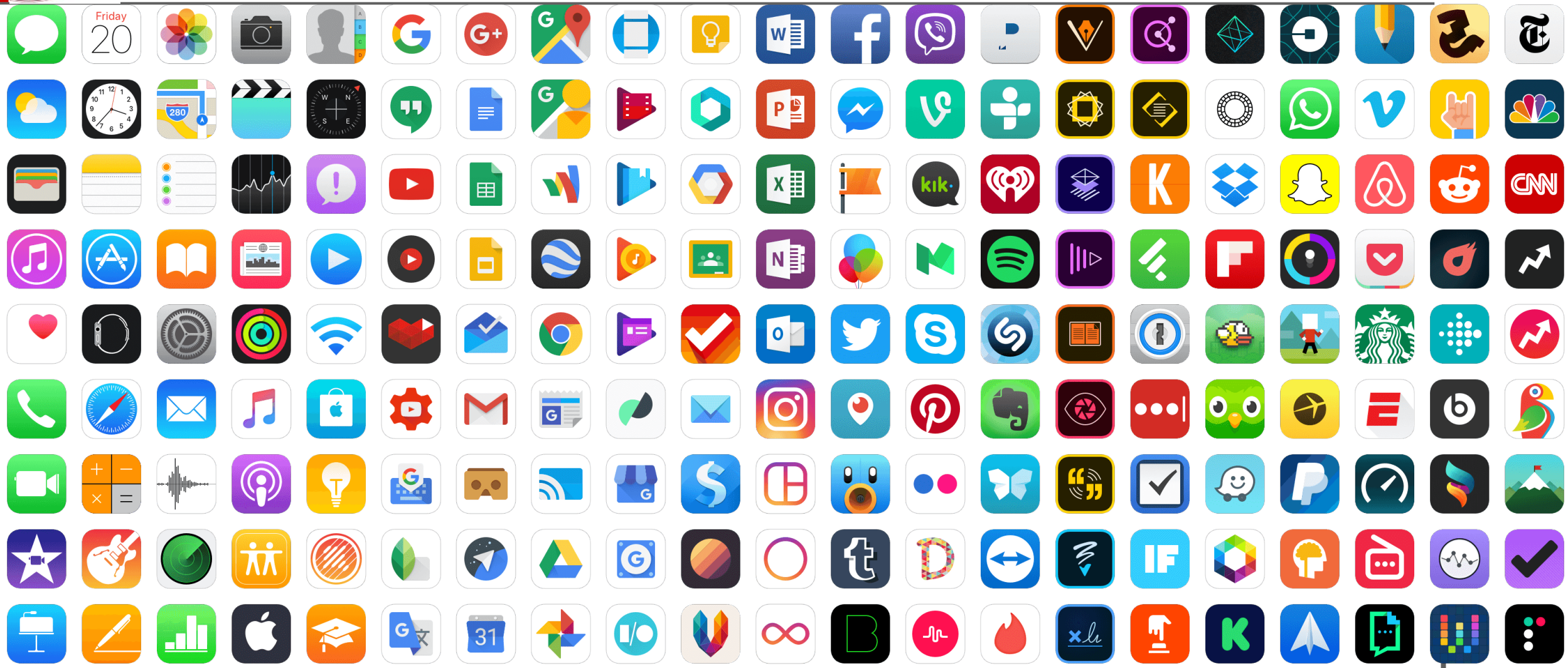
- Android comes with a number of core apps for emails, SMS messages, calendars, internet browsing, contacts, and more
- The apps included with the platform do not have a special status among the apps the user chooses to install
  - Therefore a third party app can become your default web browser, SMS messaging service, or even your default keyboard (some exceptions apply, such as the System Settings app)
- System apps work both as an app for users and to provide key functionality that developers can access from their own app
  - For example, if your app wants to deliver an SMS message, you don't need to create that functionality yourself - you can instead invoke whatever SMS app is already installed to deliver a message to the specified recipient

Apps (mobile apps)

- Software applications that allow access to the most varied services, interactive and dynamic content, games, work content, etc ...



Apps



1. Tools – 99%
2. Communication – 99%
3. Video players and editing – 96%
4. Travel and local – 95%
5. Social media – 95%
6. Productivity – 92%
7. Music – 88%
8. Entertainment – 84%
9. News – 81%
10. Photography – 75%
11. Books – 70%
12. Lifestyle – 65%
13. Personalization – 60%
14. Business – 39%
15. Shopping – 35%
16. Weather – 32%
17. Sports – 30%
18. Education – 28%
19. Finance – 25%
20. Health – 23%

Most popular Google Play store app categories

September, 2019

Link: https://www.statista.com/statistics/200855/favourite-smartphone-app-categories-by-share-of-smartphone-users/

Key Mobile App Statistics

- Mobile apps are expected to generate over $935 billion in revenue by 2023.
- The Apple App Store has 1.96 million apps available for download.
- There are 2.87 million apps available for download on the Google Play Store.
- 21% of Millennials open an app 50+ times per day.
- 49% of people open an app 11+ times each day.
- 69% of all US digital media time comes from mobile apps.
- The average smartphone owner uses 10 apps per day and 30 apps each month.

https://buildfire.com/app-statistics/

HOW MANY APPS DO WE USE?

On average, a person now has more than **80 apps** installed on their phone

175B apps are downloaded yearly

The average person uses...

9 mobile apps on a daily basis and

30 apps every month

Some fundamentals

- Native Android apps can be written in both Java language and Kotlin language (and C ++)
- The Android SDK tools compile the code along with all the data and resource files into an APK, an Android package, which is an archive file with an **.apk** suffix
- An APK file contains all the content of an Android app and is the file used by Android devices to install the app

Kotlin or Java?

Android’s Kotlin-first approach

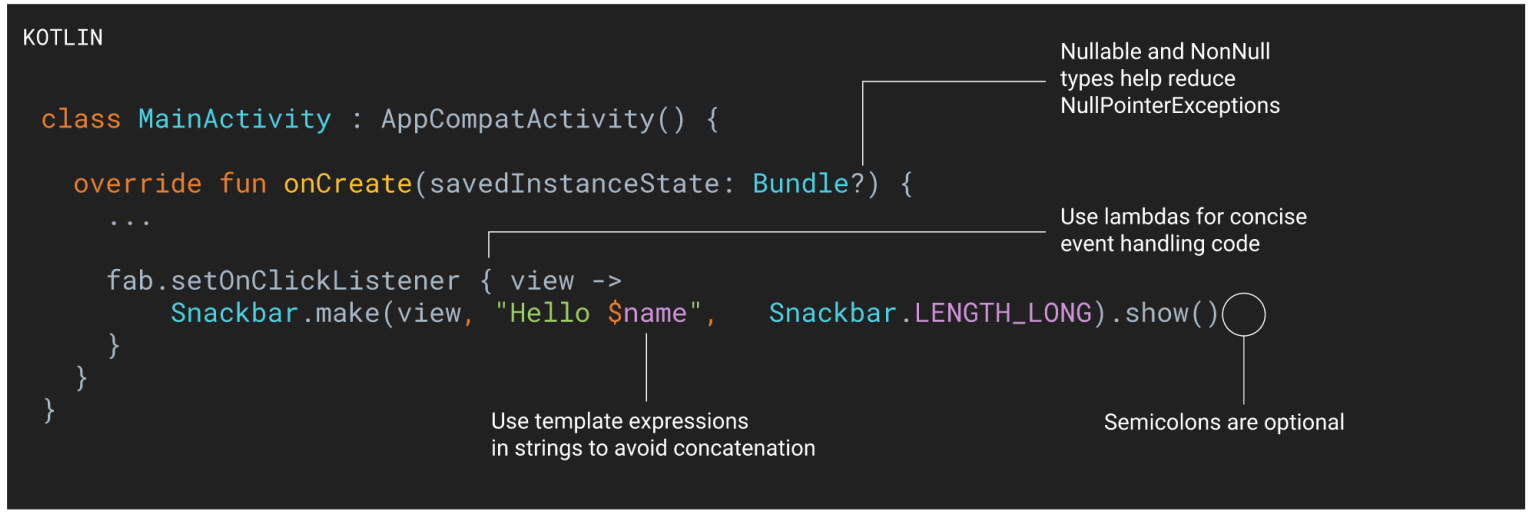
At Google I/O 2019, we announced that Android development will be increasingly Kotlin-first, and we’ve stood by that commitment. Kotlin is an expressive and concise programming language that reduces common code errors and easily integrates into existing apps. If you’re looking to build an Android app, we recommend starting with Kotlin to take advantage of its best-in-class features.

In an effort to support Android development using Kotlin, we co-founded the Kotlin Foundation and have ongoing investments in improving compiler performance and build speed. To learn more about Android's commitment to being Kotlin-first, see Android's commitment to Kotlin.

Develop Android apps with Kotlin

Write better Android apps faster with Kotlin. Kotlin is a modern statically typed programming language used by over 60% of professional Android developers that helps boost productivity, developer satisfaction, and code safety.

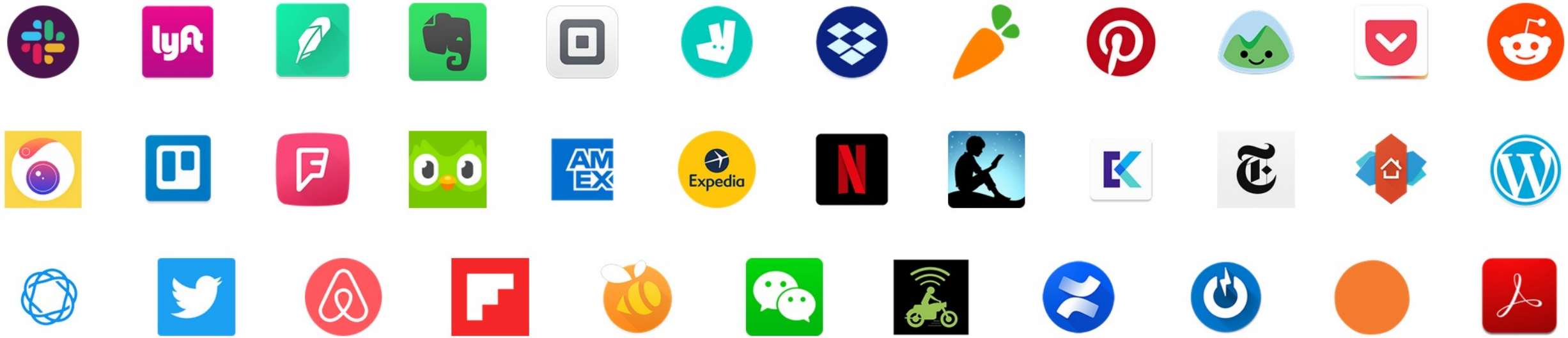
What does Kotlin code look like?



Apps built with Kotlin

Many apps are already built with Kotlin—from the hottest startups to Fortune 500 companies. Learn how Kotlin has helped their teams become more productive and write higher quality apps.

See developer stories



Some fundamentals

- Each Android app lives in its own **security sandbox**, protected by the following Android security features
  - The Android OS is a multi-user Linux system where **each app is a different user**
  - By default, the system assigns each app a unique Linux user ID (the ID is used only by the system and is unknown to the app).
  - The system sets permissions for all files in an app so that only the user ID assigned to that app can access them
  - Each process has its own virtual machine (VM), so an app's code runs separately from other apps
    - By default, each app runs in its own Linux process.
  - The Android system starts the process when one of the app components needs to run, then stops the process when it is no longer needed or when the system needs to recover memory for other apps

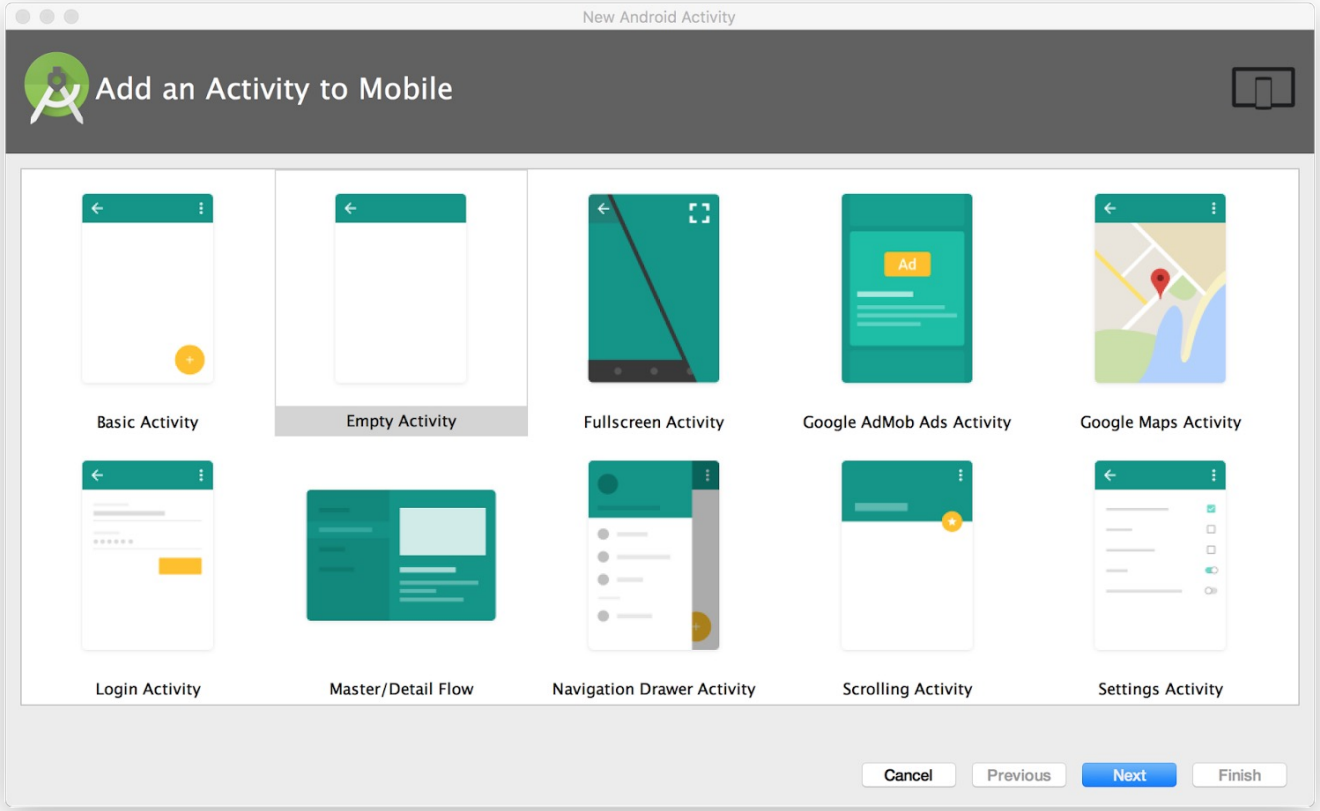
Some fundamentals

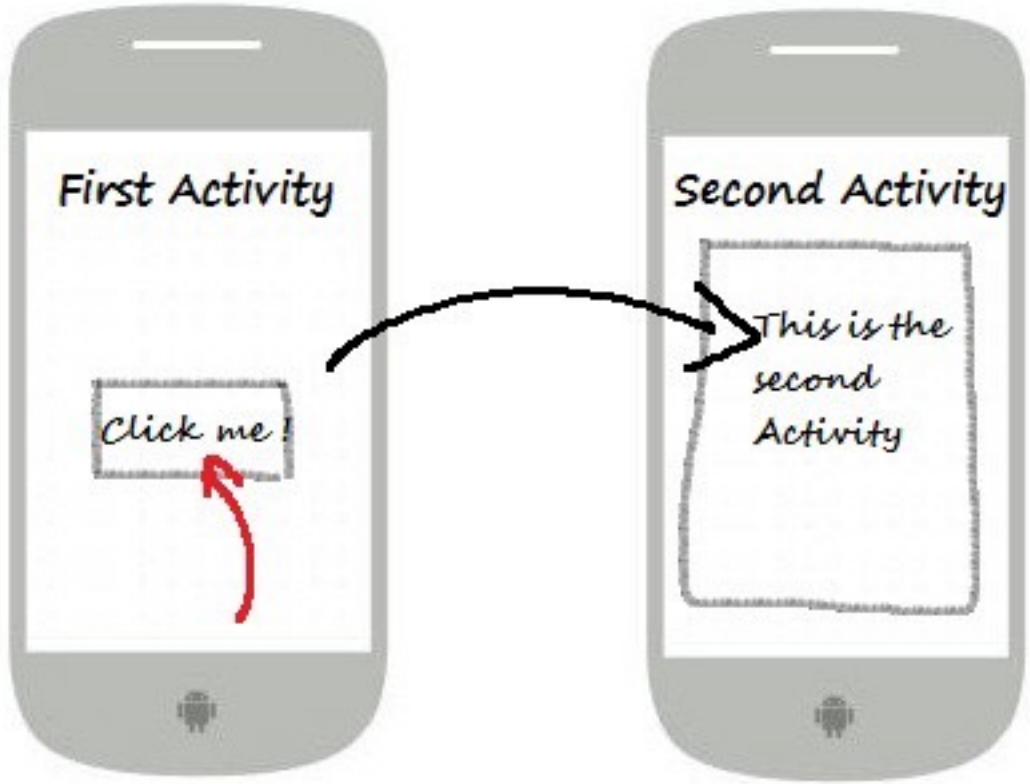
- The Android system implements the principle of least privilege each app, by default, only has access to the components it needs to do its job and no more.
- This creates a very secure environment in which an app cannot access parts of the system for which it has not been authorized
- However, an app may ask for permission to access device data such as device location, camera, and Bluetooth connection
- The user must explicitly grant these permissions

App *components*

- App Components are the fundamental blocks / bricks of an Android app
- Each component is an access point (entry point) through which the system or a user can access the app
- Some components depend on others
- There are 4 different types of components:
  - Activities
  - Services
  - Broadcast receivers
  - Content providers

Activities


New Android Activity
Add an Activity to Mobile
Basic Activity
Empty Activity
Fullscreen Activity
Google AdMob Ads Activity
Google Maps Activity
Login Activity
Master/Detail Flow
Navigation Drawer Activity
Scrolling Activity
Settings Activity
Cancel
Previous
Next
Finish


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

Activities

- An Activity is an entry point for interacting with the user
- Represents a single screen with a user interface
  - For example, an email app might have an activity showing a list of new emails, another activity for composing an email, and another activity for reading emails. While the activities work together to form a consistent user experience in the email app, each is independent of the others. Therefore, a different app can initiate any of these activities if the email app allows it. For example, a camera app can initiate the activity in the email app, write new email activity to allow the user to share an image

Activities

- An activity facilitates the following key interactions between system and app:
  - Keep track of what the user currently has on their screen to ensure that the system continues to run the process hosting the task
  - Knowing that previously used processes contain elements to which the user can return (interrupted activities) and therefore giving higher priority to maintaining those processes
  - Help the app manage its interrupted process so that the user can return to the activities with the previous state restored
  - Provide a way for apps to implement user flows with each other and for the system to coordinate these flows (the most classic example is "share")
  - An activity is implemented as a subclass of the Activity class

Services

- A Service is a generic entry point for keeping an app running in the background
  - It is a component that runs in the background to perform long-running operations or perform jobs for remote processes
- A service does not provide a user interface
  - For example, a service could play music in the background while the user is in another app, or it could retrieve data on the network without blocking the user's interaction with an activity.
- Another component, such as an activity, can start the service and leave it running or associate with it to interact with it
- There are actually three types of services that tell the system how to manage an app:
  - Foreground
  - Background
  - Bound services

Started services

Foreground Services

- They tell the system to keep them running until their work is completed, even if the user is not interacting with the app
- To play music even after the user has left the app or for notifications:
  - Music playback is something that the user is directly aware of, so in this case the system knows that it really has to do its best to keep the process of that service running, because the user will notice if it disappears

Background Services

- They are services that run in the background without the user being directly aware of them as running
- In this case, the system has more freedom in managing its process
  - It can allow it to be killed (and then restarting the service sometime later) if it needs RAM for things that are of more critical / important to the user

Services

- Because of their flexibility (for better or for worse), services have proved to be a really useful element
  - Animated wallpapers, notification listeners, screen savers, input methods, accessibility services and many other core system features are all built as services implemented by applications and the system undertakes to manage their execution when needed.
- The component is implemented as a subclass of Service

Broadcast receivers

- A Broadcast receiver is a component that allows the system to send events to the app outside of a normal user flow, allowing the app to respond to system-wide broadcast announcements
- Since broadcast receivers are a well-defined entry in the app, the system can also provide broadcasts to apps that are not currently running
  - for example, an app can schedule an alarm to display a notification to notify the user of an upcoming event ... and by delivering that alarm to an app's BroadcastReceiver, the app does not need to remain running until the the alarm goes off

Broadcast receivers

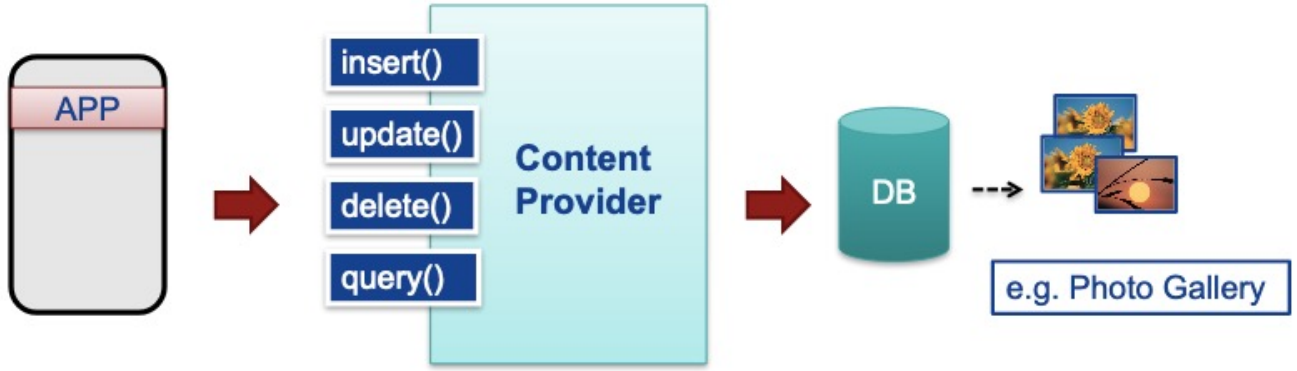
- Many broadcasts come from the system for example, a broadcast announcing that the screen is off, the battery is low, or an image has been taken
- Apps can also start broadcasts
  - for example to let other apps know that some data has been downloaded to your device and is available for use
- Although broadcast receivers do not display a user interface, they can create status bar notifications to alert the user when a broadcast event occurs

Broadcast receivers

- More commonly, however, a broadcast receiver is only a gateway to other components and is intended to do a minimal amount of work
- For example, it might schedule a job service to run some jobs based on event (with JobScheduler)
- A broadcast receiver is implemented as a subclass of BroadcastReceiver and each broadcast is delivered as an Intent object

Content providers

- A content provider manages a shared set of app data that can be stored on the file system, in an SQLite database, on the web, or in any persistent storage location that the app can access
- Through the content provider, other apps can query or modify data if the content provider allows
  - For example, the Android system provides a content provider that manages the user's contact information. Therefore, any app with the appropriate permissions can request the content provider, ContactsContract.Data, to read and write information about a particular person.



Content providers

- You can think of a content provider as an **abstraction on a database**. However, it has a different main purpose from a system design perspective
- For the system, **a content provider is an entry point into a data publishing app, identified by a URI scheme**
- Content providers are also useful for reading and writing private and unshared data in your app
  - A Content Provider is implemented as a subclass of ContentProvider and must implement a standard set of APIs that allow other apps to perform transactions

App components

- A unique aspect of the Android system design is that **any app can launch another app's component (**through **Android)**
  - For example, if you want the user to take a picture with the device's camera, there is probably another app that does and the app can use it instead of developing a new activity. This can be done simply by activating the activity in the camera app. As soon as the photo is taken, the photo is returned to the app being created to be managed. For the user, it is as if the camera were part of the app he is using

App components

- When the system starts a component, it starts the process for that app (if it's not already running) and instantiates the classes needed for the component.
  - For example, if the app starts the activity in the camera app that takes a photo, that activity is performed in the process that belongs to the camera app, not in the app process.
- Therefore, unlike apps in most other systems, **Android apps don't have a single entry point** (there is no main () function)
  - Since the system runs each app in a separate process with file permissions restricting access to other apps, **the app cannot directly activate a component from another app. However, the Android system can**. To activate a component in another app, it delivers a message to the system specifying the intention to launch a particular component. The system then activates the component for you

Intent

- To activate 3 of the 4 available components, i.e. "activities, services, and broadcast receivers" an asynchronous message called **intent** is used
  - describing an action to be performed, including what data to act on, the category of the component that should perform the action, and other instructions Intents associate individual components with each other at runtime
- They can be considered as messengers that require a component action (regardless of whether the component belongs to your app or another)
- An Intent is created with an Intent object, which defines a message to activate a specific component (explicit intent) or a specific type of component (implicit intent)

Intent

- **For activity and service, an intent defines the action to perform** (for example, to display - view - or send - send - something) and can specify the URI of the data to act on, one of the things that a started component could need to know
  - For example, an intent might broadcast a request for an activity to show an image or to open a web page
- In some cases, it is possible to start an activity to receive a result, in which case the activity also returns the result in an Intent
  - For example, it is possible to issue an intention to allow the user to choose a personal contact and return it to the user. The return intent includes a URI that points to the chosen contact
- **For broadcast receivers, the intent simply defines the announcement to be transmitted**
  - For example, a transmission to indicate that the device battery is low includes only a known action string indicating that the battery is low

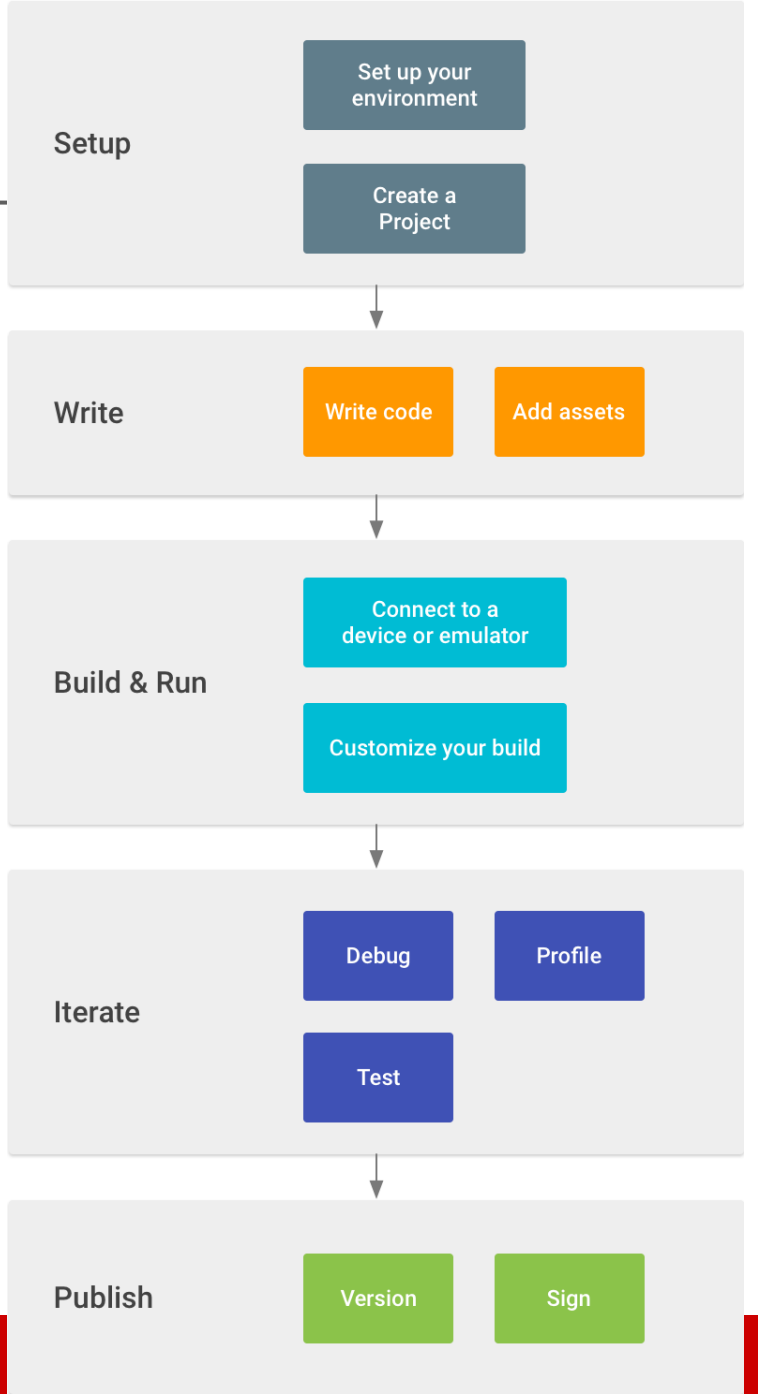
Intent e Content providers

- Unlike the other three components, content providers are not triggered by intents. Instead, they are triggered when they are targeted by a request for a ContentResolver
- The ContentResolver handles all direct transactions with the content provider in such a way that the component that is transacting with the provider does not have to do so but instead only has to call methods on the ContentResolver object
  - In other words, it leaves a layer of abstraction between the content provider and the component requesting information (for security reasons)

How to develop an Android app?

Developer workflow

- Setup
  - Install android studio and create an app
- Write the app
  - Android Studio includes a variety of tools and intelligence to help you work faster, write quality code, design a user interface, and create resources for different types of devices
- Build and run
  - During this phase, the debuggable APK package is created that can be installed and run on the emulator or on an Android device
- Debug, profile, and test
  - This is the iterative phase where you keep writing the app but focusing on eliminating bugs and optimizing the app's performance. Of course, creating tests will help these efforts
- Publish



Setup

- Download and Install Android Studio

Android Studio provides the fastest tools for building apps on every type of Android device.

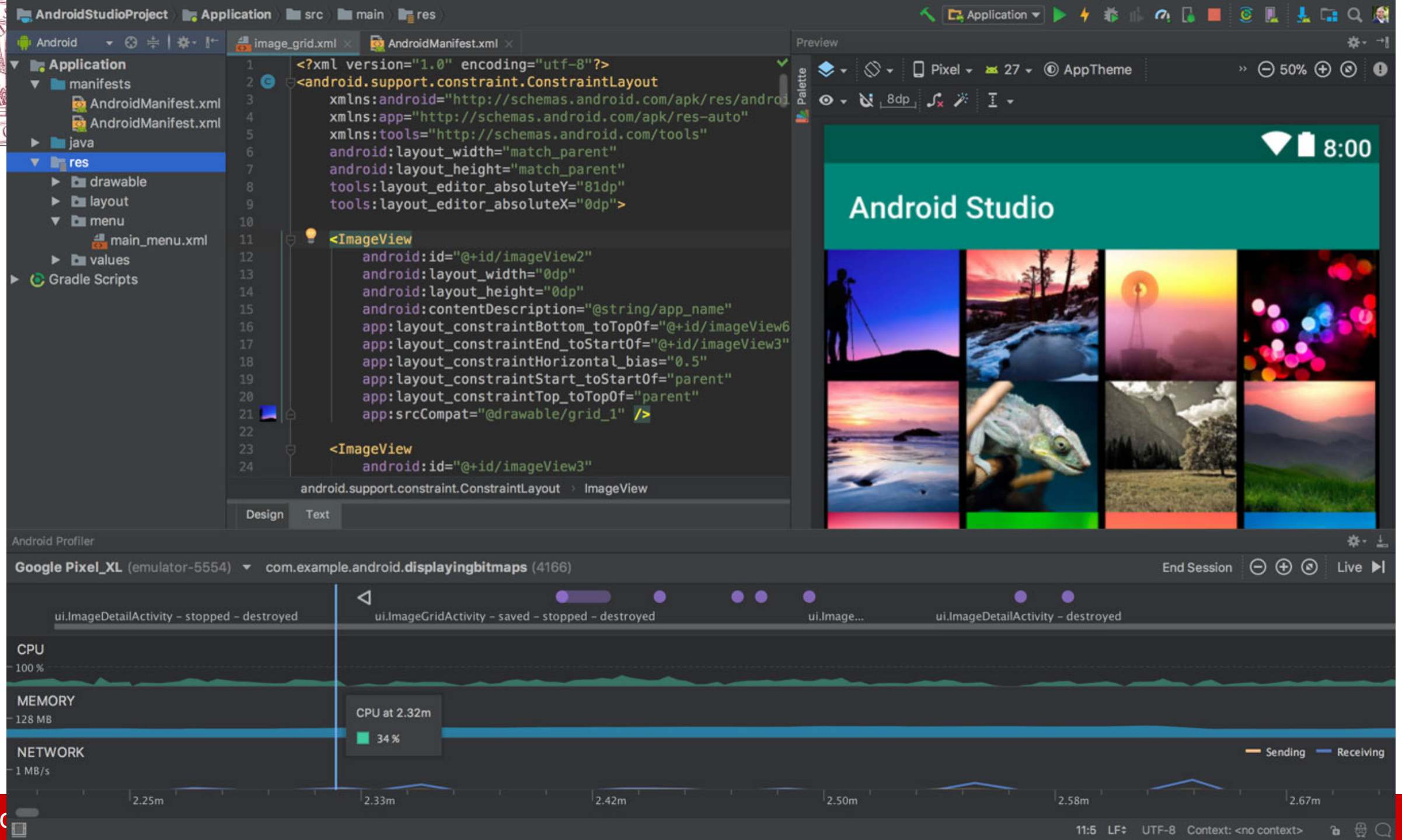
Download Android Studio

Android Studio

- Android Studio provides a complete IDE, including an advanced code editor and app templates
- It also contains tools for development, debugging, testing, and performance that make it faster and easier to develop apps
- You can use Android Studio to test your apps with a large range of preconfigured emulators, or on your own mobile device
- You can also build production apps and publish apps on the Google Play store

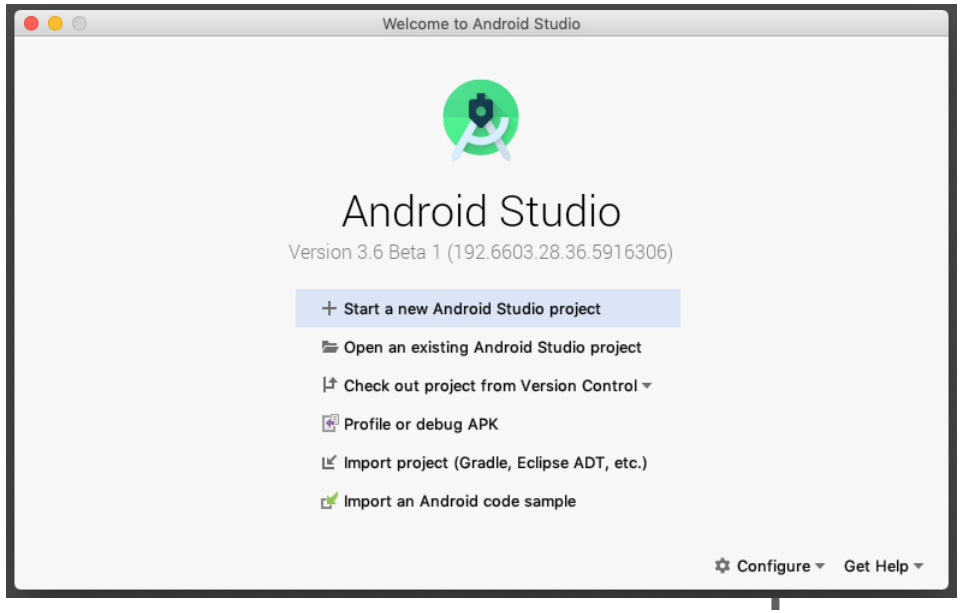
Android Studio

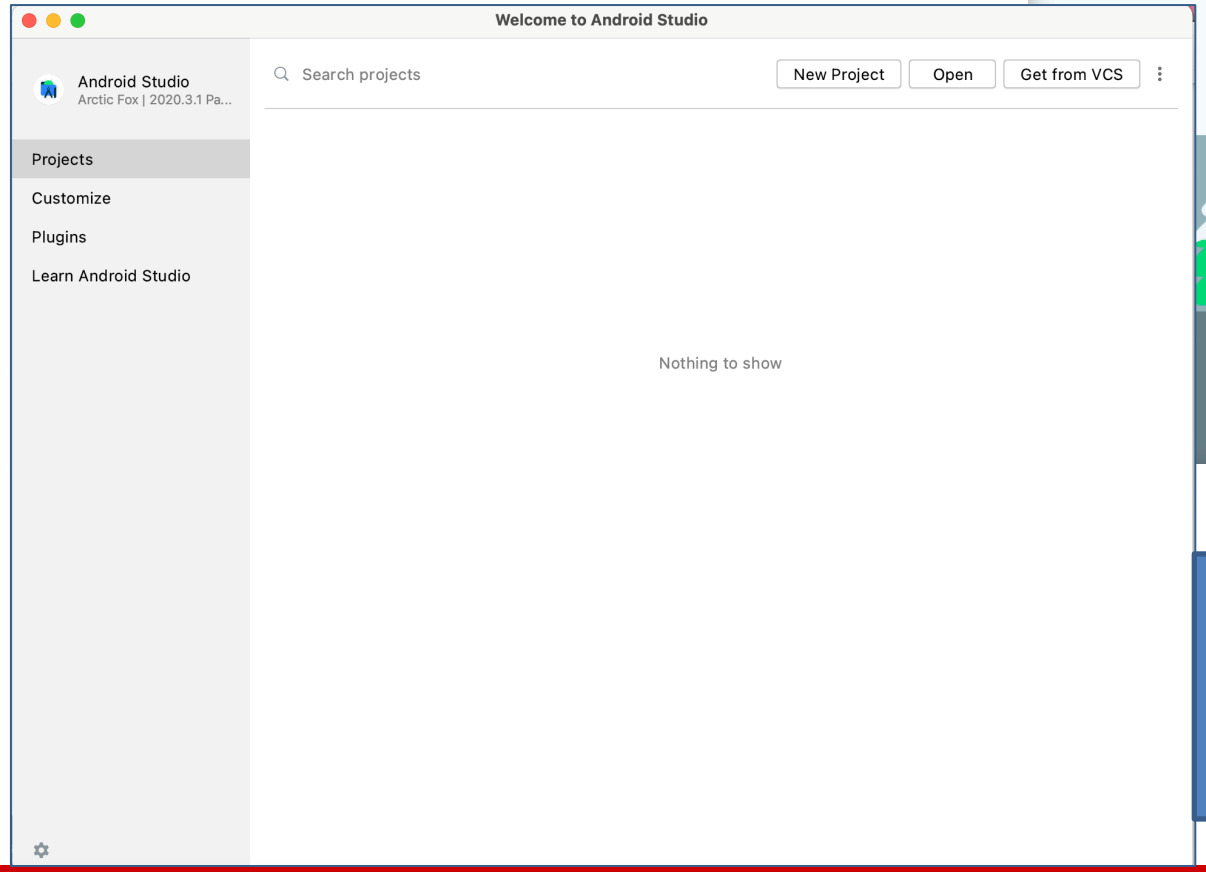
- Android Studio is available for computers running Windows or Linux, and for Mac running macOS
- The OpenJDK (Java Development Kit) is bundled with Android Studio
- The installation is similar for all platforms
  - Navigate to the Android Studio download page and follow the instructions to download and install Android Studio.
  - Accept the default configurations for all steps, and ensure that all components are selected for installation.
  - After the install is complete, the setup wizard downloads and installs additional components, including the Android SDK. Be patient, because this process might take some time, depending on your internet speed.
  - When the installation completes, Android Studio starts, and you are ready to create your first project.


AndroidStudioProject
Application
src
main
res
Application
Android
image_grid.xml
AndroidManifest.xml
Application
manifests
AndroidManifest.xml
AndroidManifest.xml
java
res
drawable
layout
menu
main_menu.xml
values
Gradle Scripts
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/androi
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:layout_editor_absoluteY="81dp"
tools:layout_editor_absoluteX="0dp">
<ImageView
android:id="@+id/imageView2"
android:layout_width="0dp"
android:layout_height="0dp"
android:contentDescription="@string/app_name"
app:layout_constraintBottom_toTopOf="@+id/imageView6
app:layout_constraintEnd_toStartOf="@+id/imageView3"
app:layout_constraintHorizontal_bias="0.5"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent"
app:srcCompat="@drawable/grid_1" />
<ImageView
android:id="@+id/imageView3"
android.support.constraint.ConstraintLayout
ImageView
Design
Text
Preview
Palette
Pixel
27
AppTheme
50%
8dp
8:00
Android Studio
Android Profiler
Google Pixel_XL (emulator-5554)
com.example.android.displayingbitmaps (4166)
End Session
Live
ui.ImageDetailActivity - stopped - destroyed
ui.ImageGridActivity - saved - stopped - destroyed
ui.Image...
ui.ImageDetailActivity - destroyed
CPU
100 %
MEMORY
128 MB
CPU at 2.32m
34 %
NETWORK
1 MB/s
Sending
Receiving
2.25m
2.33m
2.42m
2.50m
2.58m
2.67m
11:5
LF
UTF-8
Context: <no context>

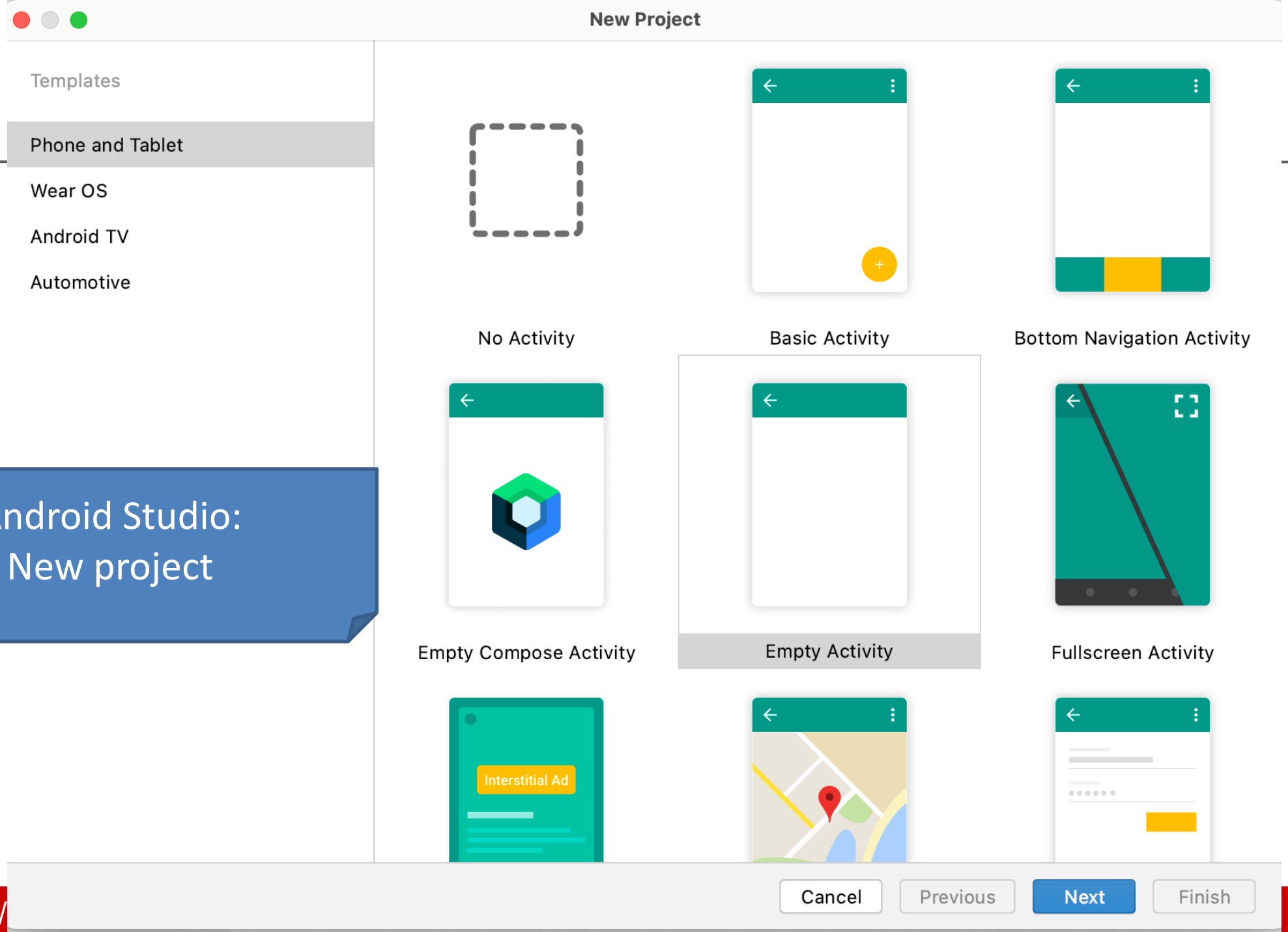
Build your first app

- **Step 1: Create a new project**
- Open Android Studio.
- In the **Welcome to Android Studio** dialog, click **Start a new Android Studio project**.





Android Studio

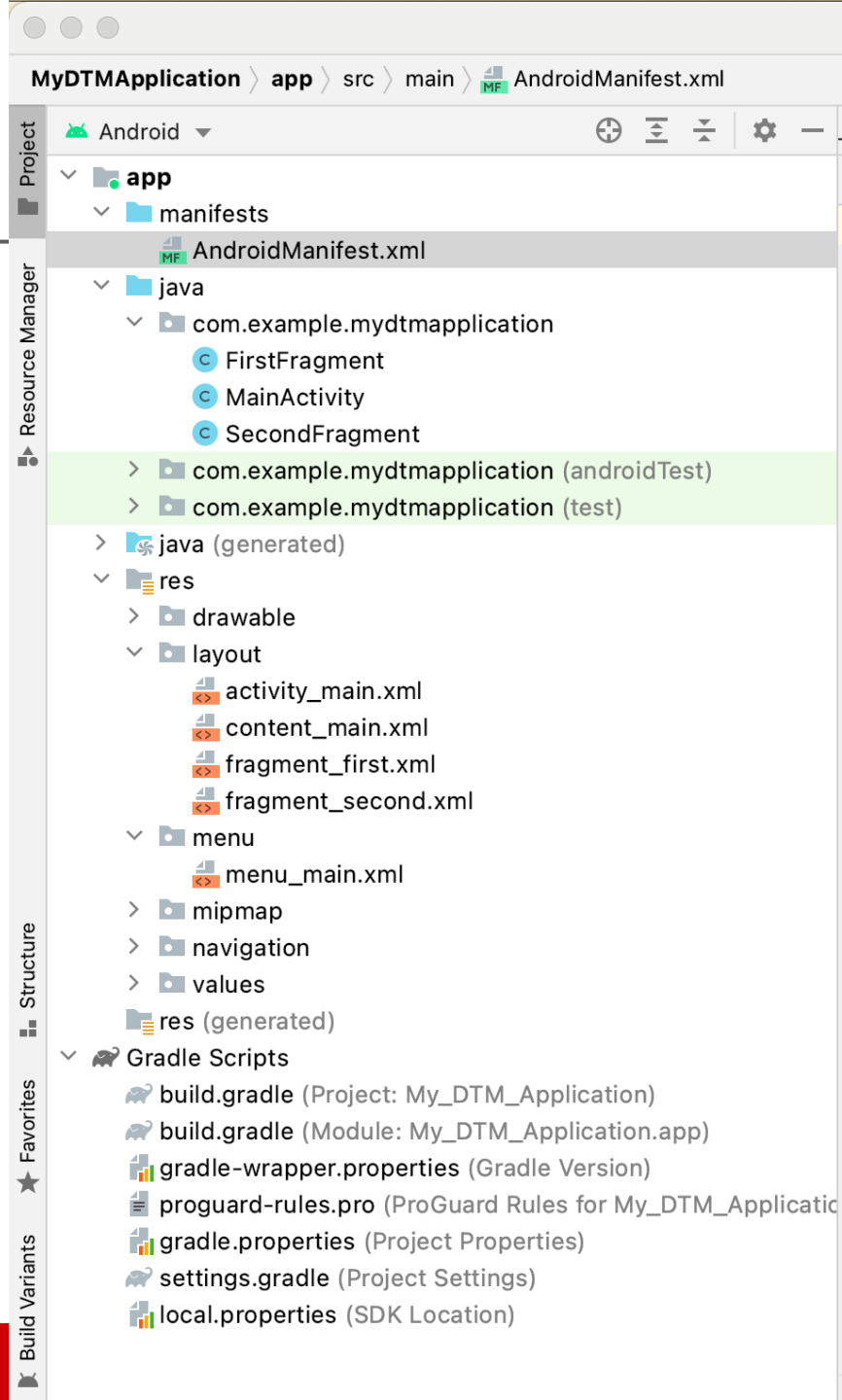


Android Studio: New project

New Project
Basic Activity
Creates a new basic activity with the Navigation component
Name
My DTM Application
Package name
com.example.mydtmapplication
Save location
/Users/catia/AndroidStudioProjects/MyDTMApplication
Language
Java
Minimum SDK
API 21: Android 5.0 (Lollipop)
Your app will run on approximately 98.0% of devices.
Help me choose
Use legacy android.support libraries
Using legacy android.support libraries will prevent you from using the latest Play Services and Jetpack libraries
Cancel
Previous
Next
Finish

Project structure

- Within each Android app module, files are shown in the following groups:
- **manifests**
  - Contains the AndroidManifest.xml file.
- **java**
  - Contains the Java source code files, separated by package names, including JUnit test code
- **Res**
  - Contains all non-code resources, such as XML layouts, UI strings, and bitmap images, divided into corresponding sub-directories



AndroidManifest.xml file

- Every app project must have an AndroidManifest.xml file (with precisely that name) at the root of the project source set. The manifest file describes essential information about your app to the Android build tools, the Android operating system, and Google Play.
- Among many other things, the manifest file is required to declare the following:
  - The **app's package name**, which usually matches your code's namespace. When packaging the app, the build tools replace this value with the application ID from the Gradle build files, which is used as the unique app identifier on the system and on Google Play.
  - The **components of the app**, which include all activities, services, broadcast receivers, and content providers. Each component must define basic properties such as the name of its Kotlin or Java class. It can also declare capabilities such as which device configurations it can handle, and intent filters that describe how the component can be started.
  - The **permissions** that the app needs in order to access protected parts of the system or other apps. It also declares any permissions that other apps must have if they want to access content from this app.
  - The **hardware and software features** the app requires, which affects which devices can install the app from Google Play.
- If you're using Android Studio to build your app, the manifest file is created for you, and most of the essential manifest elements are added as you build your app (especially when using code templates)

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="My DTM Application"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MyDTMApplication">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="My DTM Application"
            android:theme="@style/Theme.MyDTMApplication.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

A basic app

AndroidManifest.xml and Components

- To launch an app component, the Android system needs to know that the component exists
- This information must be included in the manifest
- The app must declare all its components in this file, it is the primary task of the AndroidManifest.xml file!
- The components included in the code but not declared in the manifest are not visible to the system and, consequently, can never be executed!

Intent-filter

- When declaring an activity in the manifest, you can also include intent filters that declare the activity's ability to respond to the intents of other apps
  - You can declare an intent filter for the component by adding an <intent-filter> element as a child of the component declaration element

```
<manifest ... >
    ...
    <application ... >
        <activity android:name="com.example.project.ComposeEmailActivity">
            <intent-filter>
                <action android:name="android.intent.action.SEND" />
                <data android:type="*/*" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Intent-filter

- When an app sends an intent to the system, the system finds an app component that can handle it based on the intent-filter declarations in each app's manifest file
- The system starts an instance of the corresponding component and passes the Intent object to that component
  - If more than one app can handle the intent, the user can select which app to use
- A component can have any number of intent filters (<intent-filter>), each describing a different capability of that component

Example (empty activity)

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

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

app package name

activity

Intent-filter

Manifest permission

- All authorization requests must be declared with an <uses-permission> element in the manifest
  - If the permission is granted, the app is able to use the protected features
  - Otherwise, your attempts to access those features fail

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

Manifest permission

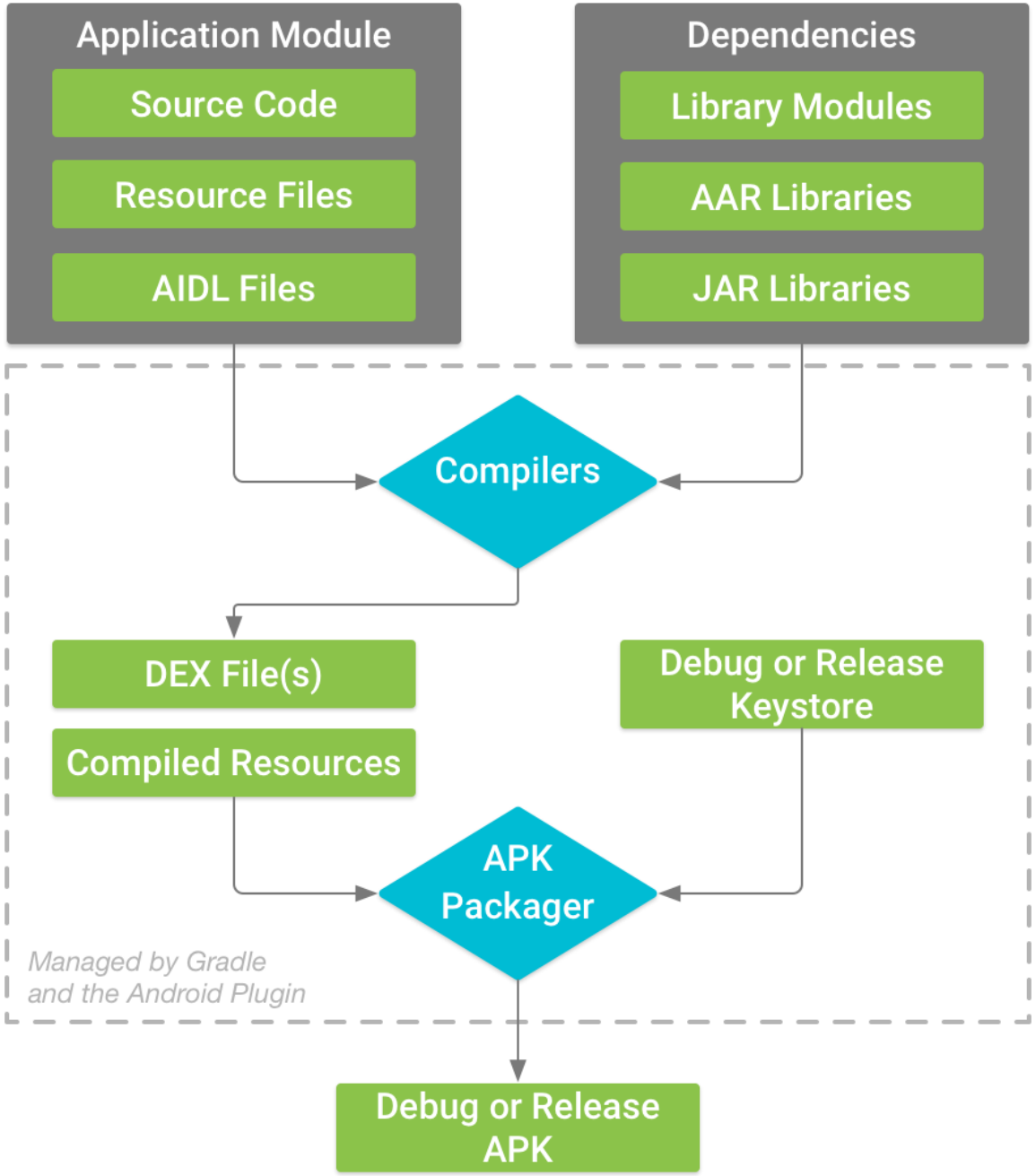
- To access hardware features (such as camera and Bluetooth) you need permissions
  - Since not all devices have the same hardware it is important to also include the <uses-feature> tag in the manifest to declare if this function is actually required
  - In the example, by declaring android: required = "false" for the function, Google Play allows the app to be installed on devices that do not have the functionality (in the example room) Note, it will then be necessary to check the presence of the functionality at runtime using PackageManager.hasSystemFeature () BUT at least the app can be installed and listed in the list of available apps

```
<uses-feature android:name="android.hardware.camera" android:required="false" />
```

build.gradle file

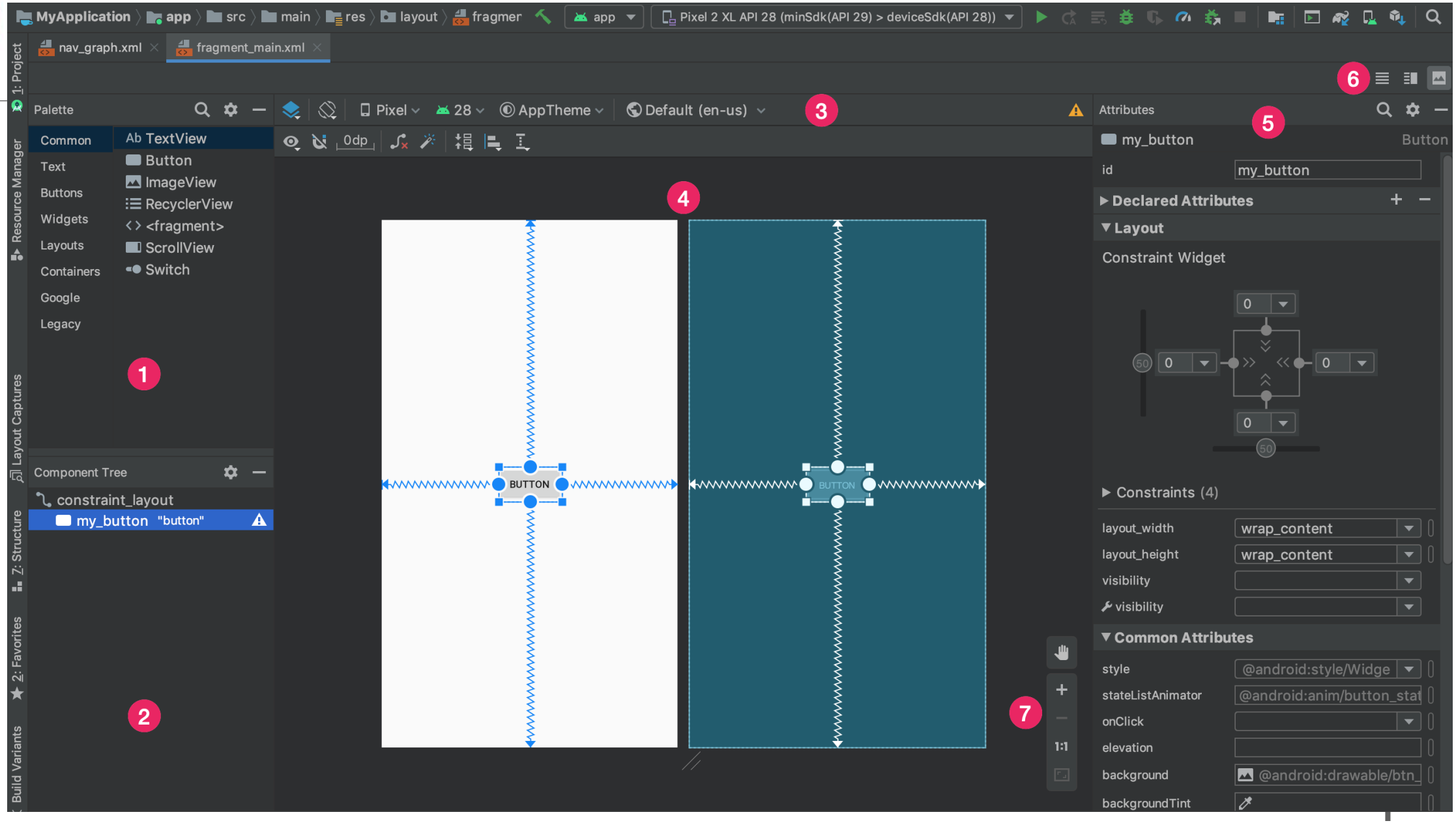
```
android {
  defaultConfig {
    applicationId 'com.example.myapp'

    // Defines the minimum API level required to run the app.
    minSdkVersion 15

    // Specifies the API level used to test the app.
    targetSdkVersion 28

    ...
  }
}
```

- The Android build system compiles app resources and source code, and packages them into APKs that you can test, deploy, sign, and distribute
- Android Studio uses Gradle, an advanced build toolkit, to automate and manage the build process, while allowing you to define flexible custom build configurations
- Each build configuration can define its own set of code and resources, while reusing the parts common to all versions of your app. The Android plugin for Gradle works with the build toolkit to provide processes and configurable settings that are specific to building and testing Android applications.


Application Module
Source Code
Resource Files
AIDL Files
Dependencies
Library Modules
AAR Libraries
JAR Libraries
Compilers
DEX File(s)
Compiled Resources
Debug or Release Keystore
APK Packager
Managed by Gradle and the Android Plugin
Debug or Release APK

Build a UI with Layout Editor

- The Layout Editor enables you to quickly build layouts by dragging UI elements into a visual design editor instead of writing layout XML by hand. The design editor can preview your layout on different Android devices and versions, and you can dynamically resize the layout to be sure it works well on different screen sizes
- https://developer.android.com/studio/write/layout-editor

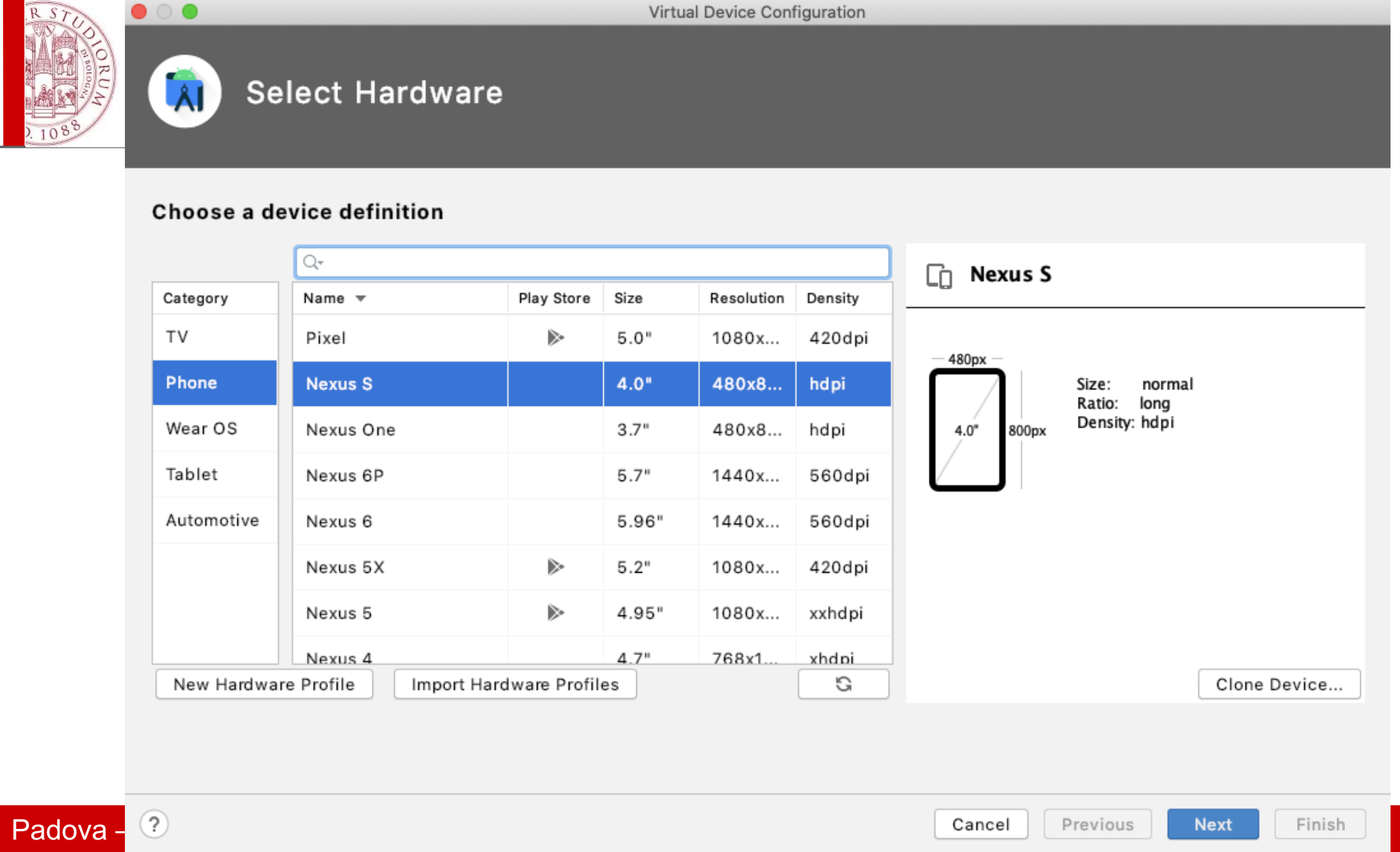

MyApplication
app
src
main
res
layout
fragmer
Pixel 2 XL API 28 (minSdk(API 29) > deviceSdk(API 28))
nav_graph.xml
fragment_main.xml
Palette
Common
Text
Buttons
Widgets
Layouts
Containers
Google
Legacy
TextView
Button
ImageView
RecyclerView
<fragment>
ScrollView
Switch
Component Tree
constraint_layout
my_button "button"
Pixel
28
AppTheme
Default (en-us)
0dp
BUTTON
BUTTON
Attributes
my_button
Button
id
my_button
Declared Attributes
Layout
Constraint Widget
Constraints (4)
layout_width
wrap_content
layout_height
wrap_content
visibility
visibility
Common Attributes
style
@android:style/Widge
stateListAnimator
@android:anim/button_stat
onClick
elevation
background
@android:drawable/btn_
backgroundTint
1:1
1
2
3
4
5
6
7

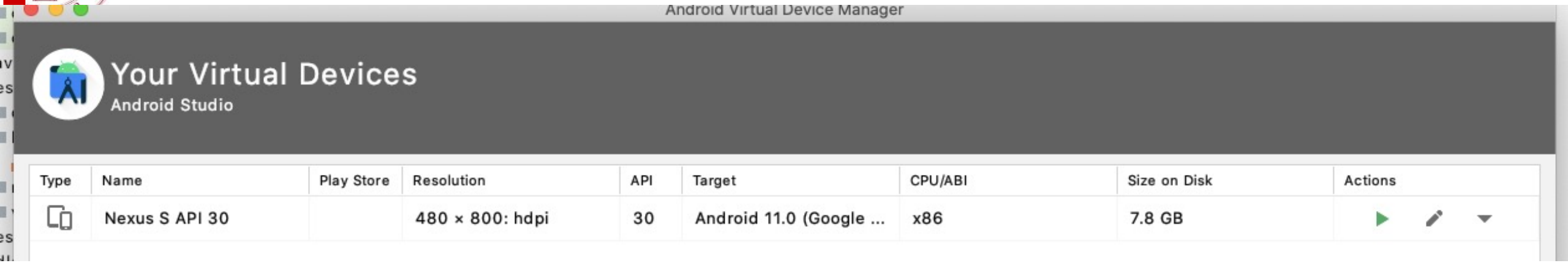
Run the app

- How to run the app?

Run the app

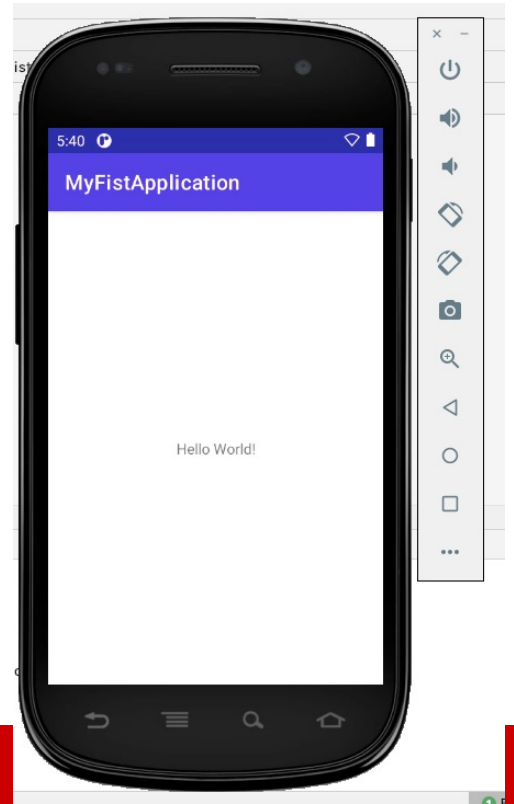
- On a real device
  - Requires an Android device to be connected via USB to the PC The USB driver needs to be installed More info: https://developer.android.com/training/basics/firstapp/running-app#RealDevice
- On the Android studio emulator
  - We will always use this solution in the labs! An "Android Virtual Device (AVD)" must be created Note: we want to have an AVD with the Android 11 system image Recommended: Nexus S


Virtual Device Configuration
Select Hardware
Choose a device definition
Category
TV
Phone
Wear OS
Tablet
Automotive
Name
Play Store
Size
Resolution
Density
Pixel
5.0"
1080x...
420dpi
Nexus S
4.0"
480x8...
hdpi
Nexus One
3.7"
480x8...
hdpi
Nexus 6P
5.7"
1440x...
560dpi
Nexus 6
5.96"
1440x...
560dpi
Nexus 5X
5.2"
1080x...
420dpi
Nexus 5
4.95"
1080x...
xxhdpi
Nexus 4
4.7"
768x1...
xhdpi
New Hardware Profile
Import Hardware Profiles
Nexus S
480px
4.0"
800px
Size: normal
Ratio: long
Density: hdpi
Clone Device...
Cancel
Previous
Next
Finish



Click “Run”

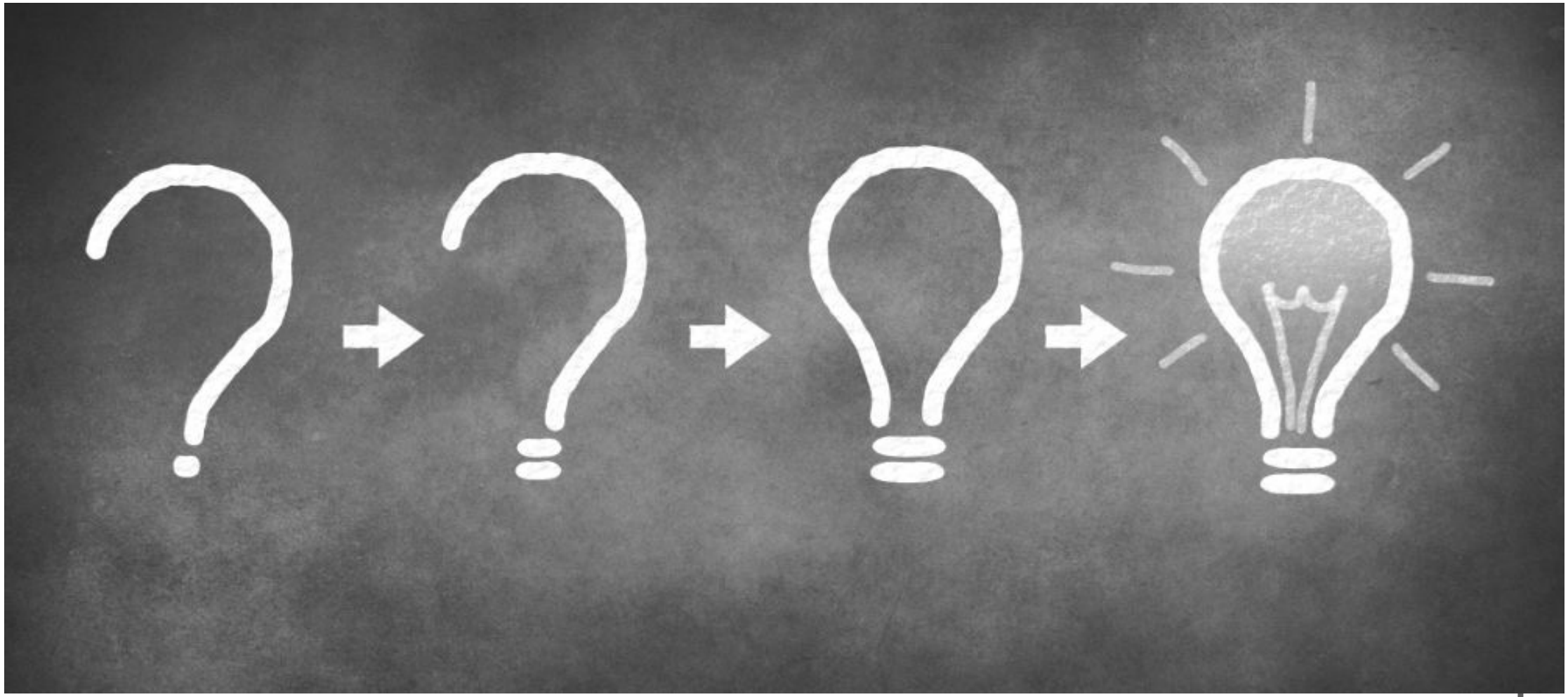
Android Studio install the app on the AVD and start the emulator! After "a few" seconds you will see "Hello, World!" displayed in your app!



Would you like to try?

- Build your first app
  - https://developer.android.com/training/basics/firstapp
- Build your second app
  - https://developer.android.com/codelabs/build-your-first-android-app#0
- Build your «n» app
  - https://codelabs.developers.google.com/codelabs/recognize-flowers-with-tensorflow-on-android#0

Questions?



Some interesting links

- https://developer.android.com/
- https://earthlymission.com/android-version-history-a-visual-timeline/
- https://developer.android.com/guide/platform
- https://developer.android.com/guide/components/fundamentals
- https://developer.android.com/studio/workflow
- https://developer.android.com/studio/install
- https://developer.android.com/studio/intro
- https://developer.android.com/studio/write/layout-editor
- https://developer.android.com/guide/topics/manifest/
- https://developer.android.com/guide/topics/permissions/overview
- https://developer.android.com/studio/run/managing-avds
- https://developer.android.com/training/basics/firstapp/running-app
- https://developer.android.com/guide
- …