Android & Kotlin Introduction

Android Lecture 1

Simona Kurňavová

About this course

Course page: https://d3s.mff.cuni.cz/teaching/nprg056/

Garant: Jan Kofroň (Contact: jan.kofron@d3s.mff.cuni.cz)

Lecturer: Simona Kurňavová (Contact: kurnavova.simona+mff@gmail.com)

Schedule for semestral projects:

- *October 1 – December 1*: Forming project groups (1-3 students) and creating project specifications
- *December 1*: The project specification has to be accepted by a lecturer
- *February 28:* Final version of the project
- *April 15*: Issues identified by lecturers fixed

Course objectives

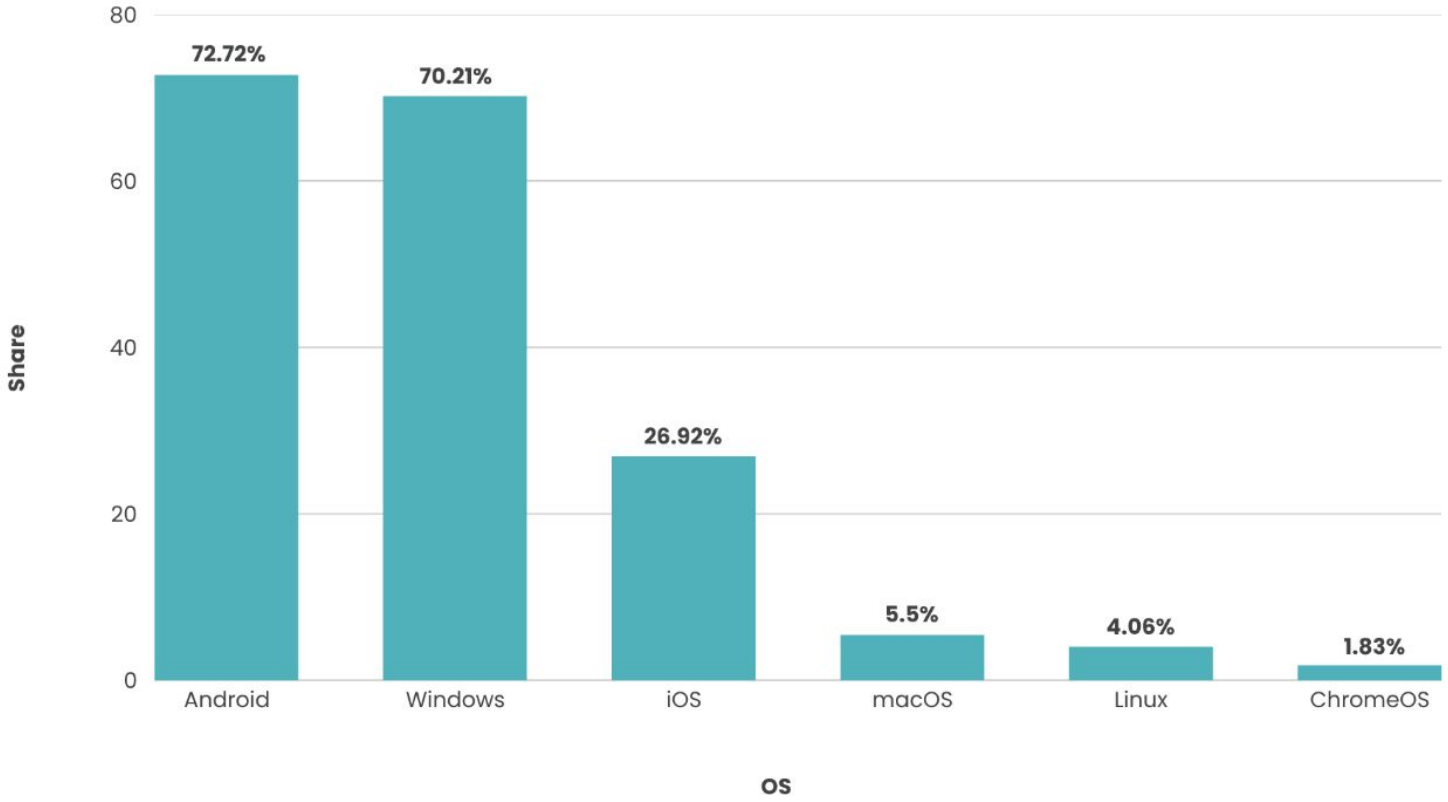
- Introduction to Mobile development, Kotlin programming language
- Basics of Android development: project, Activities
- Jetpack Compose, navigation
- ViewModels, MVI and MVVM patterns
- Clean architecture
- Kotlin Coroutines
- Networking, databases
- Build process, Release process
- Kotlin Multiplatform, introduction to Swift & SwiftUI
- *Bonus: AI inference on Android*

Today’s Agenda

- Motivation & History
- Mobile & Android development
- Kotlin
- Q&A

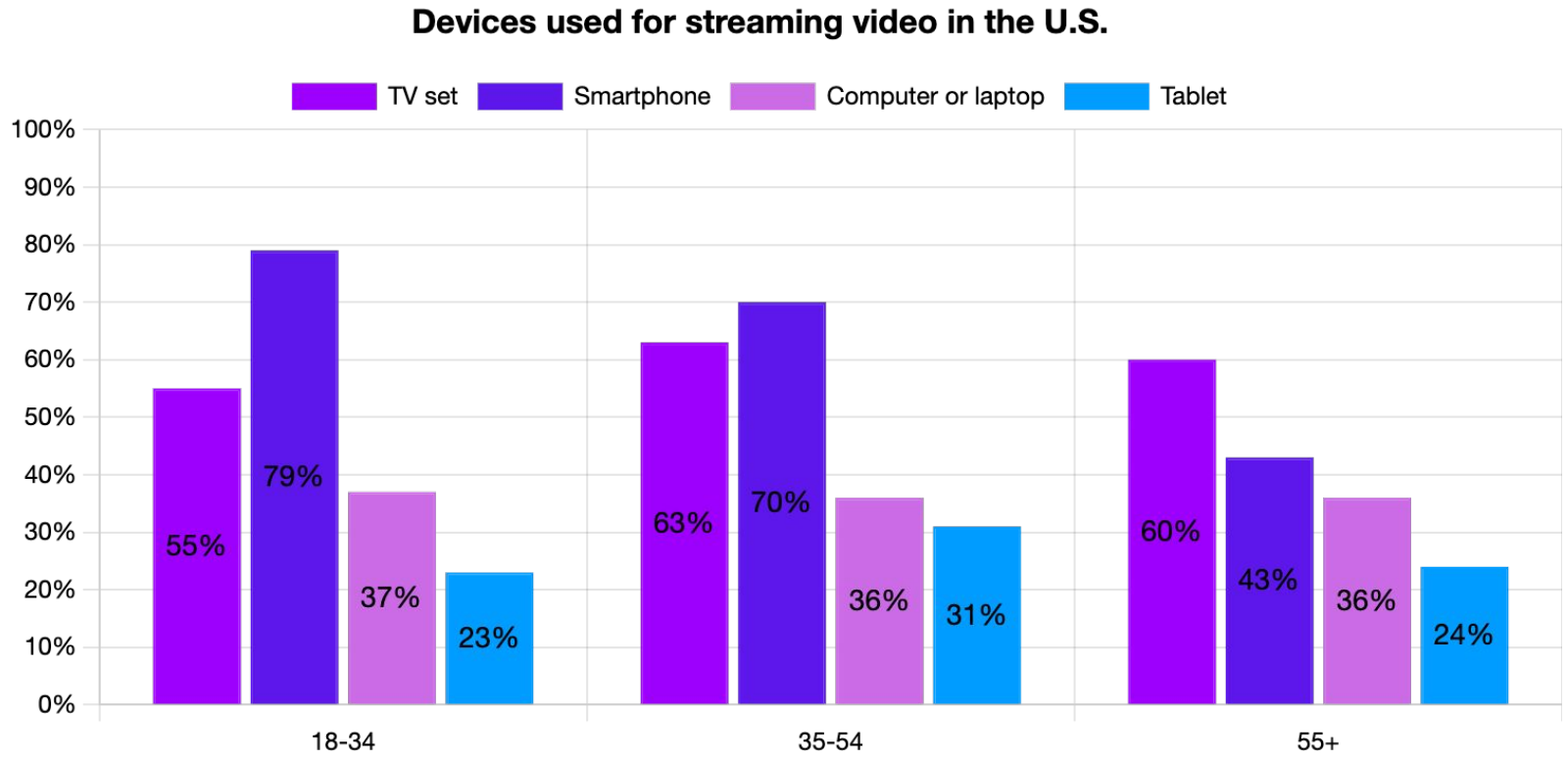
Motivation and history

Global operating system market share: 2025



Source

Devices used for streaming video in U.S in 2025



Source

Motivation for Mobile development

- Billions of active devices worldwide
- Fast access to services and information
- Features like camera, GPS, sensors
- High engagement - notifications, widgets
- Offline mode
- Security & trust
- Modern tools: Jetpack Compose, SwiftUI, …
- Easy distribution and accessibility - Google Play, App Store

Mobile device specifics

- Not enough computation power
- Changing state:
  - Screen orientation
  - Foldables
- Unstable network connection
- Small battery
- Small and variable display size
- Never ending interruptions
  - Engaging notifications
  - Alarms

History

1993

First handheld commercially available mobile phone

Motorola DynaTAC 8000x

2007

First iPhone

2008

First commercial Android phone

HTC Dream / T-Mobile G1

Along the way

Nokia 5310 expressmusic

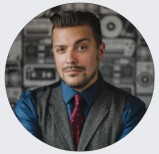
LG Chocolate BL20

BlackBerry Bold 9900

Motorola Razr

Sony Ericsson Walkman

Motorola Aura



Interesting series of videos about old fun phones: [When phones were fun](#) (MrMobile)

Modern phones

Apple iOS

iOS

- Developed by Apple
- Used on: iPhone, iPad (now iPadOS based on iOS), iPod touch, Apple TV, Apple Watch
- Unix-based OS
- Closed source (but some components like are open source)
- Most widely used mobile OS after Android (especially in the premium market)

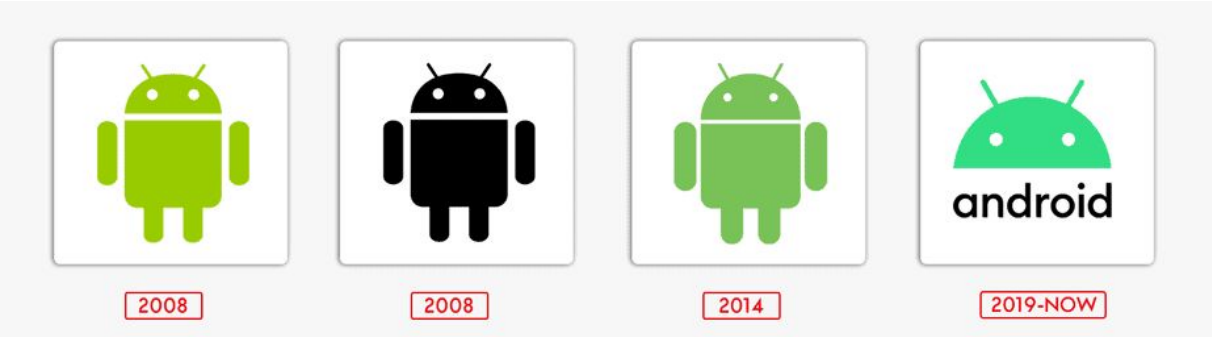
iOS

- Distribution: App Store
- Secure and closed ecosystem
- Tight integration with other Apple devices and services through features like Handoff or AirDrop.
- Apple controls hardware and software with their products

Android & tour of Android versions

android

- Developed by Google
- Linux-based OS
  - No shell, no access to root (by default)
  - Open source https://source.android.com (Just the OS, Google play store and services are proprietary)
- Most widely used mobile OS (source)
- Formerly meant as OS for digital cameras



android

- Android – Phones and tablets
- WearOS - Smartwatch
  - Extended notification center
  - Sporttester
- Chrome OS
- Android Auto
  - Mirror optimized UI to built-in infotaiment
- Android automotive
  - Standalone OS in cars
- Android Things (deprecated January 2022)
  - IoT
- Google glass
  - Glass explorer program (retired 2015)
  - Glass for enterprise (retired 2023)

Android Ecosystem

- Distribution:
  - Primary Google play store
  - Alternative stores: Samsung Galaxy store, Amazon store, …
  - Direct downloads
- Different monetization models
  - Free
  - Paid
  - In-App products (subscriptions and one-time)
- Android users not willing to pay for apps -> Ads
- Available in most countries

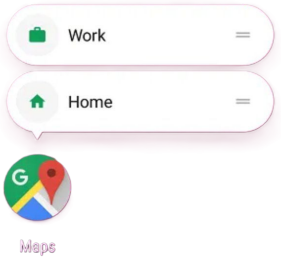
Android Nougat (API 24, 25)

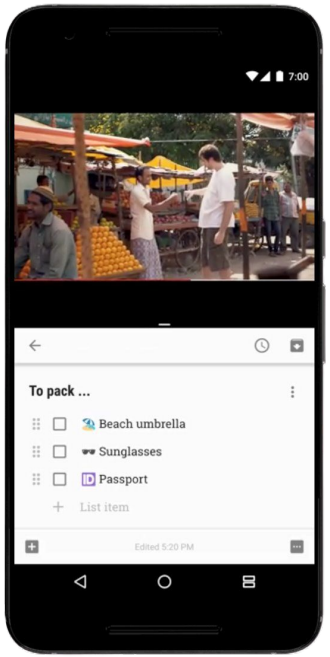
August 2016 (Android 7.0, API 24)

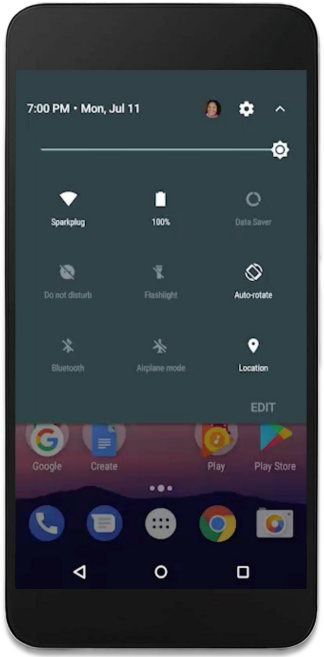
- Multi window
- Quick setting tiles
- Vulkan API

October 2016 (Android 7.1, API 25)

- App shortcuts







Source

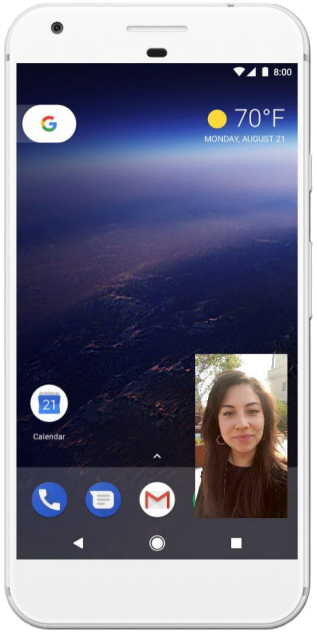
Android Oreo (API 26, 27)

August 2017 (Android 8.0, API 26)

- Picture in Picture
- Notification channels
- Custom fonts
- Autosize text view
- Multi display

October 2017 (Android 8.1, API 27)

- Neural Network API
- Cryptography update
  - Conscrypt over Bouncy castle



Source

Android Pie (API 28)

August 2018 (Android 9.0)

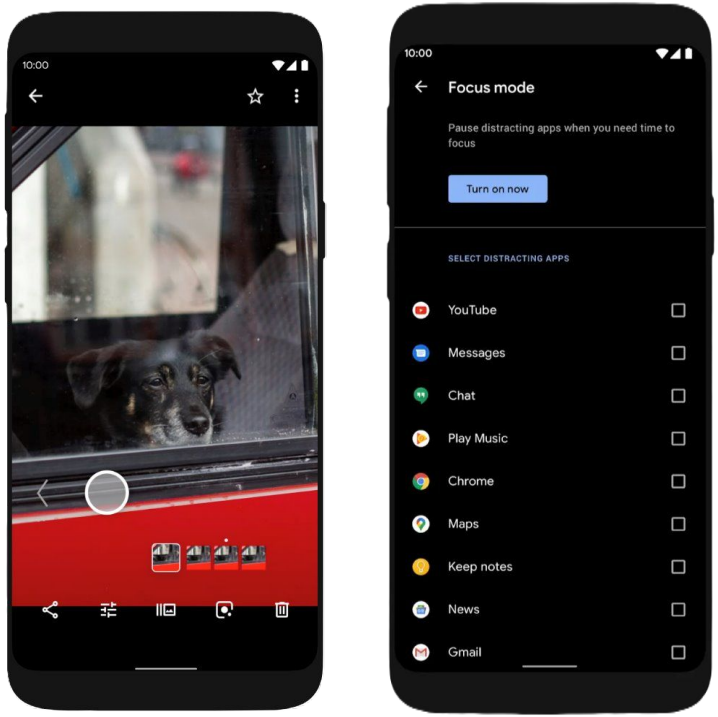
- Display cutout supports
- Notification – displays images
- Multicamera support
- Gesture navigation

Source

Android 10 (API 29)

September 2019

- Gesture navigation (new version)
- Smart replies
- Dark theme
- Foldables

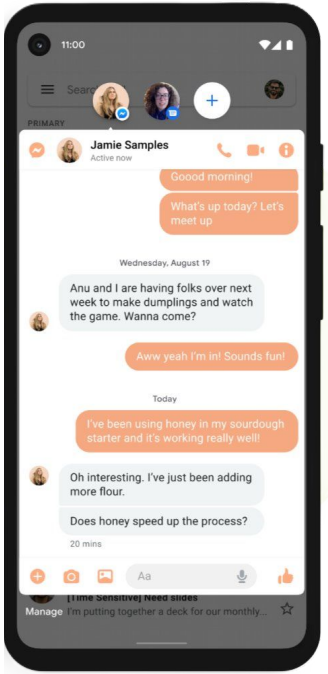


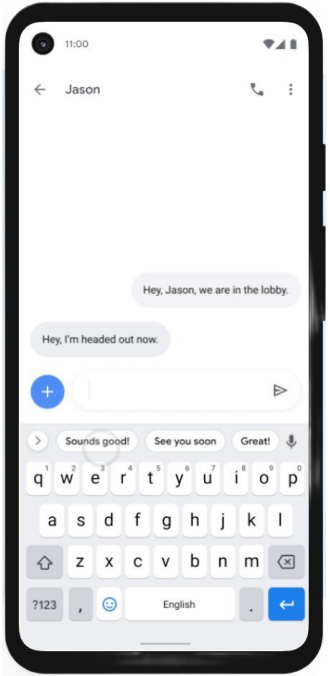
Source

Android 11 (API 30)

September 2020

- Chat bubbles
- Notification history
- One time permission
- Permission auto-reset
- 5G detection API





Source

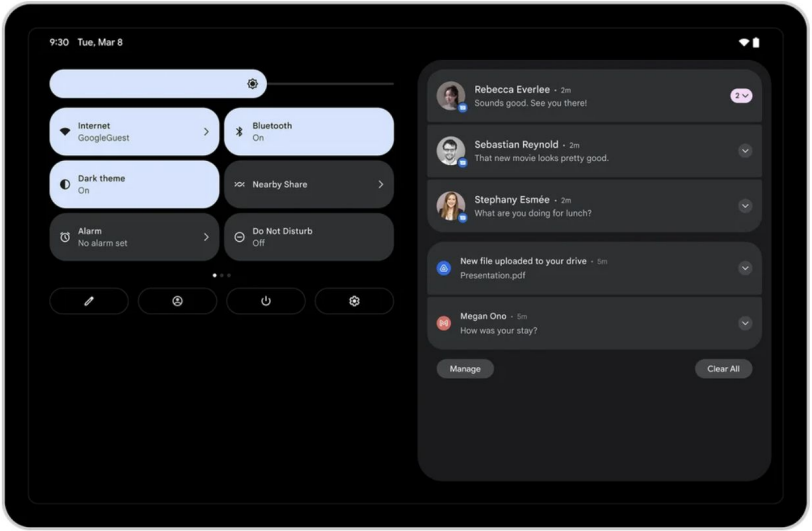
Android 12 (API 31, 32)

October 2021 (Android 12.0, API 31)

- Material You – design refresh
- Splash screen API
- Extending screenshot beyond screen
- Bluetooth permissions
  - Scan for nearby device don’t need location

March 2022 (Android 12L, API 32)

- Optimized for large screens and foldables

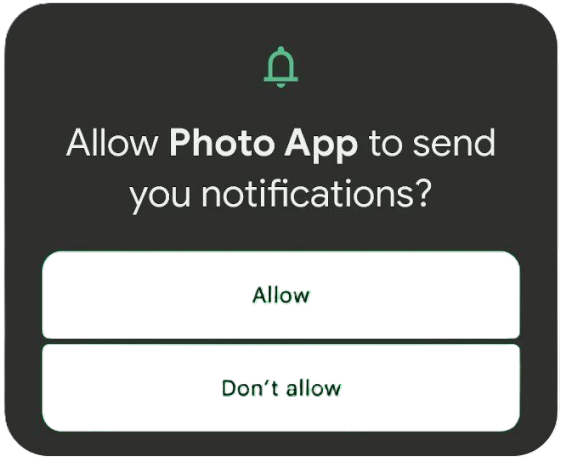


Source

Android 13 (API 33)

August 2022

- Notification permission
- Nearby wifi devices permission

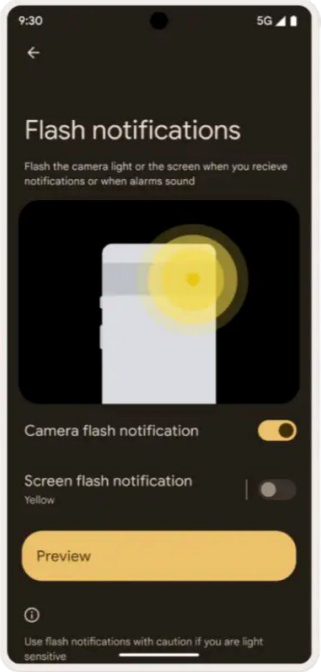


Source

Android 14 (API 34)

October 2023

- Exact alarm permission denied by default
- Partial access to photos and videos
- Change to non-dismissable notifications
- Security changes
- Accessibility improvements
  - Flash notifications
  - Better readability with font scaling



Source

Android 15 (API 35)

October 2024

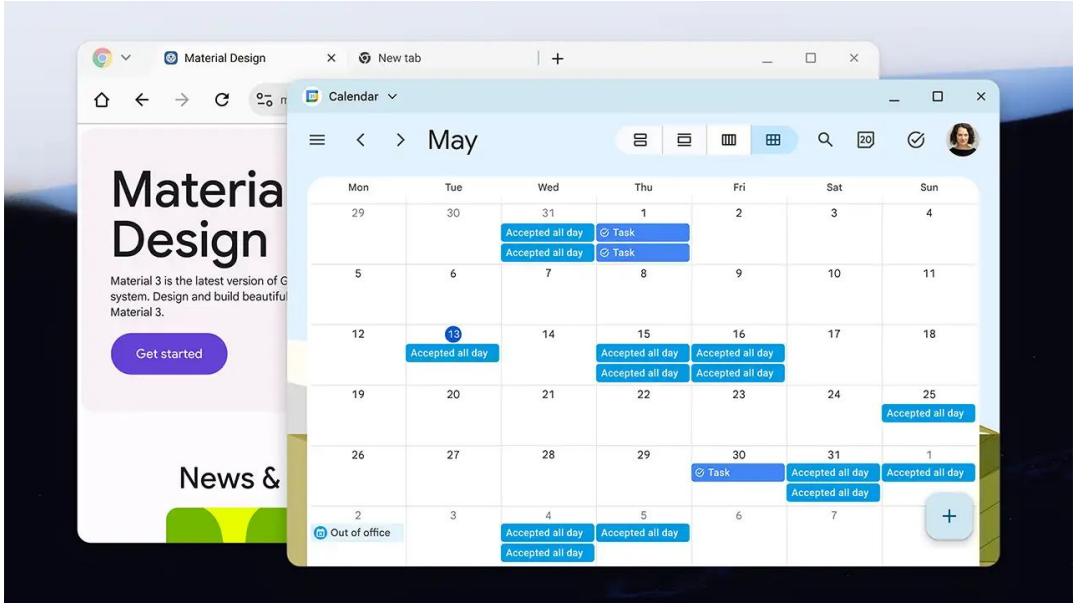
- Private space
- Screen recording detection - app is informed
- Improvements to Picture-in-picture
- Adaptive refresh rate (see)

Source

Android 16 (API 36)

June 2025

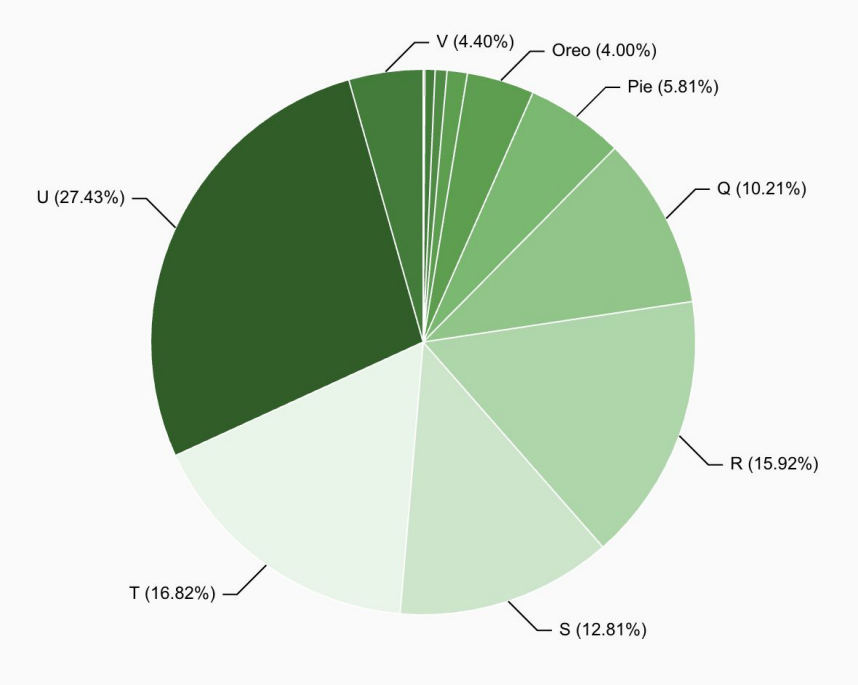
- AI features - deeper, system level integration of Gemini
- Better support for bigger screens (tablets)



Source

Android version distribution

April 1, 2025:



API Distribution

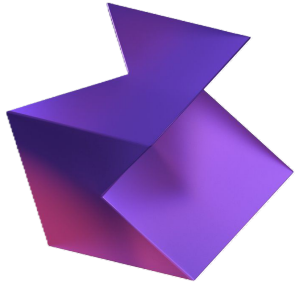
Platform Version	API	Distribution
Android 4.4 (KitKat)	19	0.1%
Android 5 (Lollipop)	21	0.1%
Android 5.1 (Lollipop)	22	0.5%
Android 6 (Marshmallow)	23	0.7%
Android 7 (Nougat)	24	0.6%
Android 7.1 (Nougat)	25	0.6%
Android 8 (Oreo)	26	1%
Android 8.1 (Oreo)	27	3%
Android 9 (Pie)	28	5.8%
Android 10 (Q)	29	10.2%
Android 11 (R)	30	15.9%
Android 12 (S)	31	12.8%
Android 13 (T)	33	16.8%
Android 14 (U)	34	27.4%
Android 15 (V)	35	4.4%

source1, source2

Mobile development

Options (iOS, Android)

- App-like mobile web
- Multi Platform frameworks:
  - Flutter (Google),
  - Xamarin, replaced by .NET (Microsoft),
  - React native (Facebook), …
- WebView based frameworks (like Ionic)
- Kotlin Multiplatform
- Native



iOS native development

- Swift: The primary language for iOS development, announced by Apple in 2014.
- Objective-C: Supported, but gradually being replaced by Swift.
- C/C++: Mainly games and libraries.

Android native development

- *Kotlin*: The primary language for Android development, support announced on GoogleIO in 2017.
- *Java:* Supported, not recommended anymore (i.e. Compose not working with Java)
- *C/C++:* Mainly games and libraries.

Android development

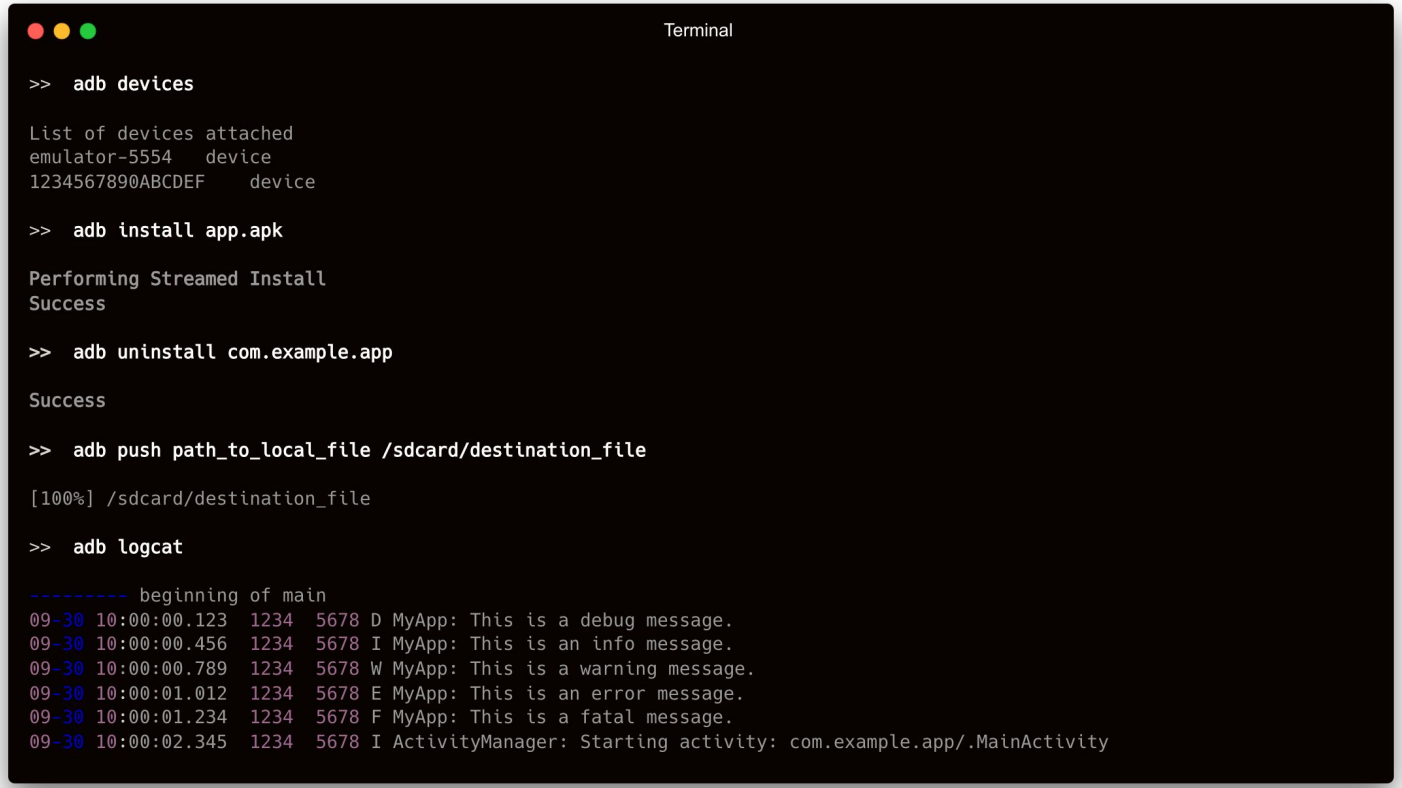
Development tools

- Android Studio - IDE
- SDK
  - Emulator
  - ADB
  - Lint - static code analysis
  - D8 (Dex Compiler) - converts Java bytecode into Dalvik bytecode (DEX)
  - aapt - Android Asset Packaging Tool
  - R8 - code shrinker and obfuscator
- NDK (C/C++ development)

Android Studio

- Primary IDE for Android & KMP development
- JetBrains
- Tools:
  - Emulator
  - Logcat
  - App inspection: Network, Scheduled work, Database
  - Terminal
  - Git management

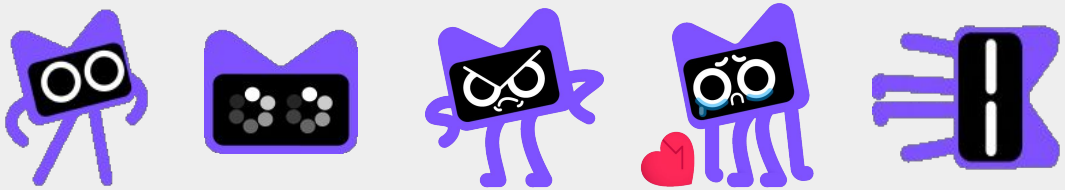
ADB (Android Debug Bridge)



- Robust build system
- Dependency management
- Compilation, packaging, testing
- Build scripts languages:
  - Kotlin DSL
  - Groovy

app/build.gradle.kts

```kotlin
plugins {
    id("com.android.application") // Android application plugin
}

android {
    compileSdk = 34 // Compile SDK version

    defaultConfig {
        applicationId = "com.example.myapp" // App package name
        minSdk = 21 // Minimum SDK version
        targetSdk = 34 // Target SDK version
        versionCode = 1 // App version code
        versionName = "1.0" // App version name
        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner" // Test runner
    }

    buildTypes {
        getByName("release") {
            isMinifyEnabled = false // Disable code shrinking
            proguardFiles(
                getDefaultProguardFile("proguard-android-optimize.txt"),
                "proguard-rules.pro" // ProGuard rules
            )
        }
    }
}

dependencies {
    implementation("androidx.core:core-ktx:1.10.1") // AndroidX core with Kotlin extensions
    implementation("androidx.appcompat:appcompat:1.6.1") // AppCompat for backward compatibility
    implementation("com.google.android.material:material:1.9.0") // Material Design components
    implementation("androidx.constraintlayout:constraintlayout:2.1.4") // ConstraintLayout for UI

    testImplementation("junit:junit:4.13.2") // JUnit for unit testing
    androidTestImplementation("androidx.test.ext:junit:1.1.5") // Android JUnit extension
    androidTestImplementation("androidx.test.espresso:espresso-core:3.5.1") // Espresso for UI testing
}
```



Kotlin

Kotlin assets

Kotlin

- **Documentation**
- Open source (Apache 2.0 license)
- First released in 2016
- Developed by JetBrains
- Tools support: IntelliJ, Android Studio, standalone compiler, ...
- Interoperable with Java (legacy code)
- Coroutines for asynchronous programming - lightweight concurrency framework
- Garbage collection
- Concise, null-safety, typesafe, type inference, immutability (List vs. MutableList)
- Kotlin playground

Kotlin current version (as of 29.9.25)

Source

Kotlin Basics

Kotlin Basics

```kotlin
var age = 25 // Mutable variable
age = 26 // Can be reassigned

val name = "Alice" // Immutable variable
// name = "Bob" // Error: Val cannot be reassigned

class Person(val name: String, var age: Int) {
    fun introduce() {
        println("My name is $name and I am $age years old.")
    }
}

val person = Person("Bob", 30)
person.introduce() // Output: My name is Bob and I am 30 years old.

object Singleton {
    fun showMessage(): String = "This is a singleton object"
}

println(Singleton.showMessage()) // Output: This is a singleton object.
```

Kotlin Basics

Values and variables:

- Statically typed
- If no type is specified, Kotlin infers the type from the initial assignment.

Classes:

- Primary constructors are directly declared in the class header (optional init block)
- Objects (Singleton class)

Functions:

- Single-expression functions: functions that omit the return keyword and braces. Body is specified after "=" symbol.
- Function return type must be specified (unless returns Unit or it is a single-expression function)

Data classes

Used mainly for data modeling

Methods generated by compiler:

- Equals() / hashCode()
- toString()
- Copy()

Data objects (since Kotlin 1.9)

Data Classes

```kotlin
data class Person(
    val name: String,
    val surname: String,
    val street: String,
    val buildingNumber: Int
)

val bruceWayne = Person(
    name = "Bruce",
    surname = "Wayne",
    street = "Wayne Manor",
    buildingNumber = 1
)

val robin = bruceWayne.copy(name = "Robin")
```

Named, default parameters, String interpolation

Default parameters: replaces method overloading

String interpolation:

- For variables: $variableName
- For expressions: ${expression}

Parameters

```
fun greetUser(name: String, age: Int = 30) {
    println("Name: $name, Age: $age") // String interpolation
}

fun main() {
    greetUser(name = "Bob")  // Output: Name: Bob, Age: 30
    greetUser(name = "Charlie", age = 25)  // Output: Name: Charlie, Age: 25
}
```

When expression

When Expression

```
val x = 15
val y = 10

when { // As statement
    x > y -> println("x is greater than y")
    x == y -> println("x is equal to y")
    else -> println("x is less than y")
}

val day = 3
val dayName = when (day) { // As expression
    1 -> "Monday"
    2 -> "Tuesday"
    3 -> "Wednesday"
    4 -> "Thursday"
    5 -> "Friday"
    else -> "Weekend"
}
```

Operator overloading

- Arithmetic operators (+, -, *, /, ...)
- "in" operator (contains)
- Indexed access operator
- Equality/Inequality operator (equals)
- Comparison operators (compareTo)

Operator overloading

```
data class ComplexNumber(val real: Double, val imaginary: Double) {

    fun plus(increment: ComplexNumber): ComplexNumber {
        return ComplexNumber(real + increment.real, imaginary + increment.imaginary)
    }
}
```

Extension functions

- Adds methods or variables to any class
- Doesn't have access to internal state of the class
- Powerful, but easy to misuse

Extensions functions

```
fun Int.isEven() = this % 2 == 0

fun Int.isOdd() = this % 2 == 1

val Int.absoluteValue: Int
    get() = if (this < 0) -this else this
```

Smartcast

- Automatically casts a variable to a specific type after checking its type with conditions

Smart Cast

```
fun smartCastDemo(x: Any) {
    when (x) {
        is Int -> println("Integer value abs(x): ${x.absoluteValue}")
        is String -> println("String x.toLowerCase: ${x.toLowerCase(Locale.getDefault())}")
        else -> println("x is not Integer or String")
    }
}
```

Lambda expressions

Function can be passed as argument or returned from other function

Lambda Expressions

```
fun <T, R> transform(item: T, transformation: (T) -> R): R {
    return transformation(item)
}

fun <T, R> T.transform(transformation: (T) -> R): R {
    return transformation(this)
}

val number = 1
println(number.transform { it.toString() })
println(transform(number) { it.toString() })
```

Companion object

- Object associated with class
- Accessed via class name
- Methods/properties do not belong to instance
- Similar to static in java
- Can implement/extend another interface/class
- Top level declaration vs. Companion object: article

Null safety

- Null safety enforced at compile time: It’s part of type system

Null Safety

```
// Safe access operator
val length: Int? = nullableString?.length

// Elvis operator
val result: String = nullableString ?: "Default Value"

// Using `let` with safe access operator
nullableString?.let {
    println("Hey, $it")
}

// This will throw a NullPointerException if nullableString is nul
nullableString!!.let { println("Hey, $it") }
```

Kotlin Multiplatform (KMP)

- **Documentation:** https://kotlinlang.org/docs/multiplatform.html
- JetBrains
- Announced in 2017

- Allows to write shared business logic in Kotlin and use it across multiple platforms

- Built on Kotlin/Native
- Primary used for iOS and Android mobile development
- Kotlin Multiplatform Gradle plugin
- Multiplatform libraries

- More information in 2023 lesson about KMP: slides
- Example of KMP project from mdevcamp workshop (contains both version: 2 separate apps, and KMP variant): repository

Kotlin

Thank you for attention

Feel free to leave feedback about this lesson: