©2025 Gonzalo Munilla Garrido. All rights reserved.

**Cross-Platform Distribution** (by design) of end-user apps

**Native Distribution** of end-user apps

## (Sandboxed) Web Browser

Developer

SC (TS) → Transpiler (to JS) → SC/Bytecode (HTML, CSS, JS, WA)

SC (C++/Rust) → Compiler (to WA Bytecode) [AOT] → SC/Bytecode (HTML, CSS, JS, WA)

Internet and external media

Web Server (Website, PWA)

Computing Device

Media Content

Static Content

Dynamic Content

Networking

CSS

HTML

JS & WA

CSS Parser

HTML Parser

CSSOM Tree

DOM Tree

Layout Engine

Rendering Engine

JS Engine [JIT] (Interpreter & JIT)

V8 (Ignition, TurboFan) (Chrome/Edge), JSCore/Nitro (Safari) SpiderMonkey (Firefox)

WA Runtime (Interpreter & JIT) V8 (Liftoff), JSCore/Nitro (WasmLLInt/FTL) SpiderMonkey (Baseline/Ion)

Binary

PWA APIs: Persistent storage, push notifications, full screen, work offline, quick app switch, inclusion on the installed apps menu, available update icon

Data Persistence: Cookies, local storage

Browser Engine: Blink (Chrome, Edge), Gecko (Firefox), WebKit

OS

Hardware

## iOS / macOS

Equivalent architecture but differences in frameworks, app distribution, etc.

SC (Swift) IDE: Xcode → (LLVM) Compiler [AOT] → Binary

IPA package

macOS sideloading

Signed

iOS/macOS App Store

App Store Connect: App Review — Human Review, Automated Review

Signed (by Apple)

Signed

Automated Review

Unsigned

Application Server

User Consent

Unprivileged/Entitled App | Apple Privileged (Preinstalled) App

**Cocoa (Touch - for iOS) – Abstracted (public/entitled/privileged) APIs** to UI-related classes/methos, push notifications, handles user input, App Store in-app purchases, web view, etc.

**Media** – Abstracted APIs for multimedia functions/classes/frameworks for apps to use ( e.g. graphics, animation, video, audio, , etc.)

**Core Services** – Abstracted APIs (Provides 'Core' services like location, networking, data (for developers to store ap data on device), threading, WebKit etc.)

**Swift/Objective-C (iOS) Runtime – (macOS offers richer RT support)**

Binary

**Core OS –** (Hybrid) **XNU Kernel** (manages system-level services like managing memory, the file system, networking, etc.)

**Hardware** (System on a chip (SoC) ARM architecture : "A" (**Mobile**, Tablets, TV) and "M" series (Desktop, **Laptop**))

## Android

Differences across Android OS OEMs are accommodated typically using the HAL and the Android Framework

SC (Java/Kotlin) IDE: e.g. Android Studio, ...

(Java and Dalvik) Compiler [AOT]

Signed

(Dex) Bytecode

APK package

Sideloading

Google Play Store

Google Play Developer Console

Google Review — Human Review, Automated Review

Optional

Signed (by Google)

Application Server (3P app store/ web loading)

External Media

Application Server

User Consent

If not signed by developer

Unprivileged App | Privileged/OEM Pre-Installed (System) App

(Public) Android APIs | (Private) System APIs

**Android Framework** (Java pre-compiled classes and interfaces upon which apps are built, surfacing: Public (third party) / System (OEM/partners) APIs)

**System Services** (written in Java/Kotlin, e.g. an activity manager service for multitasking capabilities, notifications manager, windows manager, etc.)

**Android Runtime** (based on Dalvik VM)

Interpreter — 'hot spot' — Compiler [JIT] — Binary; Native Cache; Compiler [AOT] — Binary

**Hardware Abstraction Layer** (OEM device-specific features)

**Native Deamons** (background processes, e.g. initialize the system) **and Libraries** (e.g. math, logging, cryptography)

(Monolithic) **Generic Core Kernel Module (KM) (Linux) – Google KM – Vendor KM**

(**Mobile**, Tablets, Embedded) **Hardware** (SoC ARM architecture – Snapdragon (Qualcomm), Exynos (Samsung), Tensor (Google), …)

## Windows 11 (NT Kernel)

SC (C#/C++/.NET) IDE: e.g. Visual Studio, …

Compiler [AOT]

Signed

Binary/IR

MSIX/AppX Package / EXE/MSI for sideloading

Sideloading

Windows Store

Microsoft Partner Centre App Review — Human Review, Automated Review

Signed (by Microsoft)

Application Server (3P app store/ web loading)

External Media

Application Server

User Consent

If not signed by developer

Windows Un/Privileged and Preinstalled Apps

Windows Store Apps (WS Apps) | Universal Windows Platform (UWP) Apps | Desktop Apps

Windows Runtime — Binary

.NET Framework/Win32/POSIX (Runtime) — Binary

**Executive Services** (I/O Manager, Security Reference Monitor, IPC Manager, Virtual Memory Manager, Process Manager, Power Manager, Window Manager, Plug and play manager, graphic device interface, …)

**Object Manager** (Manages system resources represented as objects, e.g. files, and threads. All other subsystems above need to pass through the OM)

(Kernel Mode) **Drivers** | (Micro) **Kernel**

**Hardware Abstraction Layer**

(**Laptop**, Desktop, Server, Tablets, Embedded) **Hardware** (x86 architecture – "Core i(5, 7)" – Intel, Ryzen – AMD, SoC ARM architecture – Snapdragon – Qualcomm…)

## Linux (Debian, Arch, Red Hat, …)

Different distributions use an equivalent architecture.

Signed/ unsigned

SC (C#, C++, Java, Kotlin, .NET, Swift…) IDE: e.g. Vim, Eclipse, Visual Studio Code, …

Compiler [AOT]

Signed/ unsigned

Binary/IR

APT, Pacman, Tarballs, AppImage, Snap, … Packages

Application Server (3P app store) web loading)

External Media

User Consent

App

**System Utilities** – Executable **preinstalled** applications like the **Shell** (i.e. UI to interact with the kernel via command-line interface like Bash or graphical user interface like GNOME), that in turn, users can run from the Shell, e.g. tar, emacs, shell (e.g. bash), grep, package managers …

**System Libraries (APIs)** – pre-compiled functions for applications to interact with the Kernel, e.g. glibc, POSIX Threads

**Runtime (rich support)**

Binary

**System Call Interface (APIs)**

(Monolithic) **Linux Kernel** – Subsystems: Process scheduler, memory management unit, virtual file system, networking subsystem, Inter-Process Communication Unit, drivers, …

(Laptop, Desktop, Server, Mobile, Embedded) **Hardware**