Introducing the Swift SDK for Android

— Marc Prud'hommeaux (marc@skip.dev)
— Founding member of Swift Android workgroup
— iOS & Android Developer since 2008
— Creator of Skip (`skip.dev` *né* `skip.tools`)
— Slides: skip.dev/talks/fosdem2026

Flash Poll: Audience Swift Focus

— iOS
— macOS
— Server
— Embedded
— Windows
— WebAssembly
— Android

Android

— Most widely-used operating system in the world
— ~4 billion Android handset device users
— OS for myriad of IoT/embedded/auto devices
— Linux Kernel (GPL)
— Android Open Source Project (AOSP) (Apache)
— Android Certified (proprietary)
    — Google Mobile Services + Google Play Services

Android for iOS Developers: Development

	iOS	Android
Modern Language	Swift	Kotlin
Legacy Language	Objective-C	Java
Build System	SwiftPM	Gradle
Platform SDK	Obj-C (+C & Swift)	Java (+C)
OS/Kernel	XNU/Darwin	non-GNU/Linux
Native Libraries	Mach-O `.dylib`	ELF `.so`

Android for iOS Developers: App Development

	iOS	Android
IDE	Xcode	Android Studio, IntelliJ, …
Virtual Device	iOS Simulator	Android Emulator
UI Framework	SwiftUI (Swift)	Jetpack Compose (Kotlin)
UI (Legacy)	UIKit (Obj-C)	XML Views (Java)
App Packaging	`.ipa`	`.apk`
Distribution Portal	App Store Connect	Google Play Console
App Marketplace	Apple App Store	Google Play Store

See Native Swift on Android, Part 1, September 2024

— Running Swift code on Android, Romain Goyet, October 14, 2015

— How we put an app in the Android Play Store using Swift, Geordie J, Jul 8, 2016

— Android Community Workgroup formed February 2025

— Android Workgroup officially formed June 2025

Contributor Hall of Fame

— andriydruk*
— compnerd*
— finagolfin*
— ephemer
— hyp
— mstokercricut
— vgorloff

How Does it all Work?

Flash Poll 2: Swift Cross-Compilation SDK Experience

— Static Linux (musl)
— Webassembly (wasm)
— Android

Swift Cross-Compilation SDK

— SE-0387: Swift SDKs for Cross-Compilation implemented in Swift 6.1 (2024)
— Self-contained `.artifactbundle` archive
— Enables building for a separate target than the host
— e.g., build for Android from macOS or Linux
— Like building for iOS from macOS
— Other Swift SDKs: WebAssembly (wasm), Static Linux (musl)

Structure of the Swift SDK for Android

Unlike the other Swift SDKs, the Swift SDK for Android is *not* self-contained. It requires an external "Native Development Kit"

1. Host Toolchain (OSS via `swiftly`, *not* Xcode)
2. Swift SDK for Android
3. Android Native Development Kit (NDK)

Structure of the Swift SDK for Android: 1. The Host Toolchain

- Host Toolchain (OSS via `swiftly`, *not* Xcode)
  - Lives in:
    - `~/Library/Developer/Toolchains/swift-6.2.3-RELEASE.xctoolchain/`
  - (NOT: `/Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain`)
  - Provides: `swiftc`

- Swift SDK for Android
  - Lives in: `~/Library/org.swift.swiftpm/swift-sdks/swift-6.2.3-RELEASE_android.artifactbundle/`

Structure of the Swift SDK for Android: 3. The Android Native Development Kit (NDK)

- Android Native Development Kit (NDK)
  - Lives somewhere like:
    - `~/Library/Android/sdk/ndk/27.0.12077973/`
  - Provides:
    - `aarch64-linux-android28-clang`, `x86_64-linux-android35-clang`, `ld`
  - Outputs: `lib/arm64-v8a/libMySwiftCode.so`

Setup

— Getting Started with the Swift SDK for Android — swift.org

— NDK not included

— Install an emulator for testing

Building

— Build for ARM (64):

```
swift build --swift-sdk aarch64-unknown-linux-android28
```

— Build for X86 (64):

```
swift build --swift-sdk x86_64-unknown-linux-android28
```

Android Architectures

— `arm64-v8a` (64-bit ARM): handsets, most popular
— `armeabi-v7a` (32-bit ARM): older handsets, IoT
— `x86_64` (64-bit Intel): Chromebooks, dev emulators
— ~~`x86`~~ (32-bit Intel)
— ~~`mips`~~ (old RISC)
— ~~`riscv`~~ (new RISC)

The Swift SDK for Android comes with a number of built-in `.swiftmodules`

Available Frameworks

— `Swift` standard library

— `Foundation`

— `Dispatch`

— `Observation`

— `Testing` & `XCTest`

— `Android` (instead of `Darwin`)

Unavailable Frameworks

Pretty much everything else…

— `CoreGraphics`, `CoreLocation`, etc.

— `CloudKit`, `StoreKit`, `GameKit`, `HealthKit`, etc.

— `UIKit`, `SwiftUI`

— *Rule of thumb*: if it isn't available on Linux, then it won't be available on Android.

Third-party Packages

— `swiftpackageindex.com`: search `"platform:android"`
— Adding Android compatibility testing blog post
— ~30% of their ~9k packages build for Android
  — swift-collections, swift-crypto, swift-log, …
  — Alamofire, Factory, Yams, GraphQL, Supabase, …

How can we get our own Swift Packages working on Android?

— Porting Swift Packages to Android – skip.dev/docs/porting/

— iOS ➜ macOS ➜ Linux ➜ Android

Porting Example

```
#if canImport(Darwin)
import Darwin // macOS, iOS, etc.
#elseif canImport(Glibc)
import Glibc // for Linux
#elseif canImport(Android)
import Android // or Bionic
#endif

import Foundation
// non-Darwin Foundation doesn't include FoundationNetworking
#if canImport(FoundationNetworking)
import FoundationNetworking
#endif
```

Running an Executable

```
swift package init --name DemoCmd --type executable
swift run DemoCmd # builds and runs on host

swiftly run swift build --swift-sdk aarch64-unknown-linux-android28 --static-swift-stdlib

adb push .build/aarch64-unknown-linux-android28/debug/DemoCmd /data/local/tmp/
adb push …/libc++_shared.so /data/local/tmp/

adb shell /data/local/tmp/DemoCmd
```

https://www.swift.org/documentation/articles/swift-sdk-for-android-getting-started.html

Testing a Library

- Local Testing

```
swift package init --type library --name DroidLib
swift test
skip android test
```

- Continuous Integration
  - Swift Android Action (Linux or macOS)
  - https://github.com/marketplace/actions/swift-android-action

Current Status of Swift SDK for Android

— Available as 6.3 and main branch nightly snapshots
— Will be part of Swift 6.3 release (Spring 2026)
— WIP: Debugging
— WIP: Testing Improvements
— WIP: IDE Support
— TODO: Packaging, Integration (Gradle, etc.)
— TODO: Size Reduction: thinning ICU?

Next Up: Java Integration

— CLIs and tests aren't an app
— Android apps need to work with the Android SDK...
    — ...which is Java
    — ...and that means the Java Native Interface (JNI)
— Mads is up next to talk about that...

How you can help

- Try it out, give feedback
  - https://forums.swift.org/c/platform/android
- Add Android support to your packages
  - add Android to your CI matrix

Thank You!

— Marc Prud'hommeaux (marc@skip.dev)
— FODEM: find me at the F-Droid kiosk (UD1)
— Slides: skip.dev/talks/fosdem2026