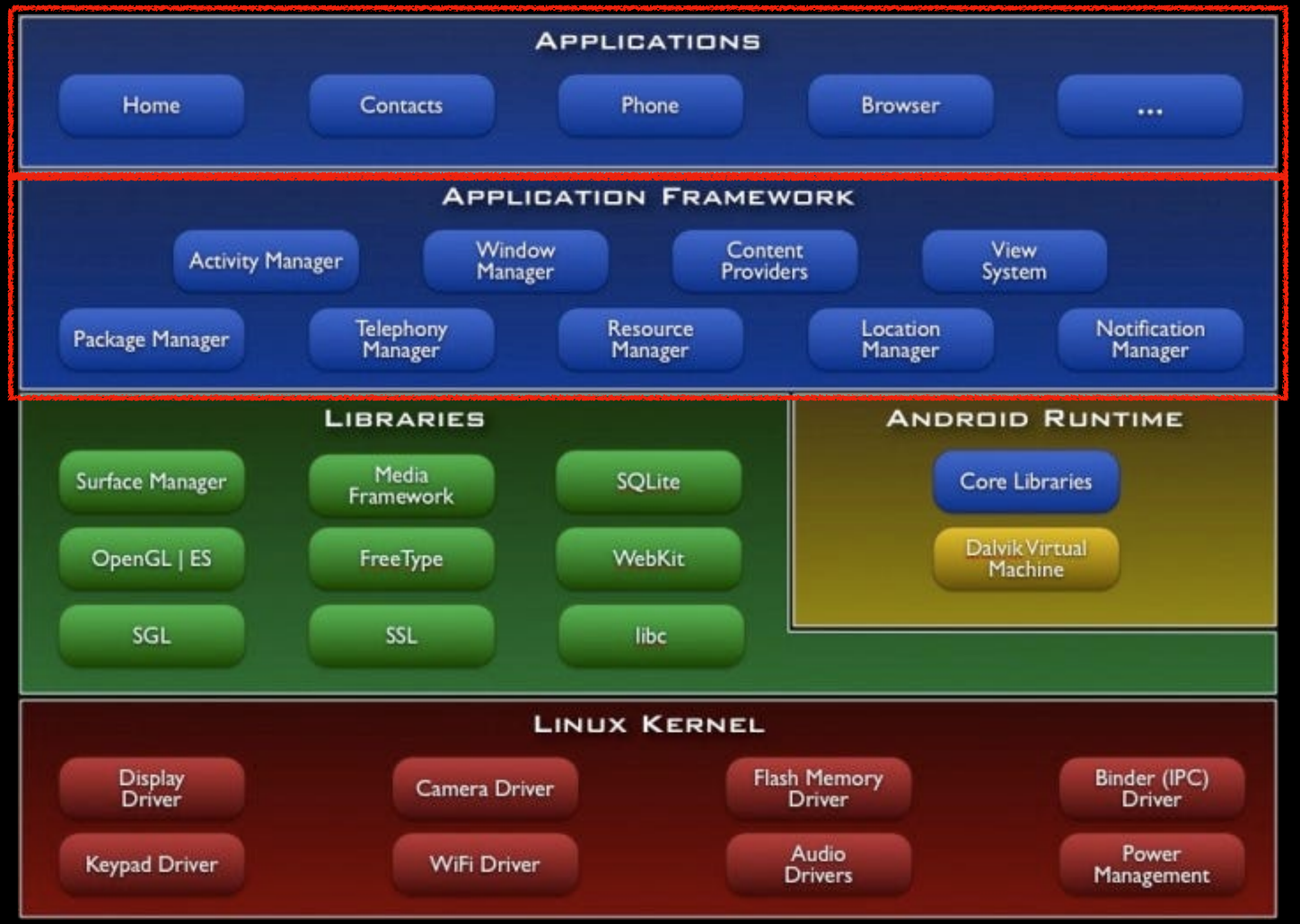
안드로이드 앱, 어떻게 실행될까?

Android Runtime의 빠른 실행을 위한 발전사 톺아보기

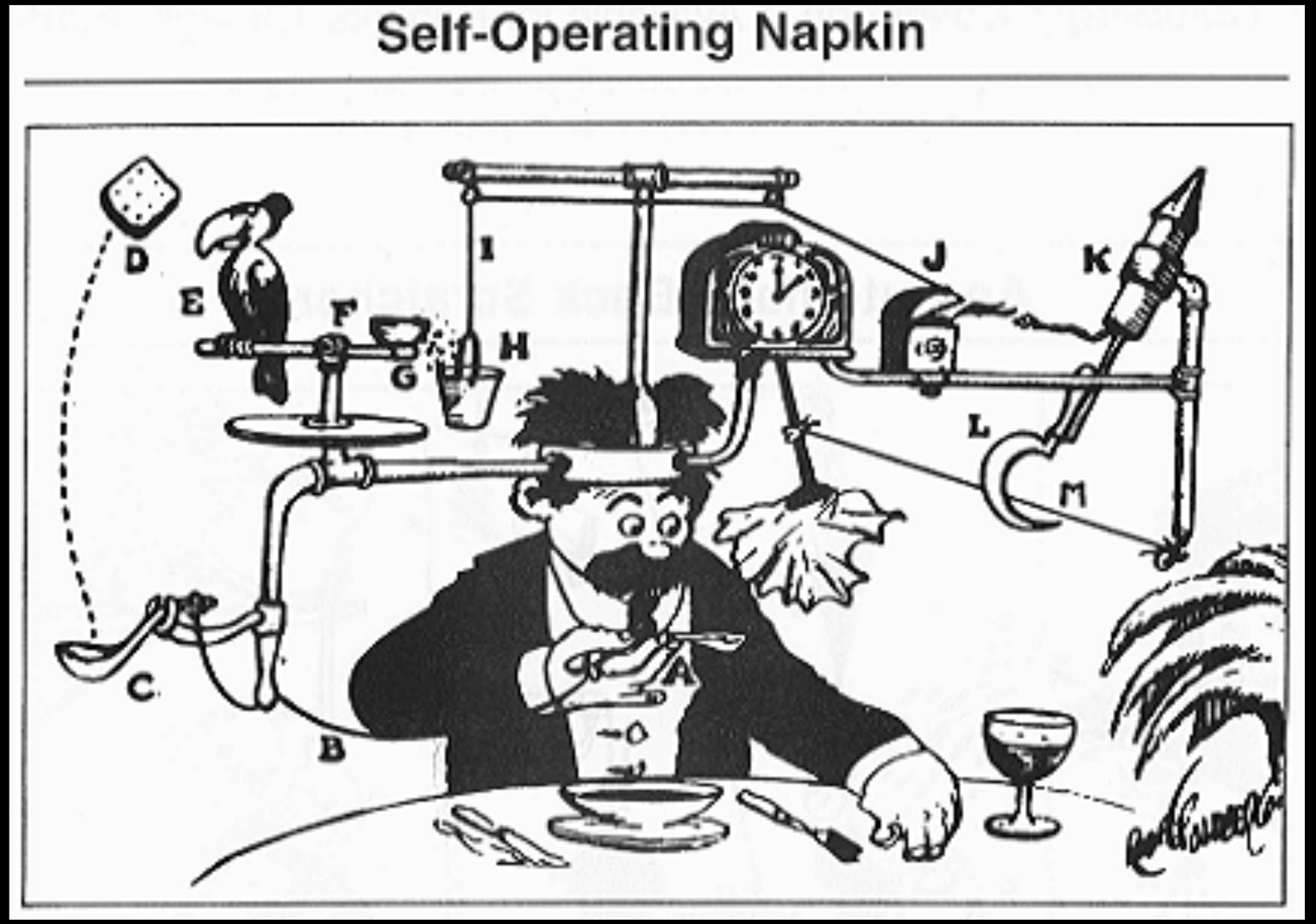
Android Developer 이용진

안드로이드... 하시는 분..?

안드로이드, 아키텍처 먼저 볼까요?



안드로이드 앱은 어떻게 실행될까?



앱 실행 거시적으로 살펴보기

1. 사용자가 홈 화면에서
앱 아이콘을 누른다

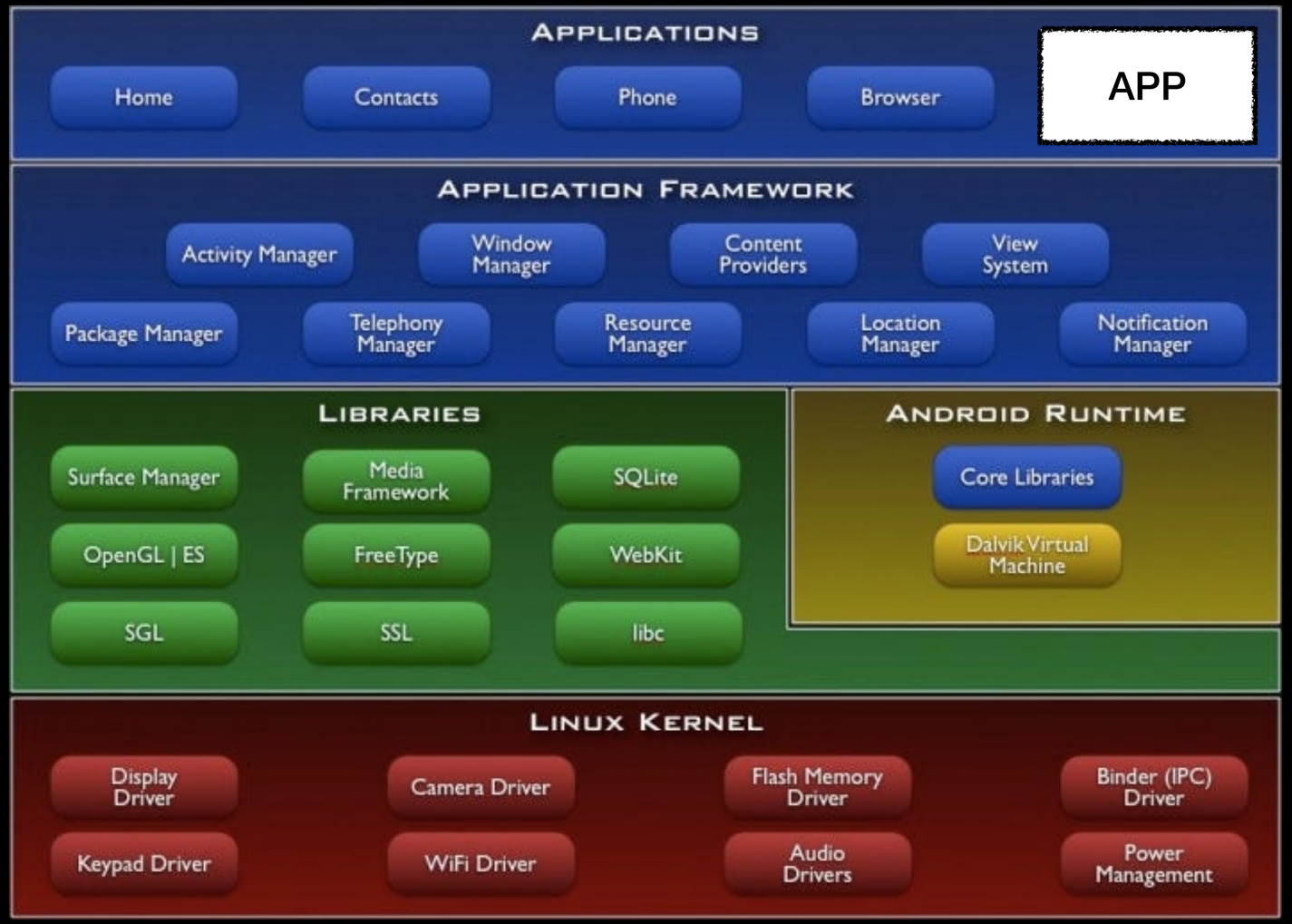
2. ActivityManager가 앱 실행 요청을 받아
Zygote라는 프로세스에 프로세스 생성을 요청한다

3. Zygote라는 프로세스가 fork() 시스템콜을 사용해
새로운 앱 프로세스를 생성한다

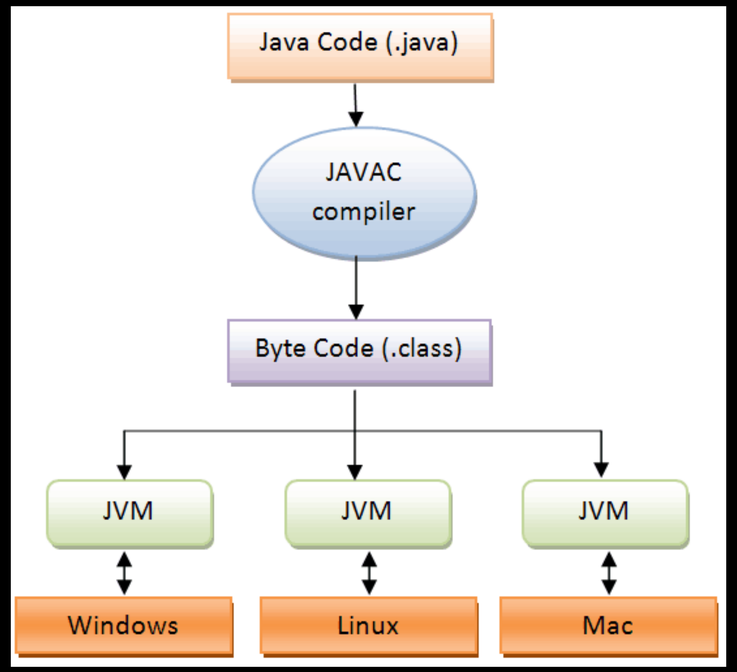
4. 앱 프로세스 내에서 Android Runtime이 시작된다.
앱의 dex 바이트코드를 로드하고 컴파일할 준비를 한다

5.ActivityThread가 생성되고
메인스레드 (UI 스레드)를 실행시킨다

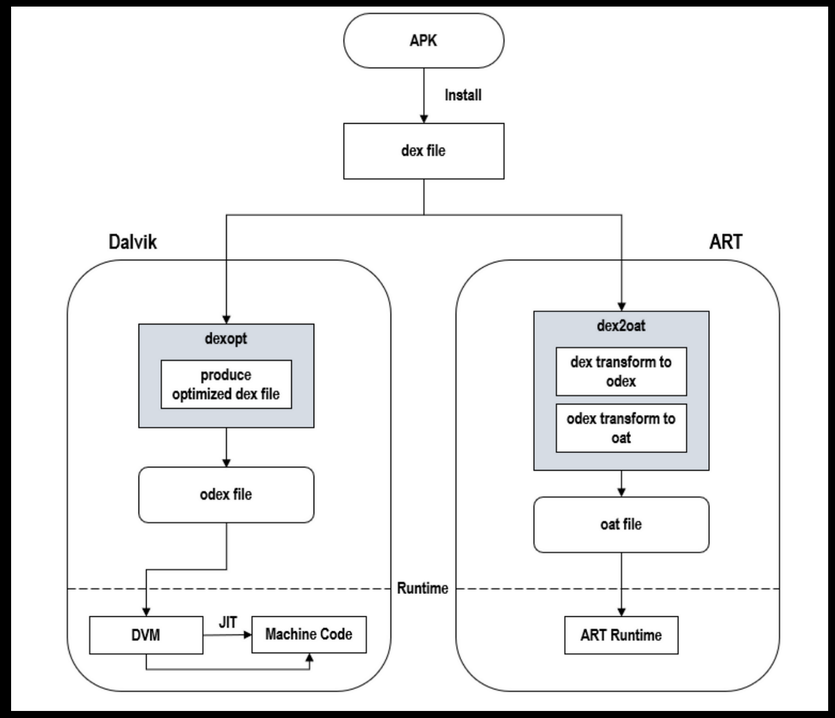
6.사용자와 상호작용하며 앱이 동작한다



Android Runtime이란?

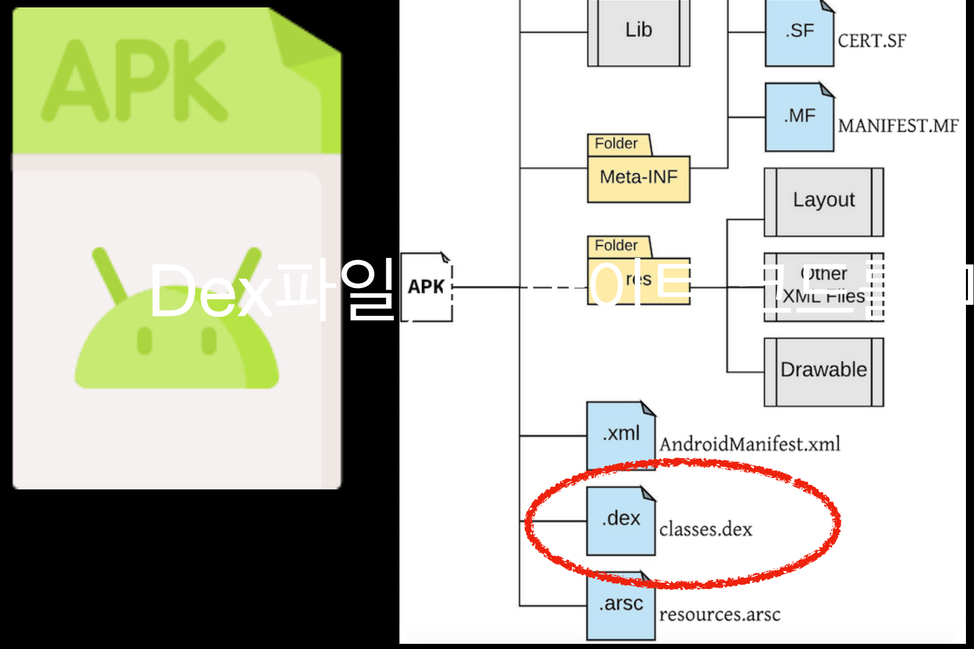


JVM



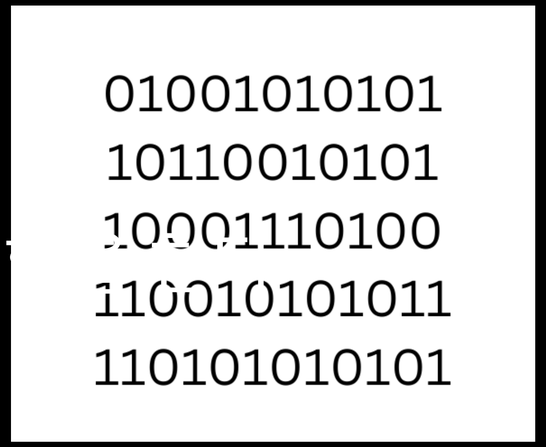
Dalvik VM, ART

Android Runtime이란?



Dex파일 [illegible] 어트 [illegible] 머신코드로 컴파일 하는 역

Android Runtime



Machine Code

Dalvik VM

Android 4.4까지 사용된 최초의 안드로이드 런타임

런타임에 바이트 코드(dex파일)를 머신코드로 컴파일했음

JIT
(Just In Time)

- 메모리를 절약할 수 있었음 😎
- 런타임 성능이 좋지 않았음 🤯
- 자주 실행되는 코드를 캐시하도록 수정 😎

ART (Android Runtime)

Android 5.0부터 사용된 안드로이드 런타임

설치 시에 바이트 코드(dex파일)를 머신코드로 미리 컴파일했음

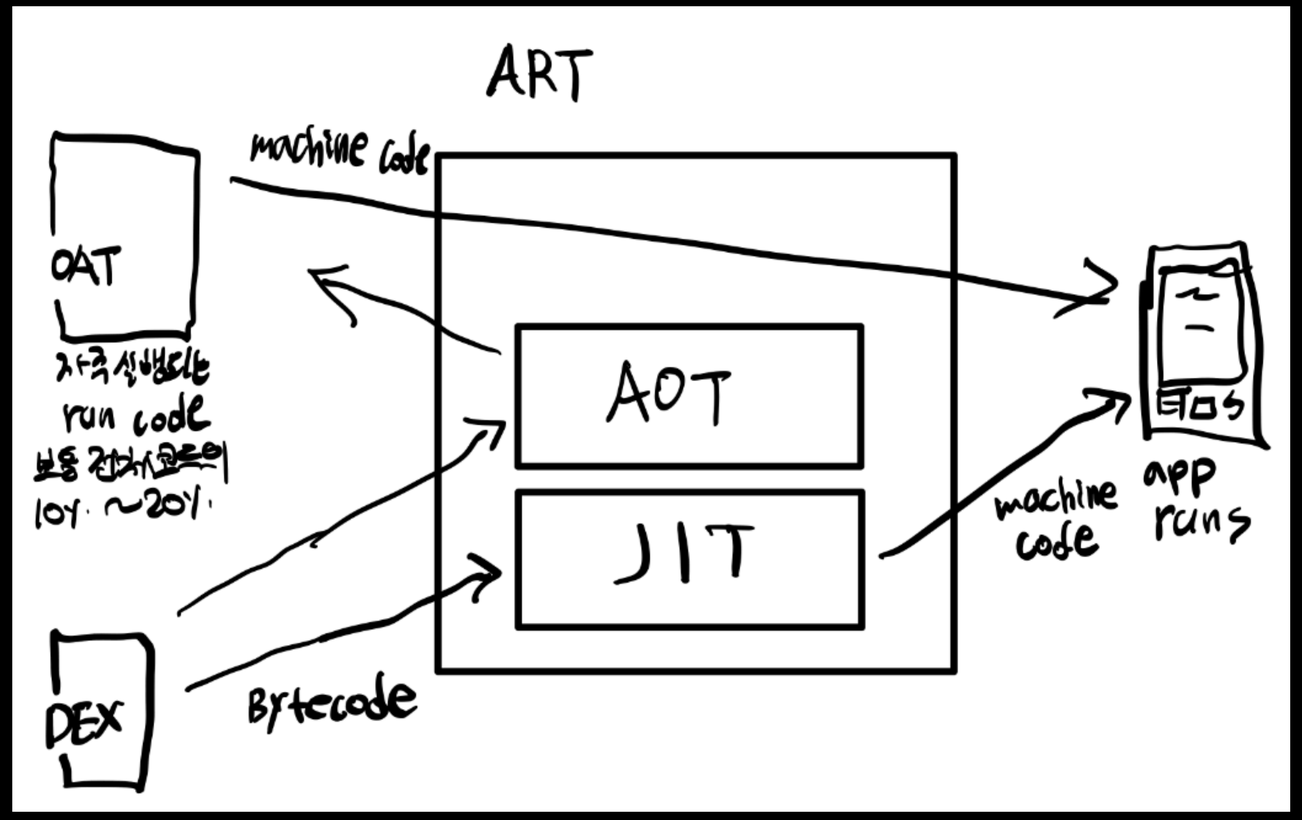
AOT
(Ahead Of Time)

- Dalvik보다 약 20배 이상 빠름 😎
- Dalvik보다 더 많은 메모리 차지 🤯
- 앱 설치 시간 증가 🤯
- OS 업데이트 시간 증가 🤯

ART (Android Runtime)

Android 7.0부터 추가된 Profile-guided 컴파일 기능

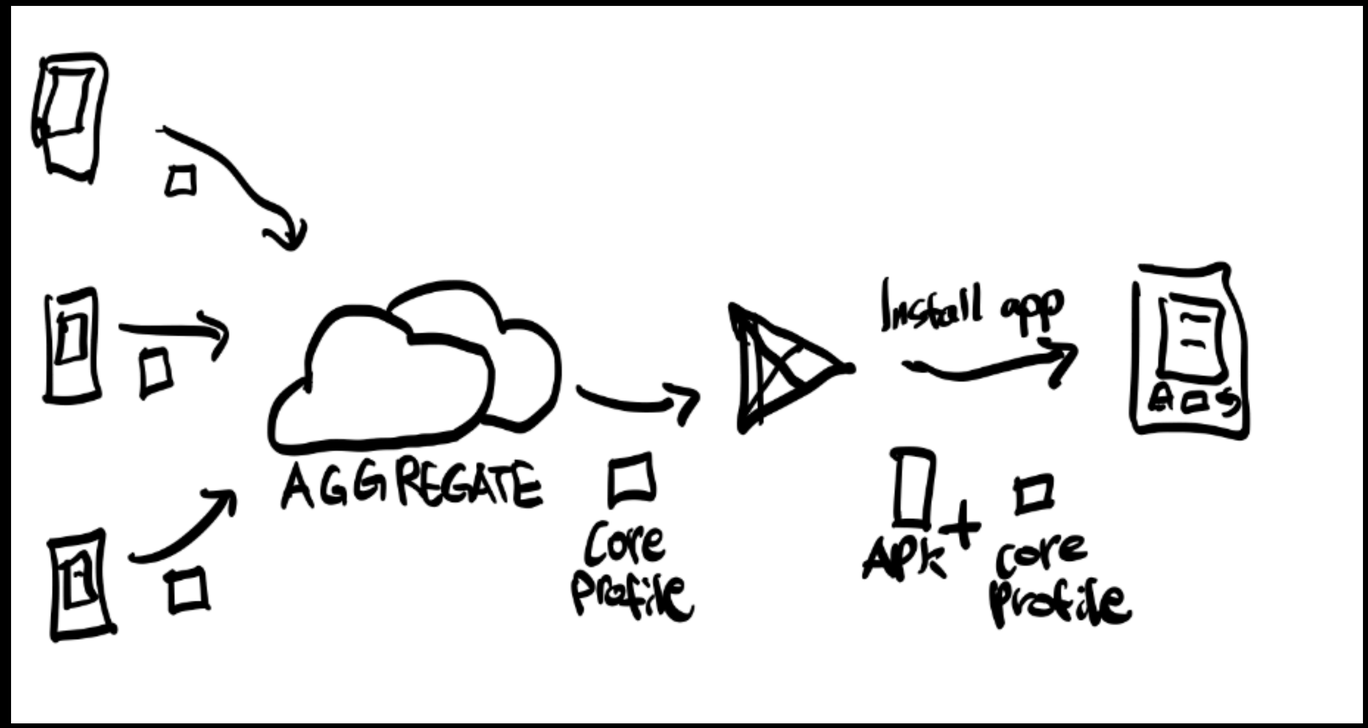
AOT + JIT
(with Profile-guided)



ART (Android Runtime)

Android 9.0부터 추가된 Cloud Profile 기능

AOT + JIT
(with Cloud Profile)



마지막 정리

- Android Runtime은 여러 역할을 맡고 있지만 그 중 하나는 바이트코드를 머신코드로 컴파일하는 역할을 함
- Android Runtime은 Dalvik ~ ART까지 계속해서 더 빠른 앱 실행을 위해 개선되어옴

1. Dalvik VM (JIT)
2. ART (AOT)
3. ART (JIT + AOT) with profile-guided
4. ART (JIT + AOT) with cloud profile

Q&A