ANDROID DEV LINE — 2009 ▶ NOW

Time-Travel in Android development

Eclipse ADTの時代から、Jetpack Composeの現在まで — 日本のAndroidアプリ開発 全駅停車

SCOPE	PERIOD	HOW TO READ	NOTE
日本のAndroidアプリ開発 国内で話題になった技術・出来事が中心	**2009年ごろ〜現在** 国内スマホ黎明期からComposeを経てAIの時代まで	**みんなで振り返る用** 各駅の「Q.」にシールを貼ってください	**諸説あります** 特に古い時期は正確性に欠ける場合があります

Kotlinがなく、Android Studioがなく、DroidKaigiすらなかった時代を、日本のAndroid開発は通ってきました。 このポスターは、**日本のAndroidアプリ開発が通ってきた「駅」を一本の路線として振り返る**ものです。 **あなたが乗車した駅はどこですか？** 各駅の「Q.」で、思い出を振り返ってみてください。

2009–2012 Donut▶ICS 始発駅：Eclipse ADTの時代

- HT-03A、Xperia…国内にAndroid端末が上陸。**ガラケーからスマホへ**
- 開発はEclipse＋ADT。**クラッシュとの戦い**（Eclipse自体の、も含む）
- Tablet対応のため**Fragment**登場。Honeycombは短命に終わりICSへ統合
- **Support Library**登場（2011〜）。新しいUI・APIを旧端末にも提供

```
// lifecycleScopeもrepeatOnLifecycleも無い。ライフサイクル管理は完全にプログラマ任せ
```

Q. EclipseでAndroid開発したことある人！

2013–2014 JB▶KitKat ライブラリ百花繚乱とHolo

- `Volley`：Google製通信ライブラリ。後にOkHttp＋Retrofitが台頭
- `ButterKnife`：`@BindView` で findViewByld を撲滅（View注入）
- `EventBus`：疎結合を実現するも、乱用でデバッグ困難な負債にも
- 青く光る**Holoデザイン**（2011登場）の円熟期。**Android Studio**発表（2013）→1.0（2014）

```
// Bitmapリーク、Fragment再生成、staticなデータ保持…クラッシュ原因は無限にあった
```

Q. Android開発を始めたとき、DroidKaigiがなかった人！

2015 Lollipop DroidKaigi初開催

- **初のDroidKaigi開催。**FatActivity対策・クラッシュ対策など知見共有が主体
- Dalvik VMから**ART**へ。AOTコンパイルでアプリの起動・実行が高速化
- **Material Design**発表、Android 5へ適用
- **Multidex問題**（64K問題）：メソッド数65,536の壁。Android 5で解消へ

Q. Android開発を始めたとき、DroidKaigiがあった人！

2016 Marshmallow Reactiveの波と、権限の時代

- **RxJava**による Reactive Programming が本格普及
- **Runtime Permission**：権限をインストール時から実行時に確認する方式へ
- **Dozeモード**：端末放置時にバックグラウンド処理を制限
- **Dagger2**によるDI、RecyclerView、Espresso。Kotlinもじわじわ話題に

2017 Nougat Kotlin正式サポート（I/O 17）

- Googleが**Kotlinを正式サポート**。正式採用前から現場は盛り上がっていた
- ORM百花繚乱のピークは2015〜16（Realm・Orma…）。**Room**登場（I/O 17）で収束へ

Q. Android開発を始めたとき、Kotlinがあった人！

2018 Oreo AAC登場、成熟の始まり

- **Android Architecture Components**登場（I/O 17発表）。LiveDataもここから
- Flux、MVC/MVP/MVI、MVVM…アーキテクチャ議論が最も熱かった時代の収束点
- Google公式指針で**MVVM系**がデファクトに（Clean Architectureの議論も活発化）
- DataBinding、Annotation Processing

Q. 始めたとき、Android Architecture Componentsがあった人！

2019 Pie Coroutines／Flowの時代へ

- Kotlin 1.3で**Coroutines**がstable。`suspend` の概念が登場
- RxJavaからCoroutines/Flowへ移行が始まる（剥がせないプロジェクトも…）
- `viewModelScope` などスコープ管理が整い、非同期処理が書きやすく

2021 Android 12 Jetpack Compose 1.0

- 2月に**beta**、7月に**1.0 stable**。宣言的UIの時代へ。「つい最近」な気がしますが、もう5年
- 「Stableからまだ1年」と思い込みたい人、正直に手を挙げてください
- Composeとともに Android Studio は**Arctic Fox**へ

Q. Android開発を始めたとき、Jetpack Composeがあった人！

2024–2025 Android 16 AIエージェント時代

- Copilot・ChatGPTの補完から一歩進み、**Gemini**がAndroid Studioに統合
- Claude Code等の**コーディングエージェント**が実用レベルに到達
- 「人が書きAIが補完する」から「**AIが書き人がレビューする**」開発へ

Q. Android開発を始めたとき、AIエージェントがいた人！

2026 NOW 終着駅（現在地）：DroidKaigiは12回目へ

あなたの「乗車駅」の思い出、ぜひこのポスターの前で聞かせてください。

tetsukay／田熊 • DeNA Co., Ltd.