



NEWTにおけるiOS18対応の進め方

令和トラベル

2024.9.30 【GO/note/令和トラベル】After iOSDC & DroidKaigi 2024

Rick



Rick
@ryu1sazae

Reiwa Travel, inc

iOS Engineer

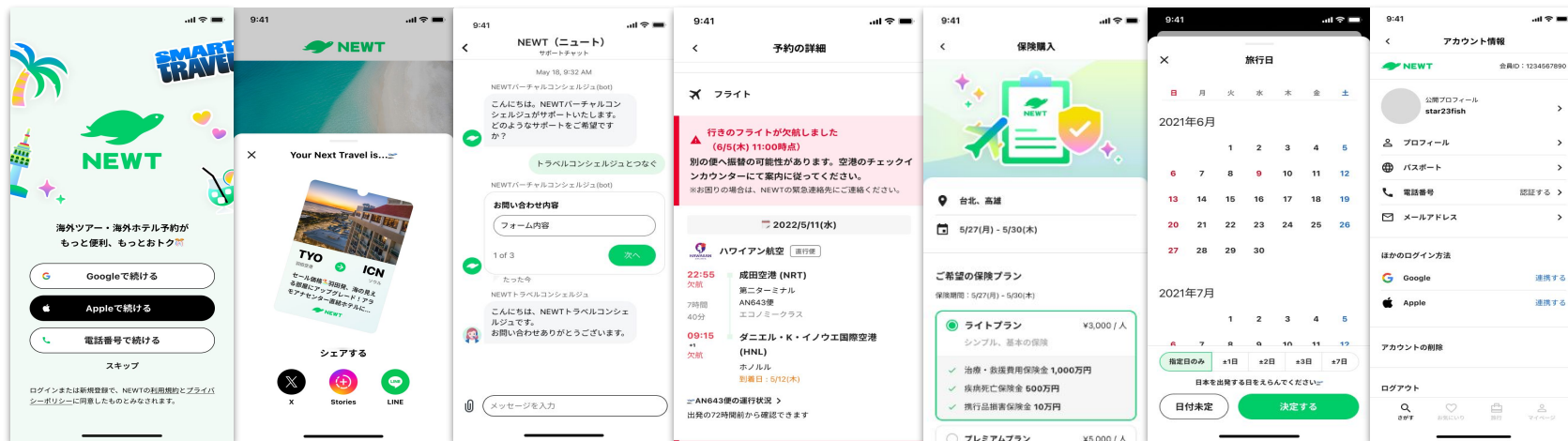


- **Xcode 15.4, Swift 5.10**
→ **Xcode 16.0 Swift 6.0**
- **サポートOS : iOS16~**
- **パッケージマネージャー : SPM**
- **フレームワーク : TCA**
- **iOSエンジニア : 2名**

1. 事業要求・品質・最新技術追従のバランスを重視
2. 段階的なSwift6対応で備える
3. iOS18対応のスケジュール
4. まとめ

1. 事業要求・品質・最新技術追従のバランスを重視
2. 段階的なSwift6対応で備える
3. iOS18対応のスケジュール
4. まとめ

急激に成長するサービスに対する事業の要求に応える



8・9月だけでも多くの施策をリリース



これらを、iOSエンジニア2名で開発する必要があった

海外旅行は多くて年一回という方がほとんど



貴重な体験を最大限素敵なものにしてもらうために、**最高の品質をカスタマーに届ける必要があります、それを保証する必要がある**



品質保証のために魅力的な技術の選択を犠牲にすることもある

- 会社のプロダクト規模で新しい技術を使う楽しさ
- プロダクトとしての表現幅の向上
- 実装容易性の向上
- 世で進化する技術とのギャップを埋めることによる負債解消

- 機能開発のスピードはなるべく通常通りを維持したいという事業目線での要求
- iOS18リリース直後でも**品質は担保必須**
- 新OS対応時は、QAチームによるテストを手厚めに実施するため、このタイミングで**可能な限りSwift6の対応を進めておきたい**というエンジニアの願望

これらのバランスを獲得するためには、
どのような手順・スケジュールで対応を進めればよいか？ 



iOS18をサポートするXcode16にて、
Swift6が正式に利用可能になった。
幸いSwift6への対応は”**段階的**”に実施することが可能 

1. 品質と最新技術追従のバランスを重視
2. **段階的なSwift6対応で備える**
3. iOS18対応のスケジュール
4. まとめ

Swift6対応終わってる方～？ 🙋

NEWTアプリは・・・？ 

48 / 57 target完了済み 🔥
(9/27時点)

段階的なSwift6対応を実施するためのフラグ



	enableUpcomingFeature(_:_:)	enableExperimentalFeature(_:_:)
利用可能Swift version	5.8~	5.8~
用途	次のメジャーバージョンで導入が決定されている機能を使用可能	文字通り、実験中の機能を使用可能
各フラグで有効にできる機能の確認方法	https://github.com/swiftlang/swift/blob/release/x.y.z/include/swift/Basic/Features.def	

Xcode16の登場時になるべく多くのTargetでSwift6モードを使用したい

→ Xcode15.xの利用時から、個々のTargetに対して可能な限りSwift6以降の機能を有効化 🔥

部分的にSwift6の利用開始を遅らせるためのMode指定



	swiftLanguageMode (SE-0435)
利用可能 Swift version	6.0~
用途	Target毎にSwiftのversionを指定 e.g. 一度にSwift6への移行が難しいTargetでは過去のversionを指定

```
// swift-tools-version:6.0
import PackageDescription

let package = Package(
    name: "NEWT",
    defaultLocalization: "ja",
    platforms: [
        .iOS(.v16),
    ],
    products: [
        .library(name: "AppFeature", targets: ["AppFeature"]),
        .library(name: "ApiClient", targets: ["ApiClient"]),
        ...
    ],
    dependencies: [
        .package(url: "https://github.com/pointfreeco/swift-composable-architecture", from: "1.15.0"),
    ],
    targets: [
        .target(
            name: "AppFeature",
            dependencies: [
                "ApiClient",
                ...
                .product(name: "ComposableArchitecture", package: "swift-composable-architecture"),
                ...
            ],
            path: "Sources/Features/App"
        ),
        .target(
            name: "ApiClient",
            dependencies: [
                "NEWTAPI",
                ...
                .product(name: "ComposableArchitecture", package: "swift-composable-architecture"),
            ],
            path: "Sources/Dependencies/ApiClient",
        ),
        ...
    ]
)
```

- Feature target

画面単位のUI・TCAのReducer

サービス内のコンテキスト単位

- Dependency target (e.g. ApiClient)

副作用 (e.g. API通信, ログ送信, etc...)

- その他のTarget

共通コンポーネント, モデル, リソース, etc...

以下の懸念から、UI・TCAのReducerを含むFeature TargetはSwift6モードの利用を、iOS18対応版リリース後に遅らせた。

① UI周りは、新OS・Xcode登場直後に不具合発生が予想されやすく・対応に時間もかかり得る

iOSエンジニア2名で機能開発も並行して行うために、避けられる工数をなるべく避けたい。**beta環境での開発だと尚更。**

② TCAへの依存から生じるコストの最小化

2024/09前半時点では、Concurrency周りのwarningやerror対応、今後の方針に対するdiscussionが活発に行われていた。この状況下でStrictConcurrencyの対応を完了させるよりも、それらが落ち着いたタイミング、または予定されているTCA v2.0の方針がもう少し固まってからの方が、ライブラリに依存するコストを最小化できると判断。

以下の懸念から、UI・TCAのReducerを含むTargetはSwift6モードの利用を、iOS18対応版リリース後に遅らせた。

① UI周りは、新OS・Xcode登場直後直後に不具合発生が予想されやすく・対応に時間もかかり得る

iOSエンジニア2名で機能開発も並行して行うために、避けられる工数をなるべく避けたかった（beta環境での開発だと尚更）

② TCAへの依存から生じる得るリスク / コストの最小化

2024/09前半時点では、Concurrency周りのwarningやerror対応、**今後の方針に対するdiscussionが活発**に行われていた。この状況下でStrictConcurrencyの対応を完了させるよりも、それらが落ち着いたタイミング、または予定されているTCA v2.0の方針がもう少し固まってからの方が、**ライブラリに依存することで発生し得るリスク / コストを最小化できると判断**。

1. 事業要求・品質・最新技術追従のバランスを重視
2. 段階的なSwift6対応で備える
- 3. iOS18対応のスケジュール**
4. まとめ

Xcode16 betaでもスムーズにSwift6モードを有効化完了

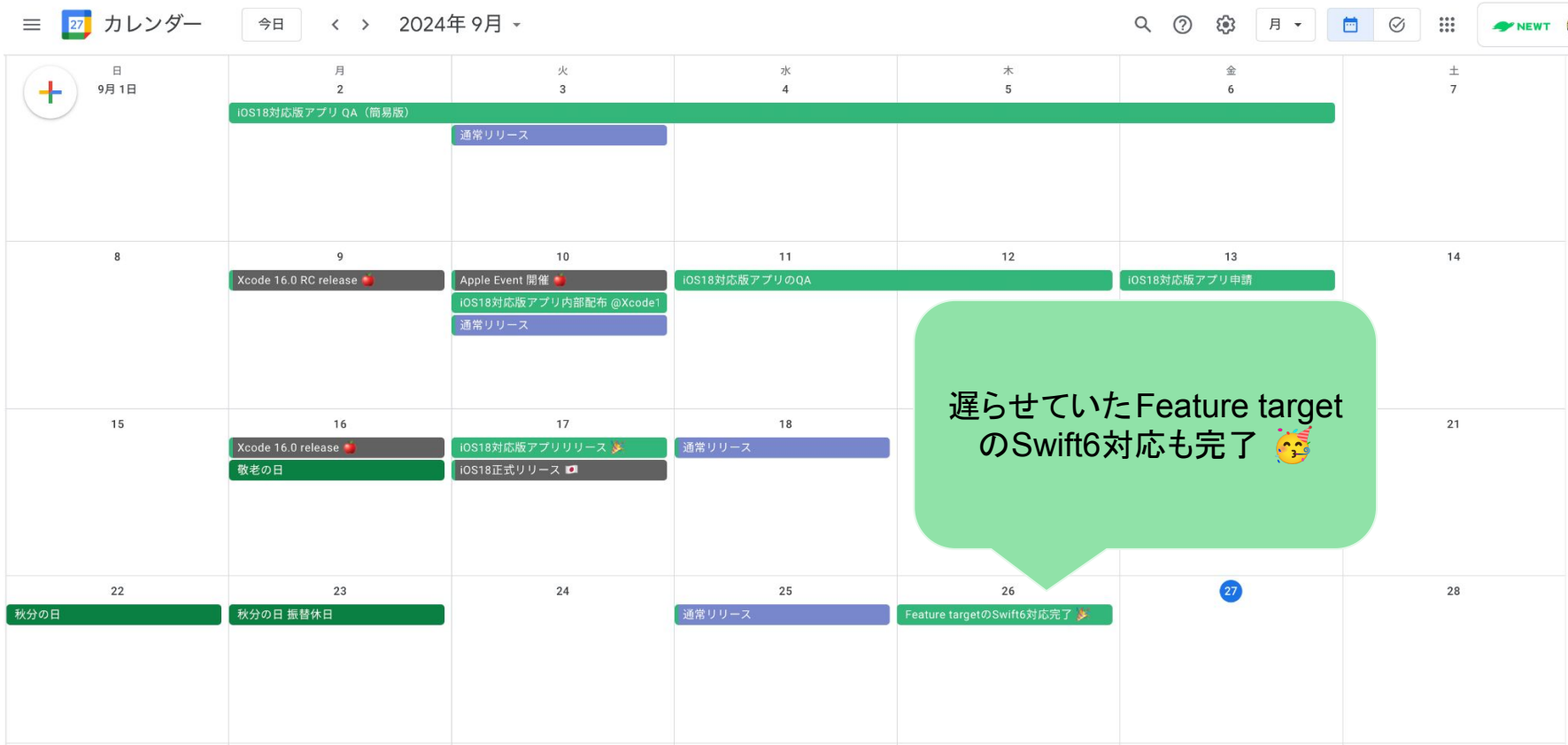


簡易版QAでリリース直前のリスクを軽減



9/10のApple Event直後に
正式リリースだと予測し、前
もって簡易版QAを実施

簡易版QAで備えていた
ため、Xcode RCでのQAも
不具合なく完了 🎉



1. 事業要求・品質・最新技術追従のバランスを重視
2. 段階的なSwift6対応で備える
3. iOS18対応のスケジュール
4. **まとめ**

- 段階的にSwift6対応を行うことで、事業要求・品質とのトレードオフ関係を断ち切る
- 自己解決の難易度が高いUI・ライブラリ依存箇所はリスクヘッジを図りながら進める
- 品質を向上させながら、事業に貢献するために、技術を活用する

Thank you!