

KMP + React Native

跨端业务逻辑共享实战

ZouLe

aweffr@foxmail.com

自我介绍 – ZouLe

- 全栈脚本仔
 - Python, JS, Kotlin, Linux
- 负责内部IM移动端
- 5年RN/Web开发经验

背景 (Motivation)

- React Native
 - 😊 高度兼容前端技术栈
 - 😊 应用广泛
 - 😊 社区成熟
 - 😓 运行效率受制于JavaScript引擎
 - 😓 单线程处理能力有限
- Kotlin Multiplatform (KMP)
 - 😊 Kotlin强大的表达能力
 - 😊 运行效率高
 - 😊 二进制调用原生平台开销低
 - 😓 社区尚未成熟
 - 💪 社区最佳实践快速迭代中



KMP + React Native: 优势互补

KMP 作为原生侧方案的优势

- 相较于安卓（Java/Kotlin）与iOS（Obj-C/Swift）分别开发
 - 节省开发成本
 - 减少业务不一致风险
- 相较于Rust、C++方案
 - JNI复杂度高（语言问题、工具链问题、.....）
 - 学习曲线陡峭
- TypeScript 和 Kotlin 共享许多编程概念，学习曲线平缓。
 - 实话：我就爱写Kotlin！

面向 Typescript 用户的 Kotlin 初印象

- 类型系统

- 编译期检查
- 类型推断
- 空安全

TypeScript

```
let number: number = 10;
let name: string | null = null;

function printName(name: string | null) {
    if (name) {
        console.log(name);
    } else {
        console.log('No name provided');
    }
}

printName(name);
```

Kotlin

```
val number: Int = 10
var name: String? = null

fun printName(name: String?) {
    if (name != null) {
        println(name)
    } else {
        println("No name provided")
    }
}

printName(name)
```

TypeScript 和 Kotlin 共享许多编程概念....

面向 Typescript 用户的 Kotlin 初印象

- 类型系统
 - 编译期检查
 - 类型推断
 - 空安全
- Lambda表达式

TypeScript

```
const numbers = [1, 2, 3, 4, 5];  
const doubled = numbers.map(number => number * 2);  
console.log(doubled); // 输出: [2, 4, 6, 8, 10]
```

Kotlin

```
val numbers = listOf(1, 2, 3, 4, 5)  
val doubled = numbers.map { it * 2 }  
println(doubled) // 输出: [2, 4, 6, 8, 10]
```

TypeScript 和 Kotlin 共享许多编程概念....

面向 Typescript 用户的 Kotlin 初印象

- 设计哲学
 - 保持兼容性与互操作性
 - JS \leftrightarrow TS
 - Java \leftrightarrow Kotlin
 - 强调简洁性和可读性
 - 实践导向

Typescript  Kotlin!

进入正题……

- KMP在RN项目中应用的典型场景有哪些
- 如何在RN项目中集成KMP

RN项目需要原生模块的三种典型场景

- 充分利用原生侧能力
 - 调用系统平台API (AudioManager, AVCaptureSession,)
 - 将耗时的计算任务交由原生代码处理, 提高性能。
- 处理异步资源
 - 通过原生模块更高效地处理异步资源, 比如网络、数据库等。
- 原生事件订阅
 - 从原生侧发送事件通知 JS引擎, 实现 UI 的响应更新。

示例项目： ReactNativeWithKMP

- Github Repo
- <https://github.com/aweffr/ReactNativeWithKMP>



Android界面



iOS界面

Step 0: 项目准备

- RN项目采用标准模板生成
- KMP项目下载自官方Wizard模板
 - <https://kmp.jetbrains.com/#newProject>

```
react-native init --version 0.73.8 RNWithKMPShowcase  
--package-name com.aweffr.rnwithkmp.showcase --skip-  
install
```

New Project


Template Gallery

Project Name

RNSharedModule

Project ID


com.aweffr.sharedmodule



Android

☒

With Compose Multiplatform UI framework based on Jetpack Compose



iOS

☒

UI Implementation

☐ Share UI (with Compose Multiplatform UI framework)

Beta

☒ Do not share UI (use only SwiftUI)

Step 1: RN 项目 gradle 改用 kts

settings.gradle

```
rootProject.name = 'com.aweffr.kotlinconf.showcase'
apply from: file("../node_modules/@react-native-community/cli-platform-
android/native_modules.gradle"); applyNativeModulesSettingsGradle(settings)
include ':app'
includeBuild('../node_modules/@react-native/gradle-plugin')
```

settings.gradle.kts

```
import groovy.lang.Closure

rootProject.name = "ReactNativeWithKMP"
apply(from = File("../node_modules/@react-native-community/cli-platform-
android/native_modules.gradle"))
val applyNativeModules: Closure<Any> = extra.get("applyNativeModulesSettingsGradle") as Closure<Any>
applyNativeModules(settings)
```

Step 1: RN 项目 gradle 改用 kts

android/
app/build.gradle

```
// ...  
  
apply from: file("../..../node_modules/@react-native-community/cli-platform-  
android/native_modules.gradle"); applyNativeModulesAppBuildGradle(project)
```

android/
app/build.gradle.kts

```
import groovy.lang.Closure  
  
// ...  
  
apply(from = file("../..../node_modules/@react-native-community/cli-platform-  
android/native_modules.gradle"))  
val applyNativeModulesAppBuildGradle: Closure<Any> = extra.get("applyNativeModulesAppBuildGradle") as  
Closure<Any>  
applyNativeModulesAppBuildGradle(project)
```

Step 1: RN 项目 gradle 改用 kts

```
// ...
dependencies {
    // classpath("com.android.tools.build:gradle")
    // classpath("org.jetbrains.kotlin:kotlin-gradle-plugin")
    classpath("com.facebook.react:react-native-gradle-plugin")
}

plugins {
    // this is necessary to avoid the plugins to be loaded multiple
    // times in each subproject's classloader
    alias(libs.plugins.androidApplication) apply false
    alias(libs.plugins.androidLibrary) apply false
    alias(libs.plugins.jetbrainsCompose) apply false
    alias(libs.plugins.kotlinMultiplatform) apply false
    alias(libs.plugins.kotlin.native.cocoapods) apply false
}

apply(plugin = "com.facebook.react.rootproject")
```

android/build.gradle

Step 2: KMP项目导入

- settings.build.gradle 注册 shared 模块
- kotlin版本对齐(1.9.24)
- shared模块使用cocapods导出iOS Framework

```
// ...
tasks.register("iosDebugPodBuild") {
    dependsOn("iosSimulatorArm64Binaries",
        "podPublishDebugXCFramework")
}
// ...
```

```
// ...
kotlin {
    androidTarget {
        compilations.all {
            kotlinOptions {
                jvmTarget = "17"
            }
        }
    }
}
// ...
```

```
// ...
cocoapods {
    summary = "Some description for the Shared Module"
    homepage = "Link to the Shared Module homepage"
    version = "0.1.1"
    ios.deploymentTarget = "12"
    framework {
        isStatic = true
        baseName = "shared"
        binaryOption("bundleId", "com.aweffr.shared")
        embedBitcode(BitcodeEmbeddingMode.BITCODE)
    }
}
// ...
```

Step 2: KMP项目导入

- Android
 - app/ 项目添加依赖
 - implementation(project(":shared"))
- iOS
 - 修改 ios/Podfile
 - 执行 pod install

```
# ...
use_react_native!(
  :path => config[:reactNativePath],
  :flipper_configuration => FlipperConfiguration.disabled,
  :hermes_enabled => false,
  :app_path => "#{Pod::Config.instance.installation_root}/.."
)

pod 'shared', :path =>
  '../android/shared/build/cocoapods/publish/debug/shared.podspec'
post_install do |installer|
  react_native_post_install(
    installer,
    config[:reactNativePath],
    :mac_catalyst_enabled => false
  )
end
# ...
```


运行演示环节

- Case 1:
 - 调用同步函数
- Case 2:
 - 调用协程函数 (Bonus: 观察两平台线程ID)
- Case 3:
 - 注册函数回调, 事件订阅模式 (Bonus: 观察两平台线程ID)

Some Naïve Tips

- Swift查找变量符号
 - 检查 Headers/ 目录产物
- Native 调用 suspend function 的主线程检查限制
 - 方案一: 切换到主线程调用
 - DispatchQueue.main.async
 - 方案二: 在编译过程中去掉限制
 - objcExportSuspendFunctionLaunchThreadRestriction=none
 - [官方文档](#)
- Suspend function 异常处理
 - 添加注释 @Throws(Throwable::class)

Further Reading

- [Multiplatform Coroutines tips and trick](#) (Medium 技术文章)
 - 覆盖了iOS端使用KMP协程的常见问题
- [reakt-native-toolkit](#) (Github Repo)
 - 通过ksp技术自动生成RN调用胶水层代码
- [Kotlin Multiplatform — React Native Bridge Modules](#) (Medium 技术文章)
 - 拆解并重新在KMP中实现RCT Marco, 将KMP直接暴露给JS侧调用

The End

ZouLe

aweifr@foxmail.com