MVI 范式在 JETPACK COMPOSE 上的应用

JAKE LIN - SENIOR MOBILE TECH LEAD @ REA GROUP
FEB 2022

架构与实现范式

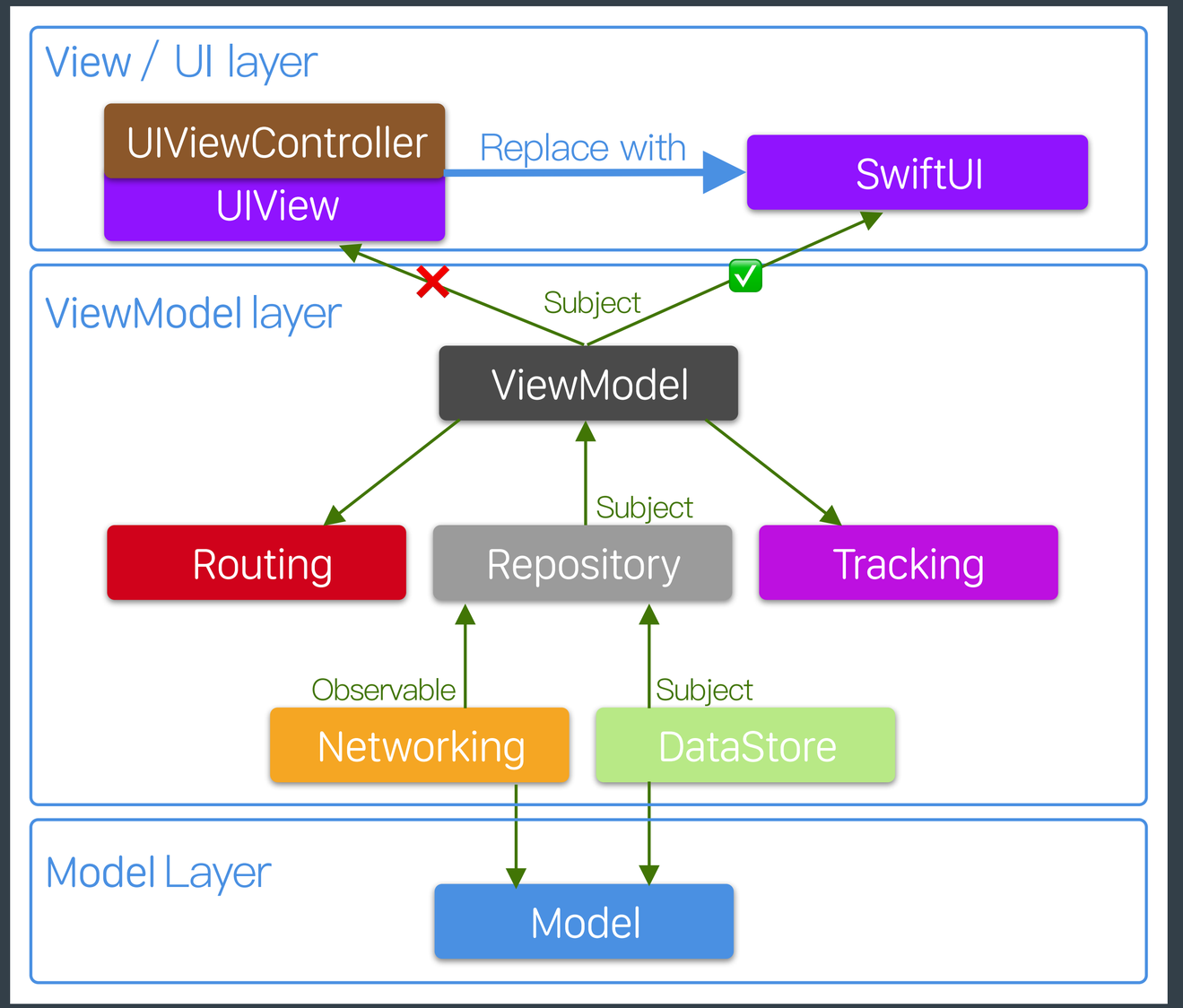
为什么要有架构与实现范式?

- 代码易于维护
- 方便代码重用
- 提高可扩展性
- 方便团队沟通

没缺点了吗?

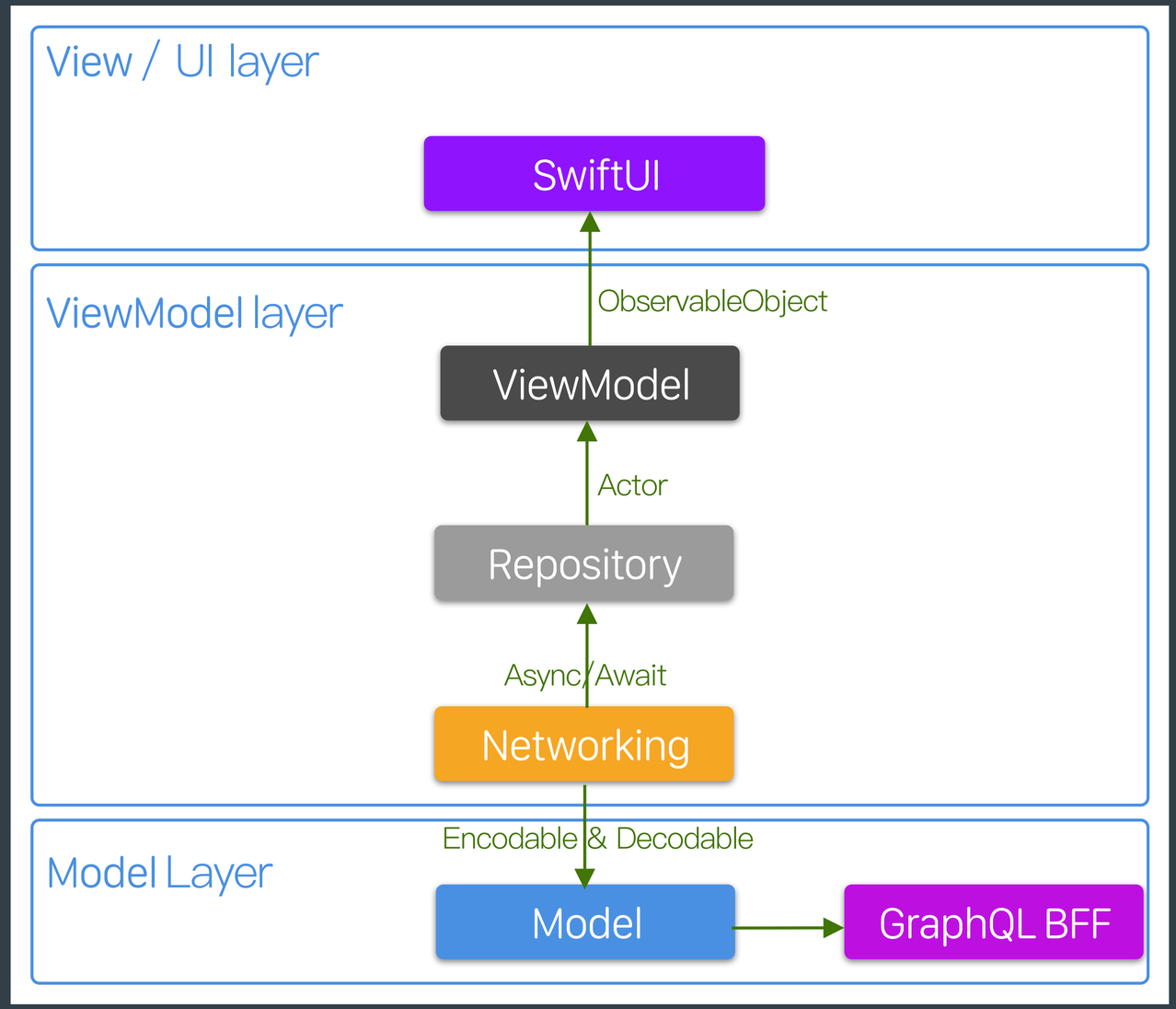
- 需要编写更多的代码，因为层多了，一个功能要写好几层
- 接手和学习成本提高，特别引人响应式编程后
- 代码变成 opinionated，只能按照预定的范式编写，相对缺乏灵活性

从 UIKit 无缝移植到 SwiftUI



https://github.com/lagoueduCol/iOS-linyongjian

MVVM 在 SwiftUI 上重新实现



https://github.com/JakeLin/Moments-SwiftUI

Android 流行范式

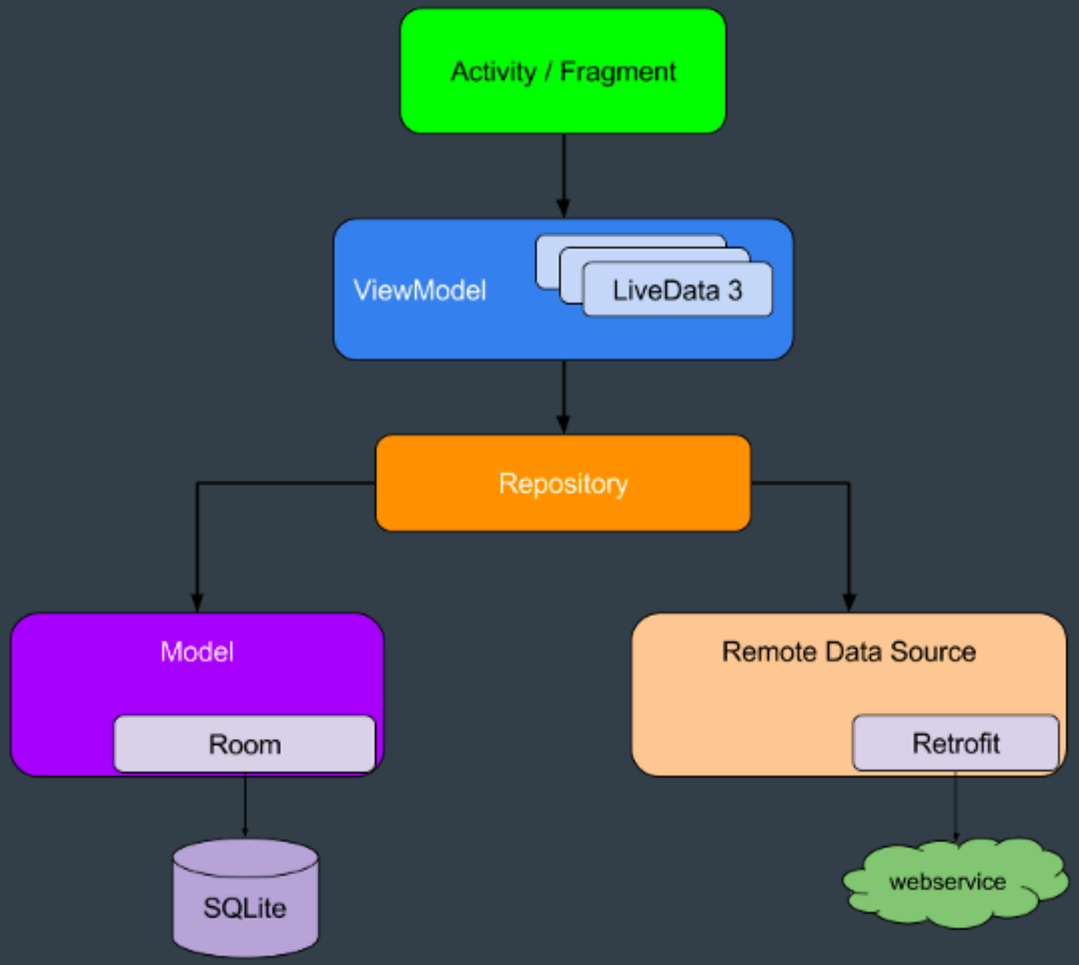
- MVC: Model View Controller
- MVP: Model View Presenter 🔥
- VIPER: View Interactor Presenter Entity Routing
- MVVM: Model View ViewModel 🔥
- CLEAN: Clean Architecture 🔥
- REDUX: ActionCreator-Action-Dispatcher-Middleware-Reducer-Store-Middleware-View
- PRNSAASPFRUICC: Production-Ready Native Single-Atom-State Purely Functional Reactive Composable UI Components
- 尼玛的，这么多，还继续发明新缩写…

这些范式到底要解决什么问题?

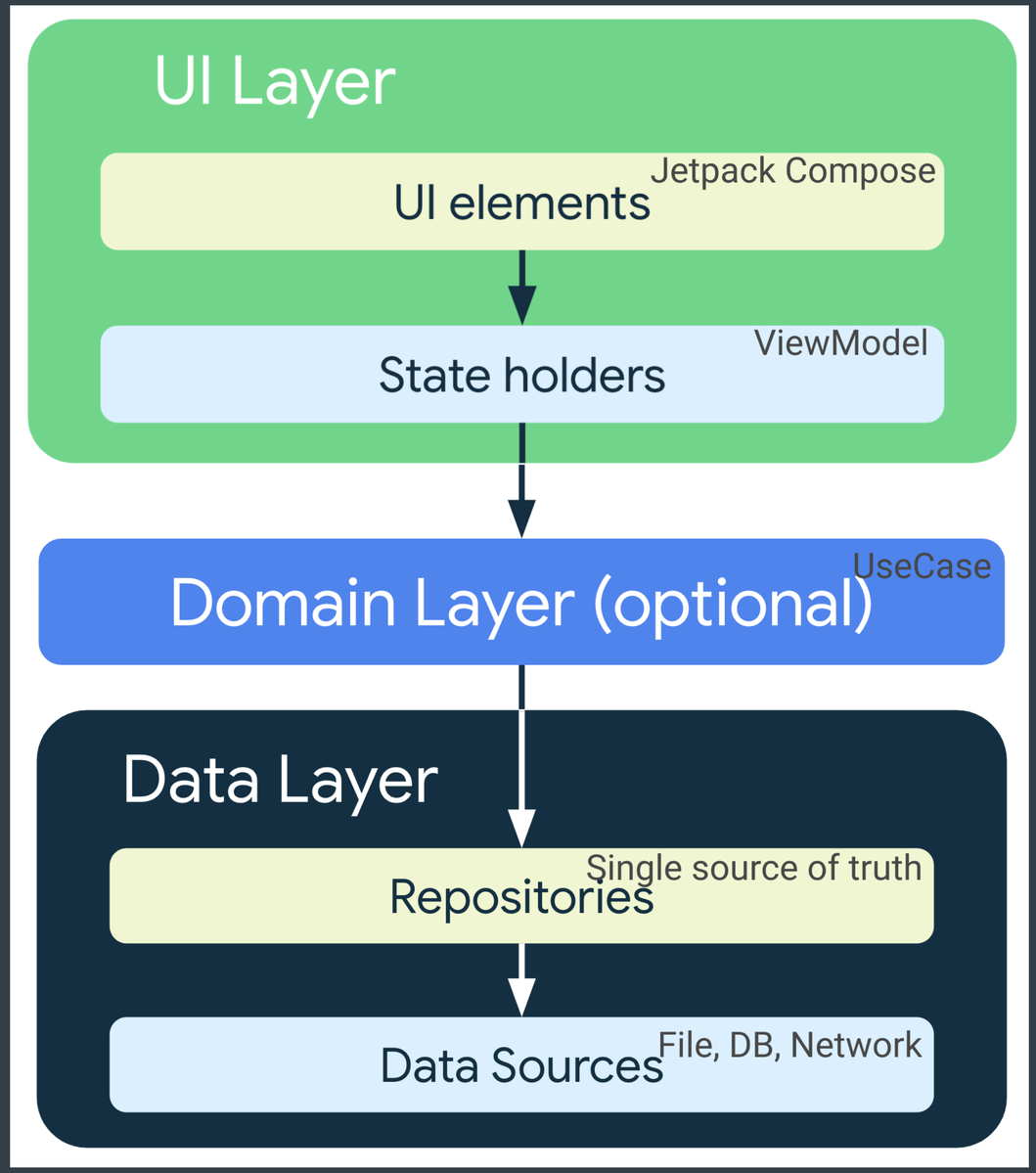
- 就业问题，保护原有职位并创造新就业
- 视图（View）与数据（Data）的分离 - SoC
- 不同功能模块之间的解耦 - 引入 DI

Google 推荐的范式 - MVVM

Android Architecture Components
LiveData + ViewModel



新的推荐范式


UI Layer
UI elements
Jetpack Compose
State holders
ViewModel
Domain Layer (optional)
UseCase
Data Layer
Repositories
Single source of truth
Data Sources
File, DB, Network

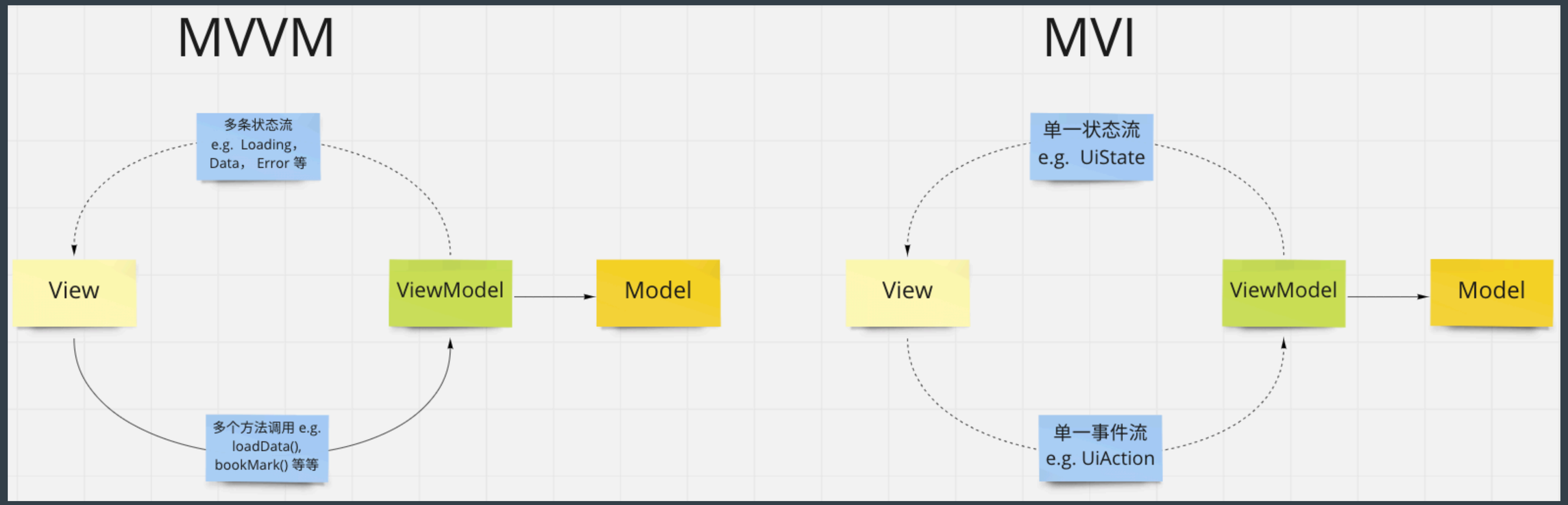
Jetpack Compose

```
@Composable
fun JetpackCompose() {
    Card {
        var expanded by remember { mutableStateOf(false) }
        Column(Modifier.clickable { expanded = !expanded }) {
            Image(painterResource(R.drawable.jetpack_compose))
            AnimatedVisibility(expanded) {
                Text(
                    text = "Jetpack Compose",
                    style = MaterialTheme.typography.h2,
                )
            }
        }
    }
}
```

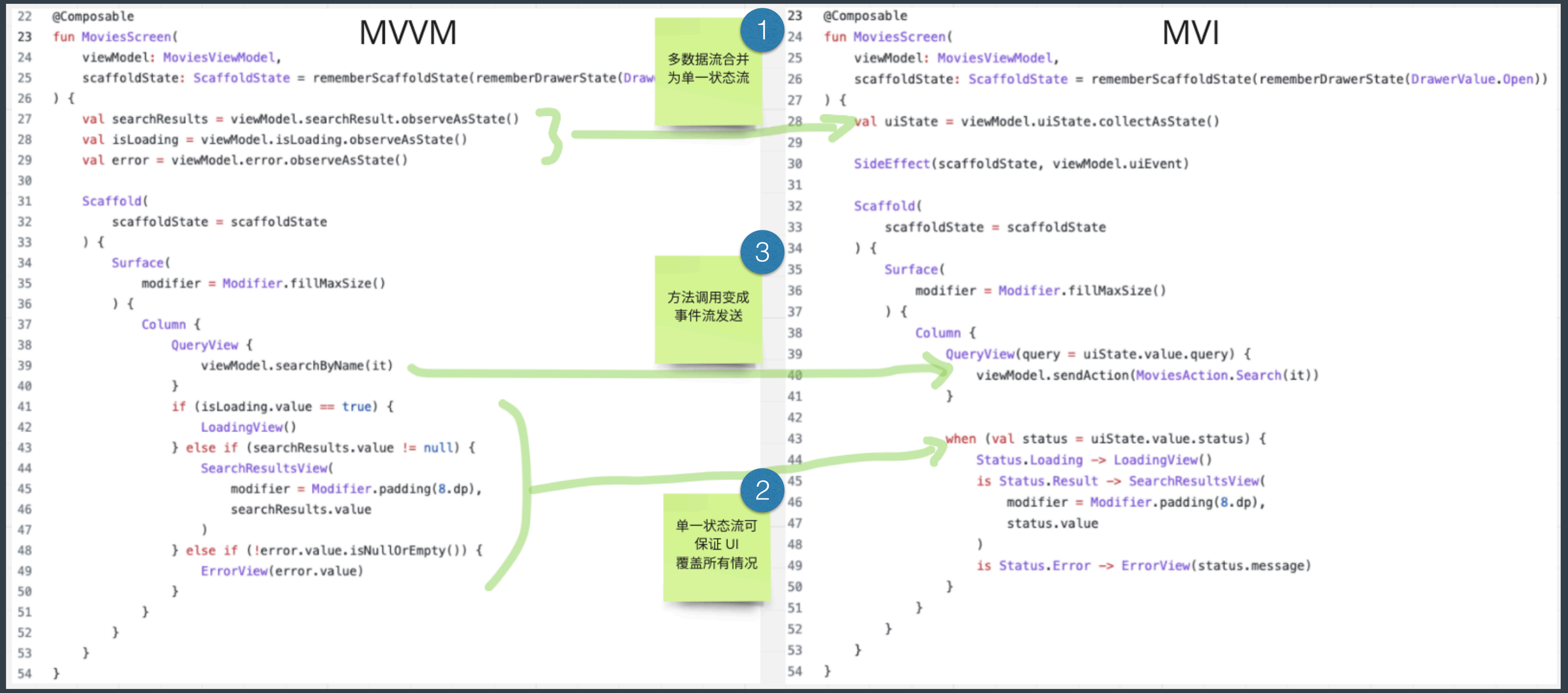
UI = renderingEngine.render(state)

MVI 范式

- MVI = Model-View-Intent
- MVI 强调了单向数据流(Unidirectional data flow)
- 也有人说 MVI = MVVM + Redux



MVVM 与 MVI 例子

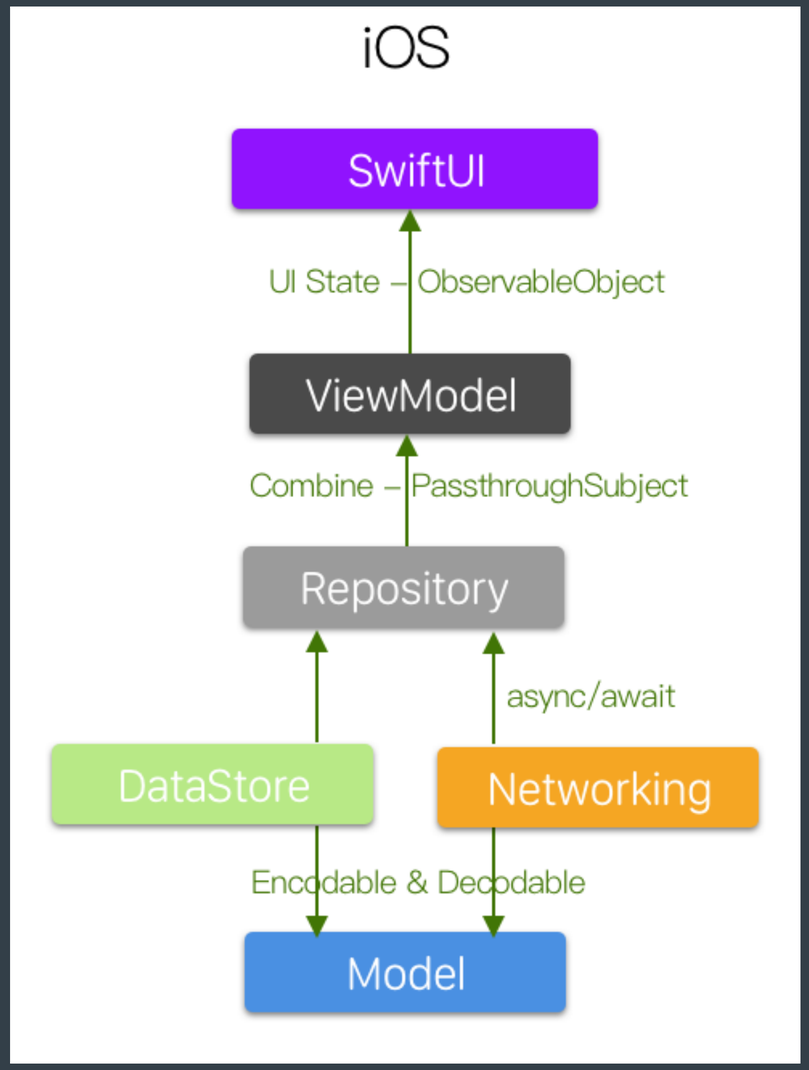


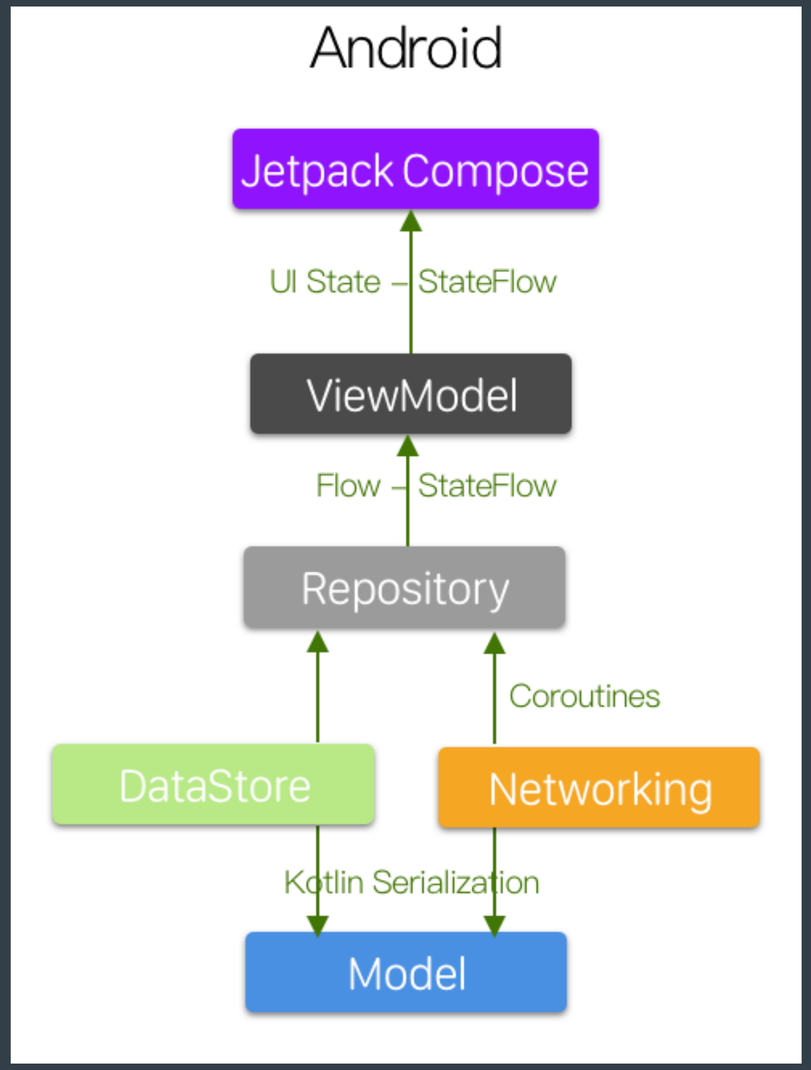

```kotlin
@Composable
fun MoviesScreen(
    viewModel: MoviesViewModel,
    scaffoldState: ScaffoldState = rememberScaffoldState(rememberDrawerState(Draw
) {
    val searchResults = viewModel.searchResult.observeAsState()
    val isLoading = viewModel.isLoading.observeAsState()
    val error = viewModel.error.observeAsState()

    Scaffold(
        scaffoldState = scaffoldState
    ) {
        Surface(
            modifier = Modifier.fillMaxSize()
        ) {
            Column {
                QueryView {
                    viewModel.searchByName(it)
                }
                if (isLoading.value == true) {
                    LoadingView()
                } else if (searchResults.value != null) {
                    SearchResultsView(
                        modifier = Modifier.padding(8.dp),
                        searchResults.value
                    )
                } else if (!error.value.isNullOrEmpty()) {
                    ErrorView(error.value)
                }
            }
        }
    }
}
```

```kotlin
@Composable
fun MoviesScreen(
    viewModel: MoviesViewModel,
    scaffoldState: ScaffoldState = rememberScaffoldState(rememberDrawerState(DrawerValue.Open))
) {
    val uiState = viewModel.uiState.collectAsState()

    SideEffect(scaffoldState, viewModel.uiEvent)

    Scaffold(
        scaffoldState = scaffoldState
    ) {
        Surface(
            modifier = Modifier.fillMaxSize()
        ) {
            Column {
                QueryView(query = uiState.value.query) {
                    viewModel.sendAction(MoviesAction.Search(it))
                }

                when (val status = uiState.value.status) {
                    Status.Loading -> LoadingView()
                    is Status.Result -> SearchResultsView(
                        modifier = Modifier.padding(8.dp),
                        status.value
                    )
                    is Status.Error -> ErrorView(status.message)
                }
            }
        }
    }
}
```

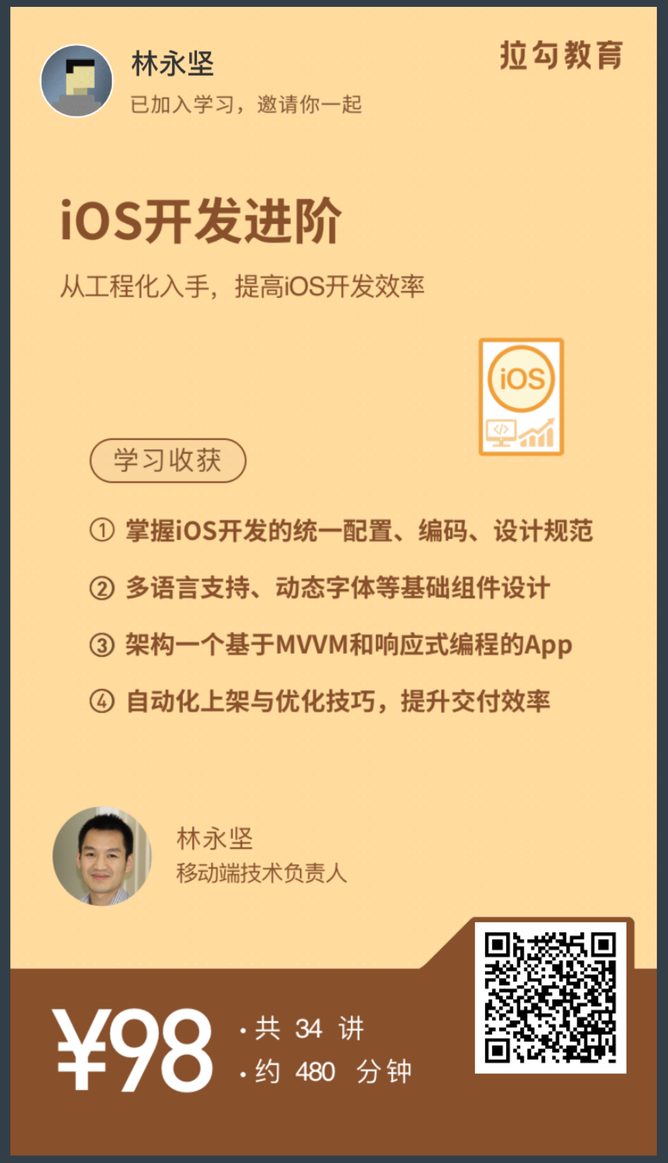
代码演示

iOS 与 Android 端对比

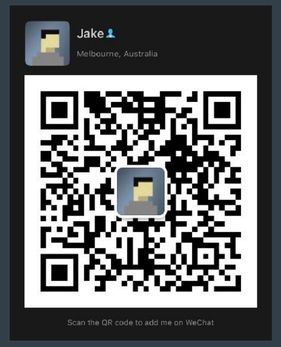




商务 aka 广告



扫码购买记住找我拿推广返现¥20



- Great Place to Work 评选澳洲最佳工作场所
- 招聘 iOS 和 Android 各种水平的开发者
- 为不同市场开发各种 App，包括 realestate.com.au, realcommercial.com.au 以及 Ignite app
- 同时开发 iOS， Android 和 BFF，多平台使用一致的架构与实现，并积极推广 SwiftUI 和 Jetpack Compose
- CI (Continuous Improvement) Fridays - 每周五可以不干项目的任务，干自己喜欢的事情，例如提高开发者体验，优化用户体验，研究新技术，提升系统健壮性等等
- 每年 12 天黑客松
- 夏天周五下午可以不上班，性别平等四个月带薪产假等等各种福利