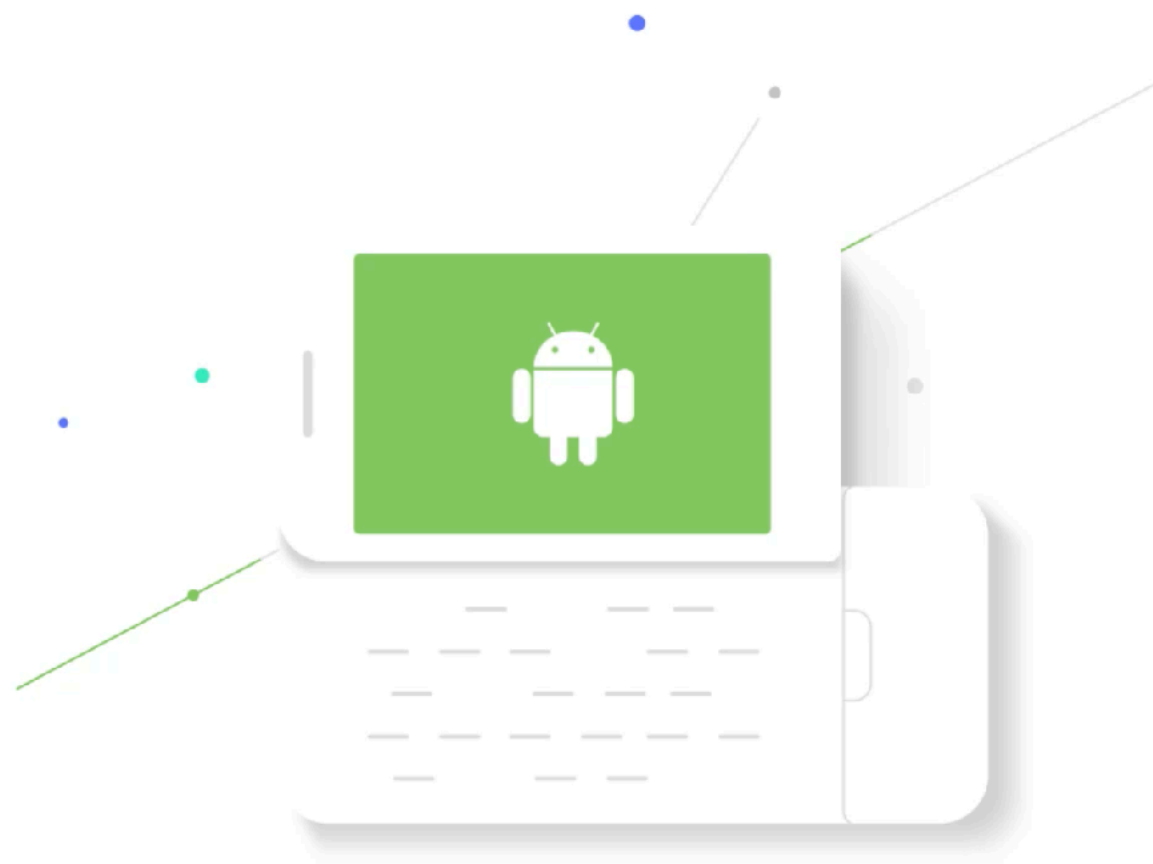


顺势而行，
让您的 Android 应用变得更加现代化

钟辉 Fred Chung

谷歌 Android 技术推广中国区负责人





T-Mobile G1



Q1 2010: 48 countries, 59 carriers, >20 devices



Q1 2010: 48 countries, 59 carriers, >20 devices



T-Mobile G1







3x



开发最佳实践



开发最佳实践

学习 Java 语言语法

把 UI 逻辑放进 Activity

用 Service 来实现后台任务

用 Receiver 来检测系统级事件

透过 Provider 来访问 SQLite



开发最佳实践

学习 Java 语言语法  新的官方语言！

把 UI 逻辑放进 Activity

用 Service 来实现后台任务

用 Receiver 来检测系统级事件

透过 Provider 来访问 SQLite



开发最佳实践

学习 Java 语言语法



新的官方语言！

~~把 UI 逻辑放进 Activity~~

太臃肿，难维护！

用 Service 来实现后台任务

用 Receiver 来检测系统级事件

透过 Provider 来访问 SQLite



开发最佳实践

学习 Java 语言语法



新的官方语言！

~~把 UI 逻辑放进 Activity~~

太臃肿，难维护！

~~用 Service 来实现后台任务~~

有时会太耗电！

用 Receiver 来检测系统级事件

透过 Provider 来访问 SQLite



开发最佳实践

学习 Java 语言语法



新的官方语言！

~~把 UI 逻辑放进 Activity~~

太臃肿，难维护！

~~用 Service 来实现后台任务~~

有时会太耗电！

~~用 Receiver 来检测系统级事件~~

对内存压力大，导致卡机！

透过 Provider 来访问 SQLite



开发最佳实践

学习 Java 语言语法



新的官方语言！

~~把 UI 逻辑放进 Activity~~

太臃肿，难维护！

~~用 Service 来实现后台任务~~

有时会太耗电！

~~用 Receiver 来检测系统级事件~~

对内存压力大，导致卡机！

~~透过 Provider 来访问 SQLite~~

太多样板代码



开发最佳实践

学习 Java 语言语法



新的官方语言！

~~把 UI 逻辑放进 Activity~~

太臃肿，难维护！

~~用 Service 来实现后台任务~~

有时会太耗电！

~~用 Receiver 来检测系统级事件~~

对内存压力大，导致卡机！

~~透过 Provider 来访问 SQLite~~

太多样板代码、样板代码

样板代码、样板代码、样板代码、

更多样板代码、更多样板代码、。。。



可能已过时的

开发最佳实践

学习 Java 语言语法



新的官方语言！

~~把 UI 逻辑放进 Activity~~

太臃肿，难维护！

~~用 Service 来实现后台任务~~

有时会太耗电！

~~用 Receiver 来检测系统级事件~~

对内存压力大，导致卡机！

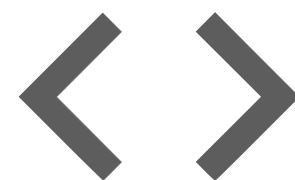
~~透过 Provider 来访问 SQLite~~

太多样板代码、样板代码

样板代码、样板代码、样板代码、

更多样板代码、更多样板代码、。。。





高效开发

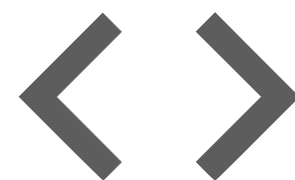


系统变更



工具改进





高效开发



系统变更



工具改进



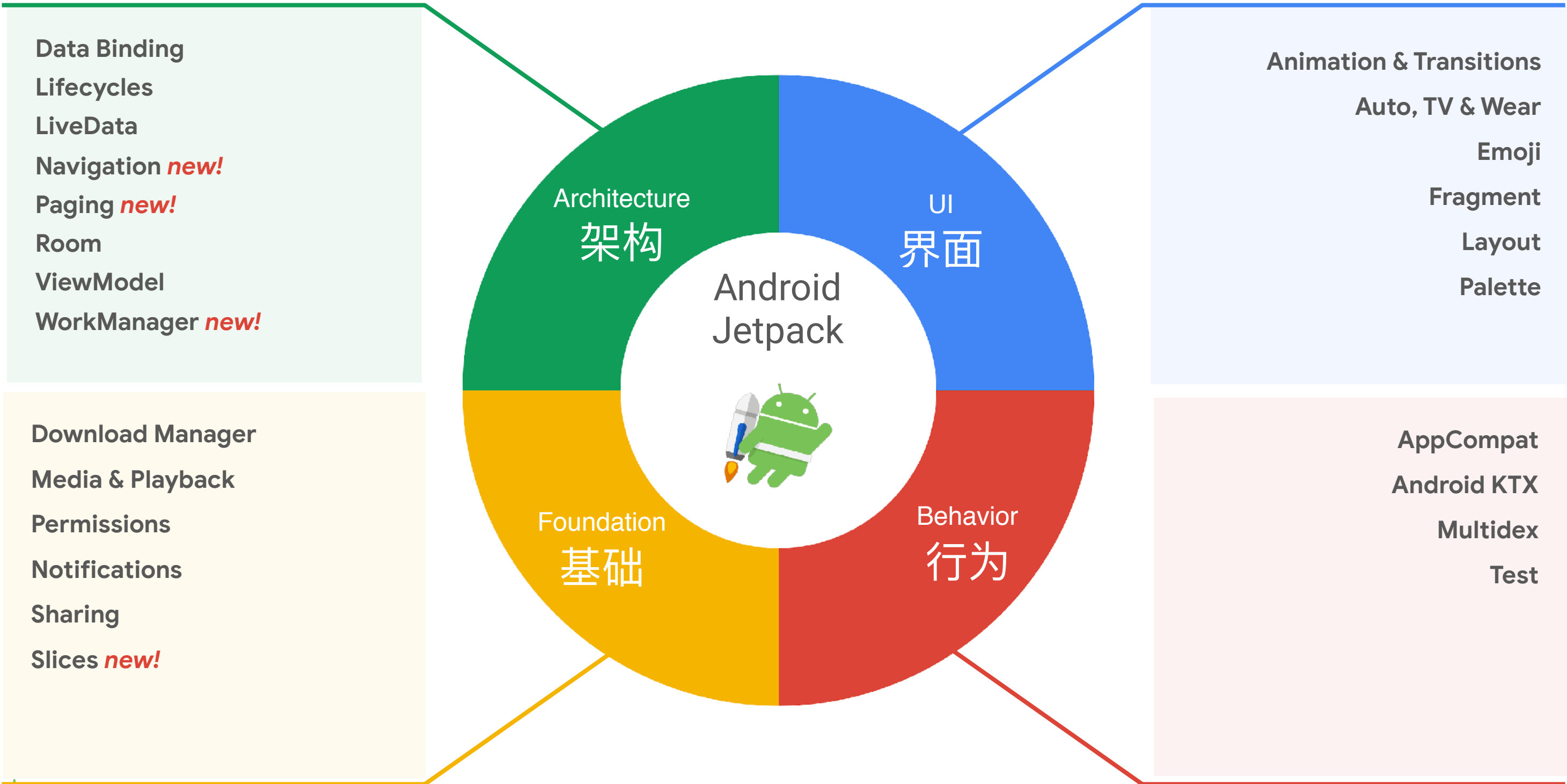
Android 开发有哪些坑？



Android Jetpack

为 Android 开发加速





Data Binding
Lifecycles
LiveData
Navigation *new!*
Paging *new!*
Room
ViewModel
WorkManager *new!*

Download Manager
Media & Playback
Permissions
Notifications
Sharing
Slices *new!*

Architecture
架构

UI
界面

Android
Jetpack



Foundation
基础

Behavior
行为

Animation & Transitions
Auto, TV & Wear
Emoji
Fragment
Layout
Palette

AppCompat
Android KTX
Multidex
Test



Data Binding
Lifecycles
LiveData
Navigation *new!*
Paging *new!*
Room
ViewModel
WorkManager *new!*

Download Manager
Media & Playback
Permissions
Notifications
Sharing
Slices *new!*

Architecture
架构

UI
界面

Android
Jetpack

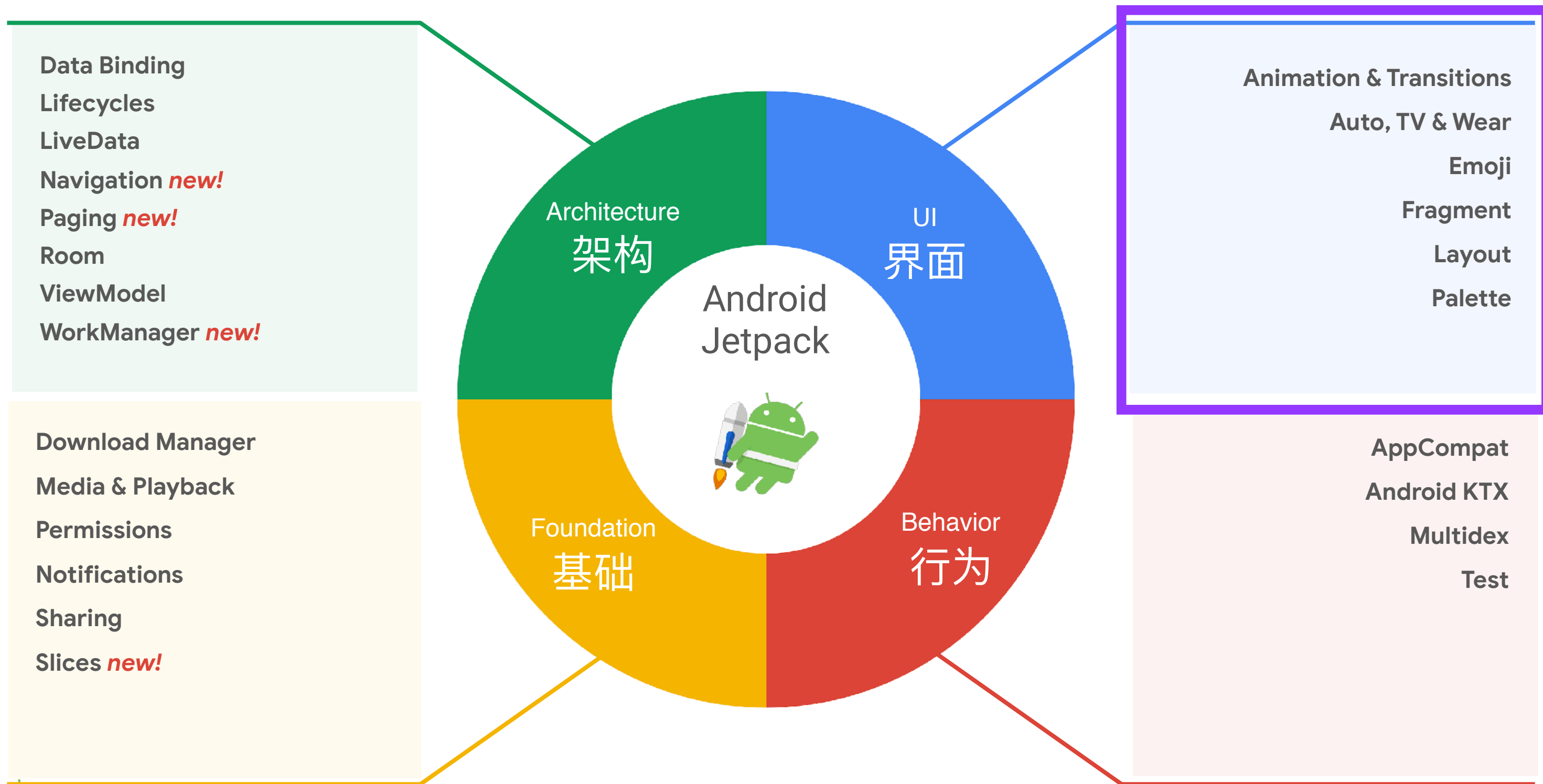


Foundation
基础

Behavior
行为

Animation & Transitions
Auto, TV & Wear
Emoji
Fragment
Layout
Palette

AppCompat
Android KTX
Multidex
Test



Data Binding
Lifecycles
LiveData
Navigation *new!*
Paging *new!*
Room
ViewModel
WorkManager *new!*

Download Manager
Media & Playback
Permissions
Notifications
Sharing
Slices *new!*

Architecture
架构

UI
界面

Android
Jetpack



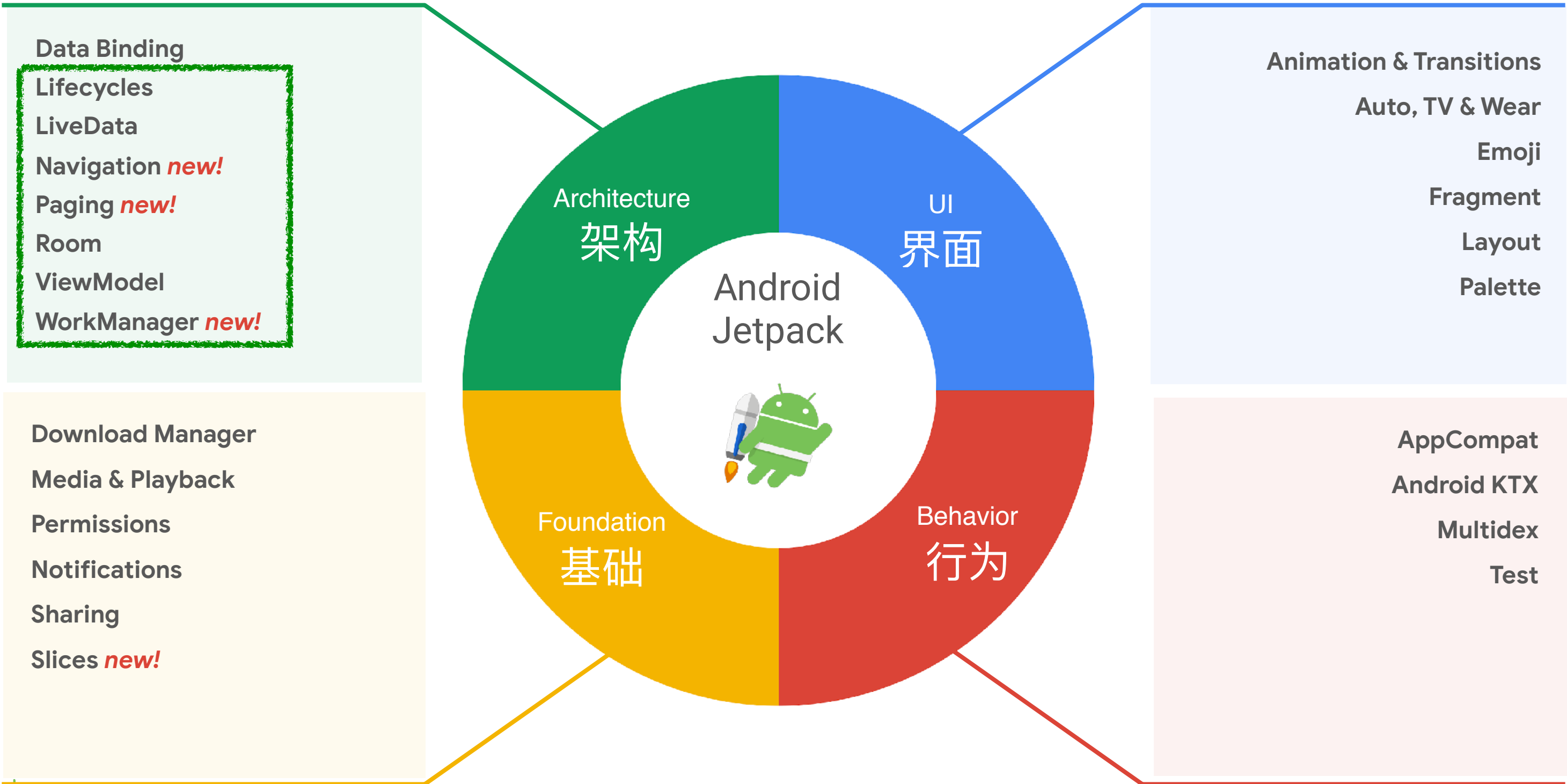
Foundation
基础

Behavior
行为

Animation & Transitions
Auto, TV & Wear
Emoji
Fragment
Layout
Palette

AppCompat
Android KTX
Multidex
Test





包名重构

```
android.support.v4.*  
android.support.v7.*  
// 其他...
```



包名重构

`androidx.*`



包名重构

`androidx.*`

Gradle 支持

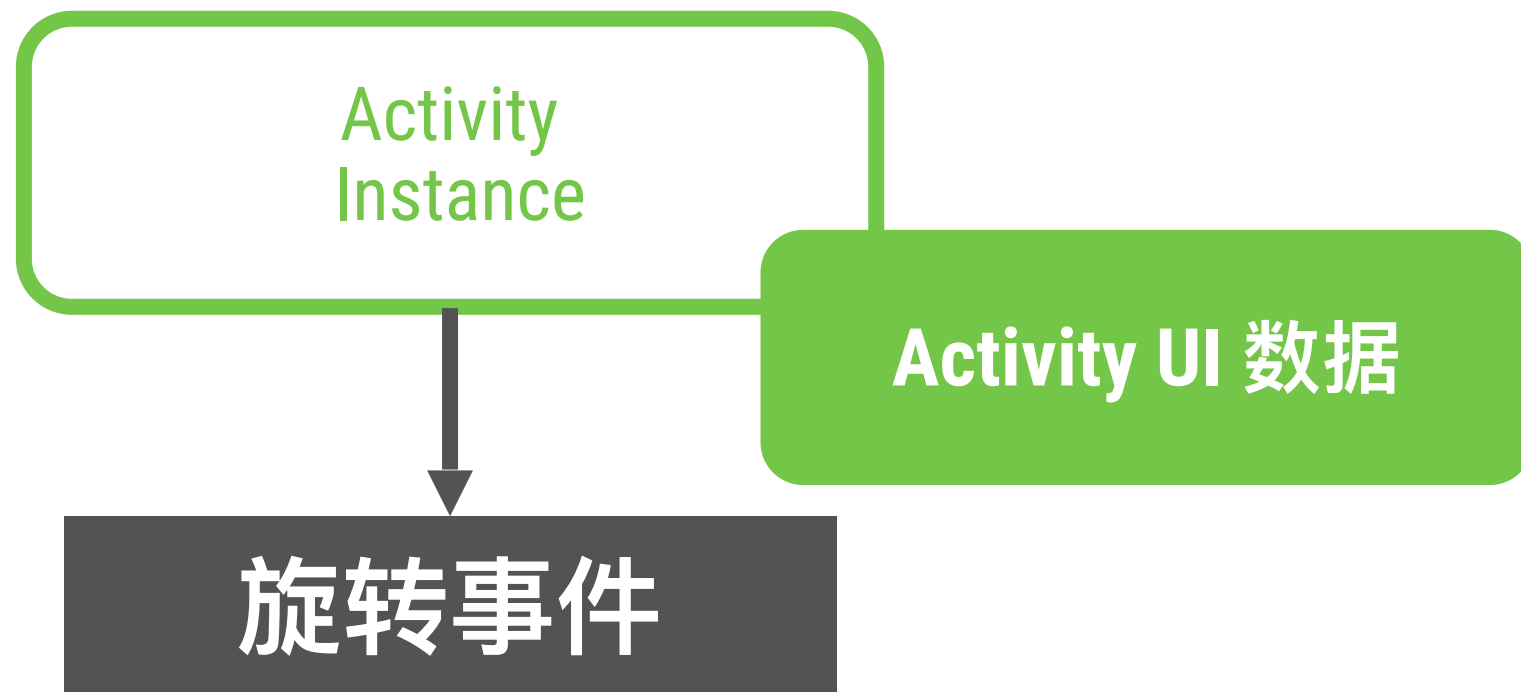
`android.useAndroidX = true`

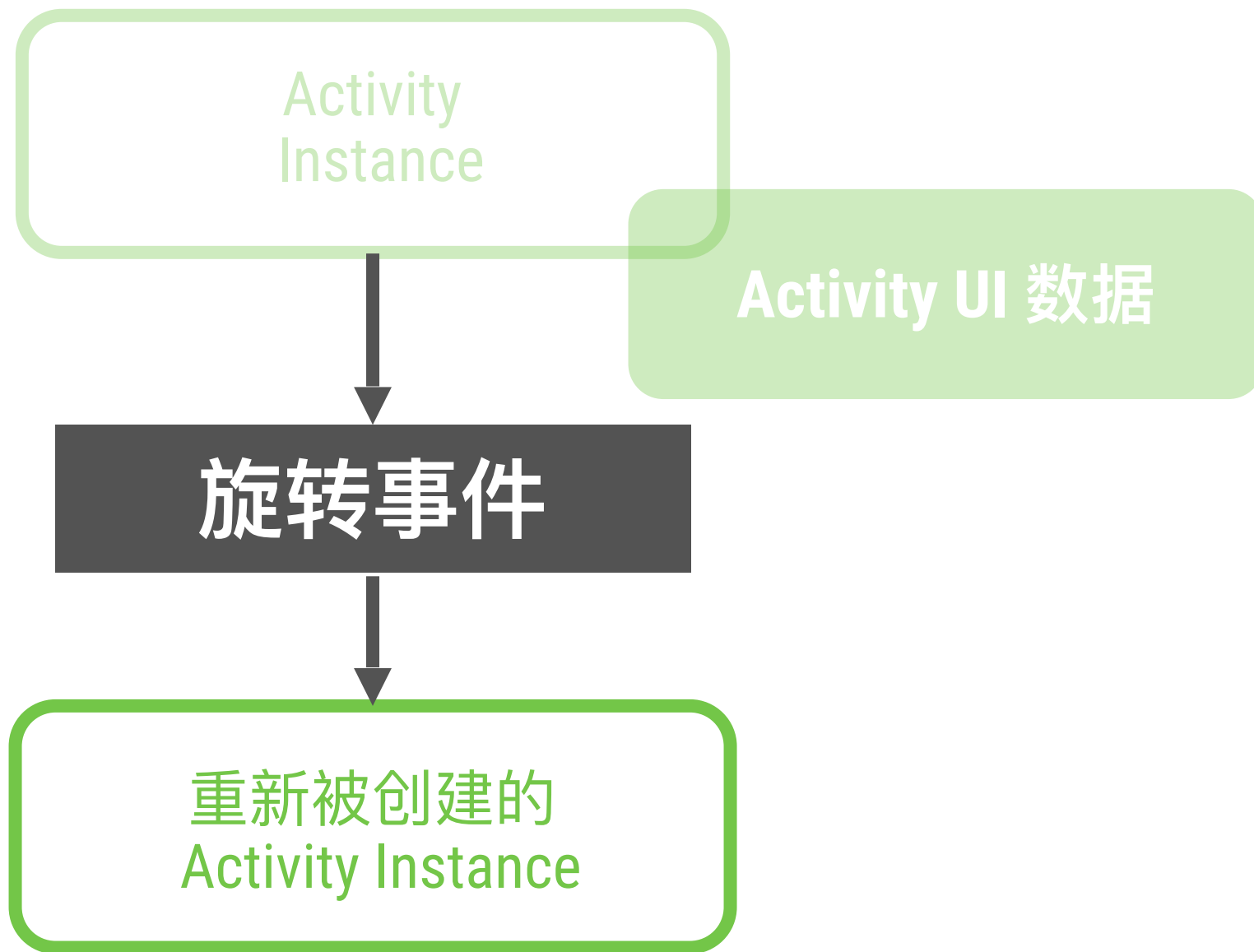


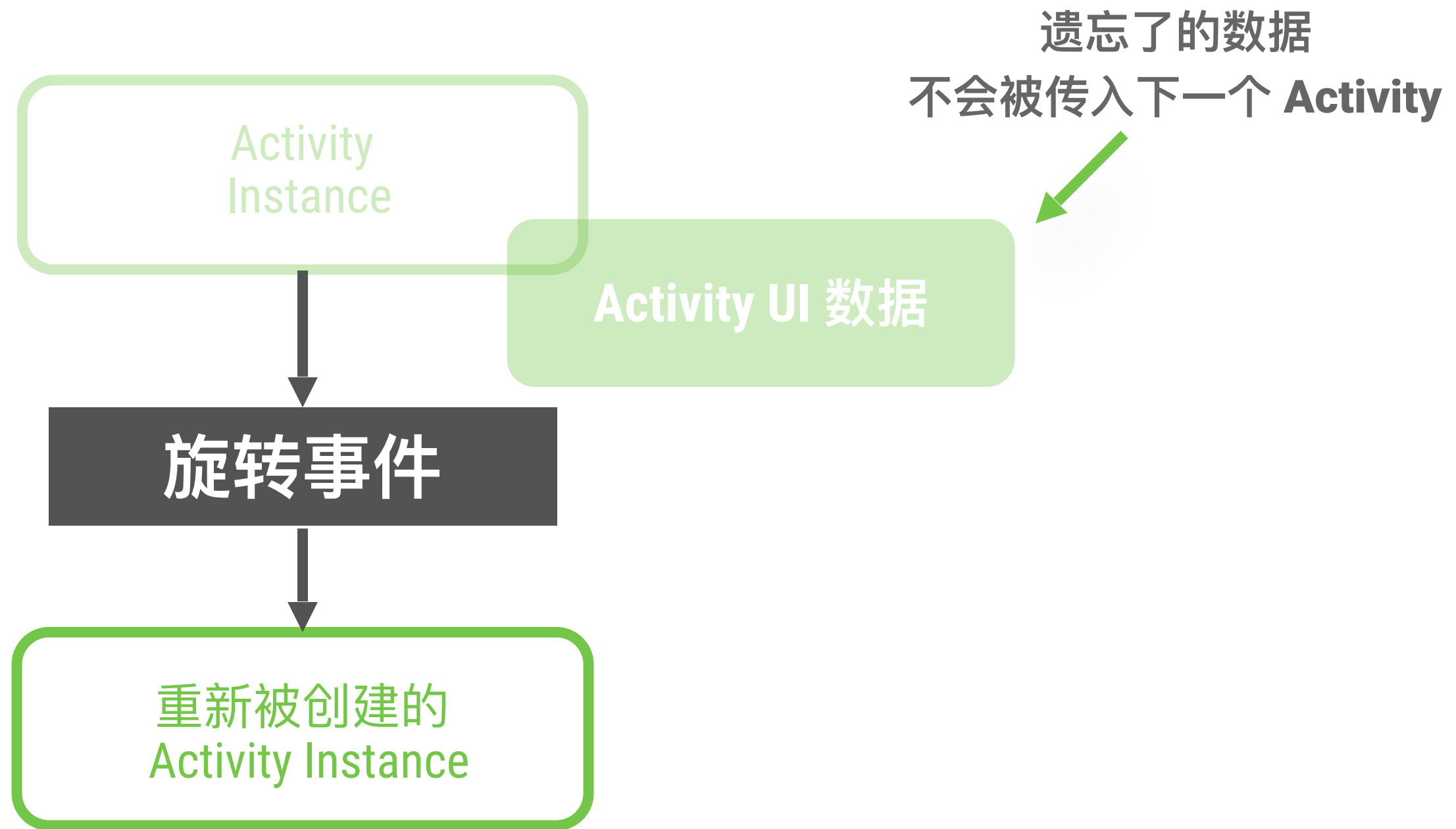
Activity
Instance

Activity UI 数据



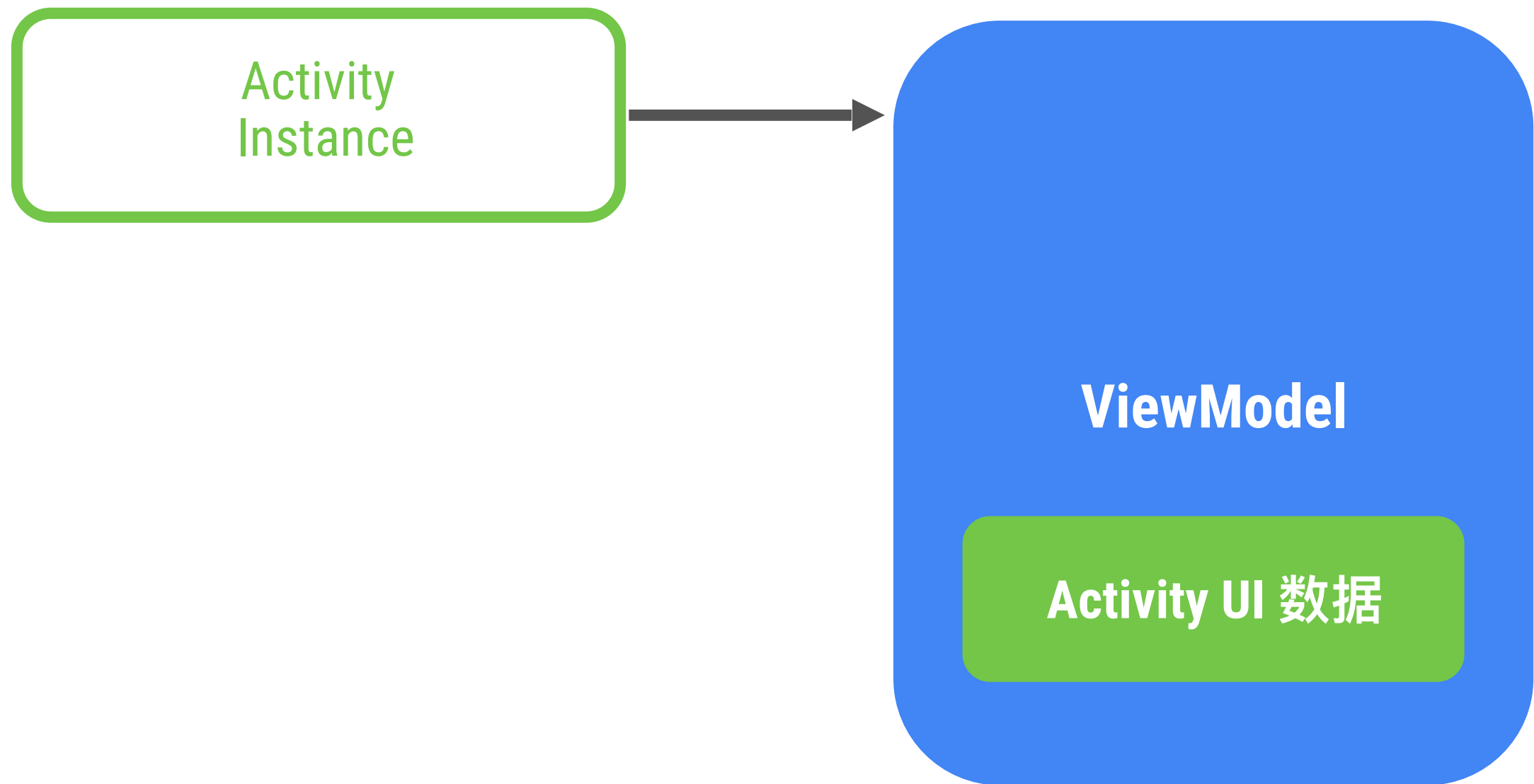


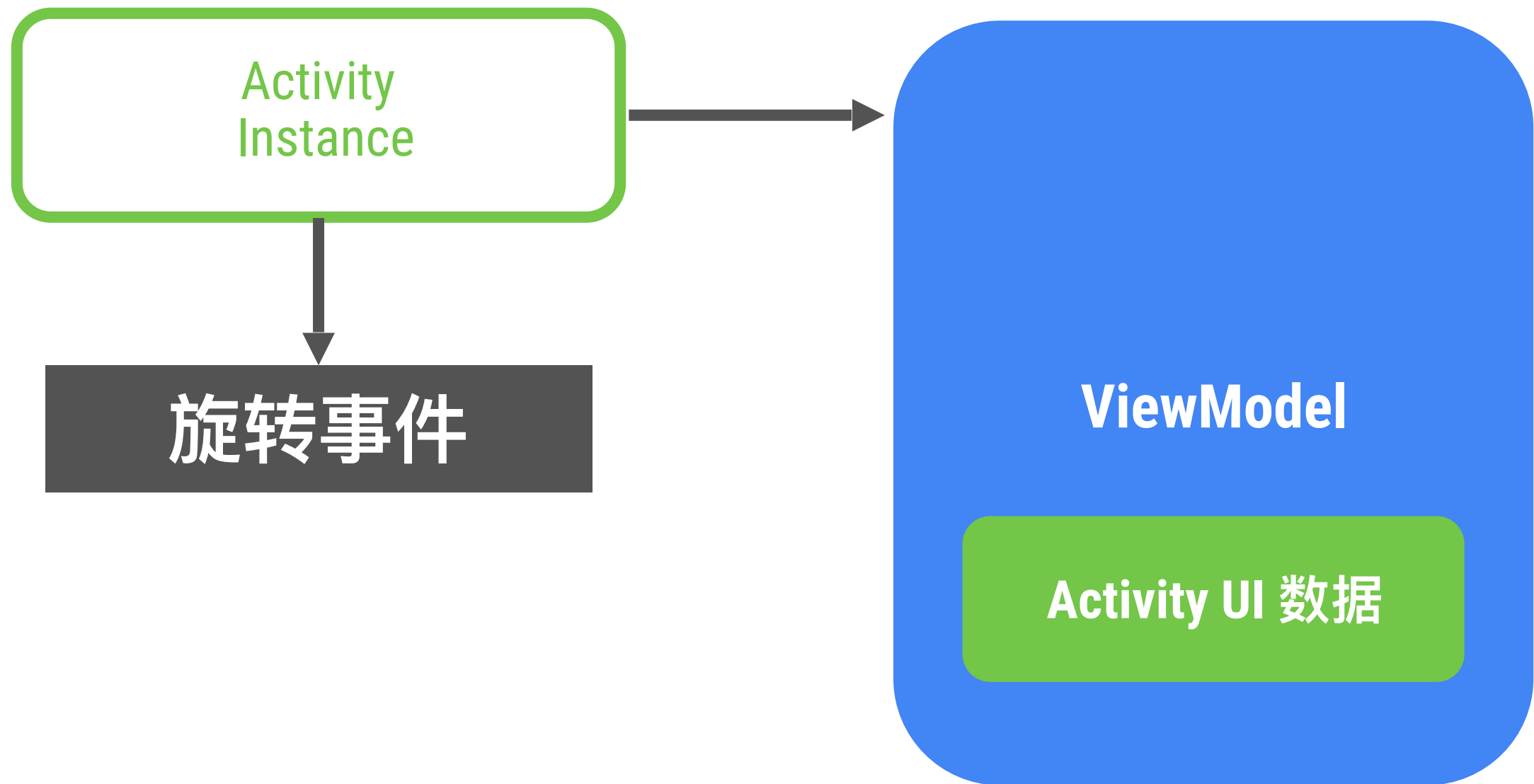


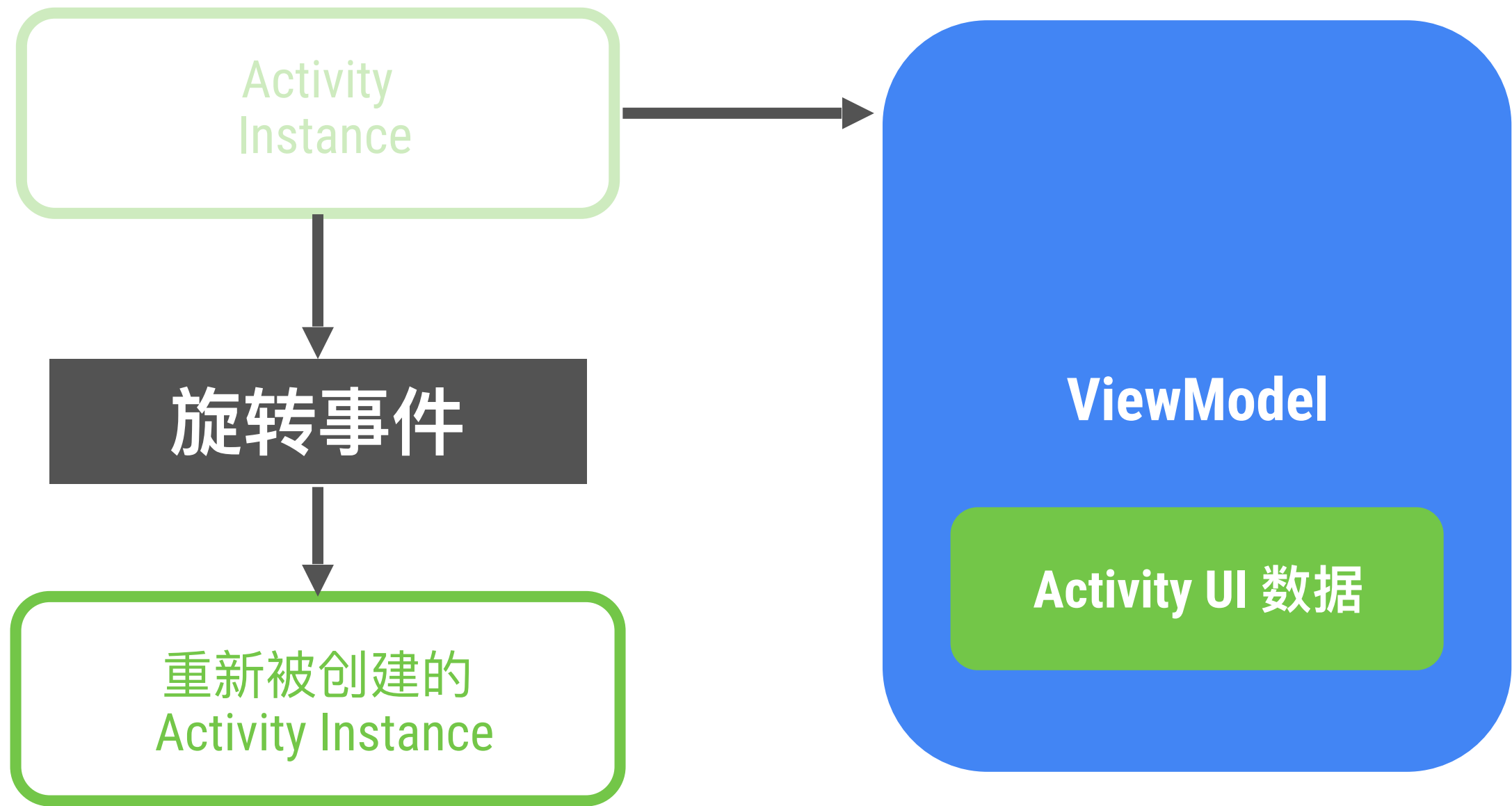


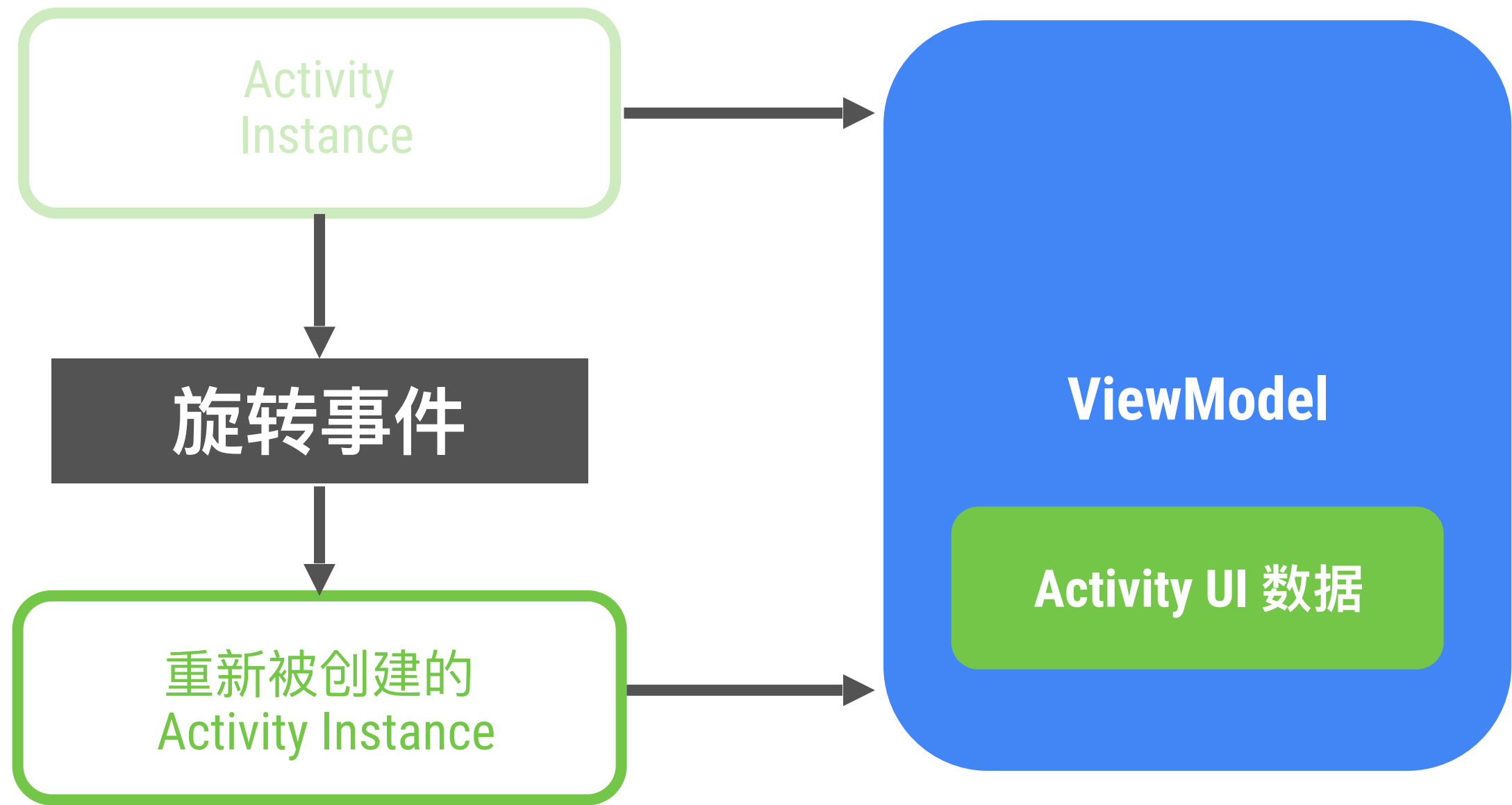
Activity
Instance









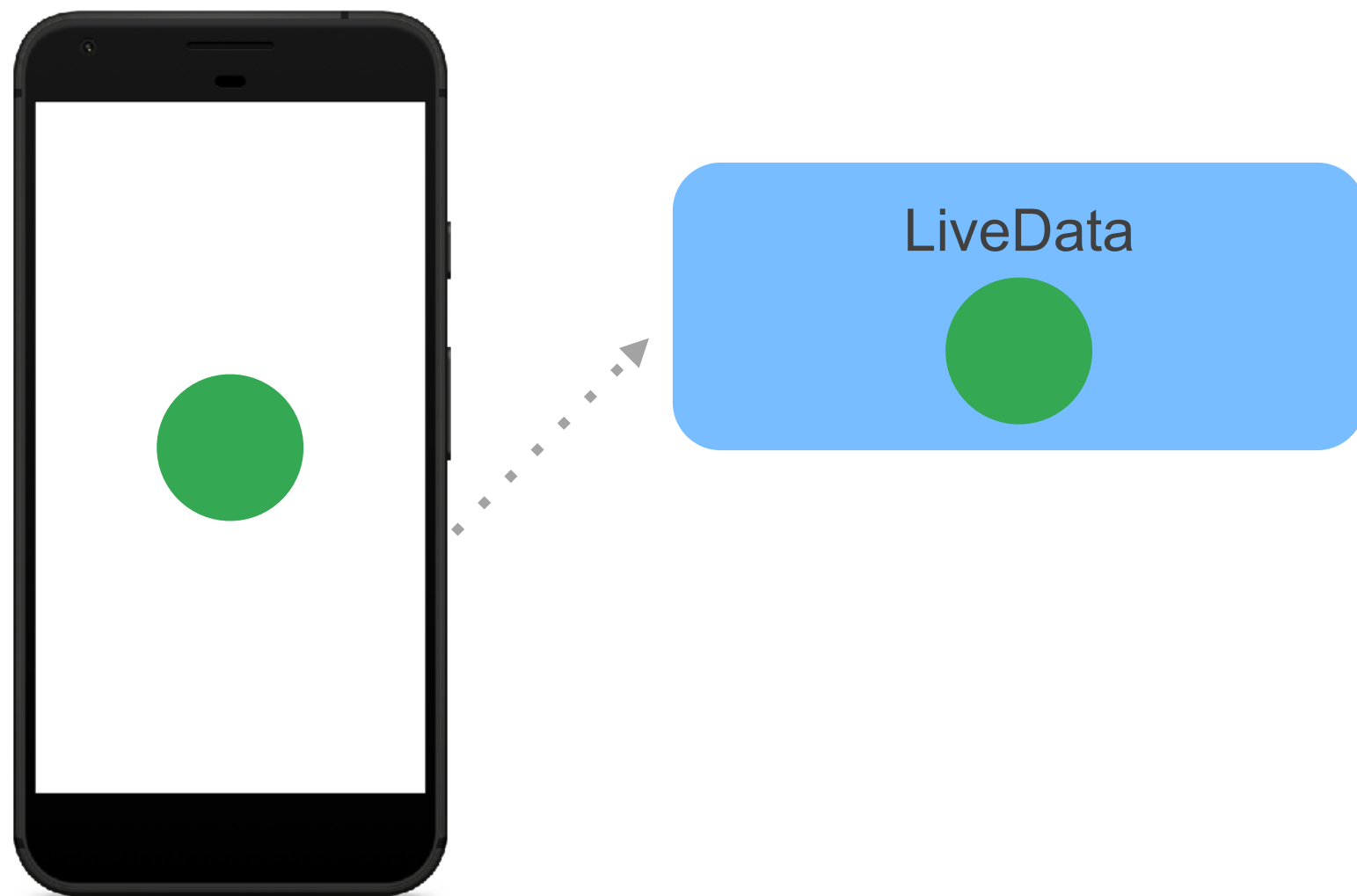


LiveData

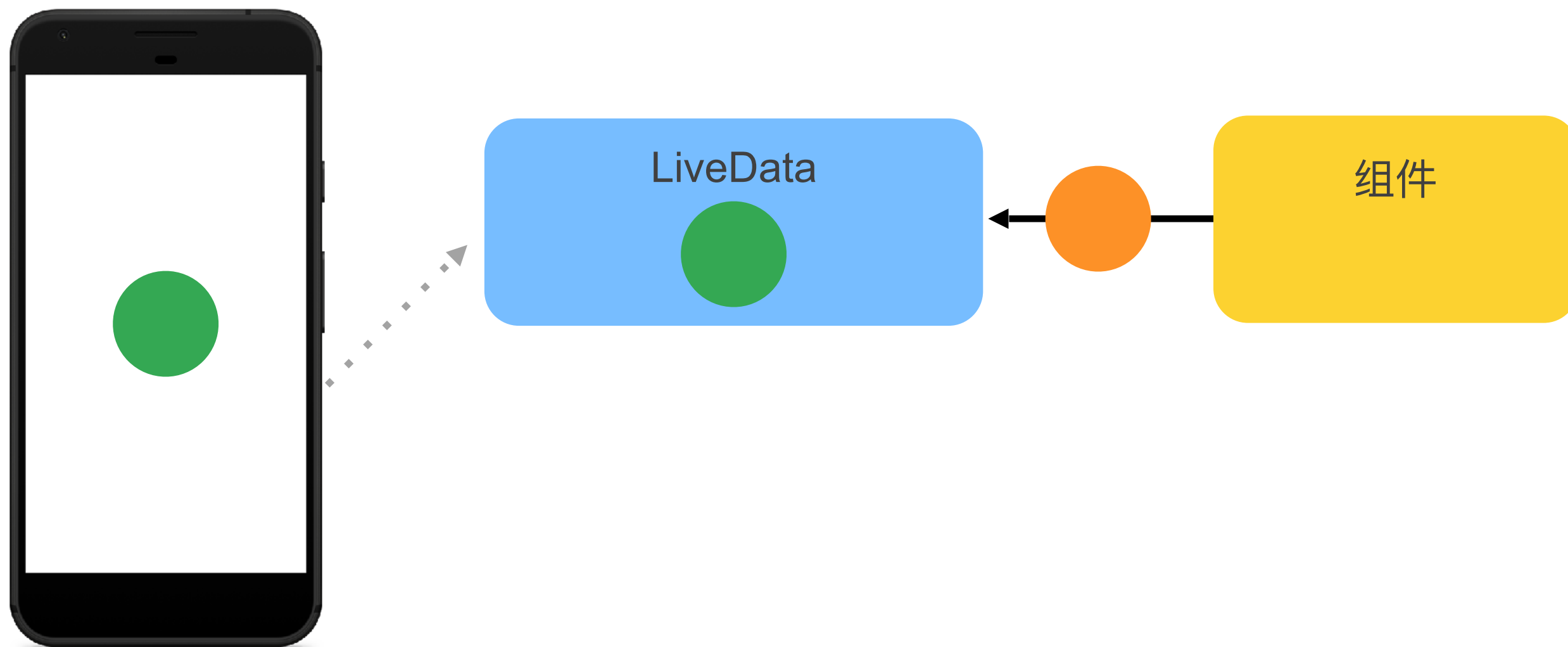
具有生命周期感知能力的可观察数据持有类



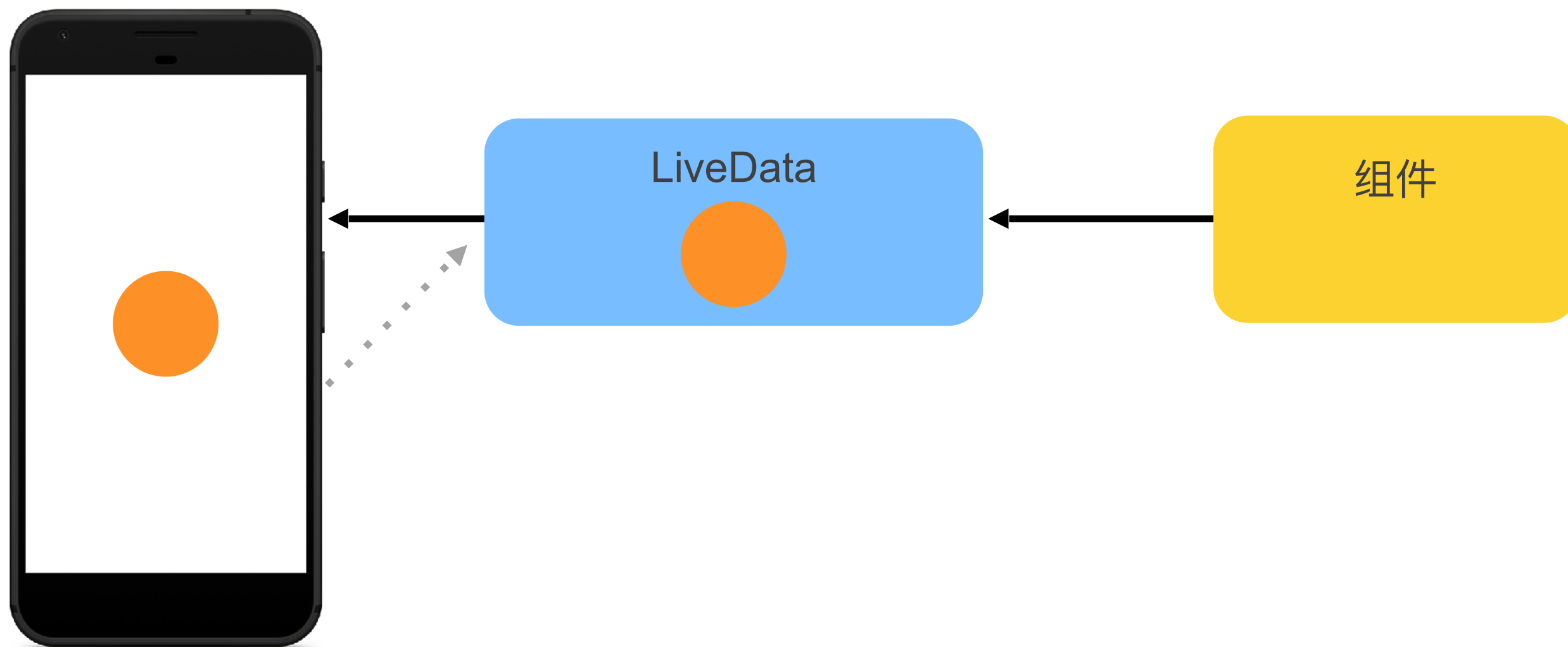
LiveData 实例



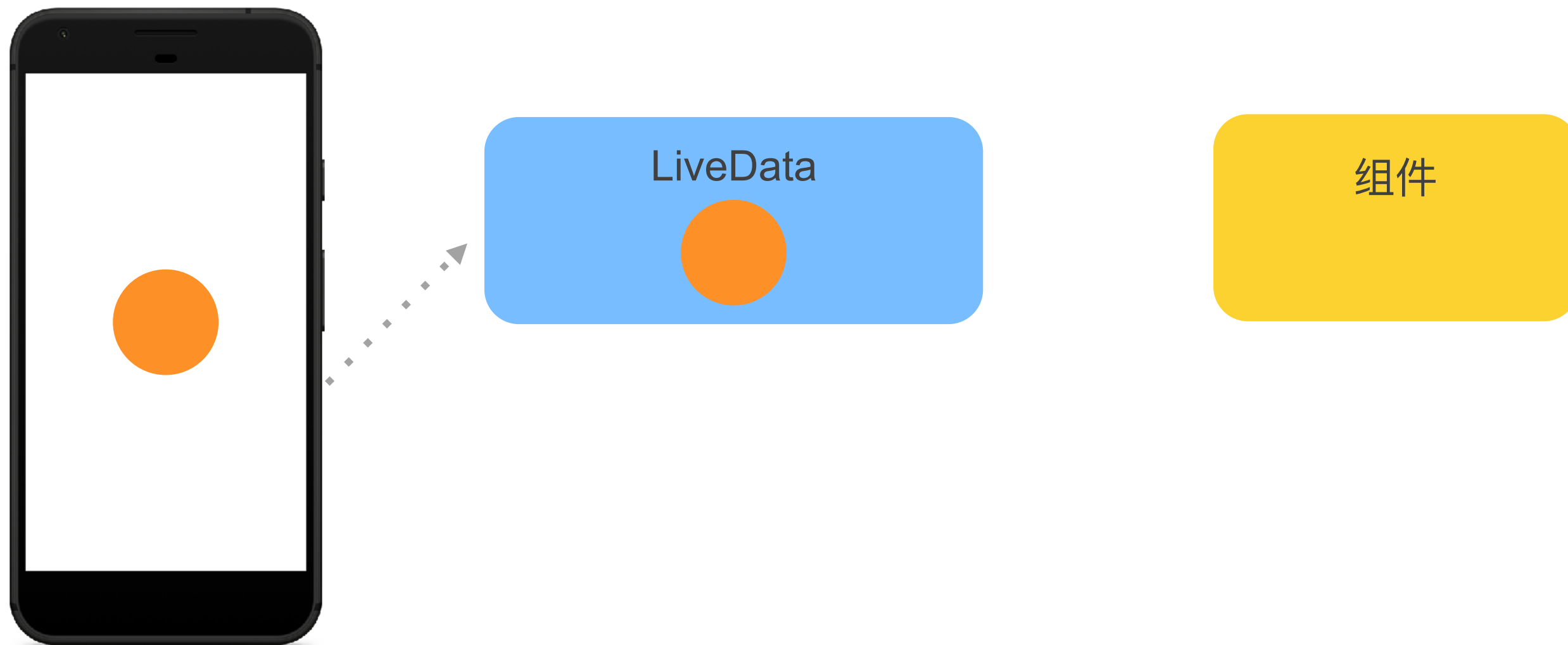
LiveData 实例



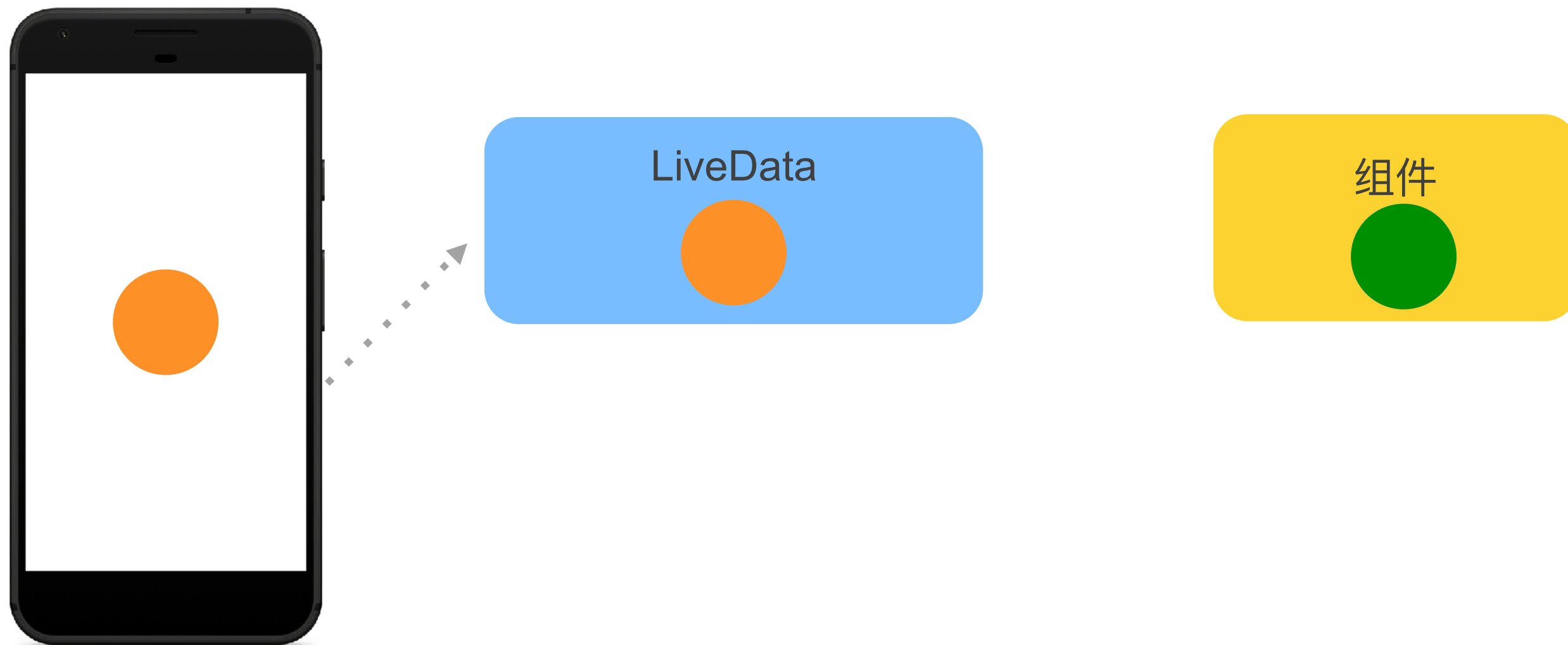
LiveData 实例



LiveData 实例



LiveData 实例



```
class UserProfileViewModel: ViewModel() {  
    //...  
}
```



```
class UserProfileViewModel: ViewModel() {  
    // _user and user are for proper  
    // encapsulation  
    private val _user = MutableLiveData<User>()  
  
    // Publicly exposed LiveData, not mutable  
    val user : LiveData<User>  
        get() = _user  
}
```



```
class UserProfileViewModel: ViewModel() {  
    // _user and user are for proper  
    // encapsulation  
    private val _user = MutableLiveData<User>()  
  
    // Publicly exposed LiveData, not mutable  
    val user : LiveData<User>  
        get() = _user  
}
```



```
override fun onCreate(savedInstanceState: Bundle?) {  
    // 拿取 ViewModel; user 变量是 LiveData  
    userViewModel.user.observe(this,  
        Observer {  
            user -> userNameTextView.text = user?.name  
        }  
    )  
}
```




```
override fun onCreate(savedInstanceState: Bundle?) {  
    // 拿取 ViewModel; user 变量是 LiveData  
    userViewModel.user.observe(this,  
        Observer {  
            user -> userNameTextView.text = user?.name  
        }  
    )  
}
```



```
override fun onCreate(savedInstanceState: Bundle?) {  
    // 拿取 ViewModel; user 变量是 LiveData  
    userViewModel.user.observe(this,  
        Observer {  
            user -> userNameTextView.text = user?.name  
        }  
    )  
}
```



```
override fun onCreate(savedInstanceState: Bundle?) {  
    // 拿取 ViewModel; user 变量是 LiveData  
    userViewModel.user.observe(this,  
        Observer {  
            user -> userNameTextView.text = user?.name  
        }  
    )  
}
```

```
// 与此同时...在另一组件  
user.setValue(newUser) // UI 线程  
user.postValue(newUser) // 后台线程
```



```
override fun onCreate(savedInstanceState: Bundle?) {  
    // 拿取 ViewModel; user 变量是 LiveData  
    userViewModel.user.observe(this,  
        Observer {  
            user -> userNameTextView.text = user?.name  
        }  
    )  
}
```

// 与此同时...在另一组件



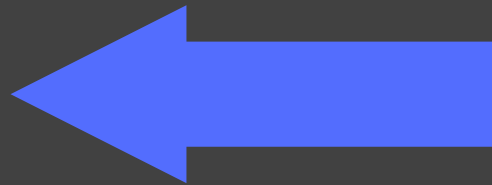
user.setValue(newUser) // UI 线程

user.postValue(newUser) // 后台线程



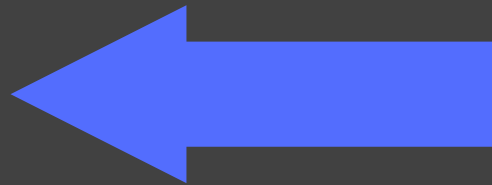
```
override fun onCreate(savedInstanceState: Bundle?) {  
    // 拿取 ViewModel; user 变量是 LiveData  
    userViewModel.user.observe(this,  
        Observer {  
            user -> userNameTextView.text = user?.name  
        }  
    )  
}
```

```
// 与此同时...在另一组件  
user.setValue(newUser) // UI 线程  
user.postValue(newUser) // 后台线程
```

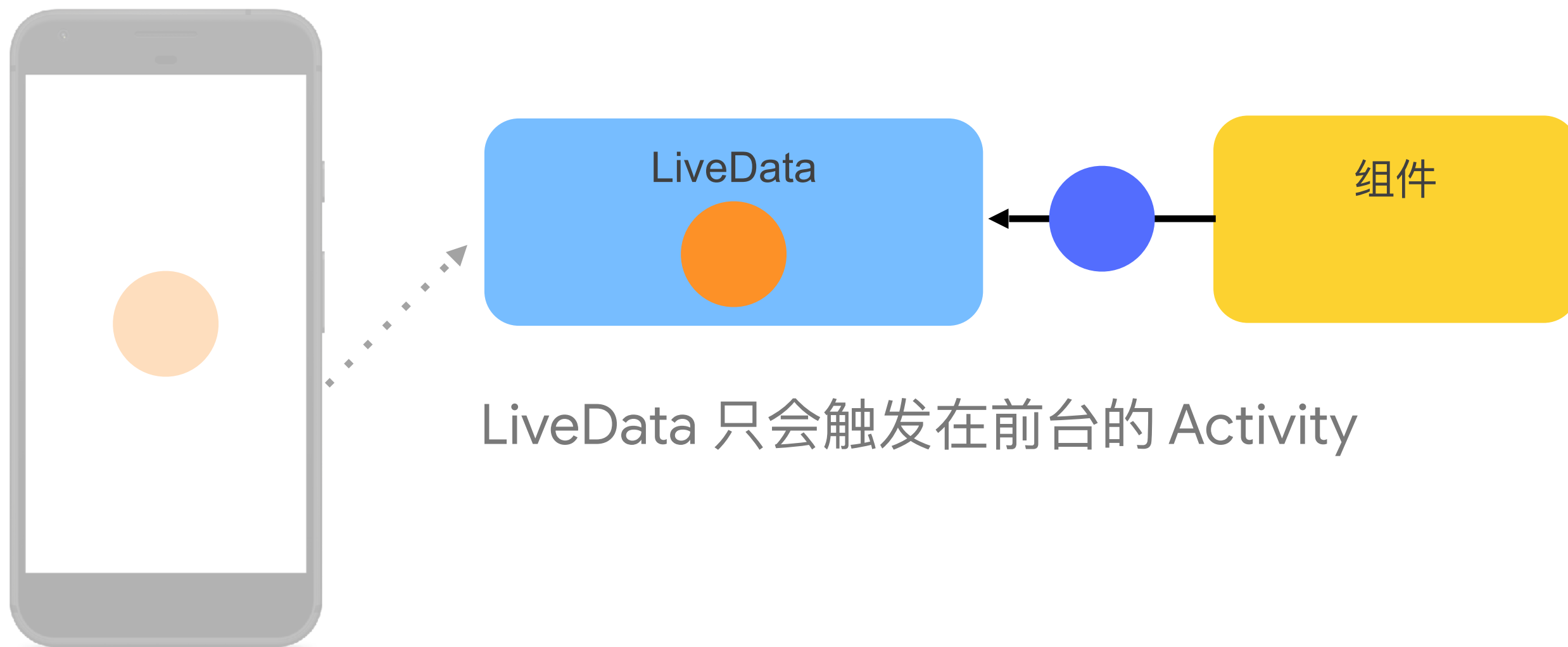


```
override fun onCreate(savedInstanceState: Bundle?) {  
    // 拿取 ViewModel; user 变量是 LiveData  
    userViewModel.user.observe(this,  
                                Observer {  
                                    user -> userNameTextView.text = user?.name  
                                })  
}
```

```
// 与此同时...在另一组件  
user.setValue(newUser) // UI 线程  
user.postValue(newUser) // 后台线程
```



LiveData 实例



LiveData 只会触发在前台的 Activity





Android Studio 3.1+

ViewModel 和 LiveData
支持 DataBinding 集成



WorkManager

预览版



WorkManager 预览版

执行后台任务的集成 API



WorkManager 预览版

执行后台任务的集成 API



WorkManager^{预览版}

执行后台任务的集成 API

可定义任务的执行约束条件 (Constraints)

如：正在充电、已连上 Wi-Fi、当某 Content Uri 有改变等



WorkManager^{预览版}

执行后台任务的集成 API

可定义任务的执行约束条件 (Constraints)

如：正在充电、已连上 Wi-Fi、当某 Content Uri 有改变等



WorkManager^{预览版}

执行后台任务的集成 API

可定义任务的执行约束条件 (Constraints)

如：正在充电、已连上 Wi-Fi、当某 Content Uri 有改变等

往后兼容到 API 14！



WorkManager^{预览版}

执行后台任务的集成 API

可定义任务的执行约束条件 (Constraints)

如：正在充电、已连上 Wi-Fi、当某 Content Uri 有改变等

往后兼容到 API 14！



// 创建需要网络约束的任务请求

```
val constraints = Constraints.Builder()  
    .setRequiredNetworkType(NetworkType.CONNECTED)  
    .build()
```

```
val request =  
    OneTimeWorkRequestBuilder<UploadPhotoWorker>()  
        .setConstraints(constraints)  
        .build()
```

```
WorkManager.getInstance().enqueue(request)
```



// 创建需要网络约束的任务请求

```
val constraints = Constraints.Builder()  
    .setRequiredNetworkType(NetworkType.CONNECTED)  
    .build()
```

```
val request =  
    OneTimeWorkRequestBuilder<UploadPhotoWorker>()  
        .setConstraints(constraints)  
        .build()
```

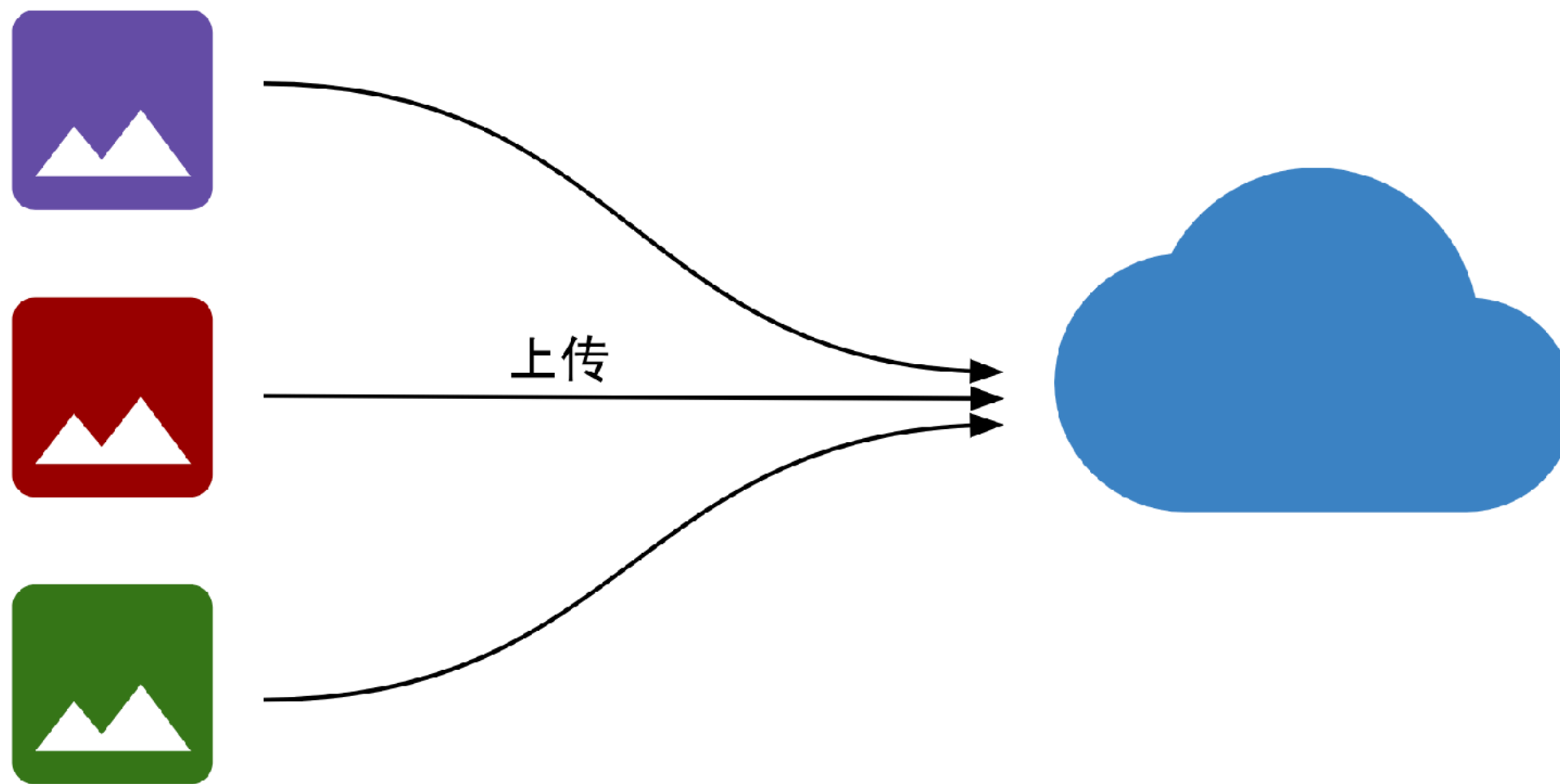
```
WorkManager.getInstance().enqueue(request)
```



串联任务



并联任务



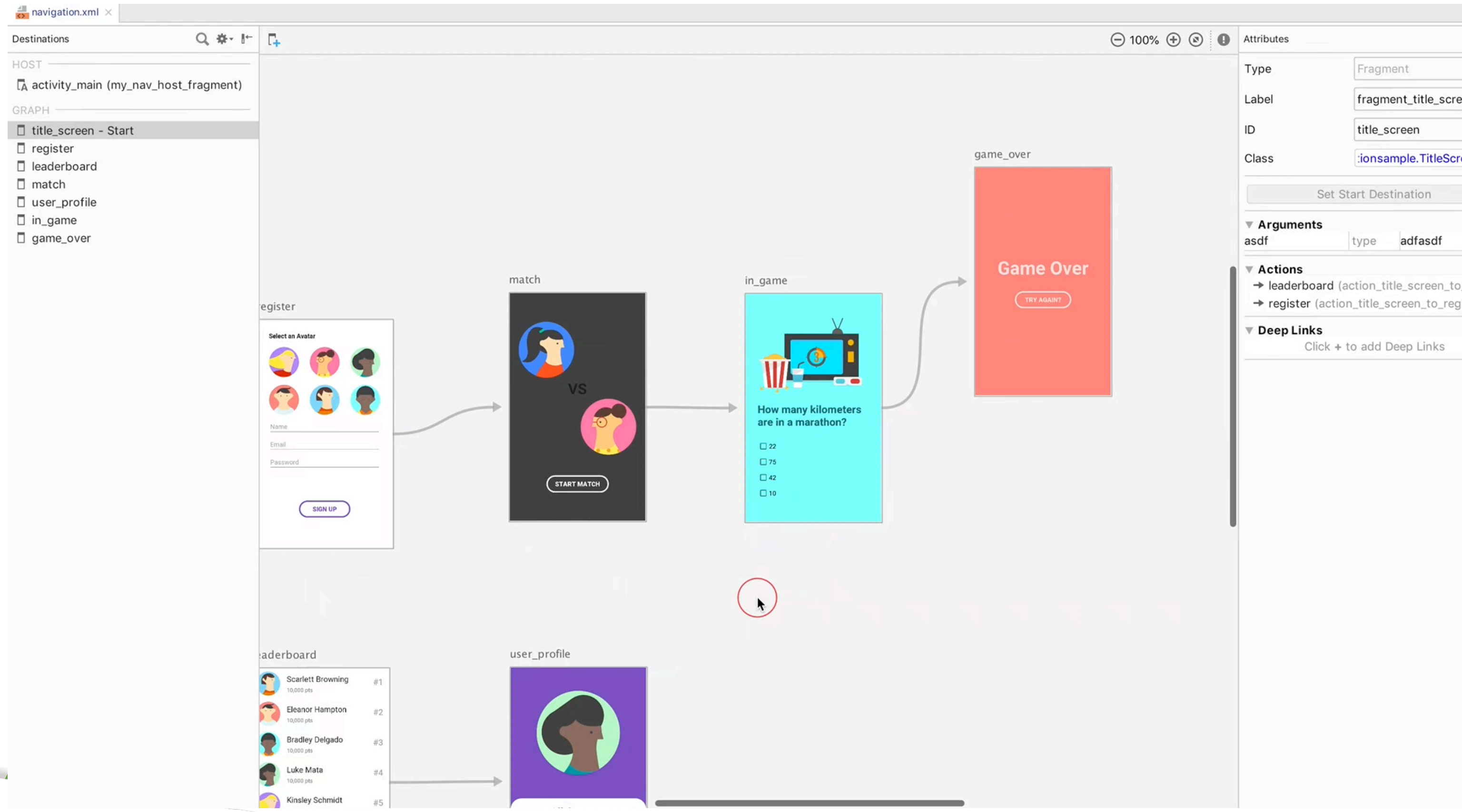
Navigation^{预览版}

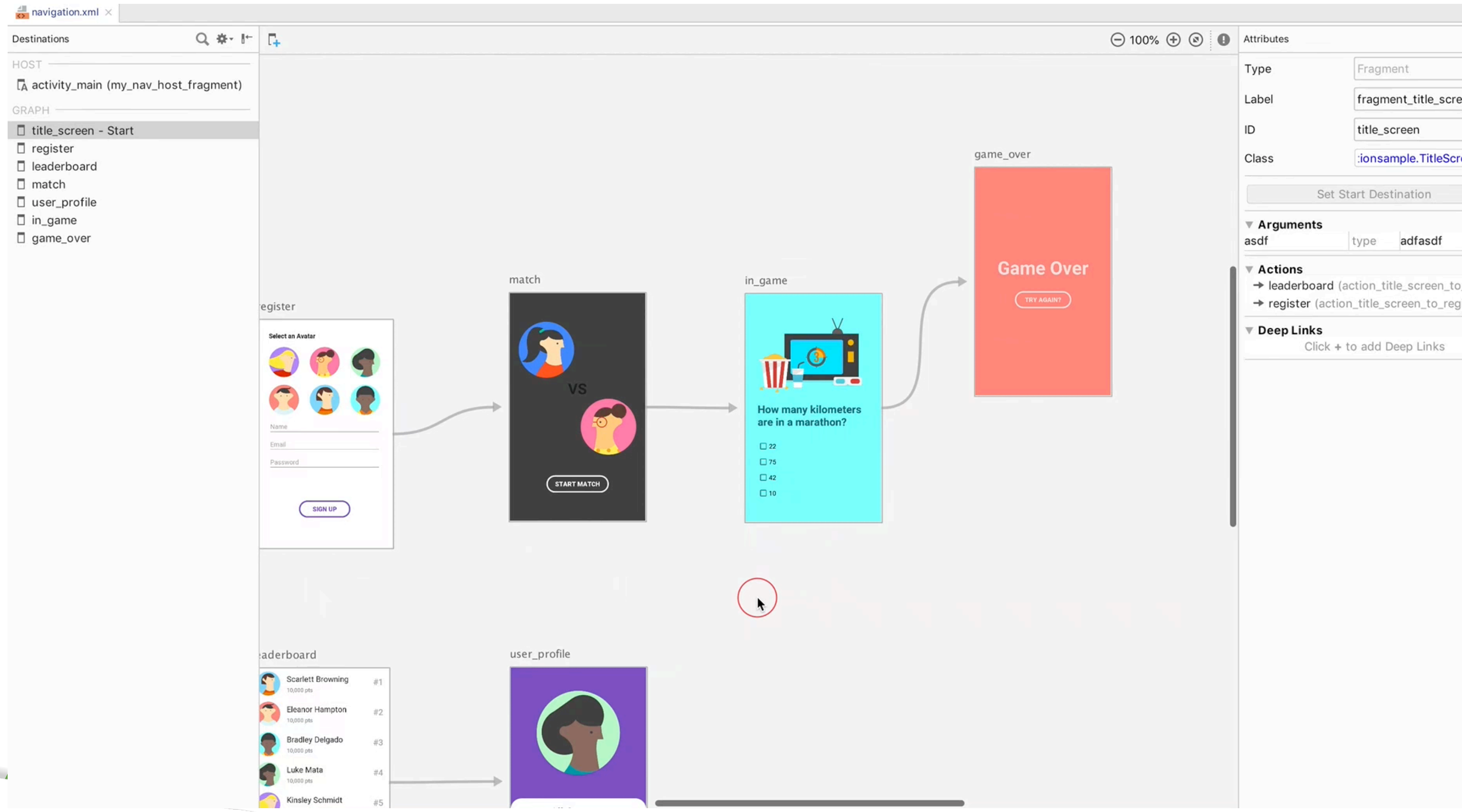
把应用内导航简单化

支持普遍导航流程：Back vs Up

支持 UI 间数据传递







Room

SQLite 对象映射抽象层

提供 Data Access Object 访问数据

大量减少 SQLite 周边样本代码



Paging



Paging

高效的逐步加载数据的分页库



Paging

高效的逐步加载数据的分页库



Paging

高效的逐步加载数据的分页库

支持加载有限、无限的 **List**



Paging

高效的逐步加载数据的分页库

支持加载有限、无限的 **List**



Paging

高效的逐步加载数据的分页库

支持加载有限、无限的 **List**

集合 RecyclerView、LiveData、RxJava 等



Paging

高效的逐步加载数据的分页库

支持加载**有限、无限的 List**

集合 RecyclerView、LiveData、RxJava 等



Paging

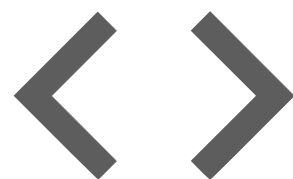
高效的逐步加载数据的分页库

支持加载有限、无限的 **List**

集合 RecyclerView、LiveData、RxJava 等

灵活支持多种数 **DataSource**





高效开发

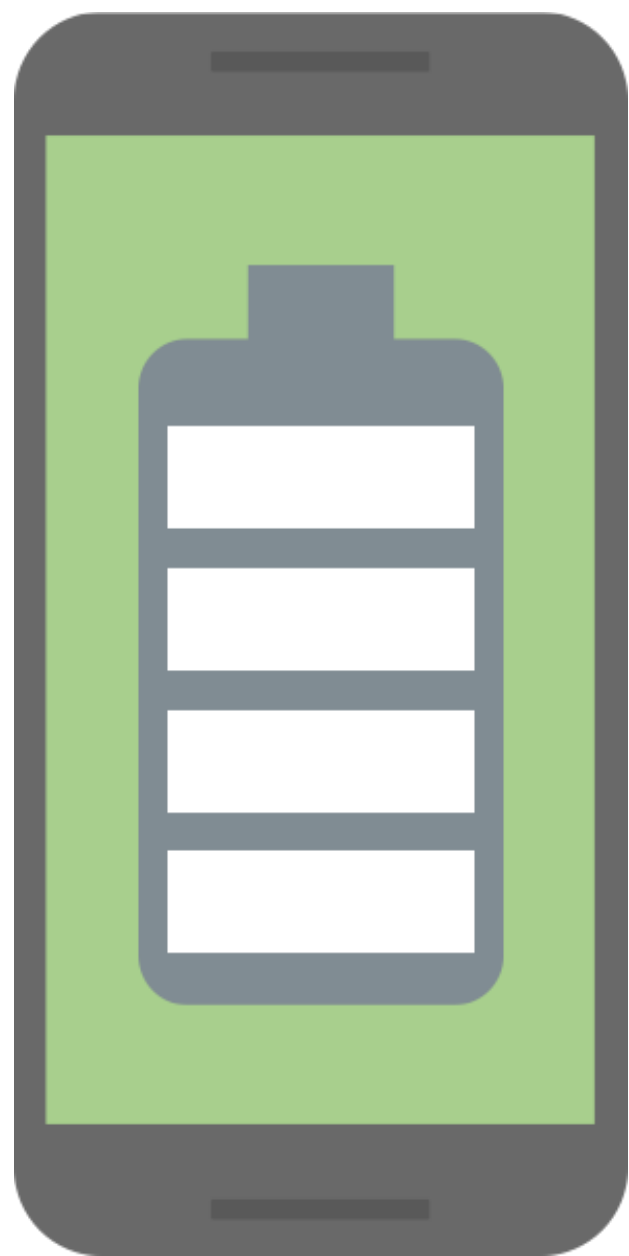


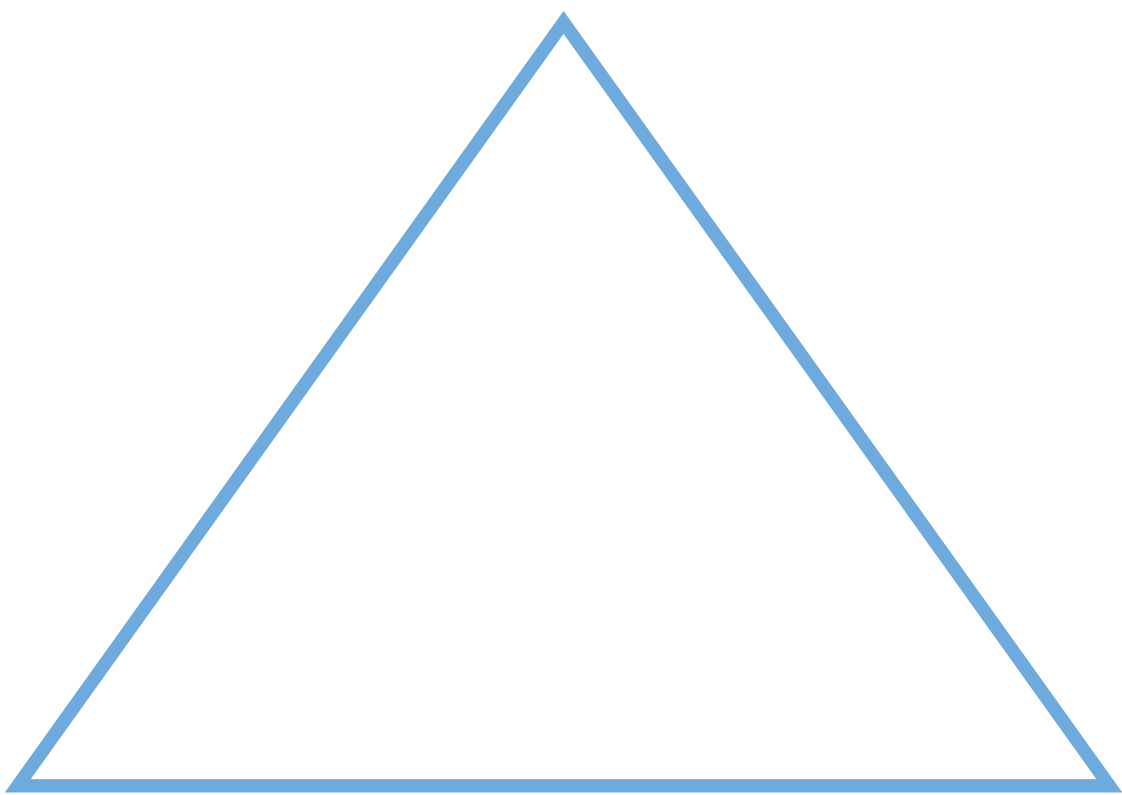
系统变更

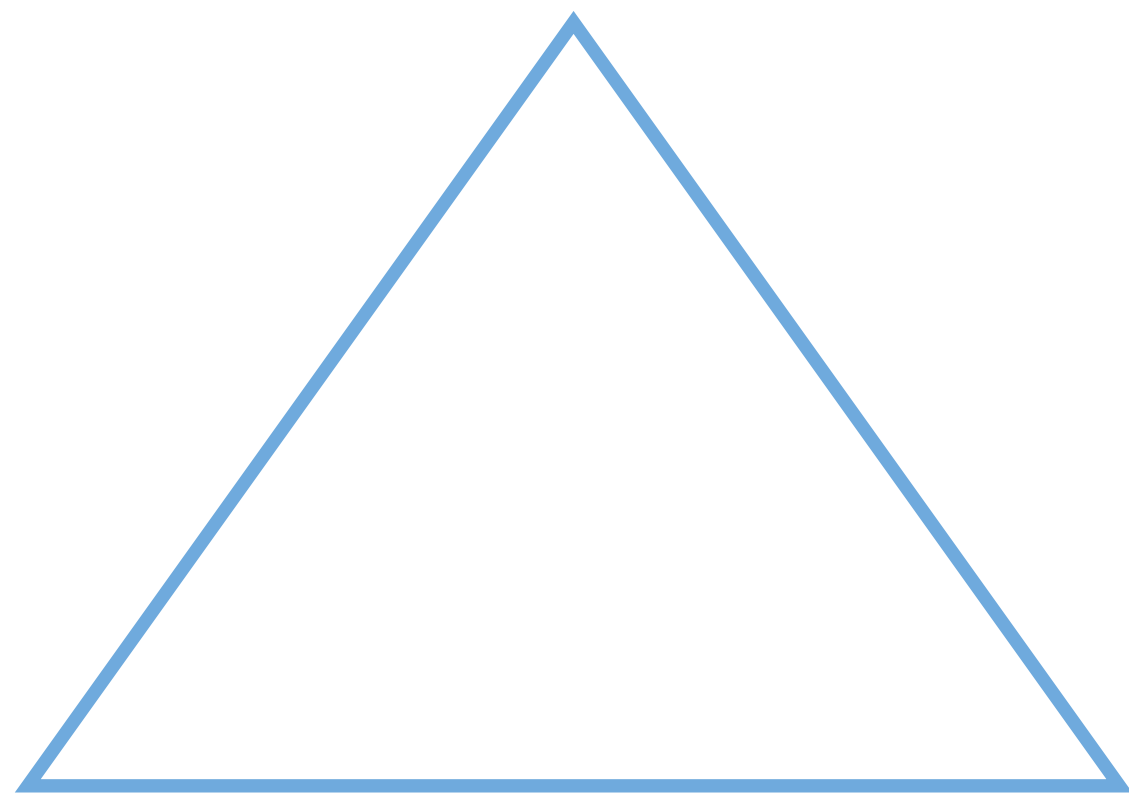


工具改进









设备状态



Doze 状态

设备状态

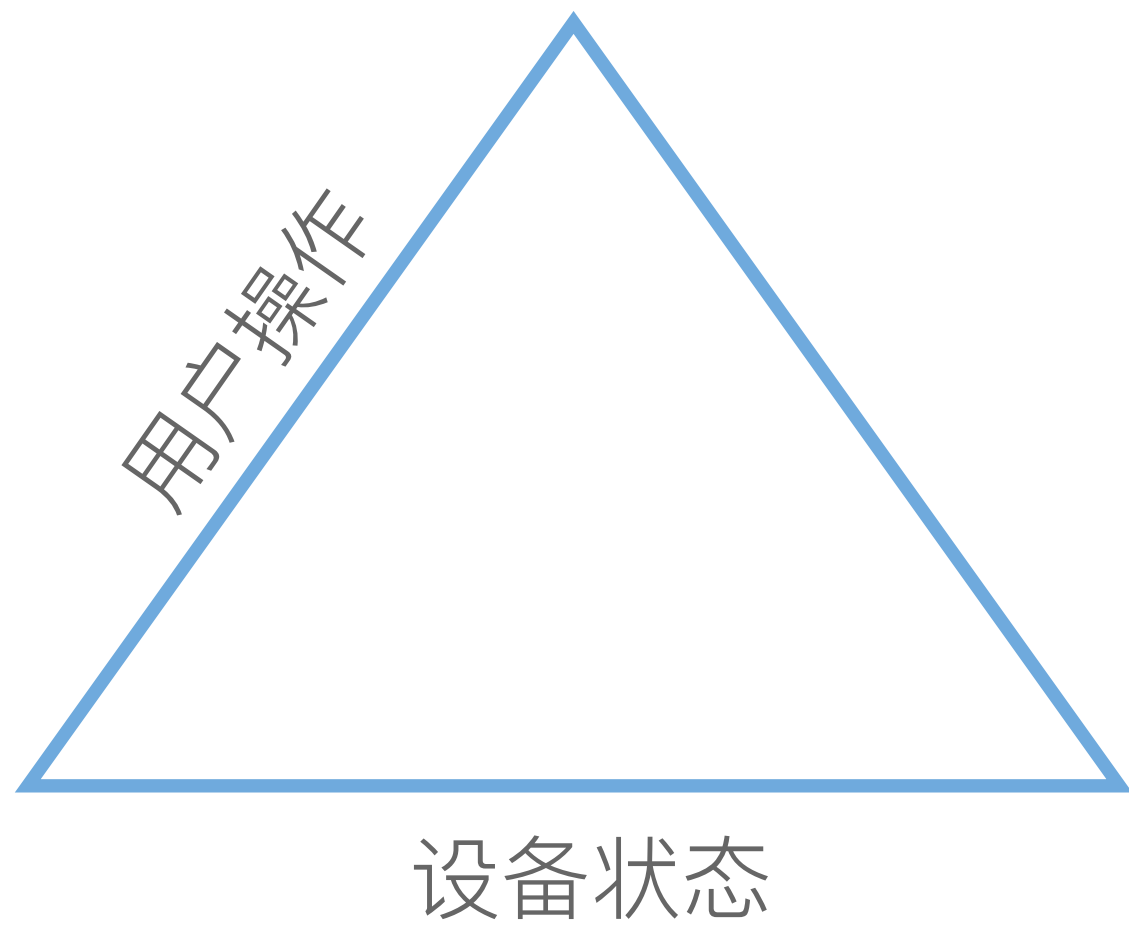


用户操作

设备状态

Doze 状态





Doze 状态

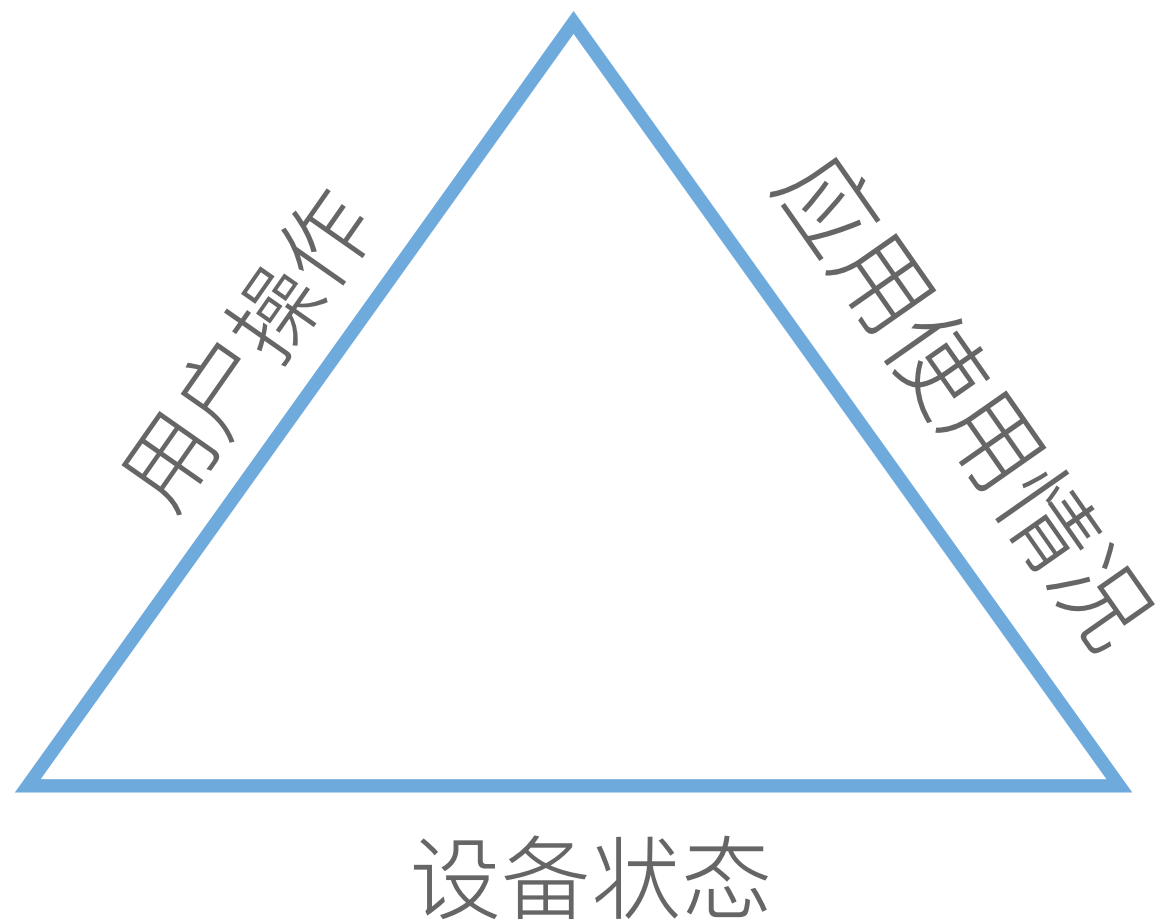


后台限制



省电模式





Doze 状态

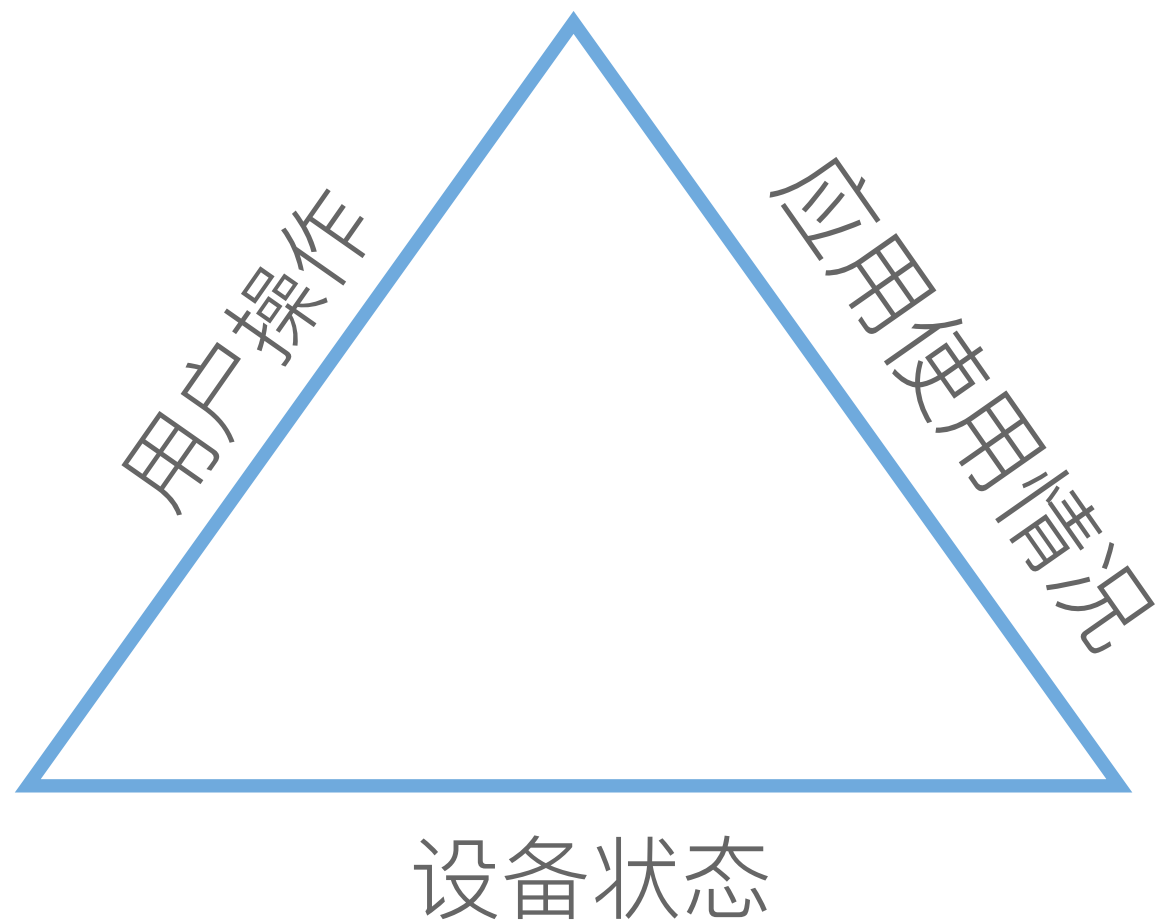


后台限制



省电模式





Doze 状态



后台限制



省电模式



应用待机分组



限制与延迟

Alarms

Jobs

网路访问

优先级

Firestore Cloud Messaging



开发者应怎应对？



开发者应怎应对？

测试，测试，再测试！

(可用 adb 命令行模拟系统限制)



应怎应对 Android 8.0
以上版本的后台限制？



最佳实践

需要在后台
执行某些任务？



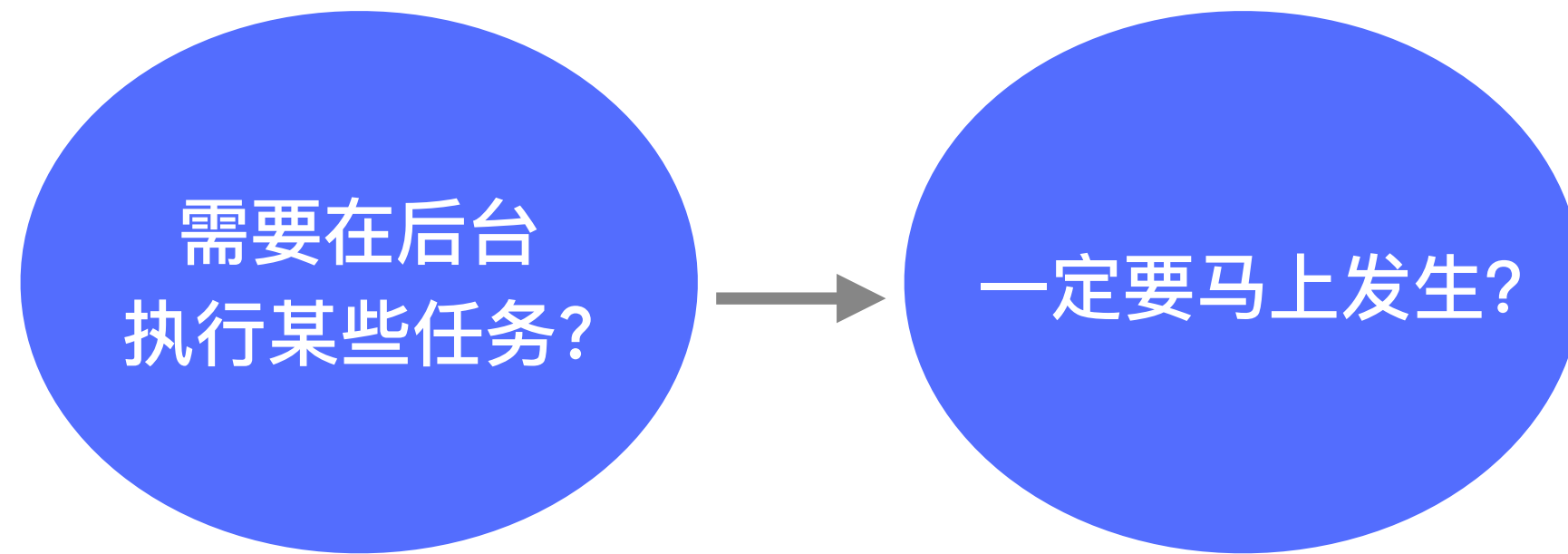
最佳实践

需要在后台
执行某些任务？

Job Scheduler



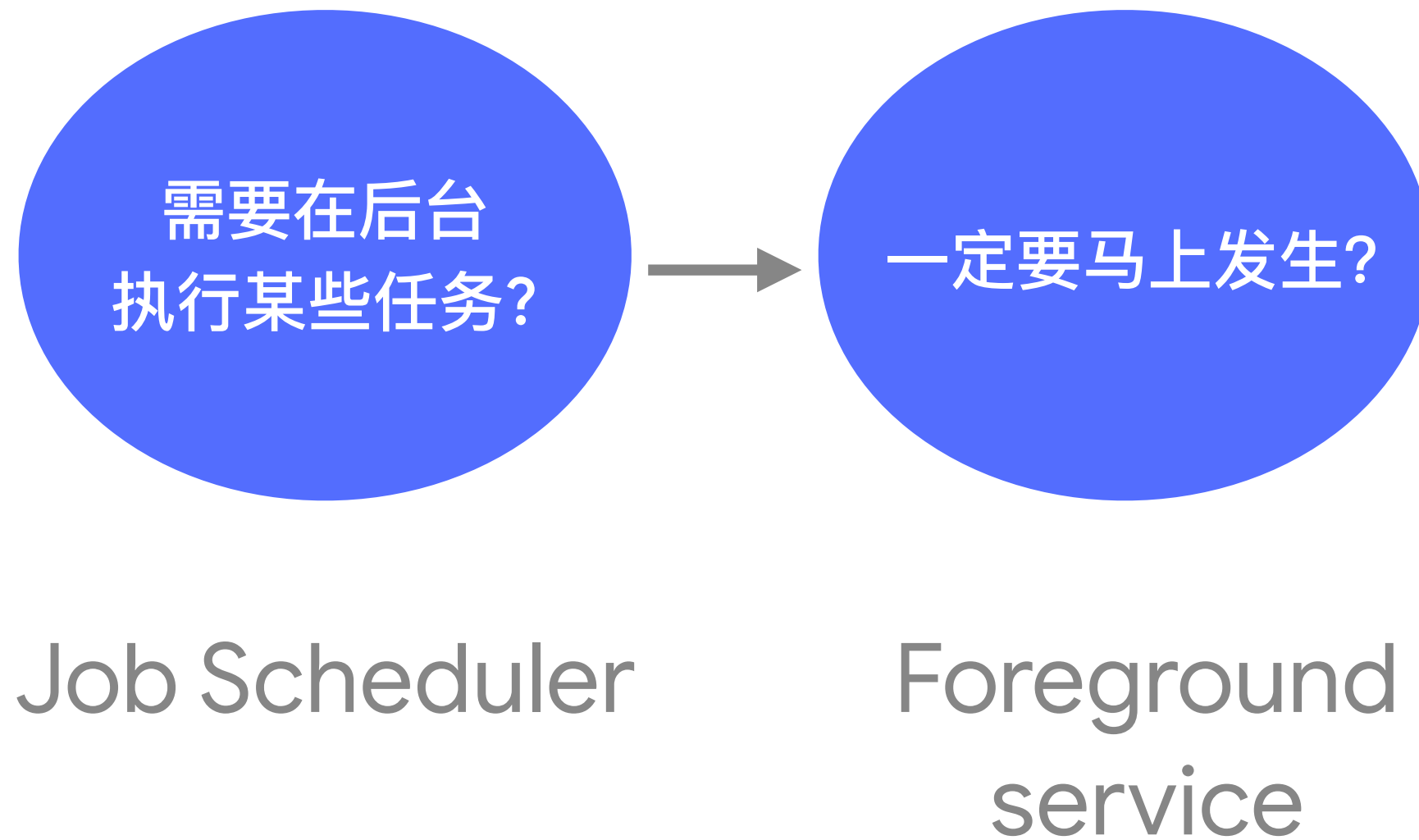
最佳实践



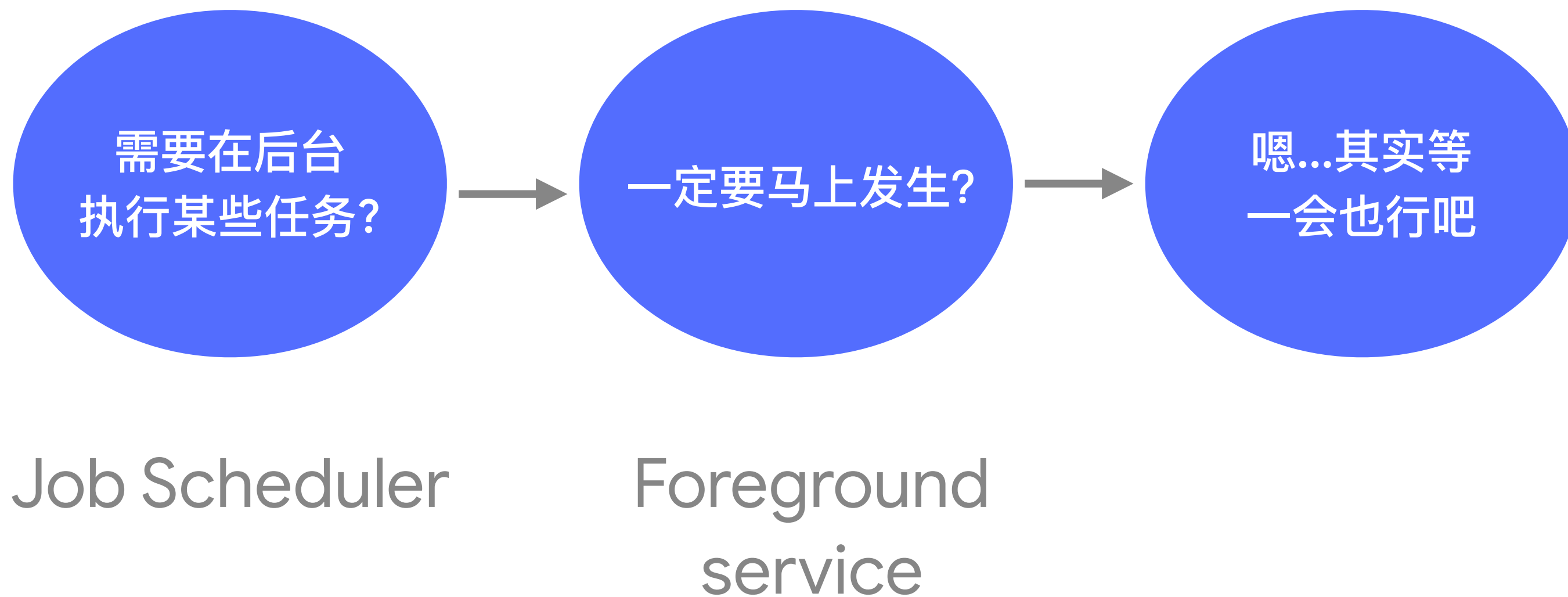
Job Scheduler



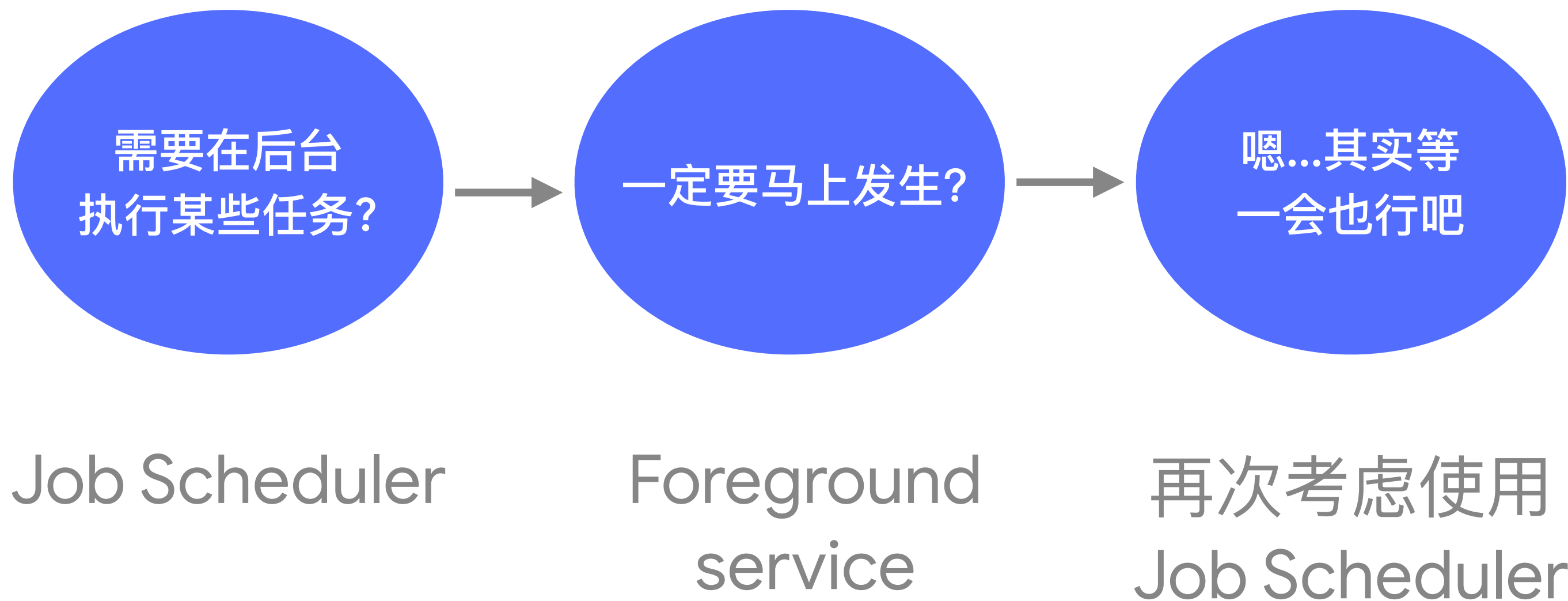
最佳实践



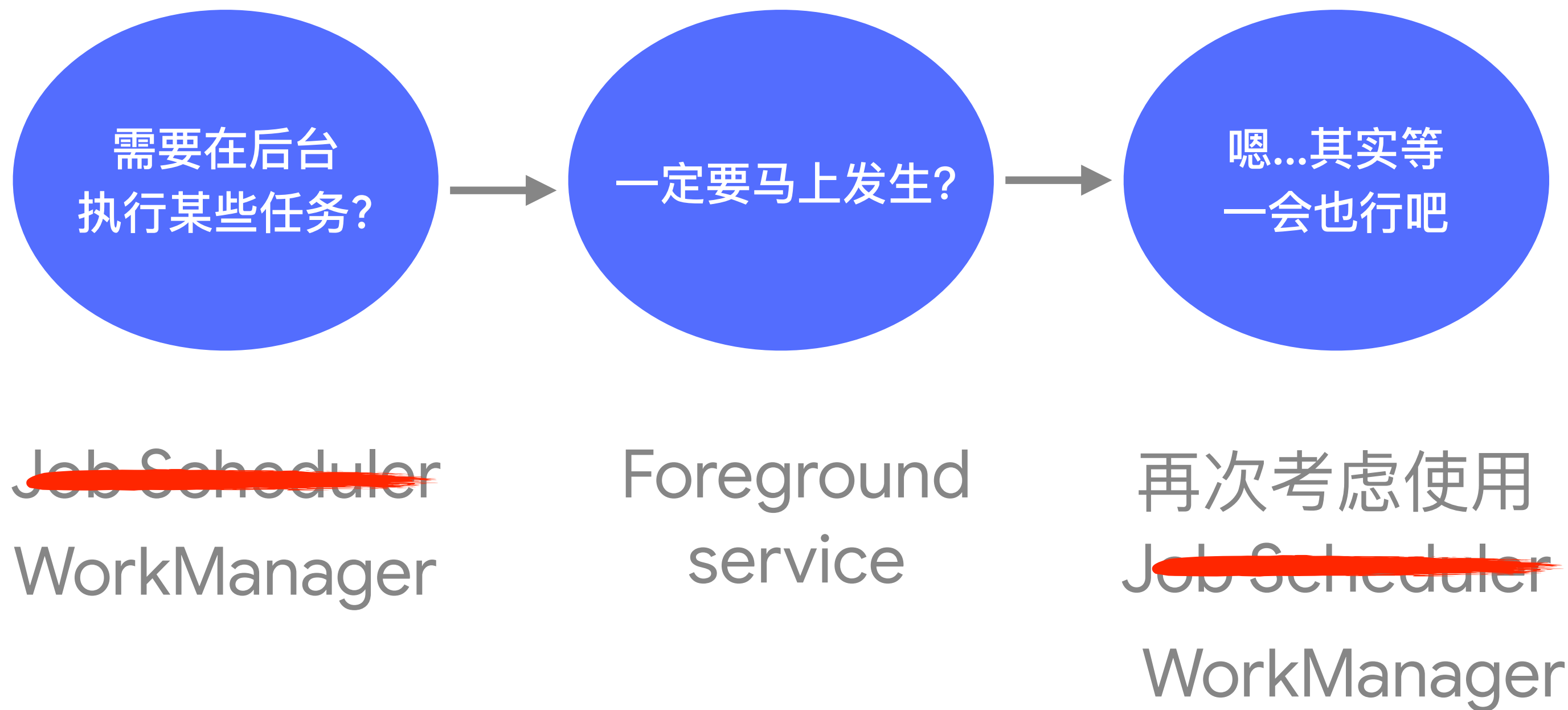
最佳实践



最佳实践



最佳实践



Android P: 使用非 SDK 接口限制

```
/* ...  
 * @hide  
 */  
public final class ActivityThread {...}
```

```
/** @hide */  
public static final boolean  
    DEBUG_BROADCAST = false;
```

```
/** @hide */  
@SystemApi  
@RequiresPermission(...)  
public int requestAudio() {...}
```



Android P: 使用非 SDK 接口限制



Android P：使用非 SDK 接口限制

- 使用公开 SDK 接口完成所有应用开发需求



Android P: 使用非 SDK 接口限制

- 使用公开 SDK 接口完成所有应用开发需求
- 非 SDK 接口仅能在系统内部使用



Android P: 使用非 SDK 接口限制

- 使用公开 SDK 接口完成所有应用开发需求
- 非 SDK 接口仅能在系统内部使用
- 保证应用在各系统和设备之间的兼容性



Android P：使用非 SDK 接口限制

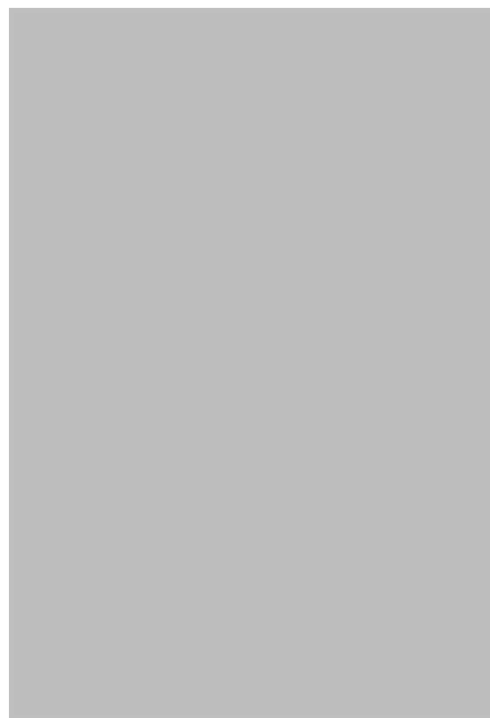
- 使用公开 SDK 接口完成所有应用开发需求
- 非 SDK 接口仅能在系统内部使用
- 保证应用在各系统和设备之间的兼容性
- 推进 Android 新系统被采用的速度



非 SDK 接口限制



白名单



灰名单

$\text{targetSdkVersion} < P$ 允许使用
 $\text{targetSdkVersion} \geq P$ 禁止使用



黑名单



怎样测试使用非 SDK 接口？

- 观察 logcat

```
“Accessing hidden field Landroid/os/Message;->flags:I  
(light greylist, JNI)”
```



怎样测试使用非 SDK 接口？

"Detected problems with API compatibility"



怎样测试使用非 SDK 接口？

"Detected problems with API compatibility"

- Toast 警告
- Debuggable 应用中会出现对话框警告



怎样测试使用非 SDK 接口？



怎样测试使用非 SDK 接口？

- 使用 `StrictMode.detectNonSdkApiUsage()`
在应用调用非 SDK 接口时获得警报



怎样测试使用非 SDK 接口?

- 使用 `StrictMode.detectNonSdkApiUsage()`
在应用调用非 SDK 接口时获得警报
- 静态扫描工具 `platform/art/tools/veridex`

```
// 编译
make appcompat

// 在 out\target\common\obj\PACKAGING 中生成三个限制名单
//   hiddenapi-blacklist.txt
//   hiddenapi-dark-greylst.txt
//   hiddenapi-light-greylst.txt

// 测试某个 APK
./art/tools/veridex/appcompat.sh --dex-file=test.apk
```



我用了非 SDK 接口该怎么办？



我用了非 SDK 接口该怎么办？

- 升级第三方库



我用了非 SDK 接口该怎么办？

- 升级第三方库
- 查看是否可以转换至公开 SDK 接口



我用了非 SDK 接口该怎么办？

- 升级第三方库
- 查看是否可以转换至公开 SDK 接口
- 合理的理由必须使用某个接口，请告诉我们

<https://goo.gl/KM1FMp>



我用了非 SDK 接口该怎么办？

- 升级第三方库
- 查看是否可以转换至公开 SDK 接口
- 合理的理由必须使用某个接口，请告诉我们

<https://goo.gl/KM1FMp>



加强隐私保护



加强隐私保护



处于后台时



加强隐私保护



处于后台时

新权限组
Call Logs



“刘海”屏幕支持



“刘海”屏幕支持



“刘海”屏幕支持

`WindowInsets.getDisplayCutout()`

`android:windowLayoutInDisplayCutoutMode`

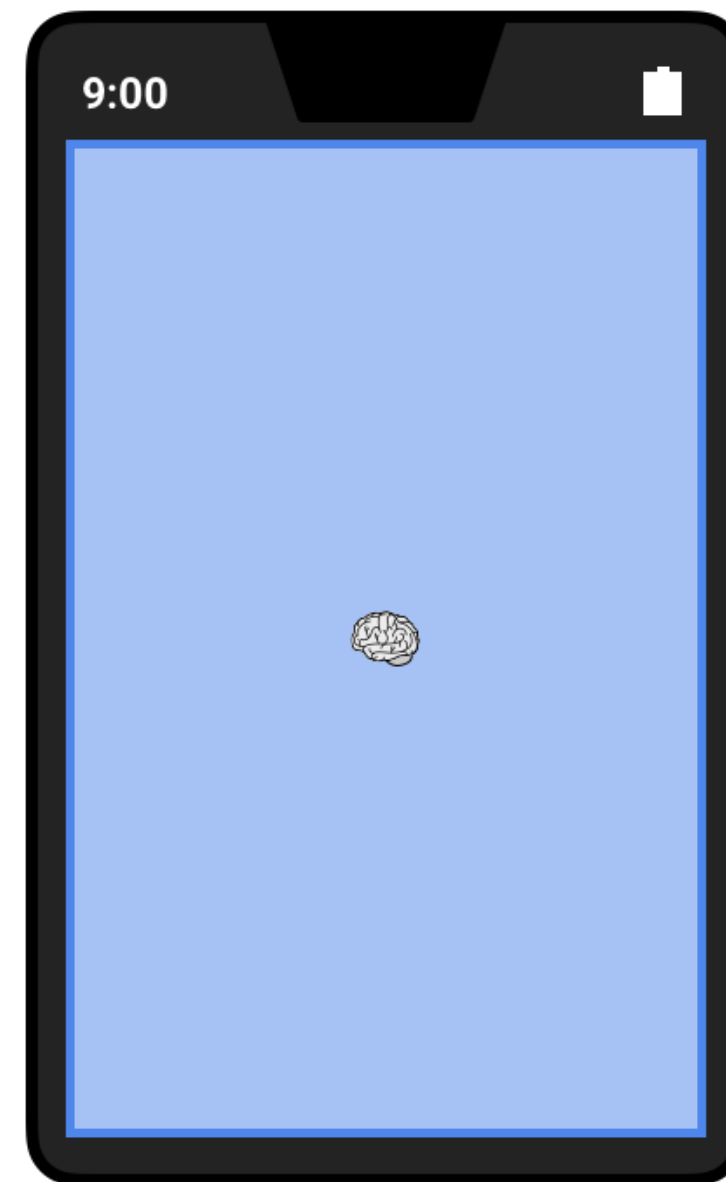


“刘海”屏幕支持

`WindowInsets.getDisplayCutout()`

`android:windowLayoutInDisplayCutoutMode`

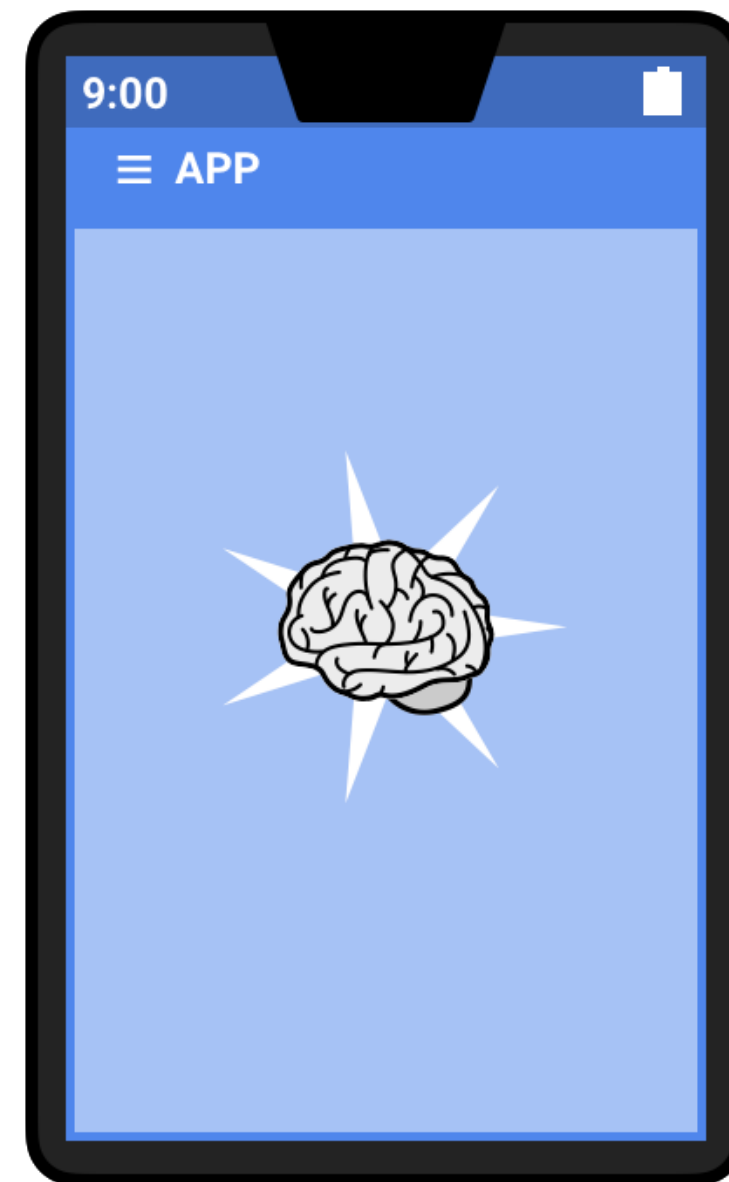
- "never"



“刘海”屏幕支持

`WindowInsets.getDisplayCutout()`
`android:windowLayoutInDisplayCutoutMode`

- "never"
- "default"



“刘海”屏幕支持

`WindowInsets.getDisplayCutout()`

`android:windowLayoutInDisplayCutoutMode`

- "never"
- "default"
- "shortEdges" + `DisplayCutout.getSafeInsets()`

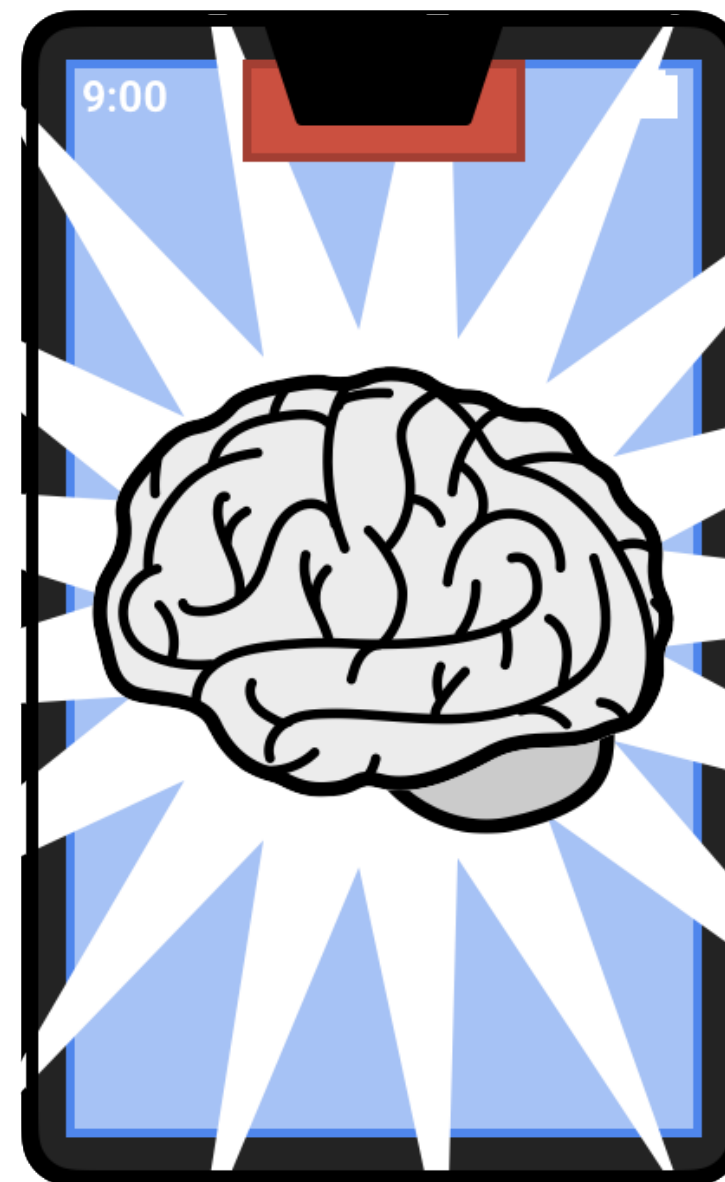


“刘海”屏幕支持

`WindowInsets.getDisplayCutout()`

`android:windowLayoutInDisplayCutoutMode`

- "never"
- "default"
- "shortEdges" + `DisplayCutout.getSafeInsets()`
- "shortEdges" + `DisplayCutout.getBounds()`



Google Play Target SDK 政策

最佳实践资源



Google Play Target SDK 政策

`targetSdkVersion ≥ 26`

最佳实践资源



Google Play Target SDK 政策

`targetSdkVersion ≥ 26`

最佳实践资源



Google Play Target SDK 政策

`targetSdkVersion ≥ 26`

新应用发布：2018 年 8 月

最佳实践资源



Google Play Target SDK 政策

`targetSdkVersion ≥ 26`

新应用发布：2018 年 8 月

应用更新发布：2018 年 11 月

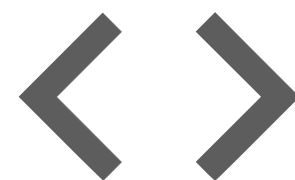
最佳实践资源





开发者预览版





高效开发



系统变更



工具改进







- 简洁语法





- 简洁语法
- 安全性高





- 简洁语法
- 安全性高
- 互相调用





- 简洁语法
- 安全性高
- 互相调用
- 工具支持



“谷歌开发者” 公众号问卷

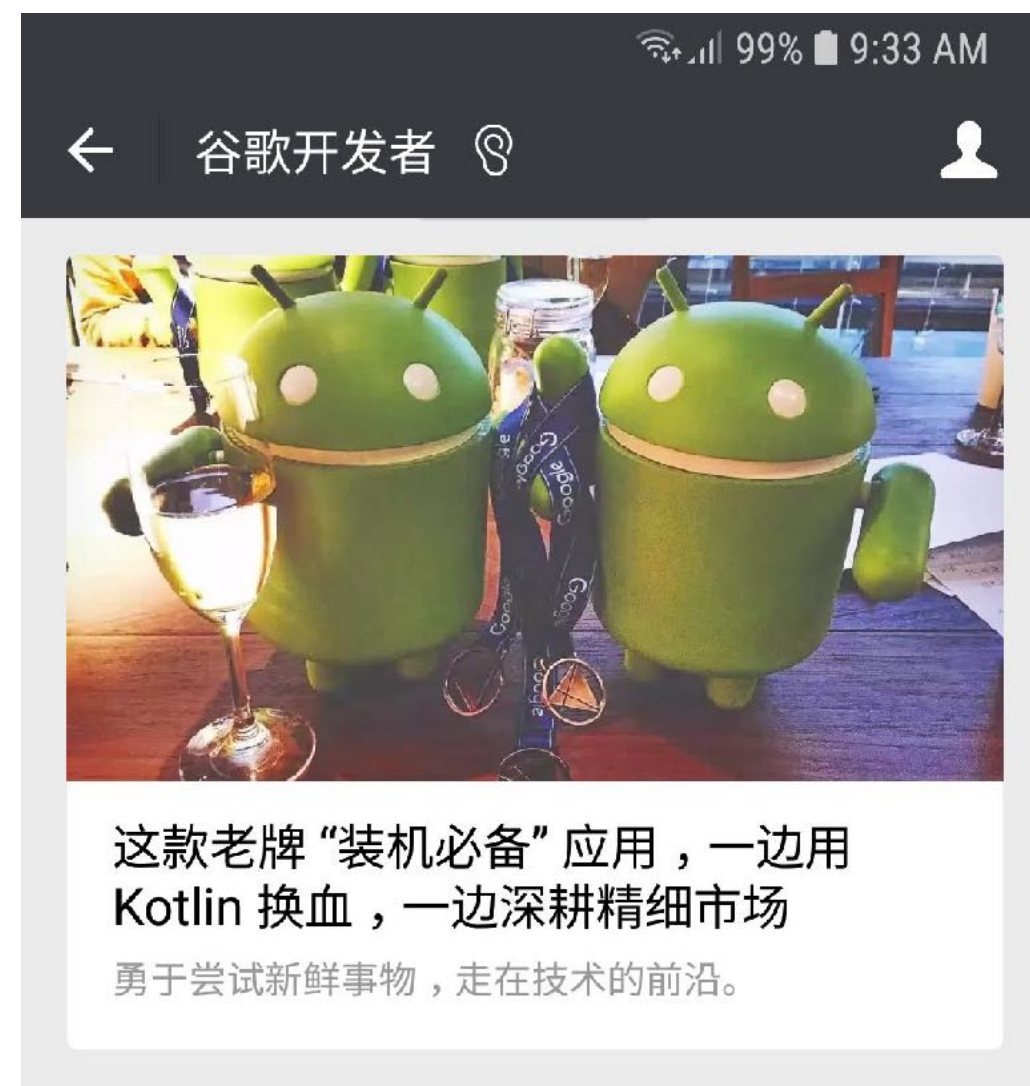
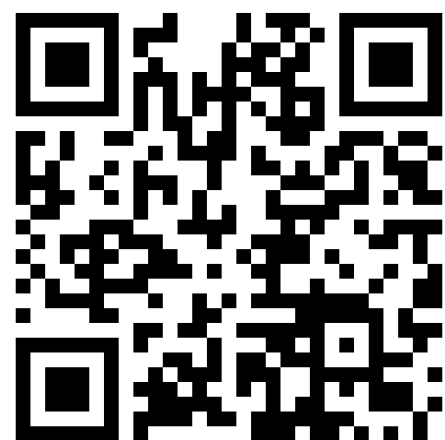
40%

问卷回答者表示
已在使用的 Kotlin



开发者故事：Camera360

“使用 *Kotlin* 显著减少
NullPointerException”



Kotlin

```
val pathDifference = Path(myPath1).apply {  
    op(myPath2, Path.Op.DIFFERENCE)  
}
```

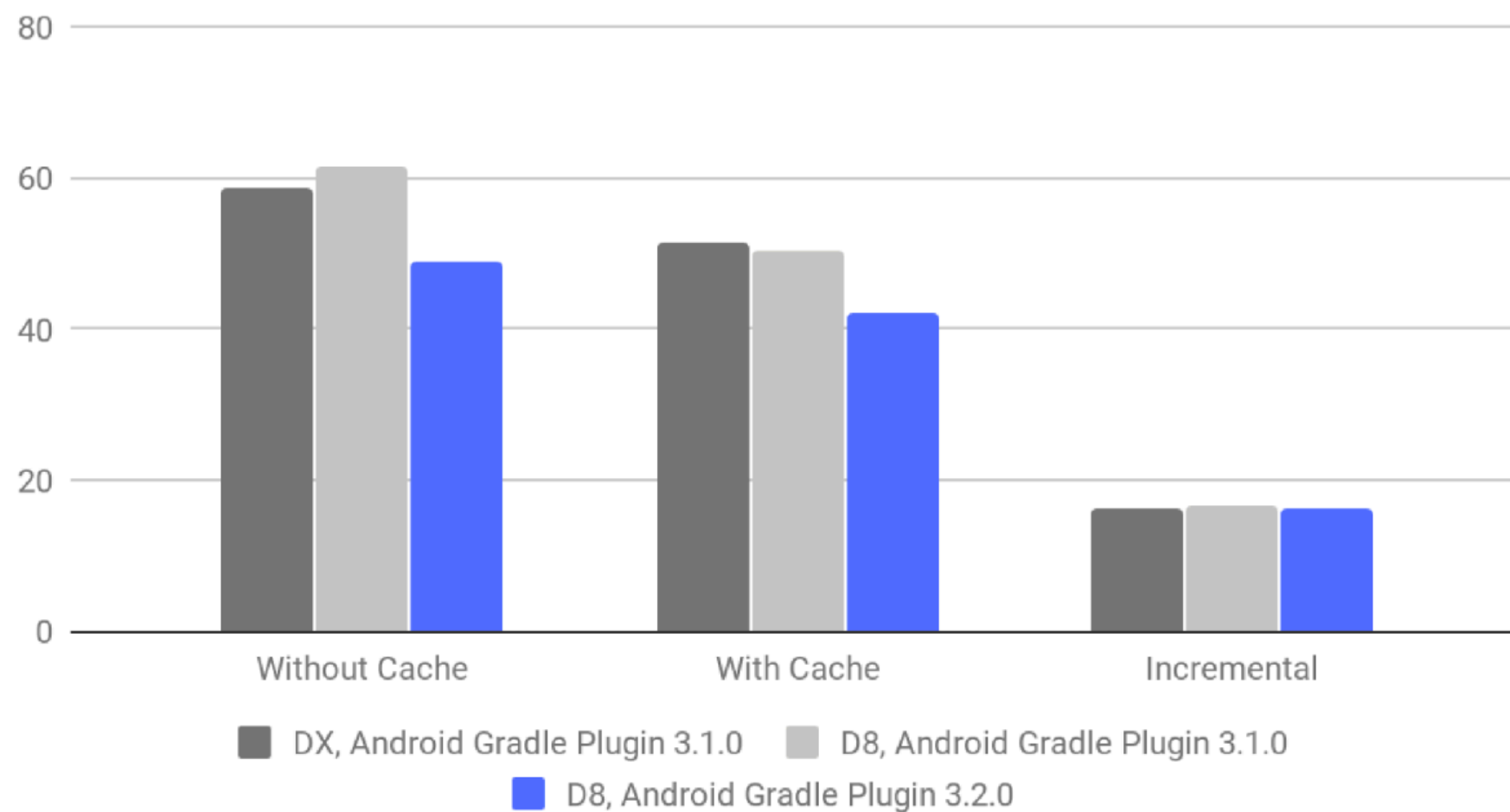
```
canvas.apply {  
    val checkpoint = save()  
    translate(0F, 100F)  
    drawPath(pathDifference, myPaint)  
    restoreToCount(checkpoint)  
}
```

With Android KTX

```
val pathDifference = myPath1 - myPath2  
  
canvas.withTranslation(y = 100F) {  
    drawPath(pathDifference, myPaint)  
}
```



DX (AGP 3.1.0) VS D8 (AGP 3.1.0) VS D8 (AGP 3.2.0)

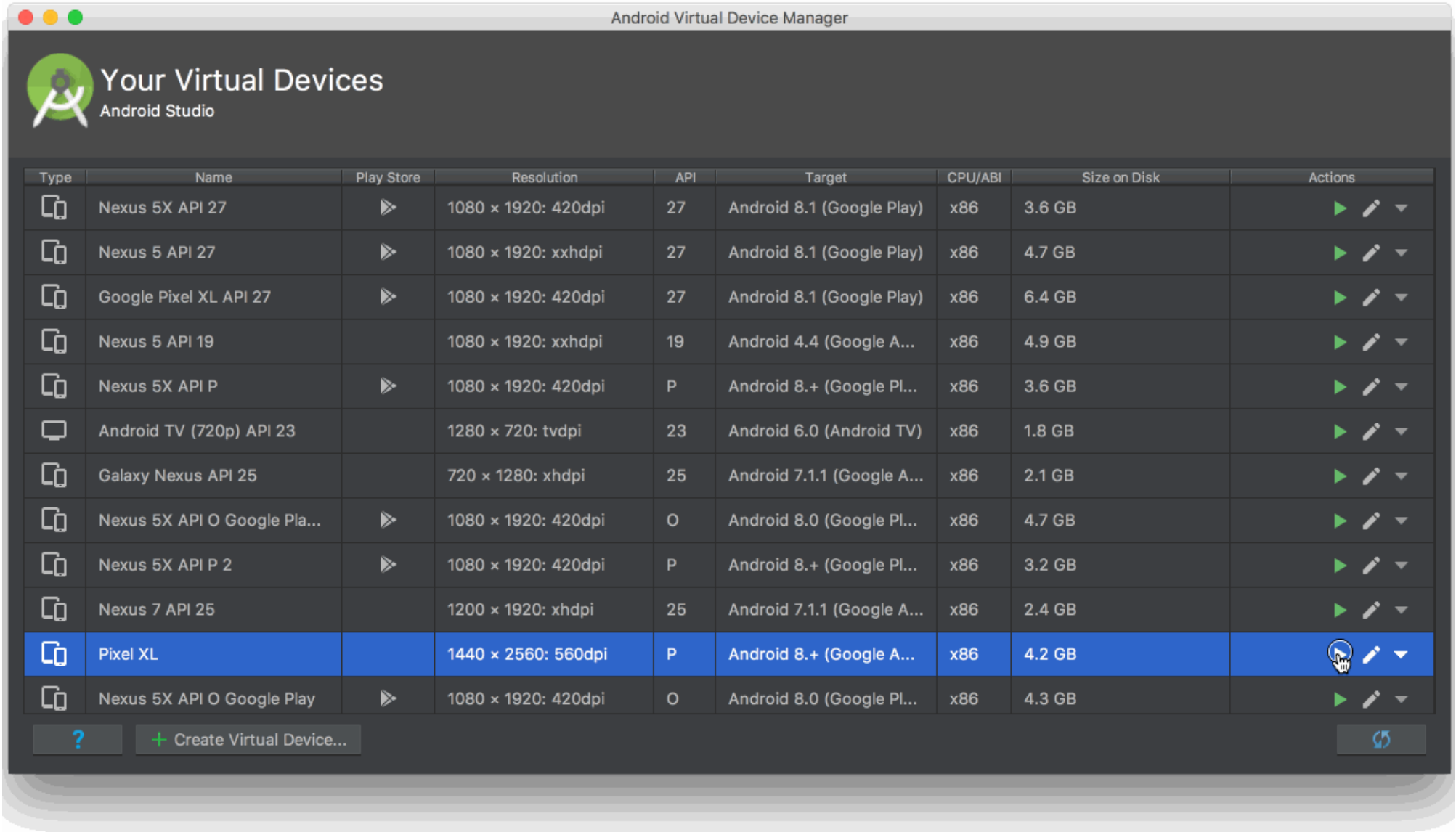


16%

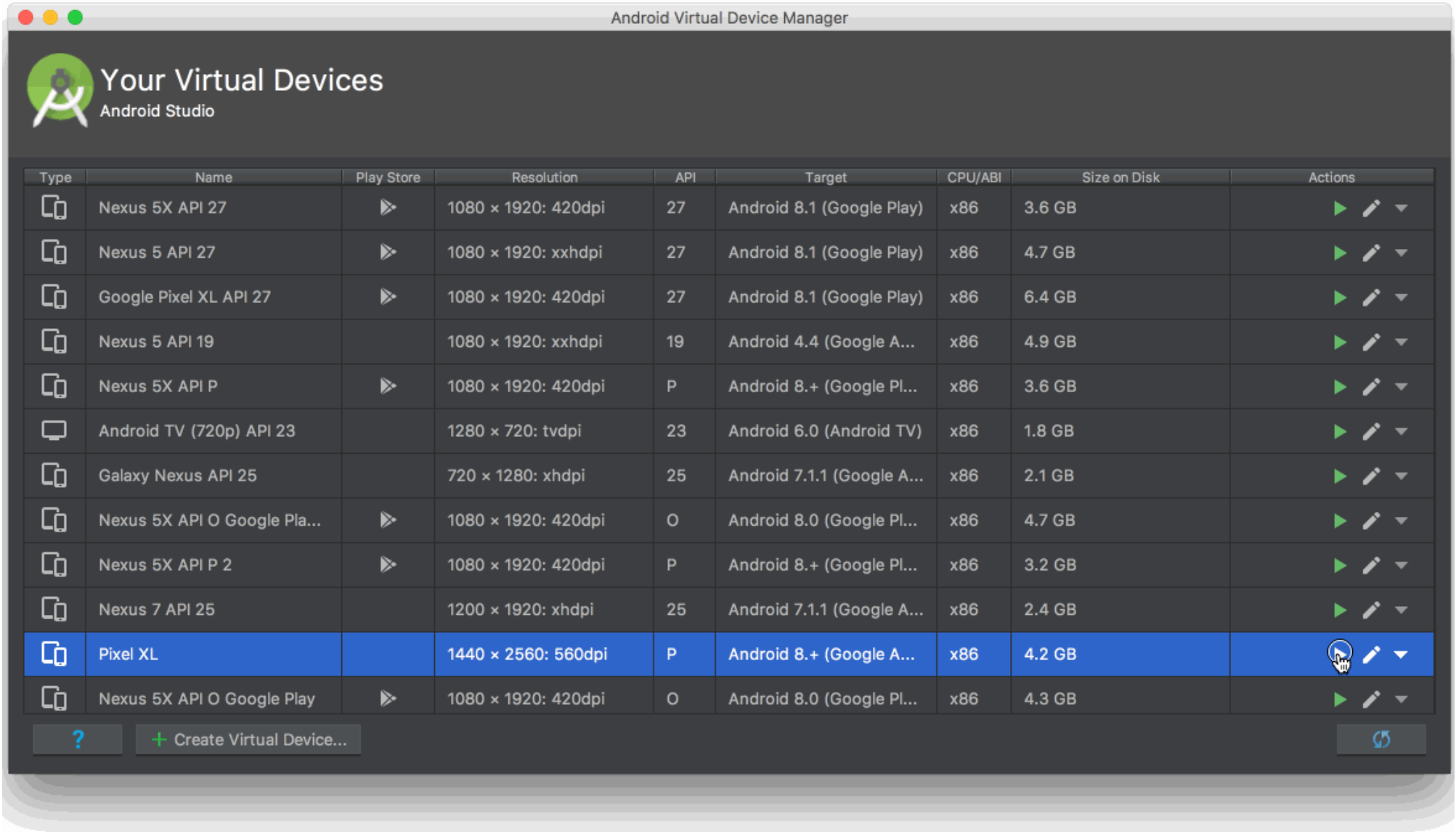
构建速度提升
(缓存关闭)



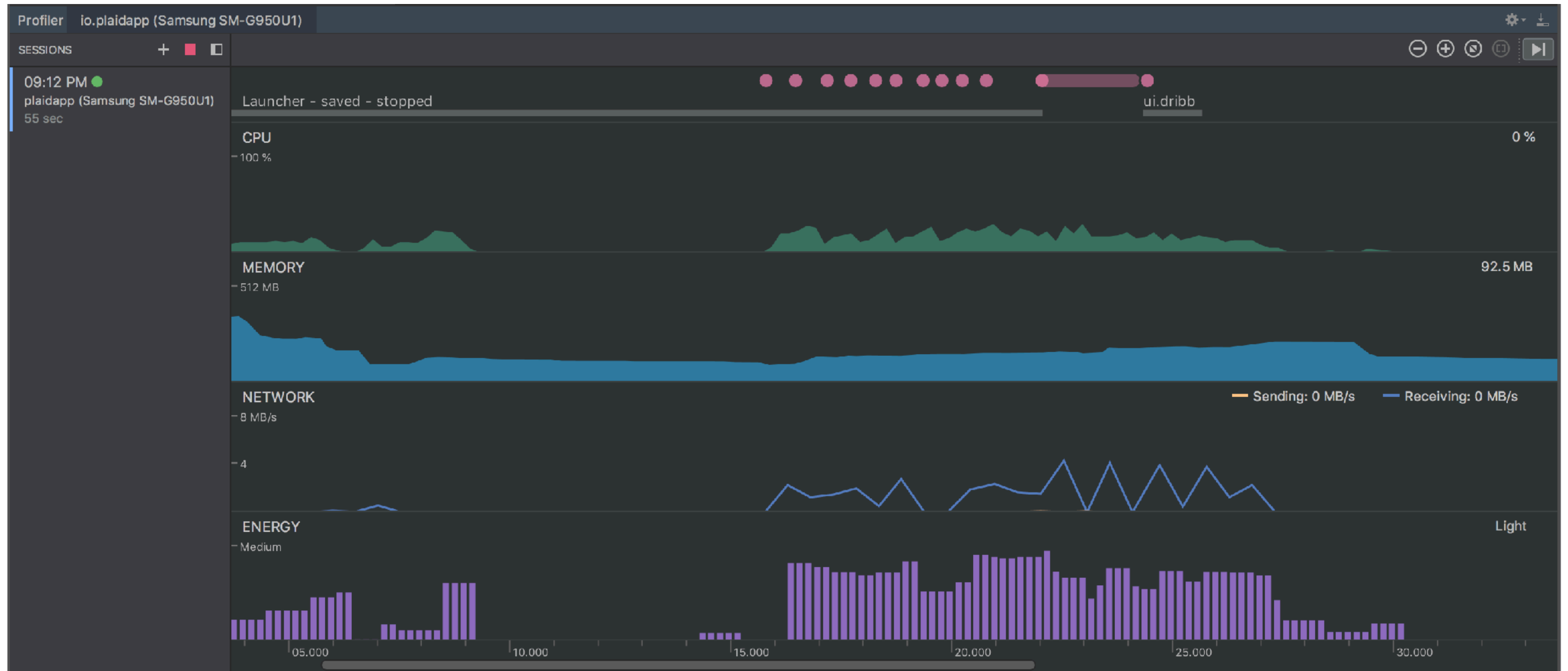
模拟器 Quick Boot

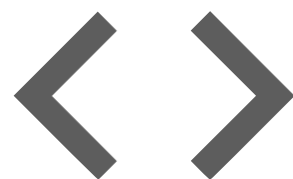


模拟器 Quick Boot



分析器





高效开发



系统变更



工具改进



Android 开发者资源



Google 开发者
微信公众号



Android 开发者中文网站
developer.android.google.cn



腾讯/优酷视频
Google 开发者频道



谢谢

