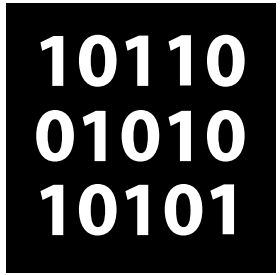
Kotlin
для написания общего кода под Android и iOS

Kotlin
для написания общего кода под Android и iOS

Святослав Щербина svyatoslav.scherbina@jetbrains.com

- Системный программист
- Интересуюсь разработкой и реализацией языков программирования
- Делаю Kotlin/Native в JetBrains
  - Интеграция с iOS



Agenda

- Проблемы с существующими кросс-платформенными инструментами
- Как это работает в Котлине?
  - Kotlin Multiplatform Programming
    - Язык, тулинг, библиотеки
  - Kotlin/Native
    - Управление памятью, модель памяти, интероперабельность
- Как с этим жить?
  - Demo app
- Как начать?
  - Ссылки на руководства, примеры etc.

Проблема

Писать общий код под Android и iOS тяжело.

Типичные ограничения разных инструментов:

- Неродной язык: JS, C# и т.п.
- Конкретный подход к UI и/или нет нативного UI
- Принудительное “write once run everywhere”

Что за Kotlin?

Современный язык программирования:

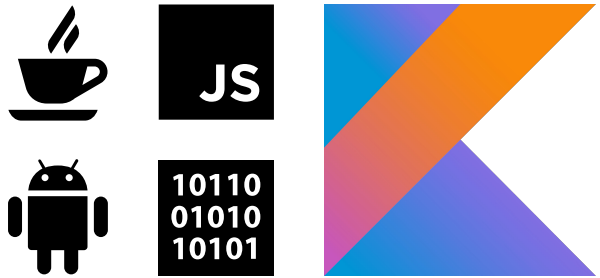
- Краткий, выразительный и безопасный
- Мультиплатформенный
  - Появился как очередной JVM-язык, сейчас компилируется в JS и Native
  - Официальный язык для Android
- Интероперабельный
- Инструментированный
  - IntelliJ IDEA, Android Studio, Gradle и т.д.

Kotlin Multiplatform Programming

Позволяет писать Kotlin-код под несколько платформ с поддержкой в тулинге

Ключевые отличия:

- “Write once, run everywhere” не обязательно, но иногда возможно
- Конкретный подход к UI не зафиксирован (но и не предлагается)



Kotlin Multiplatform Programming: язык

```
// Common
expect val deviceName: String
fun labelText() = "I'm on shiny new $deviceName!"


// Android
actual val deviceName: String get() = Build.MODEL


// iOS
actual val deviceName: String
    get() = UIDevice.currentDevice.model
```

Kotlin Multiplatform Programming: IDE

commonMain/kotlin/.../deviceName.kt

```
1 A  expect val deviceName: String
2    fun labelText() = "I'm on shiny new $deviceName!"
```

main/java/.../deviceName.kt

```
1    import android.os.Build
2
3 E  actual val deviceName: String get() = Build.MODEL
```

iosMain/kotlin/.../deviceName.kt

```
1    import platform.UIKit.UIDevice
2
3 E  actual val deviceName: String
4        get() = UIDevice.currentDevice.model
```

IDEA одновременно показывает код для всех платформ.

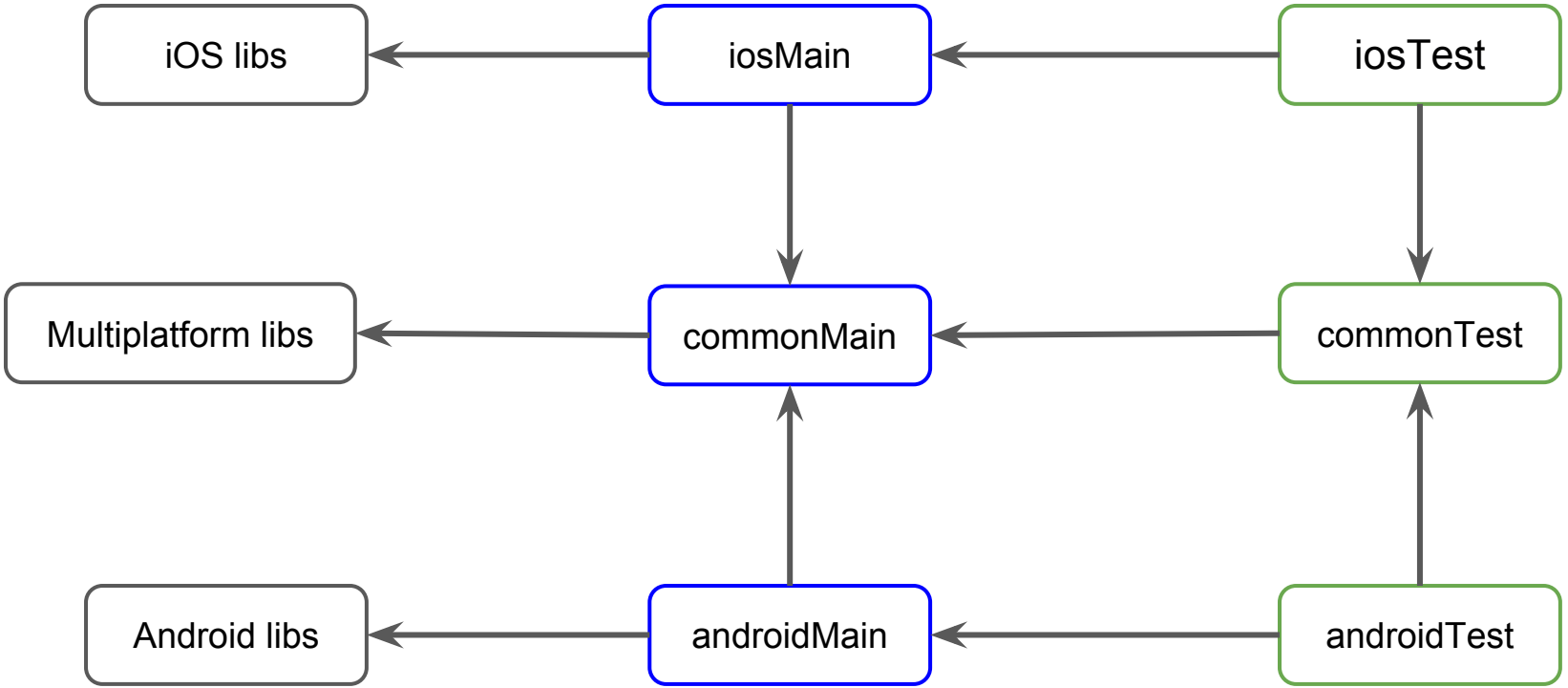
Kotlin Multiplatform Programming: Gradle

Gradle-скрипт задаёт соответствие между исходниками и платформами.

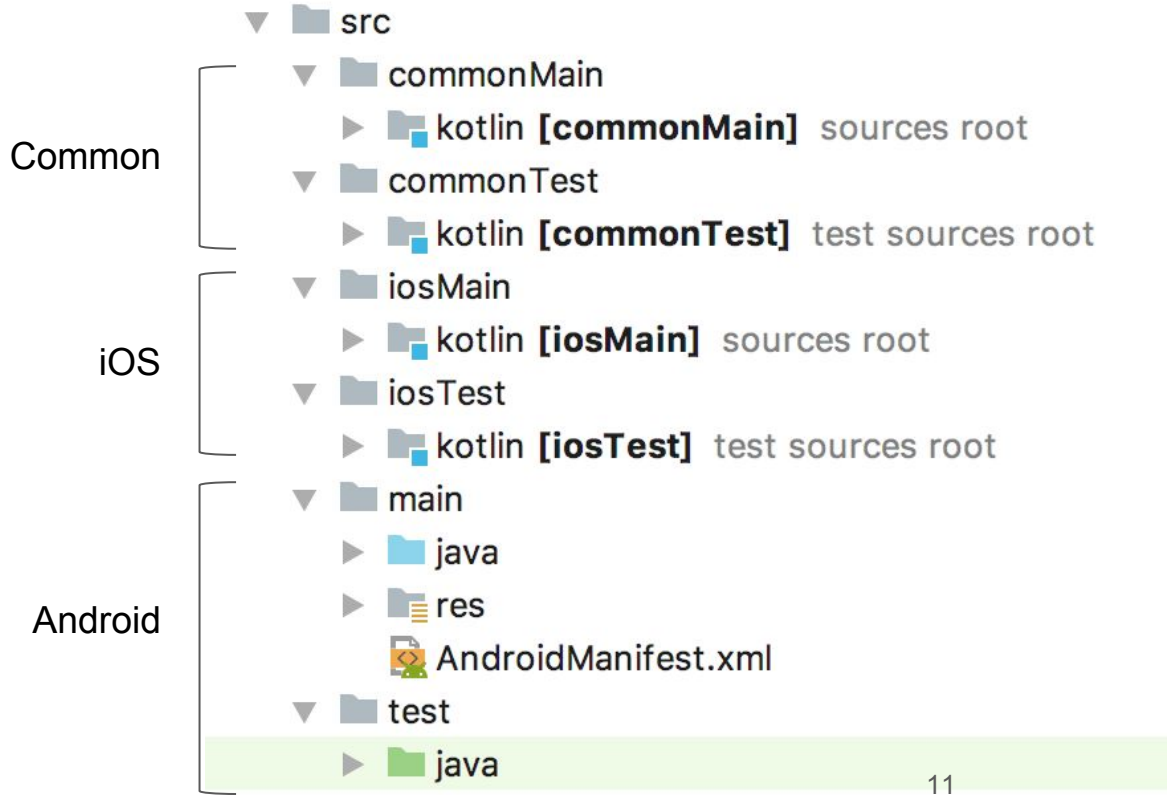
```
plugins { id 'kotlin-multiplatform' version '1.3.0' }

kotlin {
    targets {
        fromPreset(presets.android, 'android')
        fromPreset(presets.iosX64, 'ios') {
            compilations.main.outputKinds('FRAMEWORK')
        }
    }
}
```

Kotlin Multiplatform Programming: модель



Kotlin Multiplatform Programming: модель в IDE



IDEA импортирует Gradle-скрипт, чтобы получить структуру проекта.

Kotlin Multiplatform Programming: библиотеки

Платформенный код может быть спрятан в библиотеки.

Общий код может использовать мультиплатформенные библиотеки:

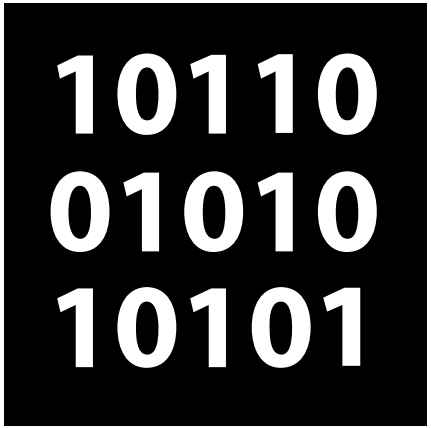
- Kotlin Standard Library
- kotlinx.coroutines – запуск и управление корутинами
- Ktor client – выполнение HTTP-запросов
- kotlinx.serialization – сериализация Kotlin-объектов в JSON и т.п.
- и т.д.

Kotlin компилируется в

JVM

JS

Android



Native:

- iOS
- WebAssembly
- Native macOS
- Native Linux (x86_64, arm32, mips32)
- Native Windows

Kotlin/Native

Технология компиляции Kotlin в нативный код.

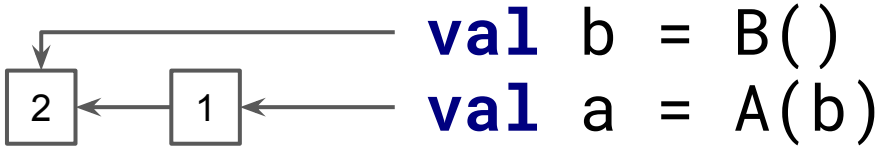
- Без JVM (больше платформ, лучше время запуска, меньше дистрибутив)
- Интеграция и взаимодействие с платформой (Swift/Obj-C на iOS)

Объявлен бетой в Kotlin 1.3

Kotlin/Native: управление памятью

Kotlin-объекты удаляются автоматически

- Подсчёт ссылок



Kotlin/Native: управление памятью

Kotlin-объекты удаляются автоматически

- Подсчёт ссылок

Kotlin/Native: управление памятью

Kotlin-объекты удаляются автоматически

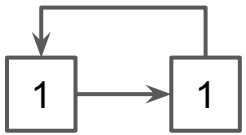
- Подсчёт ссылок

Kotlin/Native: управление памятью

Kotlin-объекты удаляются автоматически

- Подсчёт ссылок. Но что там с циклами?

Циклический мусор



Kotlin/Native: управление памятью

Kotlin-объекты удаляются автоматически

- Подсчёт ссылок
- Циклические ссылки обрабатываются техникой “trial deletion”

Objective-C-объекты удаляются автоматически

- Циклические ссылки в них не обрабатываются

Kotlin/Native: модель памяти

- Обычные объекты принадлежат единственному потоку рекурсивно
  - Владение может быть передано явно
- Глубоко неизменяемые объекты можно разделять между потоками
  - Обычные объекты можно превратить в неизменяемые явно

Kotlin/Native: модель памяти

```
val message = Message()
message.text = "Hello"

async { // (Runtime) error.
    use(message)
}
use(message)
```

```
val message = Message()
message.text = "Hello"
message.freeze() // Ok now.
async {
    use(message)
}
use(message)
```

Kotlin/Native: модель памяти

- Обычные объекты принадлежат единственному потоку рекурсивно
  - Владение может быть передано явно
- Глубоко неизменяемые объекты можно разделять между потоками
  - Обычные объекты можно превратить в неизменяемые явно
- Разделяемые изменяемые объекты могут быть реализованы специальным образом (unsafe, internal, WIP)

Kotlin/Native: взаимодействие с Objective-C и Swift

```
import platform.Foundation.*

fun NSDate.daysTo(other: NSDate): Int =
    NSCalendar(NSGregorianCalendar)
        .components(NSDayCalendarUnit, this, other, 0)
        .day.toInt()
```

Kotlin/Native: взаимодействие с Objective-C и Swift

```kotlin
// Kotlin
interface View {
    fun updateView(data: DataModel)
}

class Presenter(val repository: Repository, val view: View) {
    fun start() { ... }

    fun present() {
        view.updateView(repository.data)
    }
}
```

Kotlin/Native: взаимодействие с Objective-C и Swift

```
// Swift
class ViewController: UIViewController, View {
    lazy var presenter =
            Presenter(repository: theRepository, view: self)

    override func viewDidLoad() {
        super.viewDidLoad()
        presenter.start()
    }

    func updateView(data: DataModel) { ... }
}
```

Kotlin/Native: шаги компиляции

Kotlin-исходники

→ Kotlin IR *(Kotlin-фронтенд)*

→ LLVM bitcode *(Kotlin Native-бэкенд)*

→ Машинный код *(LLVM)*

→ Нативный бинарник *(системный линкер)*

Kotlin/Native: тулинг

Код можно собирать и публиковать через Gradle

и открывать в IDEA и Android Studio.

Для отладки используется LLDB.

Xcode можно использовать для запуска и отладки.

Kotlin/Native: ограничения

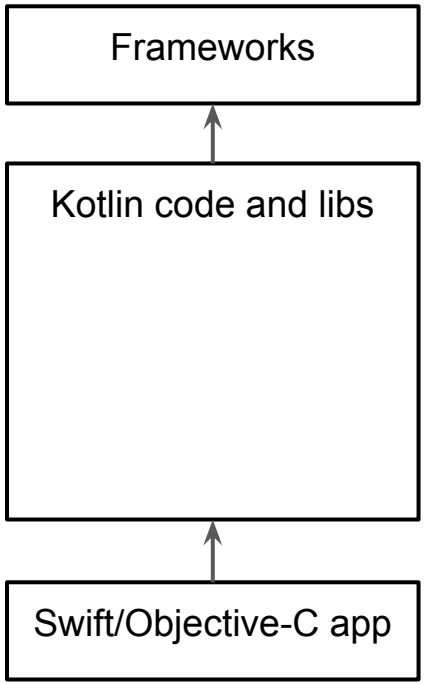
Фундаментальные

- Нет Java-зависимостей

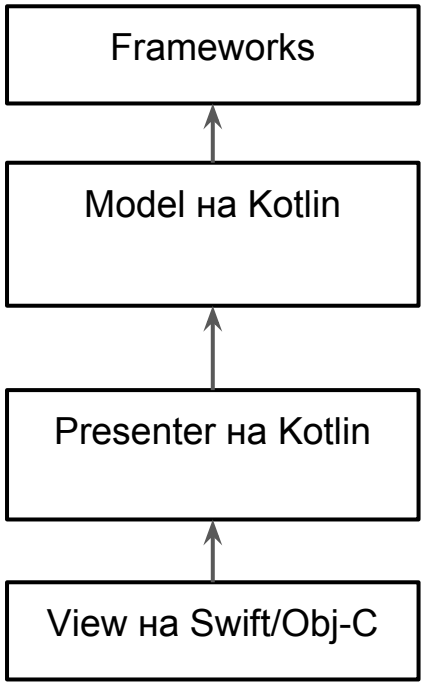
Временные

- Корутины привязаны к главному потоку
- Xcode Interface Builder не поддерживает Kotlin
- и т. д.

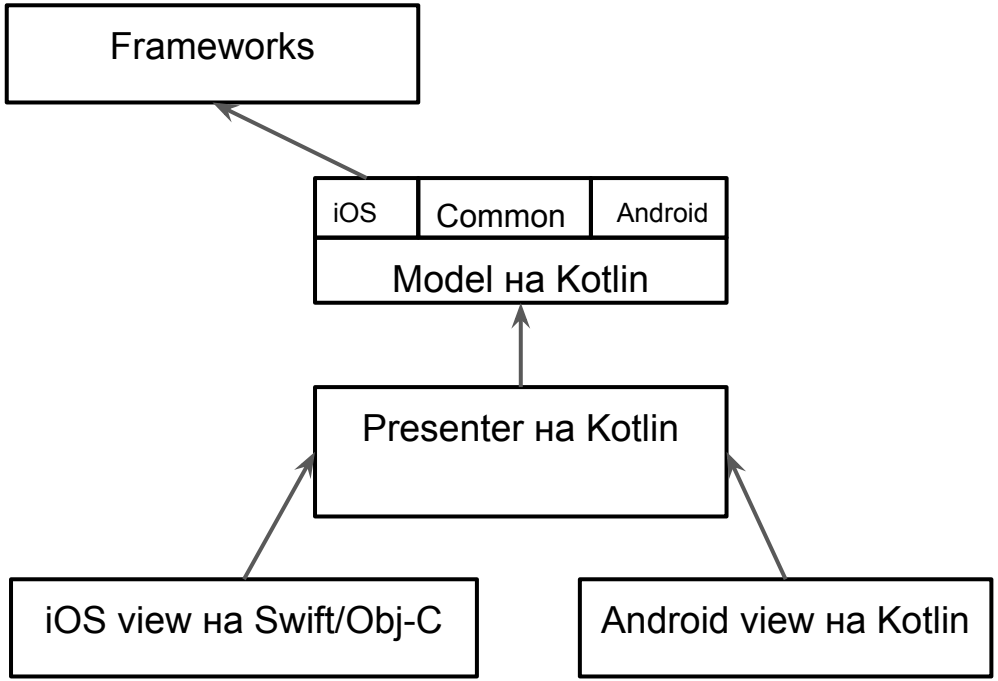
Kotlin/Native: ограничения

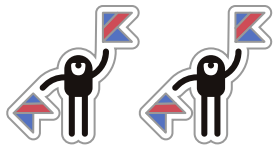


Kotlin/Native для Model-View-Presenter на iOS

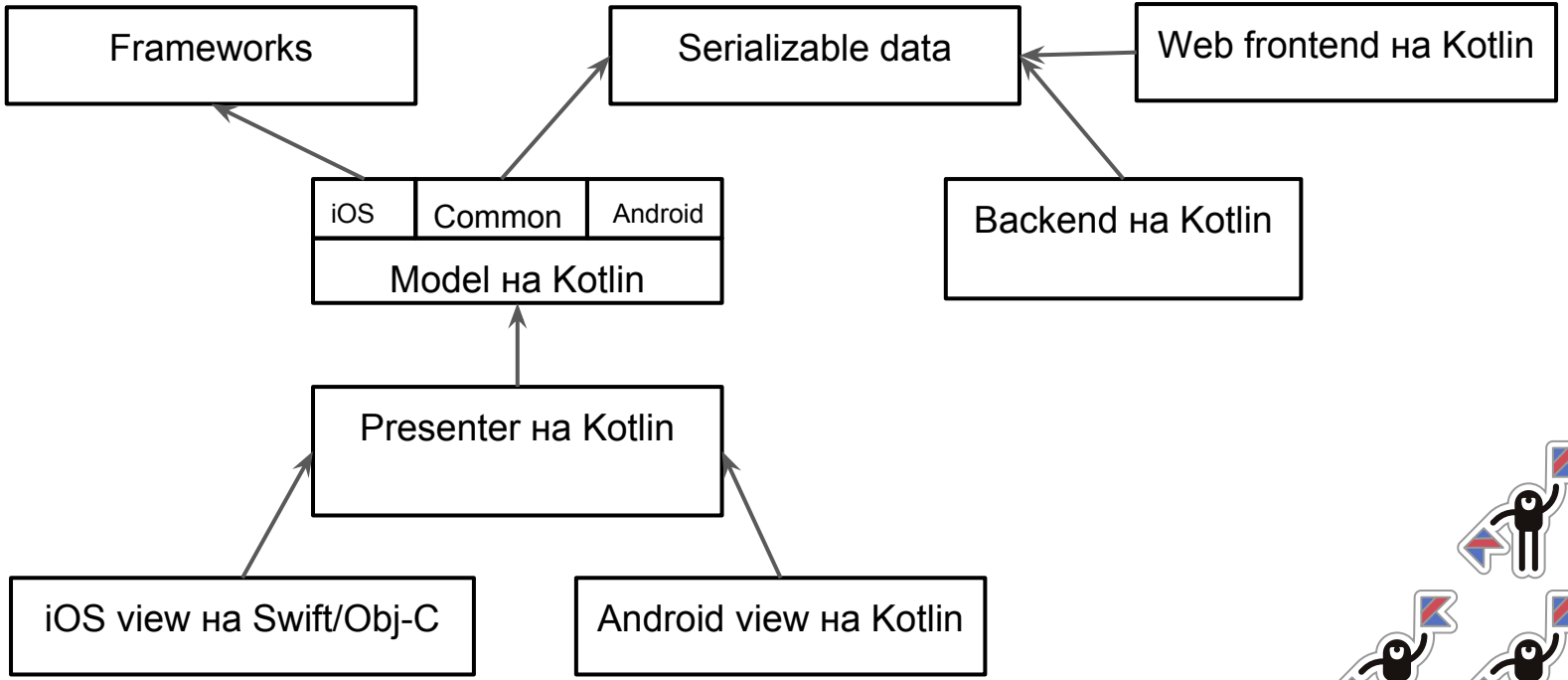


Kotlin для MVP на iOS и Android

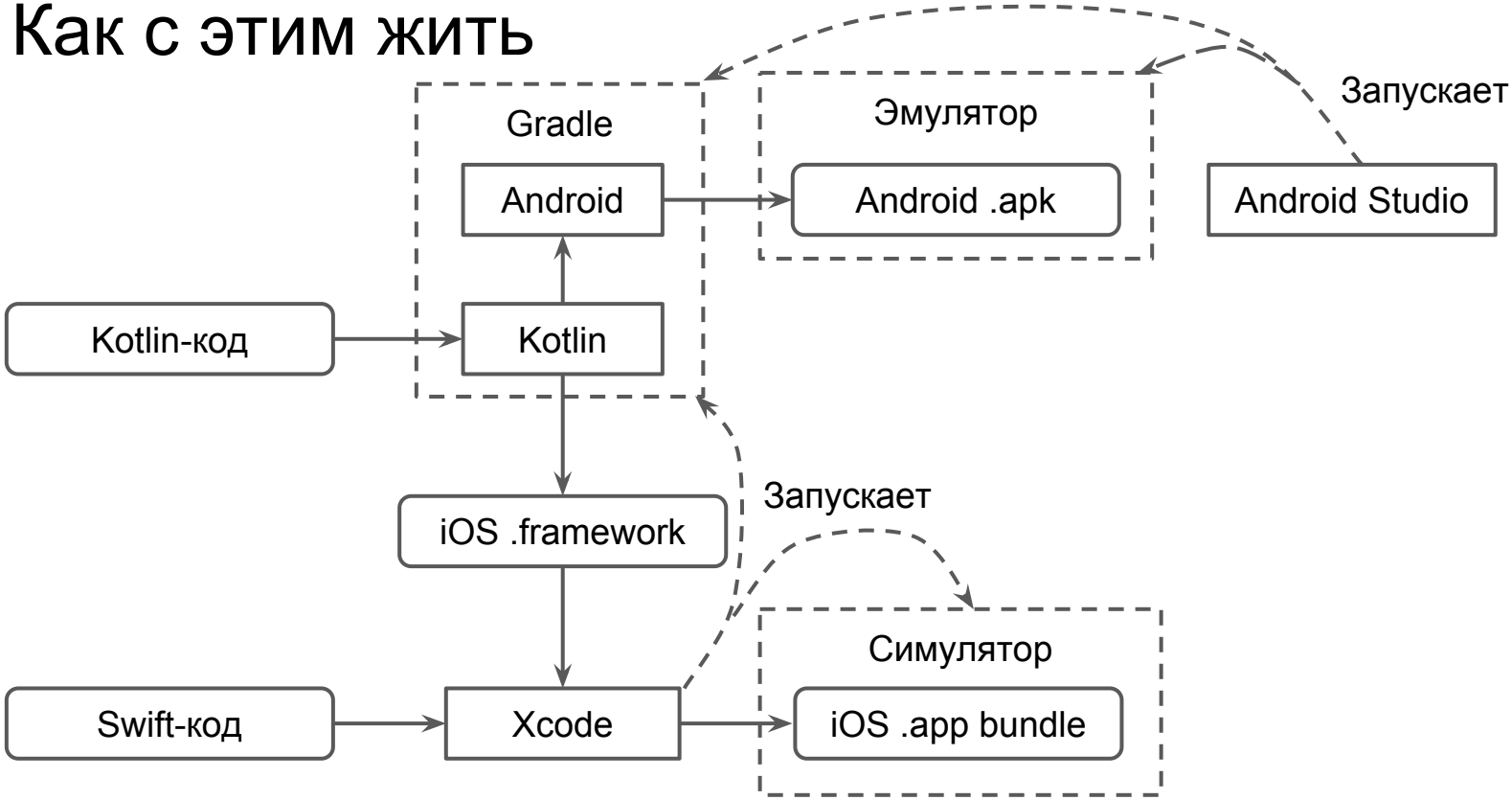




Kotlin для всего



Как с этим жить



Как с этим жить

Наконец-то код!

Резюме

Писать общий код на Kotlin под iOS и Android

- возможно, особенно для бизнес-логики
- бывает трудно, но мы работаем над этим

Ваш фидбек был бы невероятно полезен!

С чего начать?

- IntelliJ IDEA: New project → Kotlin → Kotlin (Mobile Android/iOS)
- https://kotlinlang.org/docs/tutorials/native/mpp-ios-android.html
  - https://github.com/JetBrains/kotlin-examples/tree/master/tutorials/mpp-iOS-Android
- https://github.com/JetBrains/kotlin-mpp-example
- https://github.com/SvyatoslavScherbina/Mobius-2018-Moscow-demo
- https://github.com/JetBrains/kotlinconf-app

Обратная связь:

- https://github.com/JetBrains/kotlin-native/issues
- https://youtrack.jetbrains.com/issues/KT
- #kotlin-native и #kotlin-multiplatform в https://slack.kotlinlang.org