# KOTLIN/NATIVE

*Зачем нужен еще один компилируемый язык в XXI веке?*

Николай Иготти @ JetBrains

# КОНТЕКСТ

- Управляемые среды исполнения - прекрасны!
- Портируемость, безопасность, отлаживаемость, надежность
- Докладчик - JVM-инженер с почти 20-летним стажем, автор курса по виртуальным машинам
- Так зачем же транслировать Котлин в нативный код???

# НЕМНОГО ФИЛОСОФИИ

- Разработка ПО — битва мозга инженера с растущей сложностью задачи
- Сложность задач растет неограниченно
- А инженер ограничен своими «аппаратными» ресурсами
- Можем ли мы победить?

# ЕЩЁ НЕМНОГО ФИЛОСОФИИ

- Пока мы не проиграли ✌️
- Благодаря:
  - стекам технологий, каждый уровень решает свои задачи
  - большинство разработчиков работает на высоком уровне абстракции
  - пирамида технологий, уровни ниже тестируются больше
  - все, **в основном**, работает 😉 😉 😉

# ПОЧЕМУ ЯВА МЕДЛЕННАЯ?

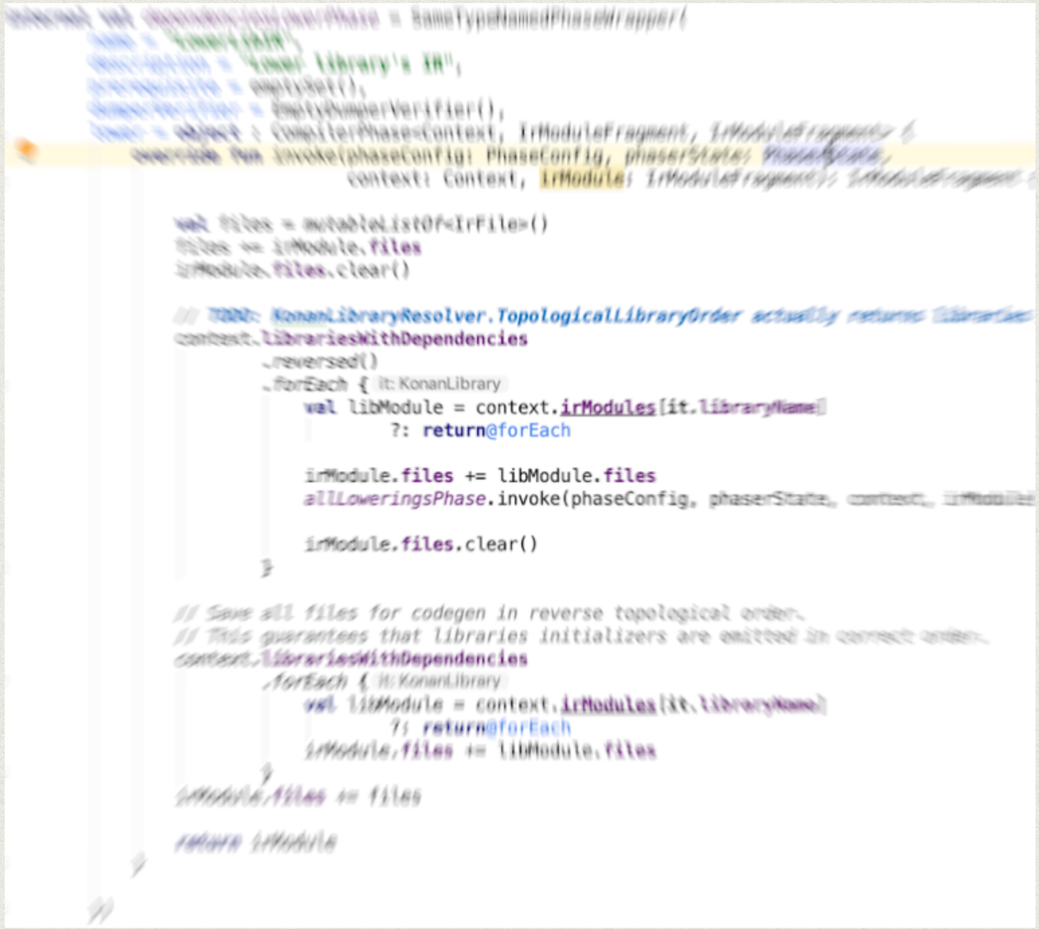
- Java-технология — продукт работы выдающихся инженеров
- Изначально задуманная для встроенных устройств 30-летней давности (Oak)
- Почему же медленная?
  - Время запуска
  - Паузы сборщика
- Фундаментальные причины
  - открытый мир
  - неограниченная простая композируемость программ
  - вера в мощь ВМ

# КАК ЖЕ ИНАЧЕ?

- iOS и macOS — экосистемы почти без ВМ
- Игровые консоли - похожая ситуация
- Закрытые системы с жесткими требованиями к производительности UI
- Разработка приложений ведется ограниченным кругом профессиональных разработчиков
- Программа в закрытом мире и самодостаточна
- Можно сэкономить один уровень абстракции!

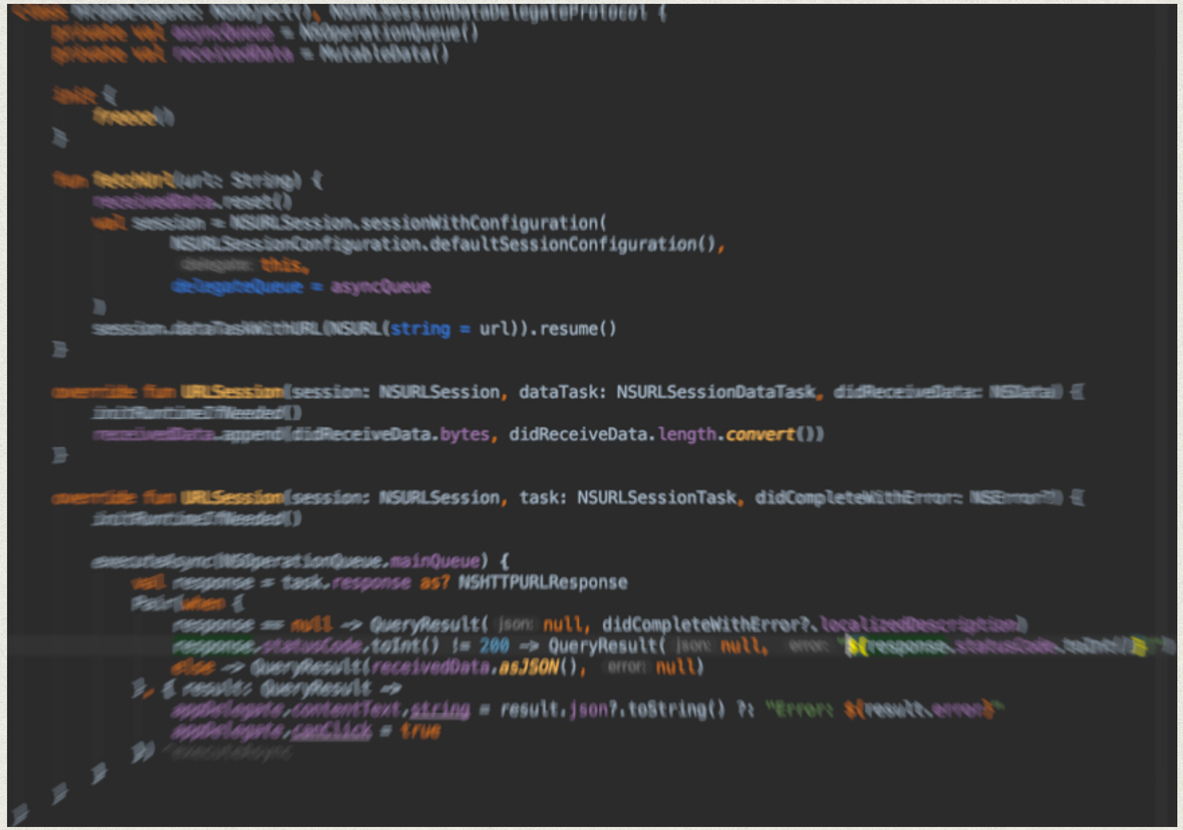
# KOTLIN

- Прагматичный высокоуровневый язык
- С функциональными и ООП конструкциями
- Поддержка в IDE от основной команды
- Хороший интероп с окружением (JVM, JS, C, Objective-C)
- Механизм мультиплатформенной разработки (MPP)



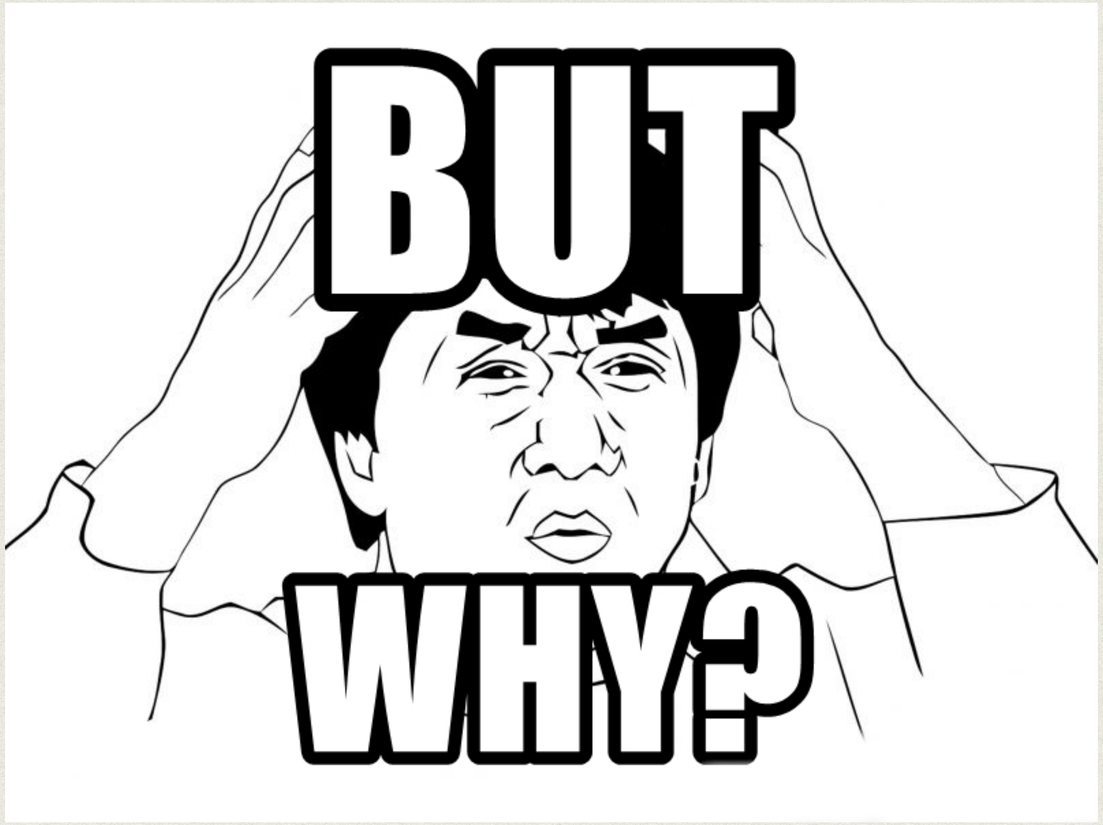
# KOTLIN/NATIVE

- Компилирует программы на чистом Kotlin в машинный код без ВМ
- Диалект Kotlin, общая stdlib, включен в MPP
- Доступен как часть дистрибутива Kotlin (также Kotlin/JVM, Kotlin/JS)
- Поддерживает iOS, macOS, Linux, Windows, Android и др.
- IDE и Gradle работает с Native



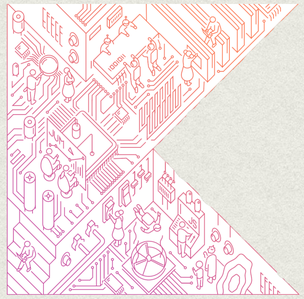
# ЗАЧЕМ KOTLIN/NATIVE?

- JVM/JSVM не всегда желательна
- Удобный доступ к существующим библиотекам на Си/Objective-C
- Компиляция в iOS/macOS
- Микросервисы
- Утилиты командной строки
- Нативные GUI приложения
- MPP таргет



# ПОЧЕМУ НЕ GRAAL VM/ ДРУГОЙ AOT КОМПИЛЯТОР?

- Удобный интероп с C/Objective-C
- Меньше приложение
- Больше поддерживаемых платформ
- Инновации в системе исполнения (заморозка, альтернативная модель конкурентного исполнения)
- Возможность для оптимизаций закрытого мира



# ВОЗМОЖНОСТИ С KOTLIN/NATIVE

- Выход за пределы JVM мира
- С привычным языком, инструментарием и окружением (IDE, Gradle, MPP библиотеки)
- Возможность комбинированных проектов (JVM++) на основе MPP
- Оставаться нативным на каждой платформе (Android, iOS, Web)

# ПРОБЛЕМЫ С КОНКУРЕНТНОСТЬЮ

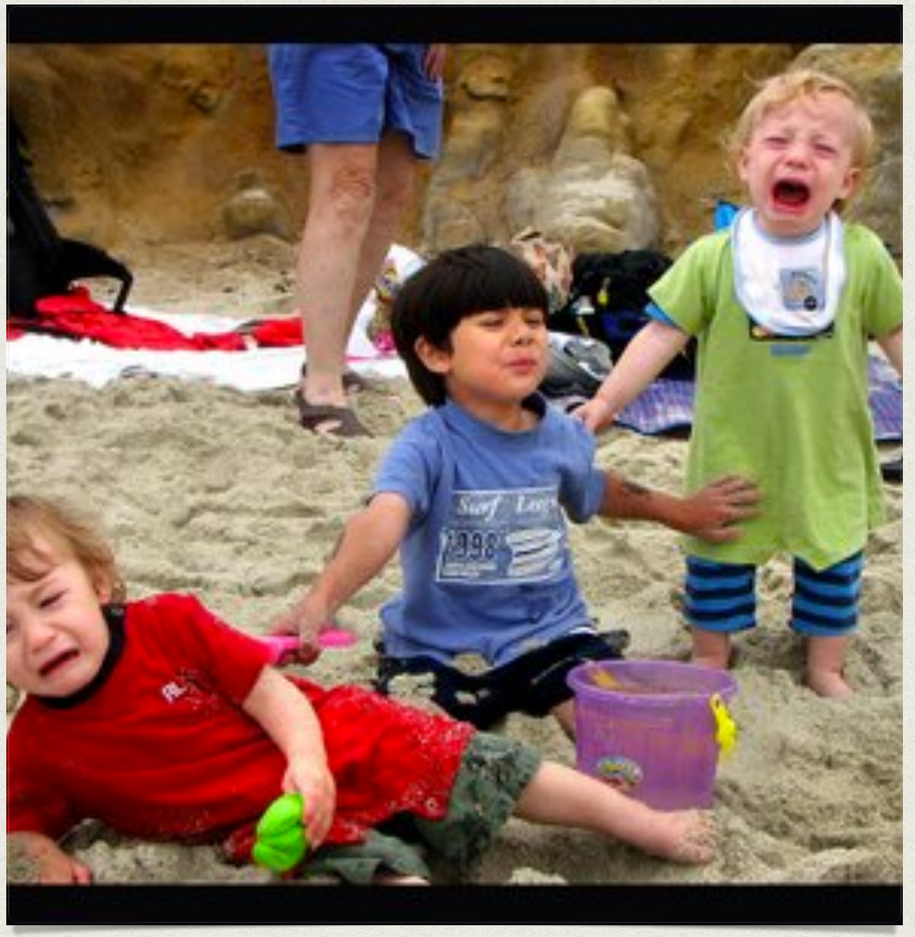
- Корректность по данным — система типов, корректность конкурентности?
- Попробуем лучше?
- Проблемы в ООП языках - одновременный мутирующий доступ к одним и тем же объектам
- **“Изменяемый XOR Разделяемый”**

# КОНКУРЕНТНОСТЬ В KOTLIN/NATIVE

- По умолчанию изменяемые объекты — локальны для потока
- Можно передавать владение подграфом объектов другому потоку
- Неизменяемые объекты можно разделять
- Изменяемый объект можно сделать неизменяемым (`Any.freeze()`)
- Сборка мусора - полностью локальная в потоке
- Простой рантайм, отсутствие глобальных  GC пауз


# СИТУАЦИЯ С ИНТЕРОПЕРАБЕЛЬНОСТЬЮ

- Много экосистем - много языков, библиотек, инструментов
- Интероперабельность обычно сложна в использовании (JNI)
- Интероперабельность обычно сложна в реализации, особенно при отсутствии общей системы исполнения (JVM, JS VM)
- Для Kotlin качественная интероперабельность с разными языками — фундаментальное свойство



# ИНТЕРОПЕРАБЕЛЬНОСТЬ В KOTLIN/NATIVE

- Полная поддержка C, Objective-C и Swift (через Objective-C) используя libclang
- Трансляция понятий между языками (указатели, управление памятью, категории, селекторы, etc.)
- Возможность создания артефактов для других языков (фреймворки, статические и динамические библиотеки и заголовки)
- Платформенные библиотеки (например, UIKit в iOS) доступны автоматически

# БУДУЩЕЕ KOTLIN/NATIVE

- Развитие вместе с Kotlin/JVM и Kotlin/JS
- Интероп с другими языками (C++, Swift, ???)
- Поддержка более привычной модели параллельных вычислений
- Новые MPP библиотеки в универсальном формате на основе IR
- Компиляторные плагины для метапрограммирования
- И...?

# ВЫ МОЖЕТЕ ПОЙТИ С НАМИ

*Это интересно, и совсем нетрудно!*

*https://bit.ly/2y3YJzI*