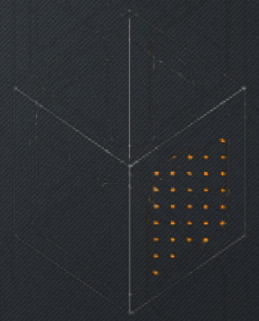

Software Engineering Conference Russia 2018

October 12-13
Moscow

KOTLIN для быстрой разработки в iOS и ANDROID

Николай Иготти

JetBrains

О чем пойдет речь

О чем пойдет речь

- Что такое Kotlin и Kotlin/Native

О чем пойдет речь

- Что такое Kotlin и Kotlin/Native
- Возможности Kotlin/Native

О чем пойдет речь

- Что такое Kotlin и Kotlin/Native
- Возможности Kotlin/Native
- Примеры взаимодействия с C, Objective-C и Swift

О чем пойдет речь

- Что такое Kotlin и Kotlin/Native
- Возможности Kotlin/Native
- Примеры взаимодействия с C, Objective-C и Swift
- Технология Kotlin/Multiplatform

О чем пойдет речь

- Что такое Kotlin и Kotlin/Native
- Возможности Kotlin/Native
- Примеры взаимодействия с C, Objective-C и Swift
- Технология Kotlin/Multiplatform
- Gradle и MPP

О чем пойдет речь

- Что такое Kotlin и Kotlin/Native
- Возможности Kotlin/Native
- Примеры взаимодействия с C, Objective-C и Swift
- Технология Kotlin/Multiplatform
- Gradle и MPP
- Мультиплатформенные библиотеки

О чем пойдет речь

- Что такое Kotlin и Kotlin/Native
- Возможности Kotlin/Native
- Примеры взаимодействия с C, Objective-C и Swift
- Технология Kotlin/Multiplatform
- Gradle и MPP
- Мультиплатформенные библиотеки
- Доступность решения

ОБЩИЕ МОМЕНТЫ: KOTLIN

```
fun generateCAdapter() {
    when {
        isFunction -> {
            val function : FunctionDescriptor = declaration as FunctionDescriptor
            cname = "_konan_function_${owner.nextFunctionIndex()}"
            val llvmFunction : LLVMValueRef = owner.codegen.llvmFunction(function)
            // If function is virtual, we need to resolve receiver properly.
            val bridge : LLVMValueRef = if (!DescriptorUtils.isTopLevelDeclaration(function) &&
                    !function.isExtension && function.isOverridable) {
                // We need LLVMGetElementType() as otherwise type is function pointer.
                generateFunction(owner.codegen, LLVMGetElementType(llvmFunction.type)!!, cname) { this: FunctionGenerationC
                    val receiver : LLVMValueRef = param( index: 0)
                    val numParams : Int = LLVMCountParams(llvmFunction)
                    val args : List<LLVMValueRef> = (0..numParams - 1).map { index -> param(index) }
                    val callee : LLVMValueRef = lookupVirtualImpl(receiver, function)
                    val result : LLVMValueRef = call(callee, args,
                            exceptionHandler = ExceptionHandler.Caller, verbatim = true)
                    ret(result)
                }
            } else {
                LLVMAddAlias(context.llvmModule, llvmFunction.type, llvmFunction, cname)!!
            }
            LLVMSetLinkage(bridge, LLVMLinkage.LLVMExternalLinkage)
        }
        isClass -> {
            // Produce type getter.
            cname = "_konan_function_${owner.nextFunctionIndex()}"
            val getTypeFunction : LLVMValueRef = LLVMAddFunction(context.llvmModule, cname, owner.kGetTypeFuncType)!!
            val builder : LLVMBuilderRef = LLVMCreateBuilder()!!
            val bb : LLVMBasicBlockRef = LLVMAppendBasicBlock(getTypeFunction, Name: "")!!
            LLVMPositionBuilderAtEnd(builder, bb)
            LLVMBuildRet(builder, (declaration as ClassDescriptor).typeInfoPtr.llvm)
            LLVMDisposeBuilder(builder)
        }
```

KOTLIN B ANDROID

KTX

KOTLIN/NATIVE

KOTLIN/NATIVE

- Компилятор Kotlin в машинный код (.so, .dll, .dylib, .exe)

KOTLIN/NATIVE

- Компилятор Kotlin в машинный код (.so, .dll, .dylib, .exe)
- Использует LLVM 6.0.1

KOTLIN/NATIVE

- Компилятор Kotlin в машинный код (.so, .dll, .dylib, .exe)
- Использует LLVM 6.0.1
- Получаемые исполняемые артефакты полностью самодостаточны (никаких ВМ, служебных библиотек и т.п.)

KOTLIN/NATIVE

- Компилятор Kotlin в машинный код (.so, .dll, .dylib, .exe)
- Использует LLVM 6.0.1
- Получаемые исполняемые артефакты полностью самодостаточны (никаких ВМ, служебных библиотек и т.п.)
- Работает на macOS, Windows, Linux

KOTLIN/NATIVE

- Компилятор Kotlin в машинный код (.so, .dll, .dylib, .exe)
- Использует LLVM 6.0.1
- Получаемые исполняемые артефакты полностью самодостаточны (никаких ВМ, служебных библиотек и т.п.)
- Работает на macOS, Windows, Linux
- Порождает исполняемый код для iOS, macOS, Linux (x86-64, ARM, MIPS), Android (sic!), Windows и WebAssembly

KOTLIN/NATIVE

- Компилятор Kotlin в машинный код (.so, .dll, .dylib, .exe)
- Использует LLVM 6.0.1
- Получаемые исполняемые артефакты полностью самодостаточны (никаких ВМ, служебных библиотек и т.п.)
- Работает на macOS, Windows, Linux
- Порождает исполняемый код для iOS, macOS, Linux (x86-64, ARM, MIPS), Android (sic!), Windows и WebAssembly
- Из коробки доступны все системные библиотеки на C/Objective-C

KOTLIN/NATIVE

- Компилятор Kotlin в машинный код (.so, .dll, .dylib, .exe)
- Использует LLVM 6.0.1
- Получаемые исполняемые артефакты полностью самодостаточны (никаких ВМ, служебных библиотек и т.п.)
- Работает на macOS, Windows, Linux
- Порождает исполняемый код для iOS, macOS, Linux (x86-64, ARM, MIPS), Android (sic!), Windows и WebAssembly
- Из коробки доступны все системные библиотеки на C/Objective-C
- Новые нативные библиотеки на на C/Objective-C добавлять очень просто

Kotlin/Native для iOS

Kotlin/Native для iOS

- Автоматическое управление памятью (как и другие платформы)

Kotlin/Native для iOS

- Автоматическое управление памятью (как и другие платформы)
- Прозрачное взаимодействие в обе стороны с Objective-C/Swift, с автоматическим управлением памятью

Kotlin/Native для iOS

- Автоматическое управление памятью (как и другие платформы)
- Прозрачное взаимодействие в обе стороны с Objective-C/Swift, с автоматическим управлением памятью
- Базовые типы и коллекции автоматически отображаются 1:1

Kotlin/Native для iOS

- Автоматическое управление памятью (как и другие платформы)
- Прозрачное взаимодействие в обе стороны с Objective-C/Swift, с автоматическим управлением памятью
- Базовые типы и коллекции автоматически отображаются 1:1
- Порождаются полноценные бинарные приложения или стандартные фреймворки с интерфейсом Objective-C

Kotlin/Native для iOS

- Автоматическое управление памятью (как и другие платформы)
- Прозрачное взаимодействие в обе стороны с Objective-C/Swift, с автоматическим управлением памятью
- Базовые типы и коллекции автоматически отображаются 1:1
- Порождаются полноценные бинарные приложения или стандартные фреймворки с интерфейсом Objective-C
- Поддержаны все основные архитектуры iOS, включая armv7

Kotlin/Native для iOS

- Автоматическое управление памятью (как и другие платформы)
- Прозрачное взаимодействие в обе стороны с Objective-C/Swift, с автоматическим управлением памятью
- Базовые типы и коллекции автоматически отображаются 1:1
- Порождаются полноценные бинарные приложения или стандартные фреймворки с интерфейсом Objective-C
- Поддержаны все основные архитектуры iOS, включая armv7
- Байндинги для всех основных SDK от Apple включены в поставку

Kotlin/Native для iOS

- Автоматическое управление памятью (как и другие платформы)
- Прозрачное взаимодействие в обе стороны с Objective-C/Swift, с автоматическим управлением памятью
- Базовые типы и коллекции автоматически отображаются 1:1
- Порождаются полноценные бинарные приложения или стандартные фреймворки с интерфейсом Objective-C
- Поддержаны все основные архитектуры iOS, включая armv7
- Байндинги для всех основных SDK от Apple включены в поставку
- Возможна отладка из XCode (полная отладочная информация в формате DWARF)

Примеры (1)

```
val windowStyle = NSWindowStyleMaskTitled or NSWindowStyleMaskMiniaturizable or
        NSWindowStyleMaskClosable or NSWindowStyleMaskResizable

window = NSWindow(windowRect, windowStyle, NSBackingStoreBuffered, false).apply {
    title = "Окошко Konan"
    opaque = true
    hasShadow = true
    preferredBackingLocation = NSWindowBackingLocationVideoMemory
    hidesOnDeactivate = false
    backgroundColor = NSColor.whiteColor()
    releasedWhenClosed = false

    delegate = object : NSObject(), NSWindowDelegateProtocol {
        override fun windowShouldClose(sender: NSWindow): Boolean {
            NSApplication.sharedApplication().stop(this)
            return true
        }
    }
}

val button = NSButton(NSMakeRect(10.0, 10.0, 100.0, 40.0)).apply {
    title = "Press me"
    target = controller
    action = NSSelectorFromString("onClick")
}
window.contentView!!.addSubview(button)
```

Примеры (2)

```
// select projection matrix
glMatrixMode(GL_PROJECTION)

// set the viewport
glViewport(0, 0, windowWidth, windowHeight)

// set matrix mode
glMatrixMode(GL_PROJECTION)

// reset projection matrix
glLoadIdentity()
val aspect = windowWidth.toDouble() / windowHeight

// set up a perspective projection matrix
gluPerspective(45.0, aspect, 1.0, 500.0)

// specify which matrix is the current matrix
glMatrixMode(GL_MODELVIEW)
glShadeModel(GL_SMOOTH)

// specify the clear value for the depth buffer
glClearDepth(1.0)
glEnable(GL_DEPTH_TEST)
glDepthFunc(GL_LEQUAL)

// specify implementation-specific hints
glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST)

glLightModelfv(GL_LIGHT_MODEL_AMBIENT, cValuesOf(0.1f, 0.1f, 0.1f, 1.0f))
glLightfv(GL_LIGHT0, GL_DIFFUSE, cValuesOf(0.6f, 0.6f, 0.6f, 1.0f))
glLightfv(GL_LIGHT0, GL_SPECULAR, cValuesOf(0.7f, 0.7f, 0.3f, 1.0f))

glEnable(GL_LIGHT0)
glEnable(GL_COLOR_MATERIAL)
glShadeModel(GL_SMOOTH)
glLightModeli(GL_LIGHT_MODEL_TWO_SIDE, GL_FALSE)
glDepthFunc(GL_LEQUAL)
glEnable(GL_DEPTH_TEST)
glEnable(GL_LIGHTING)
glEnable(GL_LIGHT0)
glClearColor(0.0f, 0.0f, 0.0f, 1.0f)
```

Примеры (3)

```
interface PartialExpressionComposer<E : Any, PE : Any> {
    fun missing(): PE
    fun ending(expression: E): PE

    fun plus(left: E, partialRight: PE): PE
    fun minus(left: E, partialRight: PE): PE
    fun mult(left: E, partialRight: PE): PE
    fun div(left: E, partialRight: PE): PE

    fun leftParenthesized(partialExpression: PE): PE
}
```

```
interface ExpressionComposer<E : Any> {
    fun number(value: Double): E
    fun plus(left: E, right: E): E
    fun minus(left: E, right: E): E
    fun mult(left: E, right: E): E
    fun div(left: E, right: E): E
}
```

```
class Calculator : ExpressionComposer<Double> {
    override fun number(value: Double) = value
    override fun plus(left: Double, right: Double) = left + right
    override fun minus(left: Double, right: Double) = left - right
    override fun mult(left: Double, right: Double) = left * right
    override fun div(left: Double, right: Double) = left / right
}
```

```
private class PartialRenderer: NSObject, KAPPartialExpressionComposer {
    func plus(left: Any, partialRight: Any) -> Any {
        return compose("+", left, partialRight)
    }

    func minus(left: Any, partialRight: Any) -> Any {
        return compose("-", left, partialRight)
    }

    func mult(left: Any, partialRight: Any) -> Any {
        return compose("*", left, partialRight)
    }

    func div(left: Any, partialRight: Any) -> Any {
        return compose("/", left, partialRight)
    }

    private func compose(_ op: String, _ left: Any, _ partialRight: Any) -> Any {
        return "\(formatDouble(left)) \(op) ".toAttributed() +
            (partialRight as! NSAttributedString)
    }
```

```
private let parser = KAPPartialParser(composer: KAPCalculator(),
                                      partialComposer: PartialRenderer())
```

Мультиплатформенная разработка в Kotlin 1.3

Мультиплатформенная разработка в Kotlin 1.3

- Поддержка JVM (сервер, Android) / JS (браузер, Node.js) / Native (iOS, Windows, Linux, macOS) в одном проекте

Мультиплатформенная разработка в Kotlin 1.3

- Поддержка JVM (сервер, Android) / JS (браузер, Node.js) / Native (iOS, Windows, Linux, macOS) в одном проекте
- Расширения языка expect/actual

Мультиплатформенная разработка в Kotlin 1.3

- Поддержка JVM (сервер, Android) / JS (браузер, Node.js) / Native (iOS, Windows, Linux, macOS) в одном проекте
- Расширения языка expect/actual
- Мультиплатформенная стандартная библиотека

Мультиплатформенная разработка в Kotlin 1.3

- Поддержка JVM (сервер, Android) / JS (браузер, Node.js) / Native (iOS, Windows, Linux, macOS) в одном проекте
- Расширения языка expect/actual
- Мультиплатформенная стандартная библиотека
- Стандартизованный Gradle DSL для JVM/JS/Native

Мультиплатформенная разработка в Kotlin 1.3

- Поддержка JVM (сервер, Android) / JS (браузер, Node.js) / Native (iOS, Windows, Linux, macOS) в одном проекте
- Расширения языка expect/actual
- Мультиплатформенная стандартная библиотека
- Стандартизованный Gradle DSL для JVM/JS/Native
- Набор стандартных MPP библиотек от JetBrains:

Мультиплатформенная разработка в Kotlin 1.3

- Поддержка JVM (сервер, Android) / JS (браузер, Node.js) / Native (iOS, Windows, Linux, macOS) в одном проекте
- Расширения языка expect/actual
- Мультиплатформенная стандартная библиотека
- Стандартизованный Gradle DSL для JVM/JS/Native
- Набор стандартных MPP библиотек от JetBrains:
  - kotlinx.coroutines — утилиты для асинхронного программирования

Мультиплатформенная разработка в Kotlin 1.3

- Поддержка JVM (сервер, Android) / JS (браузер, Node.js) / Native (iOS, Windows, Linux, macOS) в одном проекте
- Расширения языка expect/actual
- Мультиплатформенная стандартная библиотека
- Стандартизованный Gradle DSL для JVM/JS/Native
- Набор стандартных MPP библиотек от JetBrains:
  - kotlinx.coroutines — утилиты для асинхронного программирования
  - kotlinx.serialization — сериализация/десериализация в JSON и proto

Мультиплатформенная разработка в Kotlin 1.3

- Поддержка JVM (сервер, Android) / JS (браузер, Node.js) / Native (iOS, Windows, Linux, macOS) в одном проекте
- Расширения языка expect/actual
- Мультиплатформенная стандартная библиотека
- Стандартизованный Gradle DSL для JVM/JS/Native
- Набор стандартных MPP библиотек от JetBrains:
  - kotlinx.coroutines — утилиты для асинхронного программирования
  - kotlinx.serialization — сериализация/десериализация в JSON и proto
  - ktor — HTTP клиент и фреймворк

Мультиплатформенная разработка в Kotlin 1.3

- Поддержка JVM (сервер, Android) / JS (браузер, Node.js) / Native (iOS, Windows, Linux, macOS) в одном проекте
- Расширения языка expect/actual
- Мультиплатформенная стандартная библиотека
- Стандартизованный Gradle DSL для JVM/JS/Native
- Набор стандартных MPP библиотек от JetBrains:
  - kotlinx.coroutines — утилиты для асинхронного программирования
  - kotlinx.serialization — сериализация/десериализация в JSON и proto
  - ktor — HTTP клиент и фреймворк
- Сторонние MPP библиотеки, например sessionize от Kevin Galligan

Расширения языка для MPP

```
expect class Product

expect object Factory {
    fun create(config: Map<String, String>): Product
    val platform: String
}
```

```
actual class Platform actual constructor() {
    actual val platform: String = "Android"
}

actual class Product(val user: String) {
    fun androidSpecificOperation() {
        println("I am ${Build.MODEL} by ${Build.MANUFACTURER}")
    }
    override fun toString() = "Android product for ${Build.MODEL}"
}

actual object Factory {
    actual fun create(config: Map<String, String>) =
            Product(config["user"]!!)
    actual val platform: String = "android"
}
```

```
actual class Product(val user: String) {
    val model: String = memScoped {
        val systemInfo = alloc<utsname>()
        uname(systemInfo.ptr)
        ^memScoped systemInfo.machine.toKString()
    }

    fun iosSpecificOperation() {
        println("I am $model")
    }

    override fun toString() = "iOS product for $model"
}

actual object Factory {
    actual fun create(config: Map<String, String>) =
            Product(config["user"]!!)
    actual val platform: String = "ios"
}
```

Стандартизованный Gradle DSL

```
kotlin {
    sourceSets {
        commonMain {
            dependencies {
                implementation 'com.example:sample-lib:1.0'
                implementation 'org.jetbrains.kotlin:kotlin-stdlib'
            }
        }
        allJvm {
            dependsOn commonMain
            dependencies {
                implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-core:0.23.4'
            }
        }
        jvm6Main.dependsOn allJvm
        jvm8Main.dependsOn allJvm
    }

    targets {
        fromPreset(presets.jvm, 'jvm6') {
            compilations.main.source(sourceSets.allJvm)
        }
        fromPreset(presets.jvm, 'jvm8') {
            compilations.main.source(sourceSets.allJvm)
        }
        fromPreset(presets.js, 'nodeJs')

        fromPreset(presets.linuxX64, 'linux')
        fromPreset(presets.mingwX64, 'windows')
        fromPreset(presets.macosX64, 'mac')

        configure([linux, windows, mac]) {
            compilations.main.outputKinds('EXECUTABLE')
        }
    }
}
```

Библиотеки для MPP от JetBrains

- kotlinx.coroutines

```
fun produceNumbers() = produce { this: ProducerScope<Int>
    var x = 1
    while (true) send(x++) // infinite stream of integers starting from 1
}

fun main(args: Array<String>) = runBlocking<Unit> { this: CoroutineScope
    val numbers = produceNumbers() // produces integers from 1 and on
    for (i in 1..5) println(numbers.receive()) // print first five
    numbers.cancel()
}
```

- kotlinx.seralization +
- HTTP client из ktor

```
fun performRequest(path: String) {
    val client = HttpClient()
    launch(Unconfined) {
        val response = client.call {
            url {
                protocol = URLProtocol.HTTPS
                host = "jsonplaceholder.typicode.com"
                encodedPath = path
                port = 443
            }
        }.response

        if (!response.status.isSuccess()) throw IllegalStateException("HTTP response code ${response.status}")
        val body = response.content.readRemaining().readText()
        client.close()
        val json = JsonTreeParser(body).readFully()
        println(jsonContent(json))
        println(json.jsonObject.getPrimitive("key").int)
    }
}
```

Библиотеки для MPP от JetBrains

- kotlinx.coroutines

```
fun produceNumbers() = produce { this: ProducerScope<Int>
    var x = 1
    while (true) send(x++) // infinite stream of integers starting from 1
}

fun main(args: Array<String>) = runBlocking<Unit> { this: CoroutineScope
    val numbers = produceNumbers() // produces integers from 1 and on
    for (i in 1..5) println(numbers.receive()) // print first five
    numbers.cancel()
}
```

- kotlinx.seralization +
- HTTP client из ktor

```
fun performRequest(path: String) {
    val client = HttpClient()
    launch(Unconfined) {
        val response = client.call {
            url {
                protocol = URLProtocol.HTTPS
                host = "jsonplaceholder.typicode.com"
                encodedPath = path
                port = 443
            }
        }.response

        if (!response.status.isSuccess()) throw IllegalStateException("HTTP response code ${response.status}")
        val body = response.content.readRemaining().readText()
        client.close()
        val json = JsonTreeParser(body).readFully()
        println(jsonContent(json))
        println(json.jsonObject.getPrimitive("key").int)
    }
}
```

Библиотеки для MPP от JetBrains

- kotlinx.coroutines

```
fun produceNumbers() = produce { this: ProducerScope<Int>
    var x = 1
    while (true) send(x++) // infinite stream of integers starting from 1
}

fun main(args: Array<String>) = runBlocking<Unit> { this: CoroutineScope
    val numbers = produceNumbers() // produces integers from 1 and on
    for (i in 1..5) println(numbers.receive()) // print first five
    numbers.cancel()
}
```

- kotlinx.seralization +
- HTTP client из ktor

```
fun performRequest(path: String) {
    val client = HttpClient()
    launch(Unconfined) {
        val response = client.call {
            url {
                protocol = URLProtocol.HTTPS
                host = "jsonplaceholder.typicode.com"
                encodedPath = path
                port = 443
            }
        }.response

        if (!response.status.isSuccess()) throw IllegalStateException("HTTP response code ${response.status}")
        val body = response.content.readRemaining().readText()
        client.close()
        val json = JsonTreeParser(body).readFully()
        println(jsonContent(json))
        println(json.jsonObject.getPrimitive("key").int)
    }
}
```

Создание практических мобильных решений

Создание практических мобильных решений

- Поддержка MPP, включая Native, в Kotlin 1.3 прямо сейчас (осень 2018)

Создание практических мобильных решений

- Поддержка MPP, включая Native, в Kotlin 1.3 прямо сейчас (осень 2018)
- Значительную часть кода проекта можно написать на Kotlin

Создание практических мобильных решений

- Поддержка MPP, включая Native, в Kotlin 1.3 прямо сейчас (осень 2018)
- Значительную часть кода проекта можно написать на Kotlin
- Не надо ограничивать себя узким уровнем портирования - весь API платформы доступен

Создание практических мобильных решений

- Поддержка MPP, включая Native, в Kotlin 1.3 прямо сейчас (осень 2018)
- Значительную часть кода проекта можно написать на Kotlin
- Не надо ограничивать себя узким уровнем портирования - весь API платформы доступен
- Инструменты от JetBrains (IDEA, включая Community Edition) и Android Studio способны работать с MPP кодом (включая навигацию expect/actual)

Создание практических мобильных решений

- Поддержка MPP, включая Native, в Kotlin 1.3 прямо сейчас (осень 2018)
- Значительную часть кода проекта можно написать на Kotlin
- Не надо ограничивать себя узким уровнем портирования - весь API платформы доступен
- Инструменты от JetBrains (IDEA, включая Community Edition) и Android Studio способны работать с MPP кодом (включая навигацию expect/actual)
- Gradle знает о MPP проектах, может их собирать и публиковать

Создание практических мобильных решений

- Поддержка MPP, включая Native, в Kotlin 1.3 прямо сейчас (осень 2018)
- Значительную часть кода проекта можно написать на Kotlin
- Не надо ограничивать себя узким уровнем портирования - весь API платформы доступен
- Инструменты от JetBrains (IDEA, включая Community Edition) и Android Studio способны работать с MPP кодом (включая навигацию expect/actual)
- Gradle знает о MPP проектах, может их собирать и публиковать
- Доступен основной набор MPP библиотек от JetBrains

Создание практических мобильных решений

- Поддержка MPP, включая Native, в Kotlin 1.3 прямо сейчас (осень 2018)
- Значительную часть кода проекта можно написать на Kotlin
- Не надо ограничивать себя узким уровнем портирования - весь API платформы доступен
- Инструменты от JetBrains (IDEA, включая Community Edition) и Android Studio способны работать с MPP кодом (включая навигацию expect/actual)
- Gradle знает о MPP проектах, может их собирать и публиковать
- Доступен основной набор MPP библиотек от JetBrains
- Мы в JetBrains постоянно улучшаем платформу и исправляем ошибки

Более быстрая и качественная разработка мобильного приложения

Более быстрая и качественная разработка мобильного приложения

- Бизнес логику можно писать на MPP Kotlin’е прямо сейчас

Более быстрая и качественная разработка мобильного приложения

- Бизнес логику можно писать на MPP Kotlin’е прямо сейчас
- UI и логику взаимодействия с платформенными интерфейсами можно писать по желанию на Kotlin’е или Swift’е или Objective-C

Более быстрая и качественная разработка мобильного приложения

- Бизнес логику можно писать на MPP Kotlin’е прямо сейчас
- UI и логику взаимодействия с платформенными интерфейсами можно писать по желанию на Kotlin’е или Swift’е или Objective-C
- В приложении на iOS можно попробовать Kotlin постепенно, переведя на Kotlin одну компоненту (режим framework)

Более быстрая и качественная разработка мобильного приложения

- Бизнес логику можно писать на MPP Kotlin’е прямо сейчас
- UI и логику взаимодействия с платформенными интерфейсами можно писать по желанию на Kotlin’е или Swift’е или Objective-C
- В приложении на iOS можно попробовать Kotlin постепенно, переведя на Kotlin одну компоненту (режим framework)
- Можно быстрее получать работающее мобильное приложение под основные платформ

Более быстрая и качественная разработка мобильного приложения

- Бизнес логику можно писать на MPP Kotlin’е прямо сейчас
- UI и логику взаимодействия с платформенными интерфейсами можно писать по желанию на Kotlin’е или Swift’е или Objective-C
- В приложении на iOS можно попробовать Kotlin постепенно, переведя на Kotlin одну компоненту (режим framework)
- Можно быстрее получать работающее мобильное приложение под основные платформ
- Попробуйте сегодня 😍😍😍

Обратная связь

- Николай Иготти

- nikolay.igotti@jetbrains.com

- https://kotlinlang.org/docs/reference/native-overview.html
- https://ru.coursera.org/learn/vvedenie-v-yazyk-kotlin