

Gradle Kotlin DSL



Андрей
Данилов

Infrastructure Lead @ Yandex



Mobius
2023 Spring

Yandex

Рекламный SDK Яндекса

 Яндекс.Такси

Зарабатывайте до 6 000 ₽/день* в Санкт-Петербурге

Партнёр Яндекс.Такси
*Различные индивидуальные заявки от каждого конкретного случая для получения подробной информации по комиссии и другим условиям отправки заявки на подключение. Подробнее на [https://yandex.ru/about/taxi/partner/](#)

Яндекс.Директ 3+



Яндекс.Такси
taxi.yandex.ru
 228 Бесплатно

Подача машины в среднем 4-5 минут. Следите за вашим такси на карте

Указанная в приложении Яндекс.Такси расчётная стоимость поездки может быть изменена в случае изменения пункта назначения во время поездки.

[Перейти](#)

 afisha.yandex.ru РЕКЛАМА · 16+ Я

Все театры и спектакли вашего города на Яндекс.Афише!

РЕКЛАМА



realty.yandex.ru

Яндекс.Недвижимость →



Яндекс.Директ 18+

КиноПоиск: фильмы в HD и сериалы онлайн
Google Play
 302 298 Бесплатно

Новые серии в день выхода и тысячи фильмов на вечер. Без рекламы!

[Скачать](#)

Рекламный SDK Яндекса

350 000

строк кода
в SDK

11 000

Unit
тестов

1200

UI
тестов

**Unity
Flutter**
интеграции

Статистика

по коду
и билдам

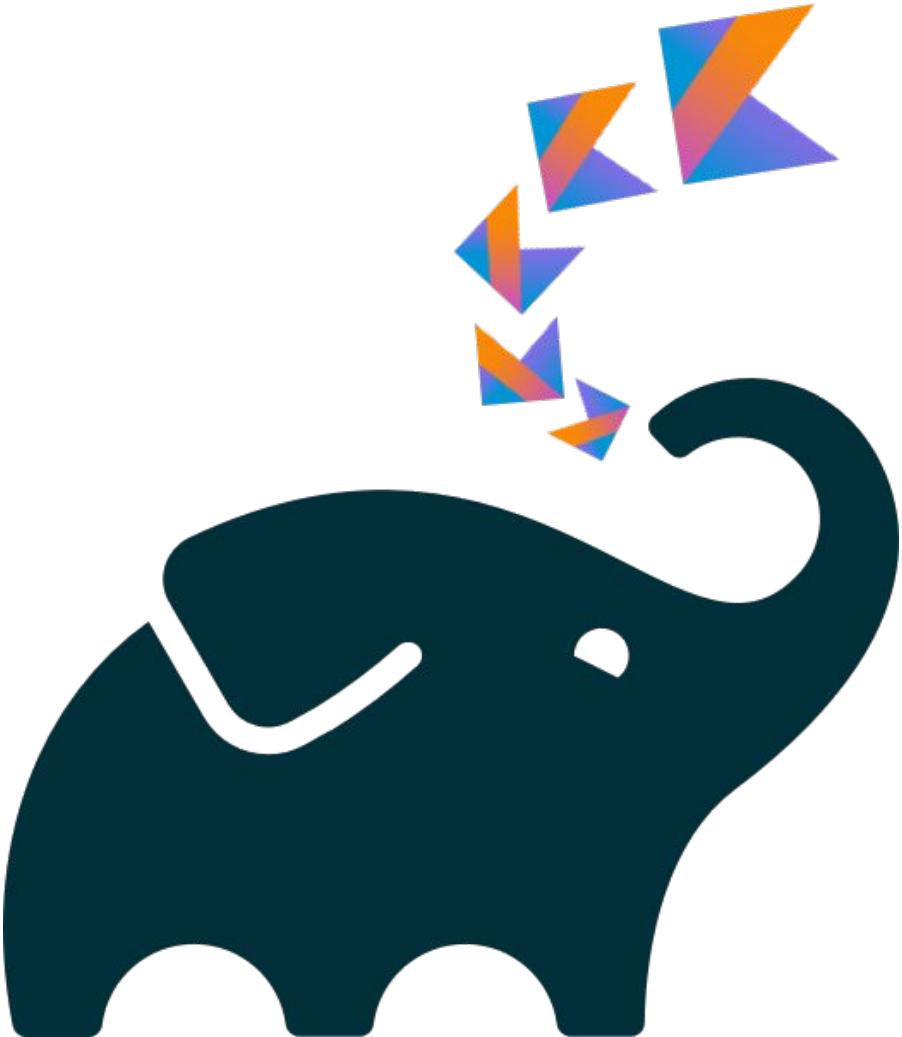
Kotlin DSL

О чём доклад

- Зачем вам вообще нужен Kotlin DSL
- С какими проблемами вы обязательно столкнетесь и как их решить

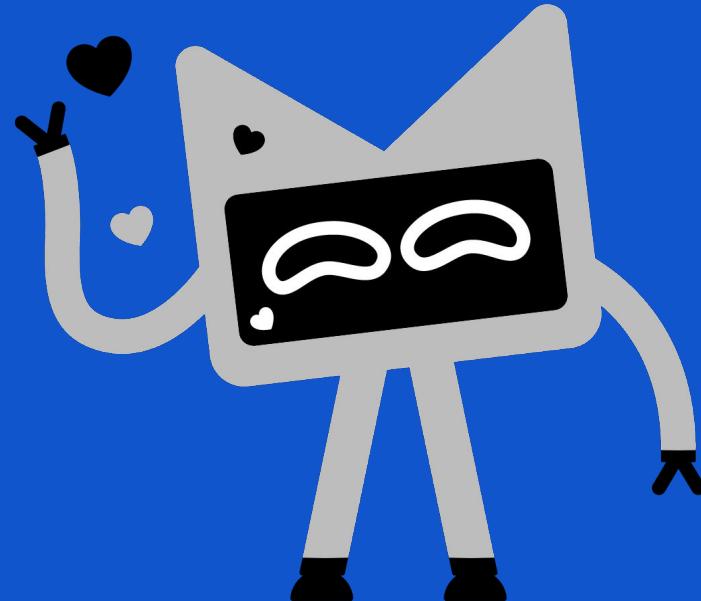
Kotlin DSL в Gradle

- Альфа в 2016
- Релиз в 2018
- ~ 200 закрытых issues
- ... и ~ 100 открытых



Зачем?

Понятный Android разработчикам язык

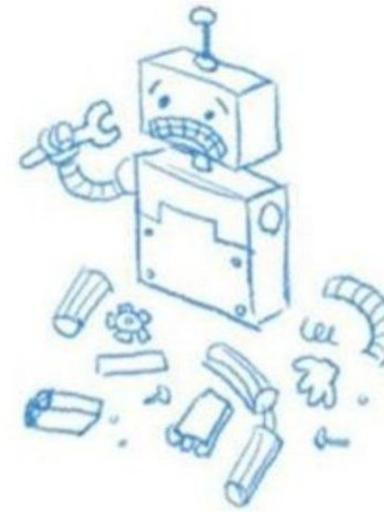


Внятный репортинг ошибок



OOPSIE WOOPSIE!!

Uwu We made a fucky wucky!! A wittle
fucko boingo! The code monkeys at our
headquarters are working VEWY HAWD
to fix this!



Внедрение репортинга ошибок: код

```
    android {  
        buildTypes {  
            release {  
                isMinifyEnabled = false  
            }  
        }  
    }
```

Ошибка в Groovy

* Exception is:

```
org.gradle.api.GradleScriptException: A problem occurred evaluating project ':groovydsl'.
```

```
<140 internal calls>
```

```
Caused by: groovy.lang.MissingMethodException: No signature of method: build_c20d142gonoqi8cogfw3aajjx.android() is applicable for argument types: (build_c20d142gonoqi8cogfw3aajjx$run_closure1) values: [build_c20d142gonoqi8cogfw3aajjx$run_closure1@38a7ba45]
```

```
at  
build_c20d142gonoqi8cogfw3aajjx.run(D:\Projects\GradleKotlinDSLExample\groovydsl\build.gradle:6)
```

```
<1 internal call>
```

```
... 140 more
```

Ошибка в Kotlin

```
> Configure project :kotlindsl

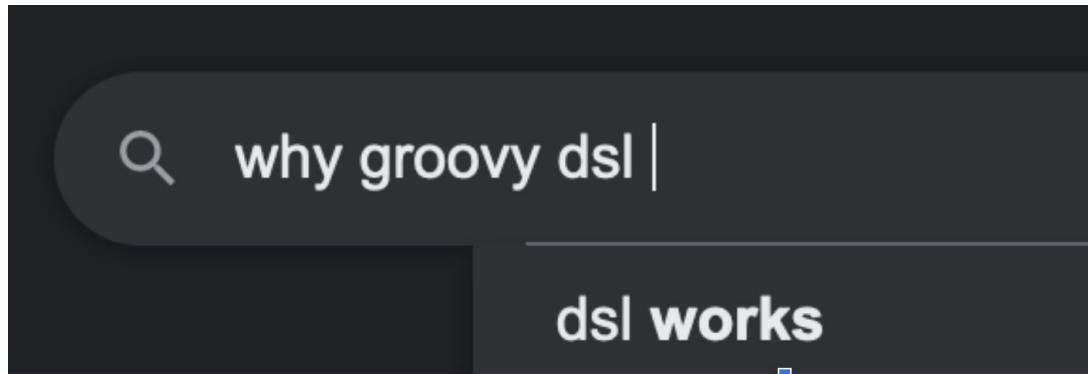
e: D:\Projects\GradleKotlinDSLExample\kotlindsl\build.gradle.kts:19:5: Unresolved
reference: buildTypess

e: D:\Projects\GradleKotlinDSLExample\kotlindsl\build.gradle.kts:20:9: Unresolved
reference. None of the following candidates is applicable because of receiver type
mismatch:

public open fun NamedDomainObjectContainer<ApplicationBuildType>.release(action:
ApplicationBuildType.() -> Unit): Unit defined in com.android.build.gradle.internal.
dsl.BaseAppModuleExtension

FAILURE: Build failed with an exception.
```

Нормальная поддержка автокомплита



Хм...



Действительно...

Поддержка Groovy DSL в Android Studio

```
jacoco {  
    repo|  
        m repositories(Closure closure) void  
        m reportsDir(File reportsDir) void  
        m reportsDir(Provider<File> reportsDir) void  
        p reportsDir File  
        p reportsDirectory DirectoryProperty  
        p repositories RepositoryHandler  
        p reporting ReportingExtension  
        m reporting(Closure configuration) ReportingExtension  
        m testReportDirName(String s) void  
        p testReportDir File  
        p testReportDirName String  
        m getReportDirName() DirectoryProperty  
    }  
Press ⇨ to insert, ⇢ to replace Next Tip ::
```

Поддержка Kotlin DSL в Android Studio

```
jacoco { this: JacocoPluginExtension
    repo|
} v reportsDirectory (from getReportsDirecto... DirectoryProperty
v reportsDir (from getReportsDir()/setReportsDir()) File
v repositories (from getRepositories()) RepositoryHandler
m repositories(configureClosure: Closure<*>) Unit
v reporting for Project in org.gradle.kot... ReportingExtension
f reporting(configure: Action<ReportingExtension>) for ... Unit
f repositories {...} (configuration: RepositoryHandler...) Unit
f reportInternalCompilerError(messageCollector: Message... Unit
f reportOnDeclaration(trace: BindingTrace, descriptor: ... Unit
f reportOnDeclarationAs(trace: BindingTrace, descriptor... Unit
f reportOnDeclarationOrFail(trace: BindingTrace, descri... Unit
f reportSuspensionPointInsideMonitor(element: V+Element) Unit
Press ⇨ to insert, ⌘ to replace Next Tip :::
```

Уже стандарт для Android

"Starting with Android Studio Giraffe, new projects use the Kotlin DSL by default for build configuration. This offers a better editing experience than the Groovy DSL with syntax highlighting, code completion, and navigation to declarations."

<https://developer.android.com/studio/build/migrate-to-kts>

Проблемы

Скорость

Как замеряем?

- Создаем пустой новый проект
- Генерим 100 модулей
- Ставим нужную версию Gradle
- Везде вырубаем daemon:

```
org.gradle.daemon = false
```

Как замеряем?

- Автоматизируем build scan, чтобы не подтверждать каждый раз

```
if (hasProperty("buildScan")) {  
    extensions.findByName("buildScan")?.withGroovyBuilder {  
        setProperty(  
            "termsOfServiceUrl",  
            "https://gradle.com/terms-of-service"  
        )  
        setProperty("termsOfServiceAgree", "yes")  
    }  
}
```

Что замеряем?

Groovy
DSL

Kotlin
DSL Hot

Kotlin
DSL Cold

Как запускаем?

- Groovy DSL / Kotlin DSL Hot запускаем через

```
./gradlew assembleDebug --scan
```

Как запускаем?

- Kotlin DSL Cold запускаем через

```
rm -rf ~/.gradle/caches/7.0/kotlin-dsl  
rm -rf ~/.gradle/caches/7.0/gradle-kotlin-dsl  
./gradlew assembleDebug --scan
```

Что быстрее в среднем за 5 запусков?

A

Kotlin
DSL

B

Groovy
DSL

С каждой новой версией Gradle performance

A

Деградирует

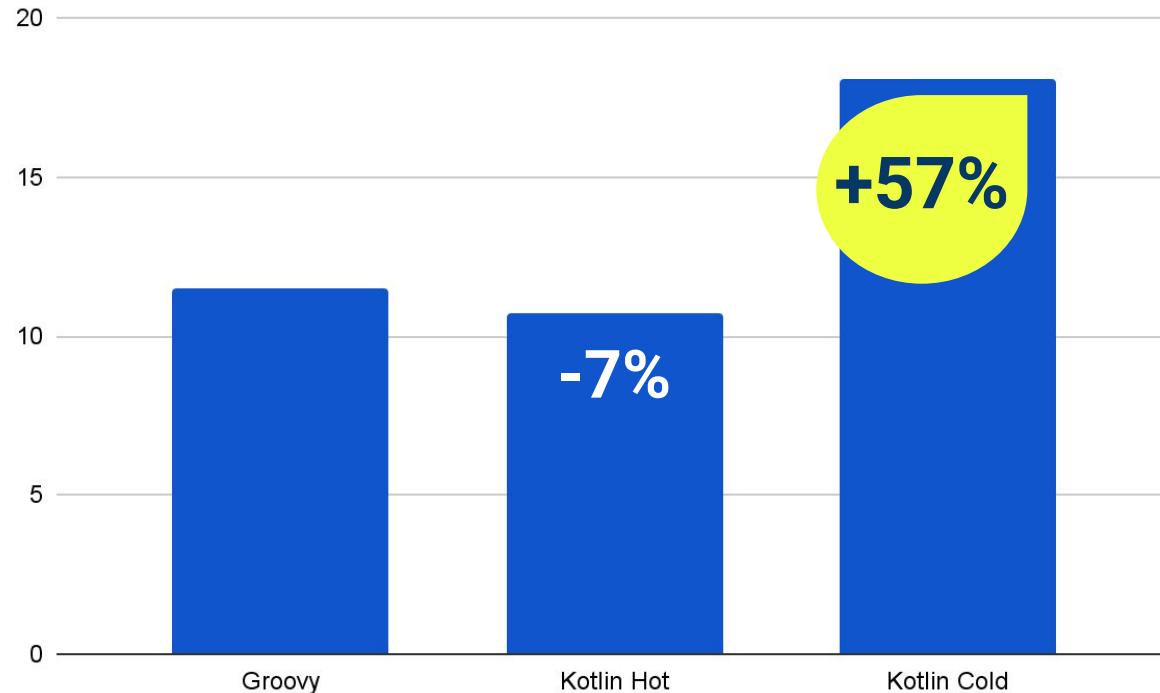
B

Улучшается

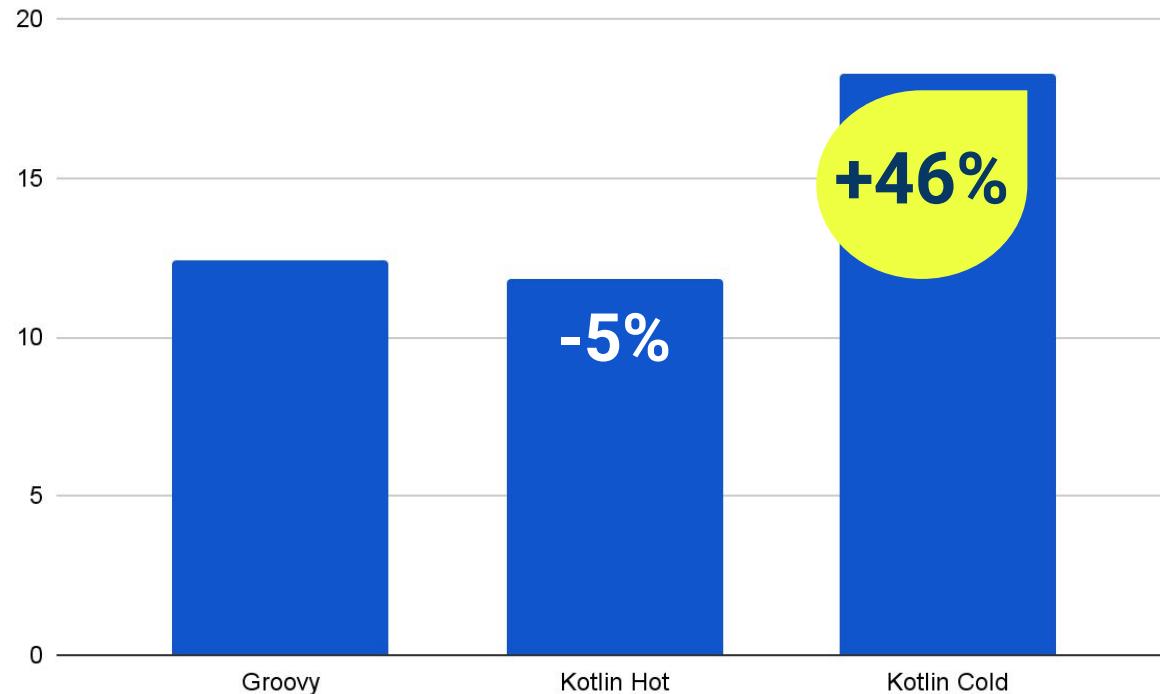
C

Не меняется

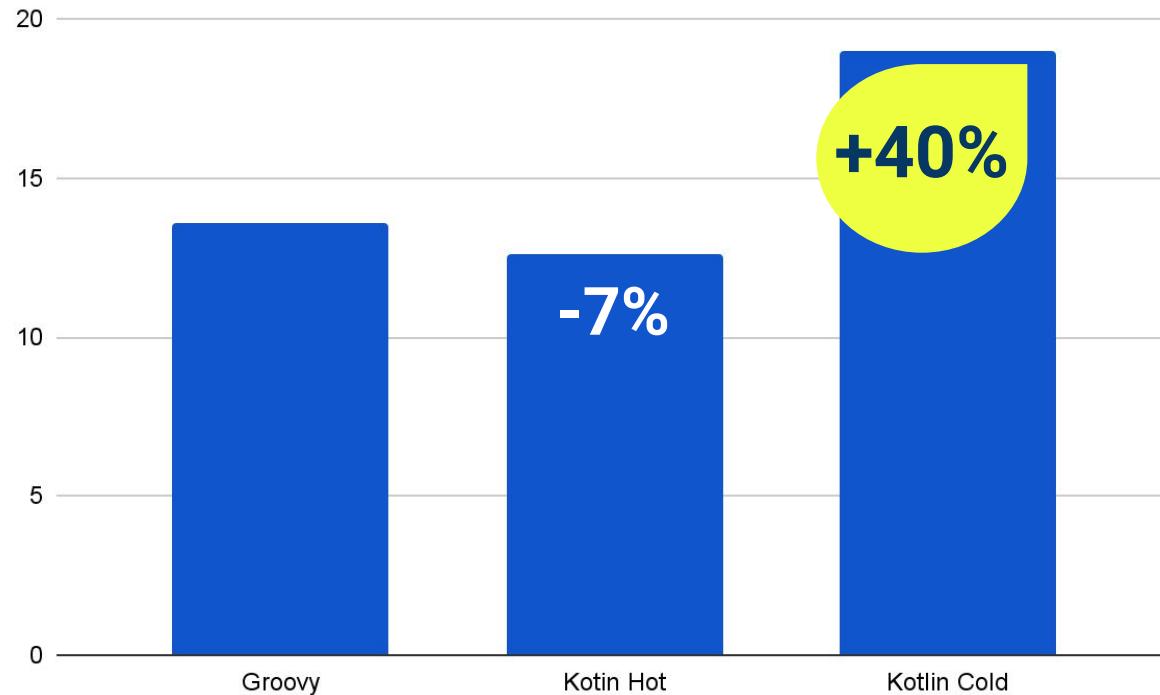
Gradle 6.7 performance



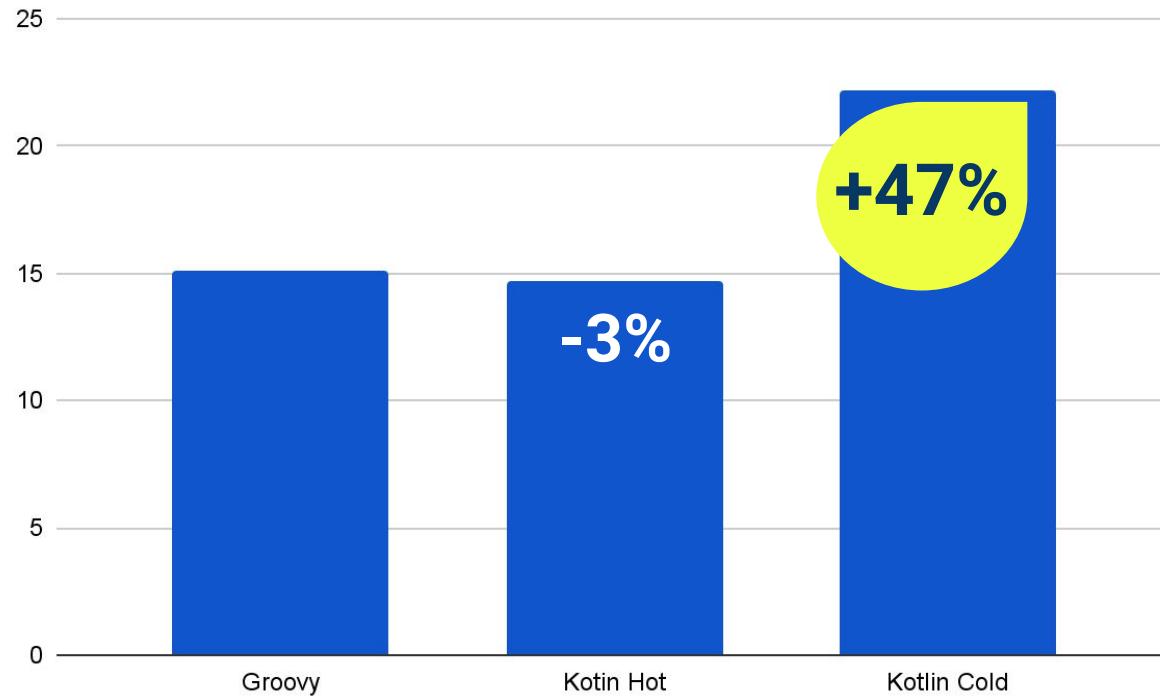
Gradle 7.0 performance



Gradle 7.6.1 performance



Gradle 8.0.1 performance



Kotlin DSL performance

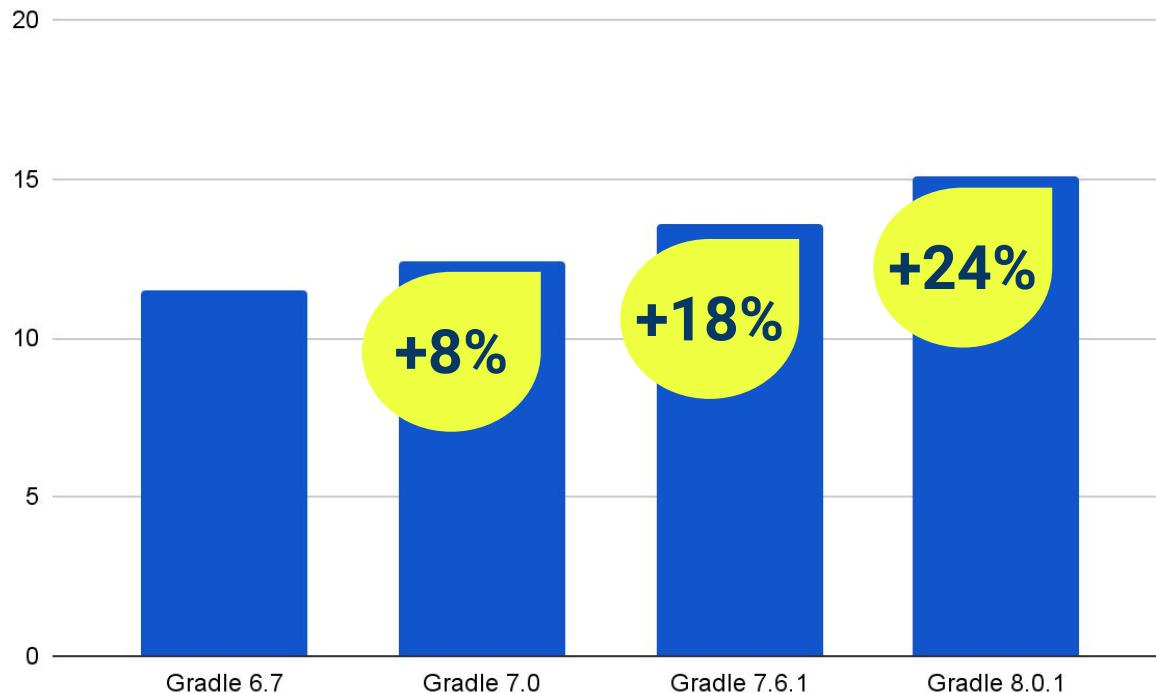
7

раз нужно запускать
конфигурацию,
чтобы Kotlin DSL был
быстрее Groovy DSL на
Gradle 7.6.1

Kotlin DSL медленнее

- Компиляция все же требует ресурсов
- Kotlin DSL при первом запуске будет медленнее Groovy в любом случае

Groovy DSL performance



А чего Groovy DSL все медленнее?!

Gradle Version	Groovy Version
6.7.1	2.5.12
7.0	3.0.7
7.6.1	3.0.10
8.0.1	3.0.13

А чего Groovy все медленнее?!



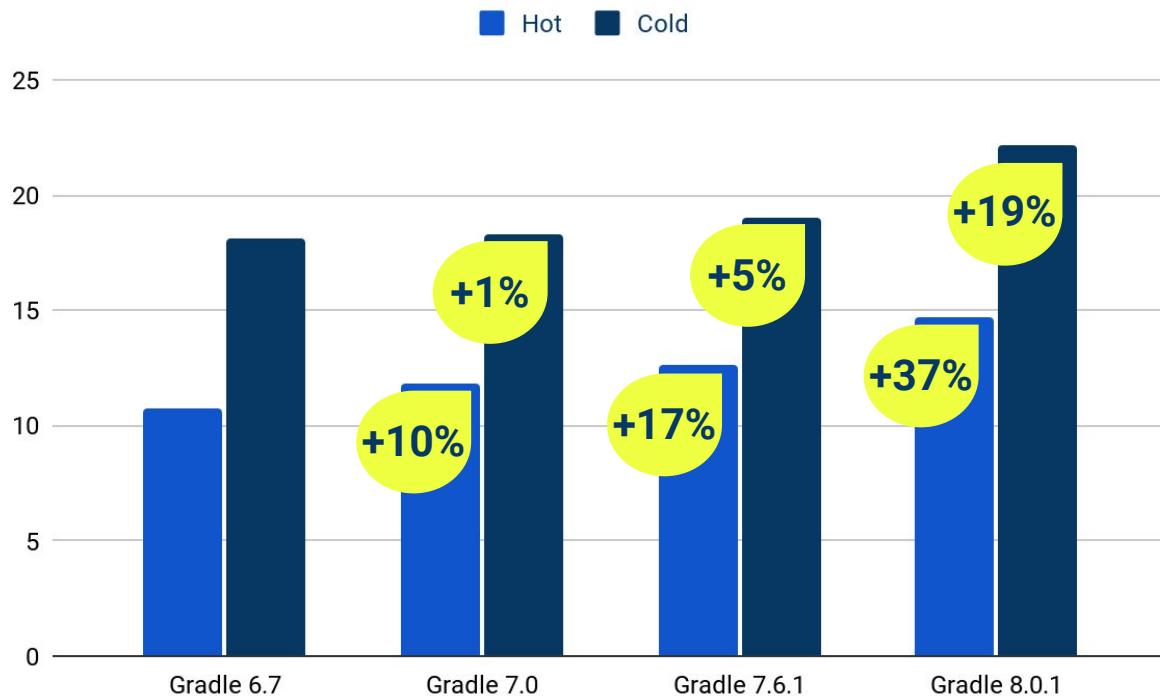
Groovy / GROOVY-9588

groovyCompile 6x slower in 3.0.4 than 2.5.6

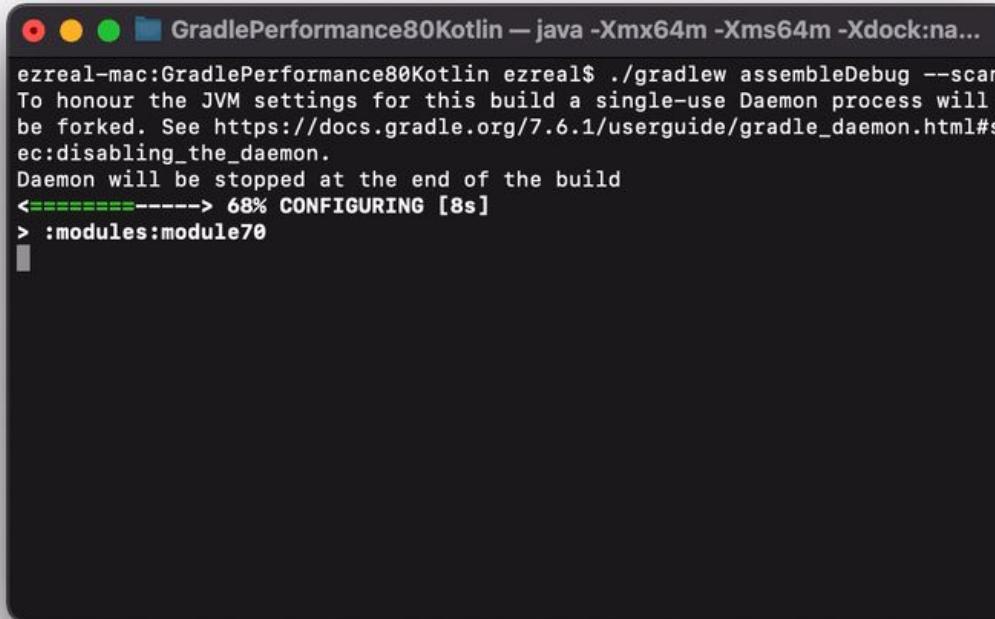
▼ Details

Type:	Bug	Status:	CLOSED
Priority:	Major	Resolution:	Fixed
Affects Version/s:	3.0.4	Fix Version/s:	4.0.0-alpha-1, 3.0.5
Component/s:	parser-antlr4		
Labels:	None		

Kotlin DSL performance



Build Scan таки не очень

A screenshot of a macOS terminal window titled "GradlePerformance80Kotlin — java -Xmx64m -Xms64m -Xdock:na...". The window contains the following text:

```
ezreal-mac:GradlePerformance80Kotlin ezreal$ ./gradlew assembleDebug --scan
To honour the JVM settings for this build a single-use Daemon process will
be forked. See https://docs.gradle.org/7.6.1/userguide/gradle_daemon.html#s
ec:disabling_the_daemon.
Daemon will be stopped at the end of the build
<===== 68% CONFIGURING [8s]
> :modules:module70
```

Build Scan таки не очень

Build	Configuration	Dependency resolution	Task execution	Build cache
Total build time				24.275s
Initialization & configuration				15.141s
Startup				2.016s
Settings				1.529s
Loading projects				0.109s
Configuration				11.487s
Execution				9.134s
Task execution				9.007s
End of build				0.127s

Build Scan таки не очень

Build	Configuration	Dependency resolution	Task execution
<hr/>			
	Total configuration time		13.016s
	Script compilation		0.000s
	Building included plugins		0.000s
	Model configuration		9.386s
	Task graph calculation		3.174s
	Other		0.456s

Документация

Что есть по документации

- 2 страницы на docs.gradle.org
- 1 страница по миграции на developer.android.com
- Статьи на habr и medium



Код в файлах

Типичный проект

app

build.gradle



Типичный проект

app

build.gradle

someScripts.gradle



Типичный проект

app

build.gradle

someScripts.gradle

someScripts2.gradle



Типичный проект

app

build.gradle

scripts

someScripts.gradle

someScripts2.gradle

someScripts3.gradle



Вынесенная логика в Groovy

```
// внутри androidConfiguration.gradle

android {

    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }

}
```

Вынесенная логика

```
// внутри build.gradle / build.gradle.kts

android {

    ...

}

apply(from = " .. /androidConfiguration.gradle")
```

Вынесенная логика в Kotlin DSL

```
// внутри androidConfiguration.gradle.kts

android {

    compileOptions {

        sourceCompatibility = JavaVersion.VERSION_1_8
        targetCompatibility = JavaVersion.VERSION_1_8
    }

}
```

Вынесенная логика в Kotlin DSL

e: D:\Projects\GradleKotlinDSLExample\androidConfiguration.gradle.kts:1:1: Unresolved reference: android

e: D:\Projects\GradleKotlinDSLExample\androidConfiguration.gradle.kts:2:5: Unresolved reference: compileOptions

e: D:\Projects\GradleKotlinDSLExample\androidConfiguration.gradle.kts:3:9: Unresolved reference: sourceCompatibility

e: D:\Projects\GradleKotlinDSLExample\androidConfiguration.gradle.kts:4:9: Unresolved reference: targetCompatibility

e: D:\Projects\GradleKotlinDSLExample\androidConfiguration.gradle.kts:6:5: Unresolved reference: kotlinOptions

e: D:\Projects\GradleKotlinDSLExample\androidConfiguration.gradle.kts:7:9: Unresolved reference: jvmTarget

e: D:\Projects\GradleKotlinDSLExample\androidConfiguration.gradle.kts:7:21: Too many characters in a character literal ''1.8''

Почему так?

```
// внутри build.gradle.kts

plugins {
    id("com.android.library")
}

android {
    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_1_8
        targetCompatibility = JavaVersion.VERSION_1_8
    }
}
```

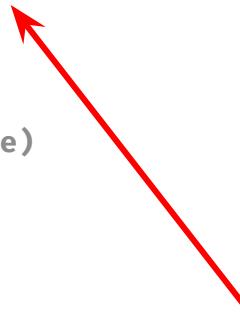
Accessors

```
android { ... }
```

```
fun org.gradle.api.Project.`android`(  
    configure: Action<com.android.build.gradle.LibraryExtension>  
)  
: Unit =  
  
(this as org.gradle.api.plugins.ExtensionAware)  
  
.extensions.configure("android", configure)
```

Можно ли использовать в обычных gradle.kts?

```
fun org.gradle.api.Project.`android`(  
    configure: Action<com.android.build.gradle.LibraryExtension>  
): Unit =  
  
(this as org.gradle.api.plugins.ExtensionAware)  
  
.extensions.configure("android", configure)
```



Unresolved reference

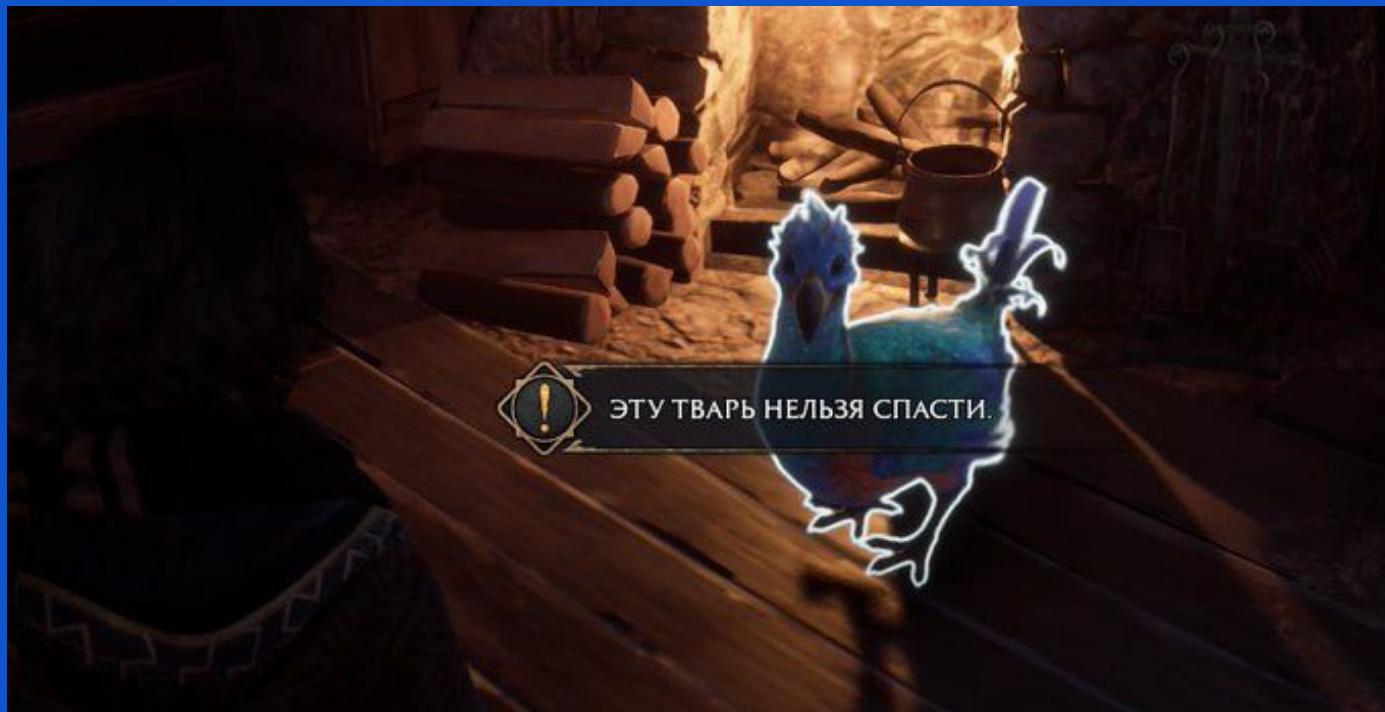
Можно ли использовать в обычных gradle.kts?

- gradle.kts (кроме build.gradle.kts) отрабатывают в изолированном classloader
- у изолированного classloader нет доступа к classpath

Неужели совсем нельзя?

- Технически можно засунуть в extra нужный объект
- Вынуть его в своем gradle.kts файле
- Выставить все что хочется через рефлексию
- ...
- Вам не захочется такое делать =)

Статус обычных *.gradle.kts



А почему в groovy dsl работает?

- совсем другой механизм работы, в том числе есть разница в работе classloader
- если сильно упростить - то там механизм близкий к обычному инлайнингу

Fun fact

```
private static final List<ScriptingLanguage> ALL =  
    Collections.unmodifiableList(  
        Arrays.asList(  
            scriptingLanguage(".gradle", null),  
            scriptingLanguage(  
                ".gradle.kts",  
                "org.gradle.kotlin.dsl.provider.KotlinScriptPluginFactory"  
            ),  
        );
```

Fun fact

`myFile.gradle // default`

`myFile.whateverIwant // still works`

Что делать?

- Оставаться на Groovy
- Convention plugin
- Обычный plugin

Convention plugin

```
// внутри convention.android.gradle.kts

plugins {
    id("com.android.library")
    id("kotlin-android")
}

android {
    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_1_8
        targetCompatibility = JavaVersion.VERSION_1_8
    }
}
```

Convention plugin

```
// внутри build.gradle.kts модуля  
  
plugins {  
    id("convention.android")  
}
```

Обычный plugin

```
class RegularPlugin: Plugin<Project> {

    override fun apply(project: Project) {
        project.extensions.configure(LibraryExtension::class) {
            compileOptions {
                sourceCompatibility = JavaVersion.VERSION_1_8
                targetCompatibility = JavaVersion.VERSION_1_8
            }
        }
    }
}
```

Обычный plugin

```
// внутри build.gradle.kts модуля
plugins {
    id("com.android.library")
    id("kotlin-android")
    id("regular-plugin")
}
```

Не стоит все класть в BuildSrc

- До сих пор во многих статьях и гайдах
- Любое изменение полностью инвалидирует build cache, включая remote build cache
- Провоцирует инфраструктурную помойку
- Давно не является рекомендованной практикой
- Гораздо лучше все класть в composite builds

Composite Builds

- По сути та же модуляризация, но для precompiled scripts
- Инвалидируется только то, что должно
- Компилируются и конфигурируются отдельно

Composite Builds: settings.gradle.kts

```
pluginManagement {  
    repositories { ... }  
}  
  
dependencyResolutionManagement {  
    repositories { ... }  
}  
  
rootProject.name = "myCompositeBuildName"
```

Composite Builds: build.gradle.kts

```
plugins {  
    `kotlin-dsl`  
    `java-gradle-plugin`  
}  
  
gradlePlugin {  
    plugins.register("myPlugin") {  
        id = "myPlugin"  
        implementationClass = "com.typical.package.MyPlugin"  
    }  
}
```

В корневом settings.gradle.kts

```
includeBuild("example")  
// или  
includeBuild("plugins/example")
```

Конфигурация плагинов

Через plugins

```
// внутри build.gradle.kts модуля
plugins {
    id("example-plugin")
}

example {
    enabled = true
}
```

Через apply

```
// внутри build.gradle.kts модуля  
apply(plugin = "example-plugin")
```

```
example {  
    enabled = true  
}
```

e: ../../build.gradle.kts:9:1: Unresolved reference: example

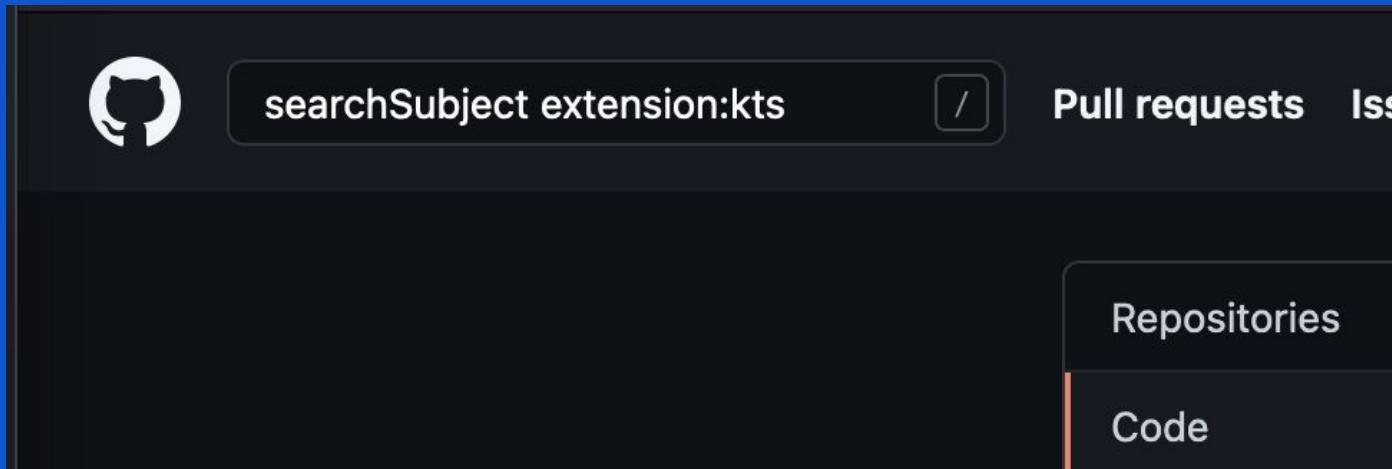
Через apply

```
// внутри build.gradle.kts модуля  
  
apply(plugin = "example-plugin")  
  
configure<com.yandex.plugins.ExamplePluginExtension> {  
    enabled = true  
}
```

А как вообще узнать extension name/package?

- Исходники, автокомплит (только если plugin написан на Kotlin/Java)
- Github
- Stackoverflow
- Github Copilot

Github в помощь



Лямбды

Kotlin lambda

```
public interface Function0<out R> : Function<R> {  
    public operator fun invoke(): R  
}  
  
public interface Function1<in P1, out R> : Function<R> {  
    public operator fun invoke(p1: P1): R  
}
```

Groovy lambda

```
public abstract class Closure<V> extends GroovyObjectSupport implements Cloneable, Runnable,  
GroovyCallable<V>, Serializable {  
  
    // over 1200 lines  
  
}
```

Kotlin lambda

```
uploadBeta {  
    getApplicationFilename = { variantData →  
        // ...  
    }  
}
```

Kotlin lambda

```
uploadBeta {  
    getApplicationFilename = KotlinClosure1<ApplicationVariant, String>({  
        // ...  
    })  
}
```

Строки

Внезапный Quiz

```
abstract class ExamplePluginExtension {  
    abstract var someString: String  
}
```

Внезапный Quiz

```
class ExamplePlugin: Plugin<Project> {

    private var pluginExtension: ExamplePluginExtension? = null

    override fun apply(project: Project) {

        createExtension(project)

        project.afterEvaluate {

            val value: String = pluginExtension!!.someString

            println(value)

        }
    }

    private fun createExtension(project: Project) { ... }

}
```

Что будет при выполнении?

```
plugins { id 'example' }

def world = "World"

def helloWorld = "Hello ${world}"

example {

    someString = helloWorld

}
```

A

Hello World

B

Hello \${world}

C

Ссылка
на объект

D

Ошибка

Внезапный Quiz

```
abstract class ExamplePluginExtension {  
    abstract var listOfStrings: List<String>  
}
```

Внезапный Quiz

```
class ExamplePlugin: Plugin<Project> {  
  
    private var pluginExtension: ExamplePluginExtension? = null  
  
    override fun apply(project: Project) {  
  
        createExtension(project)  
  
        project.afterEvaluate {  
  
            pluginExtension!!.listOfStrings.forEach {  
  
                println(it)  
  
            }  
  
        }  
  
    }  
  
    private fun createExtension(project: Project) { ... }  
  
}
```

Что будет при компиляции?

```
plugins { id 'example' }

def world = "World"

def helloWorld = "Hello ${world}"

example {

    listOfStrings = [helloWorld]

}
```

A

Hello World

B

Hello \${world}

C

Ссылка
на объект

D

Ошибка

Почему так?

```
plugins { id 'example' }

def world = "World" // String

def helloWorld = "Hello ${world}" // GString

example {

    someString = helloWorld // неявный каст

    listOfStrings = [helloWorld] // List<String> = List<GString>

}

class org.codehaus.groovy.runtime.GStringImpl cannot be cast to class java.lang.String
```

Configuration on demand

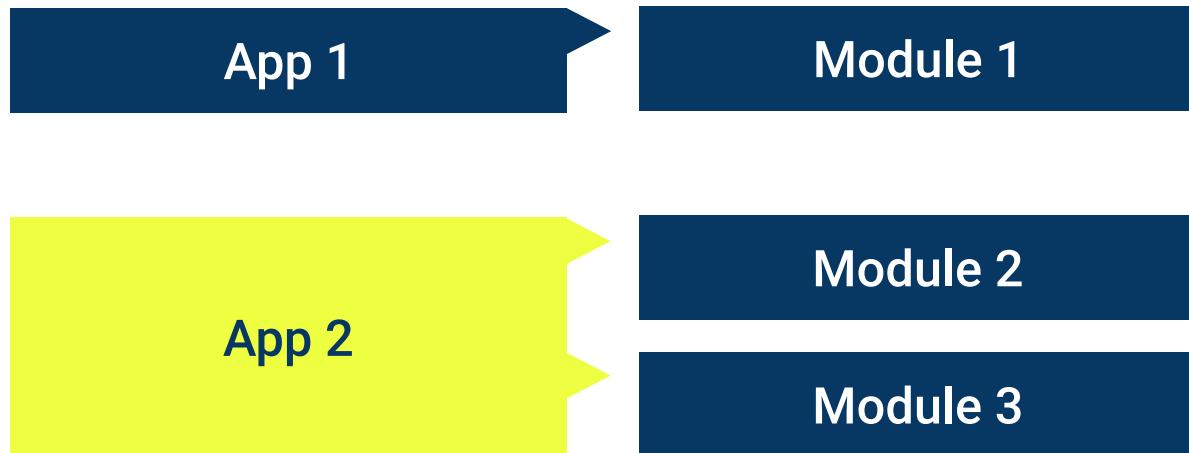
Configuration on demand

```
org.gradle.configureondemand = true
```

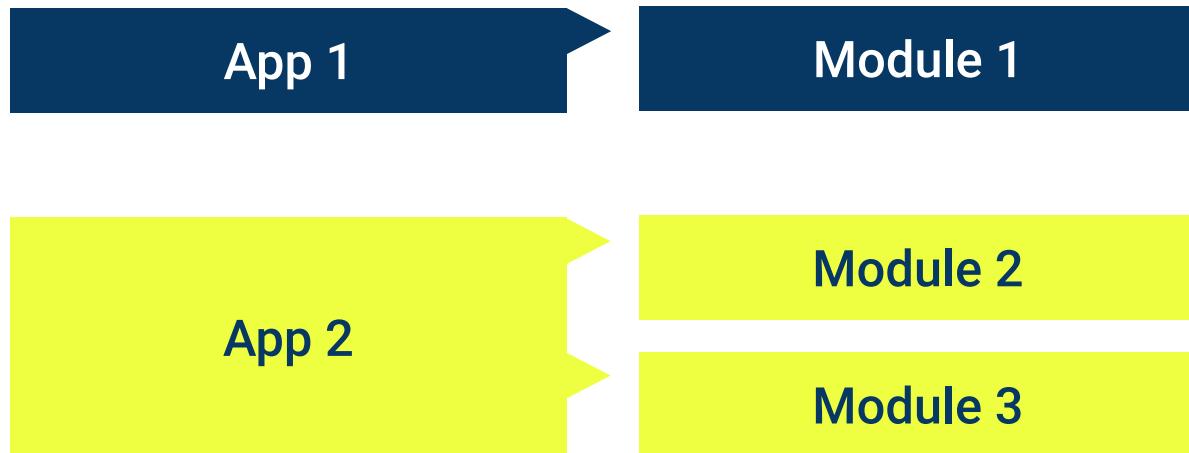
Configuration on demand



Configuration on demand



Configuration on demand



Configuration on demand

"We recommend against enabling the incubating configuration on demand feature as it can lead to very hard-to-diagnose problems."

https://docs.gradle.org/current/userguide/kotlin_dsl.html#kotdsl:limitations

Configuration on demand

Почему
не вырубить
для Kotlin DSL?



Что еще за
hard-to-diagnose
problems?

Gradle 6.7

- * What went wrong:

```
Could not determine the dependencies of task ':app2:mergeDebugAssets'.
> Could not resolve all dependencies for configuration ':app2:debugRuntimeClasspath'.
    > A problem occurred configuring project ':modules:module1'.
        > Could not open cache directory 82086lrkyos98e0oo17×0i5qm
          (/Users/me/.gradle/caches/6.7.1/gradle-kotlin-dsl/82086lrkyos98e0oo17×0i5qm).
            > org.gradle.api.internal.initialization.DefaultClassLoaderScope@1379833d
              must be locked before it can be used to compute a classpath!
```

Gradle 7.6

```
Line 006: android {  
    ^ Unresolved reference: android  
  
Line 007:     compileSdk = 31  
    ^ Unresolved reference: compileSdk  
  
Line 009:     defaultConfig {  
    ^ Unresolved reference: defaultConfig  
  
...
```

Когда воспроизводится

- Модулей десятки
- И в gradle.properties есть следующее:

```
org.gradle.daemon = true  
org.gradle.configureondemand = true
```

Вывод

Выводы

- Все плохо
- Проблем много, но они либо некритичные, либо решаемые
- Не стоит обновлять версию Gradle если не нужны новые фичи
- Kotlin DSL в Gradle это не серебряная пуля, а меньшее зло

Вопросы?