# KOTLIN SCRIPT

## ДЛЯ КОГО, ЗАЧЕМ И КАК

# ОБО МНЕ

- Техлид JVM Backend в Банке Центр-инвест
- Пишу на Kotlin больше 5 лет
- Фанат Kotlin

# ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ

# BUILD.GRADLE.KTS

```
val revealKtVersion = "0.2.4"

group = "dev.limebeck"
version = "1.0.0"

plugins {
    kotlin("jvm") version "1.9.10"
}

repositories {
    mavenLocal()
    mavenCentral()
}

dependencies {
    implementation("dev.limebeck:revealkt-script-definition:$revealKtVersion")
}
```

# JETBRAINS SPACE CI/CD DSL

```
@file:DependsOn("com.squareup.okhttp:okhttp:2.7.4")

import com.squareup.okhttp.*

job("Get example.com") {
    container(image = "amazoncorretto:17-alpine") {
        kotlinScript {
            val client = OkHttpClient()
            val request = Request.Builder().url("http://example.com").build()
            val response = client.newCall(request).execute()
            println(response)
        }
    }
}
```

# GITHUB WORKFLOWS KT

```
#!/usr/bin/env kotlin
@file:DependsOn("io.github.typesafegithub:github-workflows-kt:1.13.0")

import ...

workflow(name = "Build", on = listOf(PullRequest()), sourceFile = __FILE__.toPath()) {
    job(id = "build", runsOn = UbuntuLatest) {
        uses(action = CheckoutV4())
        uses(action = SetupJavaV3())
        uses(
            name = "Build",
            action = GradleBuildActionV2(
                arguments = "build",
            )
        )
    }
}.writeToFile()
```

# ПРОСТОЕ CLI ПРИЛОЖЕНИЕ

```
#!/usr/bin/env kotlin
@file:DependsOn("org.jetbrains.kotlinx:kotlinx-cli-jvm:0.3.6")

import kotlinx.cli.*;
import java.io.File;

val parser = ArgParser("copyIndexed")
val input by parser.option(ArgType.String, shortName = "i",
    description = "Input file").required()
parser.parse(args)
val file = File(input)
file.useLines {
    it.mapIndexed { index, line -> "$index - $line" }
        .forEach { File("${file.name}-copy").appendText(it) }
}
```

# LIVE-PLUGIN
# (ПРОСТЫЕ ПЛАГИНЫ ДЛЯ IDEA)

```
import com.intellij.openapi.actionSystem.AnActionEvent
import liveplugin.*

registerAction(id = "Insert New Line Above", keyStroke = "ctrl alt shift ENTER") { event ->
    val project = event.project ?: return@registerAction
    val editor = event.editor ?: return@registerAction
    executeCommand(editor.document, project) { document ->
        val caretModel = editor.caretModel
        val lineStartOffset = document.getLineStartOffset(caretModel.logicalPosition.line)
        document.insertString(lineStartOffset, "\n")
        caretModel.moveToOffset(caretModel.offset + 1)
    }
}
show("Loaded 'Insert New Line Above' action<br/>Use 'ctrl+alt+shift+Enter' to run it")
```

# ПРЕЗЕНТАЦИИ С REVEAL-KT

```
title = "Kotlin Script: для кого, зачем и как"

configuration {
    controls = false
    progress = false
}

slides {
    slide {
        autoanimate = true
        +title { "Kotlin Script: для кого, зачем и как" }
    }
}
```

# ПОЗИЦИОНИРОВАНИЕ KOTLIN SCRIPTING ОТ JETBRAINS

# KEEP: KOTLIN SCRIPTING SUPPORT APPLICATIONS

- Build scripts (Gradle/Kobalt)
- Test scripts (Spek)
- Command-line utilities
- Routing scripts (ktor)
- Type-safe configuration files (TeamCity)
- In-process scripting and REPL for IDE
- Consoles like IPython/Jupyter Notebook
- Game scripting engines
- ...

# ОБОБЩИМ

- Read-Eval-Print Loop (REPL)
- замена BASH-скриптов
- встраивание скриптового движка
- компиляция скриптов с исходниками

# ОБОБЩИМ

- Read-Eval-Print Loop (REPL)
- замена BASH-скриптов
- встраивание скриптового движка
- ~~компиляция скриптов с исходниками~~

# READ-EVAL-PRINT LOOP (REPL)

# READ-EVAL-PRINT LOOP (REPL)

- Читает (парсит)
- Исполняет
- Выводит ответ
- И снова

# READ-EVAL-PRINT LOOP (REPL)

# ЗАЧЕМ ОН НУЖЕН?

# ЗАЧЕМ ОН НУЖЕН?

- Обучение основам

# ЗАЧЕМ ОН НУЖЕН?

- Обучение основам
- Прототипирование

# ЗАЧЕМ ОН НУЖЕН?

- Обучение основам
- Прототипирование
- Быстрая обратная связь

# READ-EVAL-PRINT LOOP (REPL)
# АЛЬТЕРНАТИВЫ

# READ-EVAL-PRINT LOOP (REPL)

# JSHELL

```
> jshell
|  Welcome to JShell -- Version 17.0.4.1
|  For an introduction type: /help intro

jshell> var hello = "Hello, world!"
hello ==> "Hello, world!"

jshell> System.out.println(hello)
Hello, world!
```

# READ-EVAL-PRINT LOOP (REPL)

# GROOVY SHELL

```
> groovysh
Groovy Shell (4.0.20, JVM: 17.0.4.1)
Type ':help' or ':h' for help.
------------------------------------------------
groovy:000> hello = 'Hello, World!'
===> Hello, World!
groovy:000> println hello
Hello, World!
===> null
```

# READ-EVAL-PRINT LOOP (REPL)

# KSCRIPT (REPL MODE)

```
> kscript --interactive kscript.kts
[kscript] Resolving com.fasterxml.jackson.module:jackson-module-kotlin:2.17.0...
[kscript] Creating REPL
Welcome to Kotlin version 1.9.22 (JRE 22+37)
Type :help for help, :quit for quit
>>> println("Hello, World!")
Hello, World!
>>>
```

kscript.kts

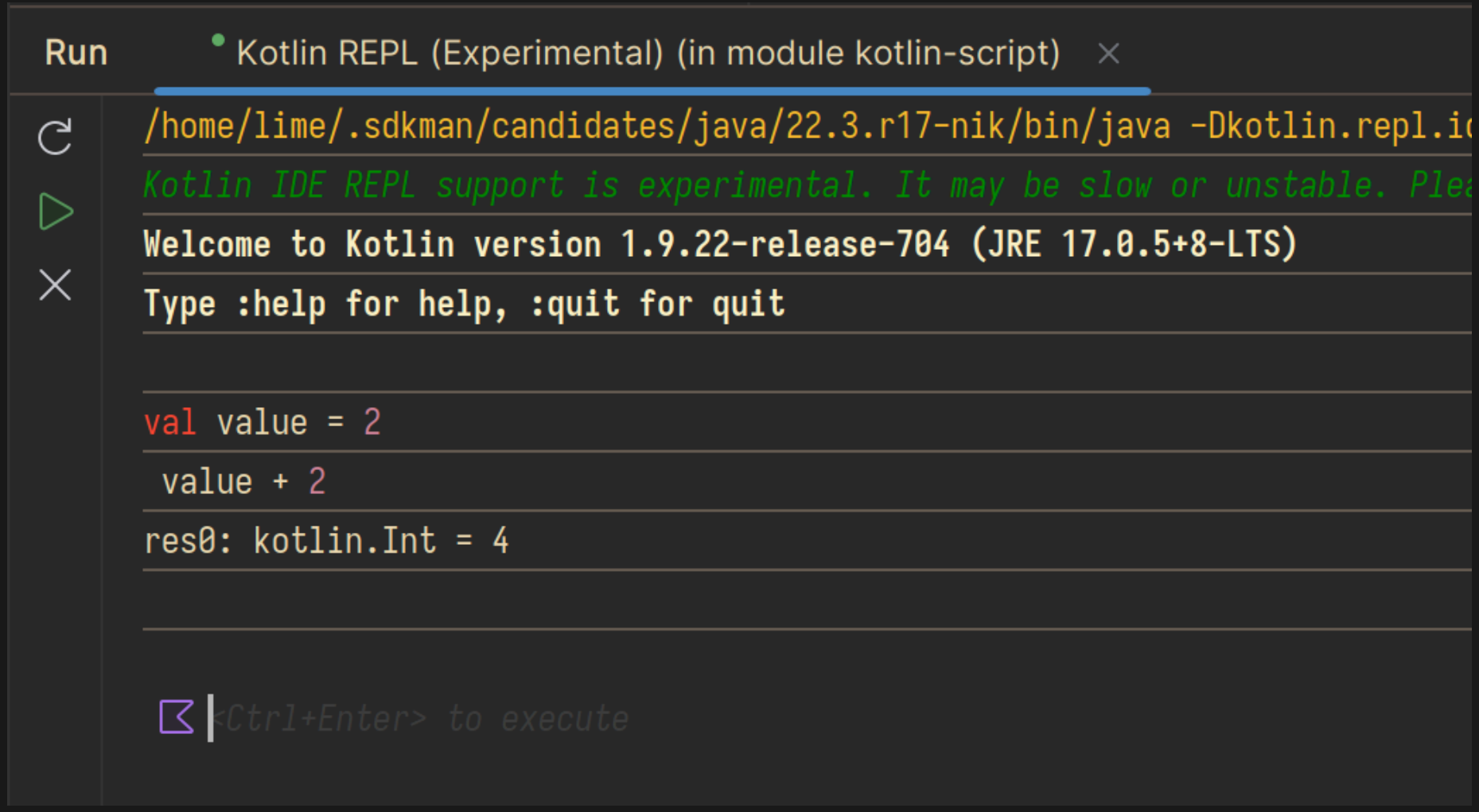
```
@file:DependsOn("com.fasterxml.jackson.module:jackson-module-kotlin:2.17.0")
```

# KOTLIN REPL

# KOTLIN SHELL

```
> kotlin
Welcome to Kotlin version 1.9.22 (JRE 17.0.4.1+1-LTS)
Type :help for help, :quit for quit
>>> val hello = "Hello, World!"
>>> println(hello)
Hello, World!
```

# IJ IDEA KOTLIN REPL




IJ IDEA > Tools > Kotlin > Kotlin REPL (Experimental)

# KOTLIN ДЛЯ JUPITER NOTEBOOK

# KOTLIN ДЛЯ JUPITER NOTEBOOK

jupyter Kotlin Kernel Examples Last Checkpoint: Last Thursday at 15:16 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Kotlin

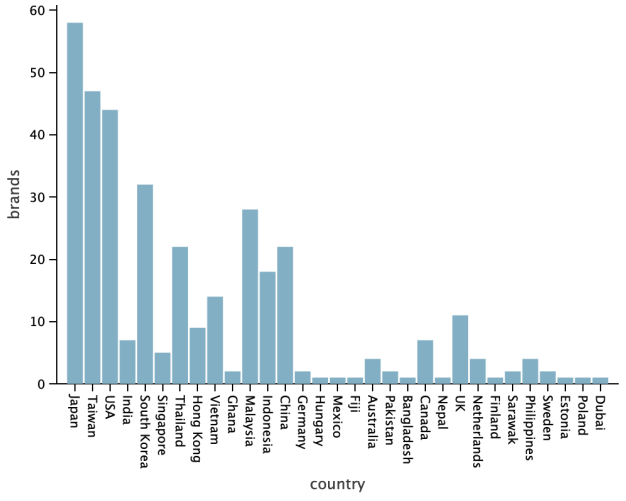
Run Code

In [1]:
```
%use lets-plot, krangl
```

In [2]:
```
val df = DataFrame.readCSV("ramen-ratings.csv")
val processedDF = df.filter({ it["Stars"].isMatching <String>{ !startsWith("Un")  }})
    .addColumn("StarsAsDouble") { it["Stars"].map <String> { it.toDouble()}}
```

In [3]:
```
val distinctBrandsPerCountry = processedDF.groupBy("Country").distinct("Brand").groupBy("Country").count()
val (xs, ys) = distinctBrandsPerCountry.rows.map { row -> (row["Country"] as String) to (row["n"] as Int) }.unzip()
val p = lets_plot(mapOf("country" to xs, "brands" to ys))

val layer = geom_bar (stat=Stat.identity, fill = "#78B3CA") {
    x = "country"
    y = "brands"
}

p + layer
```

Out[3]:



# Ноутбуки Kotlin для обучения и прототипирования

Александр
Нозик
МФТИ

JPoint
2022

# ЗАМЕНА BASH-СКРИПТОВ В АВТОМАТИЗАЦИИ ЗАДАЧ

# ЗАЧЕМ ЗАМЕНЯТЬ BASH?

# ЗАЧЕМ ЗАМЕНЯТЬ BASH?

## Bash

```
# Нумеруем строки в файле
sed = a.txt | sed 'N; s/^/ /; s/ *\(.\{4,\}\)\n/\1 /'
```

## Kotlin

```
import java.io.File

File("a.txt").useLines {
    it.forEachIndexed { i, l -> println("nulli: nulll") }
}
```

# ЗАЧЕМ ЗАМЕНЯТЬ BASH?

- Сложные скрипты на Bash - боль

# ЗАЧЕМ ЗАМЕНЯТЬ BASH?

- Сложные скрипты на Bash - боль
- Bash - write-only код

# ЗАЧЕМ ЗАМЕНЯТЬ BASH?

- Сложные скрипты на Bash - боль
- Bash - write-only код
- Зависимости на Bash?

# ПОЧЕМУ ИМЕННО KOTLIN SCRIPT

- Богатая экосистема JVM

# ПОЧЕМУ ИМЕННО KOTLIN SCRIPT

- Богатая экосистема JVM
- Зависимости не в системе

# ПОЧЕМУ ИМЕННО KOTLIN SCRIPT

- Богатая экосистема JVM
- Зависимости не в системе
- Удобство Kotlin DSL

# ПОЧЕМУ ИМЕННО KOTLIN SCRIPT

- Богатая экосистема JVM
- Зависимости не в системе
- Удобство Kotlin DSL
- Типобезопасность на уровне компиляции

# АЛЬТЕРНАТИВЫ

# JAVA 11 (JEP 330)

```
///usr/bin/env java "$0" "$@" ; exit $?

class Prog {
    public static void main(String[] args) { Helper.run(); }
}

class Helper {
    static void run() { System.out.println("Hello!"); }
}
```

```
> ./Prog.java
Hello!
```

# SHEBANG ОБЫЧНЫЙ

```
#!/usr/bin/env java
```

# SHEBANG ДЛЯ .JAVA

```
///usr/bin/env java "null0" "null@" ; exit null?
```

# JAVA 22 (JEP 458)

```
///usr/bin/env java "$0" "$@" ; exit $?

class Prog {
    public static void main(String[] args) { Helper.run(); }
}
```

```
class Helper {
    static void run() { System.out.println("Hello!"); }
}
```

```
> ./Prog.java
Hello!
```

# JAVA 22 (+ JEP 463)

```
///usr/bin/env java --enable-preview --source 22 "$0" "$@" ; exit $?

public static void main() { Helper.run(); }
```

```
class Helper {
    static void run() { System.out.println("Hello!"); }
}
```

```
> ./ProgMain.java
Hello!
```

# GROOVY

```
#!/usr/bin/env groovy
@Grab('com.fasterxml.jackson.core:jackson-databind:2.17.0')

import com.fasterxml.jackson.databind.ObjectMapper

def value = new ObjectMapper().with {
    readValue('{"key":"Hello, World!"}', Map.class)["key"]
}
println value
```

```
> ./ScriptWithDeps.groovy
Hello, World!
```

# JBANG (JAVA)

```java
///usr/bin/env jbang "$0" "$@" ; exit $?
//DEPS com.fasterxml.jackson.core:jackson-databind:2.17.0

import com.fasterxml.jackson.databind.ObjectMapper;
import java.util.Map;

class Main {
    public static void main(String[] args) throws Exception {
        ObjectMapper mapper = new ObjectMapper();
        var value = mapper.readValue("{\"key\":\"Hello, World!\"}", Map.class);
        System.out.println(value.get("key"));
    }
}
```

```
> ./JBangEx.java
[jbang] Resolving dependencies...
[jbang]    com.fasterxml.jackson.core:jackson-databind:2.17.0
[jbang] Dependencies resolved
[jbang] Building jar for JBangEx.java...
Hello, World!
```

# JBANG (GROOVY)

```groovy
///usr/bin/env jbang "$0" "$@" ; exit $?
//DEPS com.fasterxml.jackson.core:jackson-databind:2.17.0

import com.fasterxml.jackson.databind.ObjectMapper

def value = new ObjectMapper().with {
    readValue('{"key":"Hello, World!"}', Map.class)["key"]
}
println value
```

```
> ./JBangEx.groovy
[jbang] Downloading Groovy 4.0.14. Be patient, this can take several minutes...
[jbang] Installing Groovy 4.0.14...
[jbang] Resolving dependencies...
[jbang]    com.fasterxml.jackson.core:jackson-databind:2.17.0
[jbang]    org.apache.groovy:groovy:4.0.14
[jbang] Dependencies resolved
[jbang] Building jar for JBangEx.groovy...
Hello, World!
```

# JBANG (KOTLIN)

```
///usr/bin/env jbang "$0" "$@" ; exit $?
//DEPS com.fasterxml.jackson.module:jackson-module-kotlin:2.17.0

import com.fasterxml.jackson.module.kotlin.jacksonObjectMapper
import com.fasterxml.jackson.module.kotlin.readValue

val value = jacksonObjectMapper().readValue<Map<String, String>>(
    """{"key":"Hello, World!"}"""
)

fun main() {
    println(value.get("key"))
}
```

```
> ./JBangEx.kt
[jbang] Downloading Kotlin 1.8.22. Be patient, this can take several minutes...
[jbang] Installing Kotlin 1.8.22...
[jbang] Resolving dependencies...
[jbang]    com.fasterxml.jackson.module:jackson-module-kotlin:2.17.0
[jbang] Dependencies resolved
[jbang] Building jar for JBangEx.kt...
Hello, World!
```

# KSCRIPT

```
#!/usr/bin/env kscript

@file:DependsOn("com.fasterxml.jackson.module:jackson-module-kotlin:2.17.0")

import com.fasterxml.jackson.module.kotlin.jacksonObjectMapper
import com.fasterxml.jackson.module.kotlin.readValue

val value = jacksonObjectMapper()
    .readValue<Map<String, String>>("""{"key":"Hello, World!"}""")

println(value["key"])
```

```
> ./kscript.kts
[kscript] Resolving com.fasterxml.jackson.module:jackson-module-kotlin:2.17.0...
Hello, World!
```

# ЗАМЕНА BASH-СКРИПТОВ В АВТОМАТИЗАЦИИ ЗАДАЧ

# KOTLIN SCRIPT

# СКРИПТЫ .KTS

```
#!/usr/bin/env kotlin

import java.io.File

fun complexProcess(text: String): String = TODO()

val file = File("path/to/my.file")
val text = file.readText()
val newText = complexProcess(text)
file.writeText(newText)
```

# СКРИПТЫ .MAIN.KTS

- Подключение репозиториев и библиотек
- Конфигурация комплятора в самом скрипте
- Кэширование между запусками
- Поддержка в IDE "из коробки"

# СКРИПТЫ .MAIN.KTS

```
#!/usr/bin/env kotlin
@file:DependsOn("org.jetbrains.kotlinx:kotlinx-cli-jvm:0.3.6")

import kotlinx.cli.*;
import java.io.File;

val parser = ArgParser("copyIndexed")
val input by parser.option(ArgType.String, shortName = "i",
    description = "Input file").required()
parser.parse(args)
val file = File(input)
file.useLines {
    it.mapIndexed { index, line -> "$index - $line" }
        .forEach { File("${file.name}-copy").appendText(it) }
}
```

# СКРИПТЫ .MAIN.KTS

```
#!/usr/bin/env kotlin
@file:DependsOn("org.jetbrains.kotlinx:kotlinx-cli-jvm:0.3.6")

import kotlinx.cli.*;
import java.io.File;

val parser = ArgParser("copyIndexed")
val input by parser.option(ArgType.String, shortName = "i",
    description = "Input file").required()
parser.parse(args)
val file = File(input)
file.useLines {
    it.mapIndexed { index, line -> "$index - $line" }
        .forEach { File("${file.name}-copy").appendText(it) }
}
```

# СКРИПТЫ .MAIN.KTS

```
#!/usr/bin/env kotlin
@file:DependsOn("org.jetbrains.kotlinx:kotlinx-cli-jvm:0.3.6")

import kotlinx.cli.*;
import java.io.File;

val parser = ArgParser("copyIndexed")
val input by parser.option(ArgType.String, shortName = "i",
    description = "Input file").required()
parser.parse(args)
val file = File(input)
file.useLines {
    it.mapIndexed { index, line -> "$index - $line" }
        .forEach { File("${file.name}-copy").appendText(it) }
}
```

# СКРИПТЫ .MAIN.KTS

```
#!/usr/bin/env kotlin
@file:DependsOn("org.jetbrains.kotlinx:kotlinx-cli-jvm:0.3.6")

import kotlinx.cli.*;
import java.io.File;

val parser = ArgParser("copyIndexed")
val input by parser.option(ArgType.String, shortName = "i",
    description = "Input file").required()
parser.parse(args)
val file = File(input)
file.useLines {
    it.mapIndexed { index, line -> "$index - $line" }
        .forEach { File("${file.name}-copy").appendText(it) }
}
```

# СКРИПТЫ .MAIN.KTS

```
> ./test.main.kts
Value for option --input should be always provided in command line.
Usage: example options_list
Options:
    --input, -i -> Input file (always required) { String }
    --debug, -d [false] -> Turn on debug mode
    --help, -h -> Usage info
```

# ВСТРАИВАНИЕ СКРИПТОВОГО ДВИЖКА В ПРИЛОЖЕНИЕ

# ЗАЧЕМ?

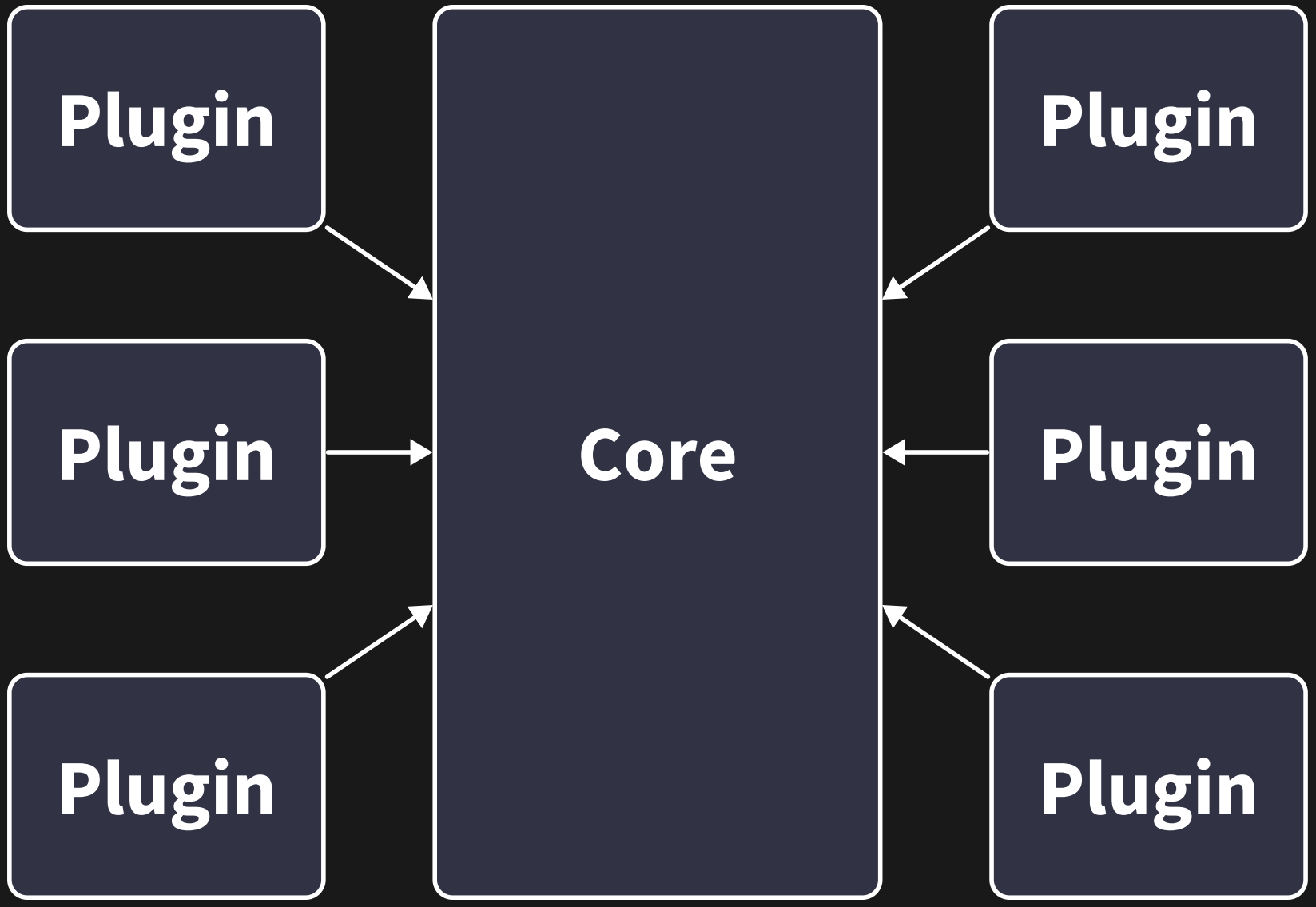
# КОНФИГУРАЦИЯ ЧЕРЕЗ "TYPESAFE DSL"

```
val revealKtVersion = "0.2.4"

group = "dev.limebeck"
version = "1.0.0"

plugins {
    kotlin("jvm") version "1.9.10"
}

repositories {
    mavenLocal()
    mavenCentral()
}

dependencies {
    implementation("dev.limebeck:revealkt-script-definition:$revealKtVersion")
}
```

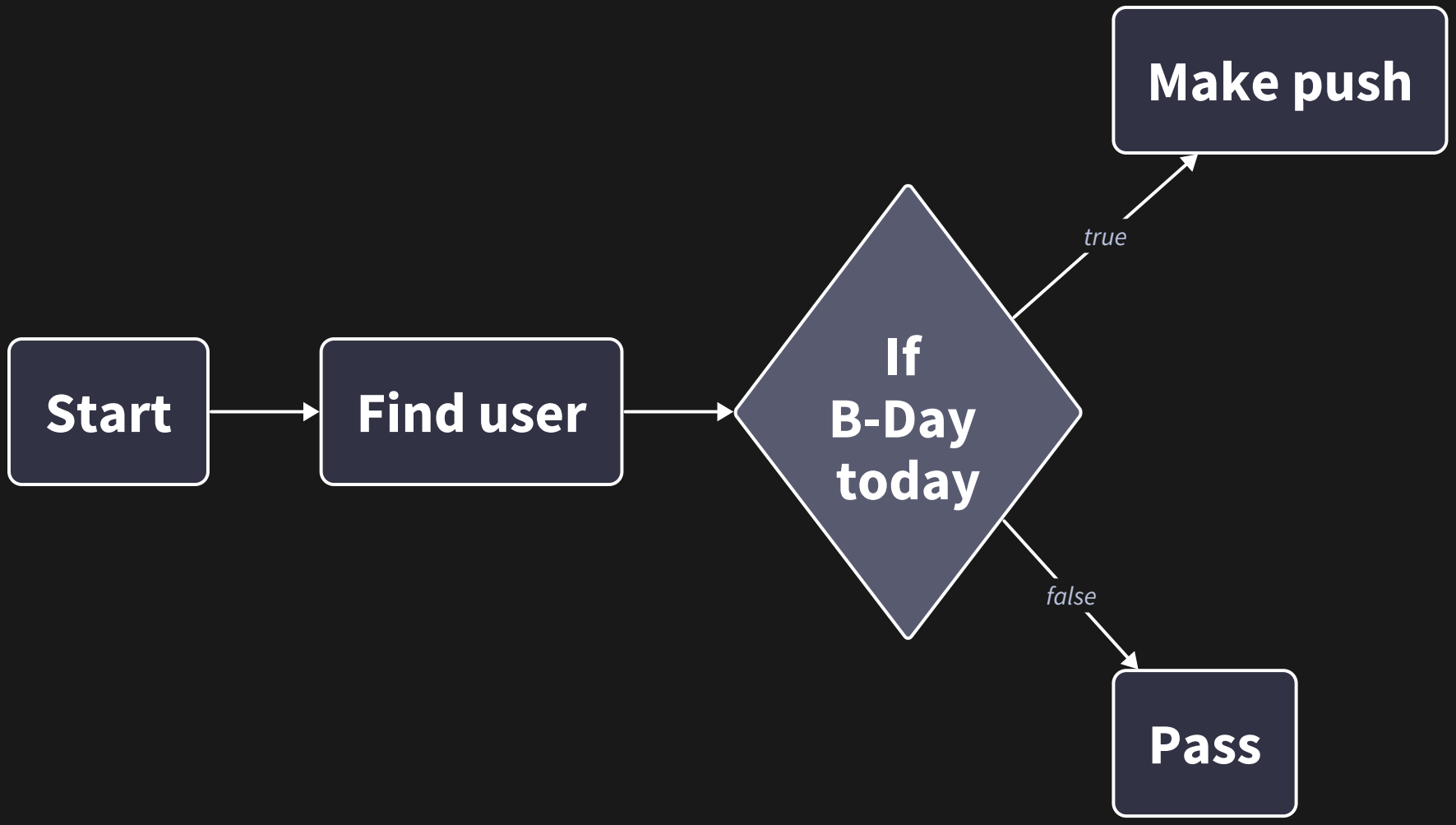
# КОНФИГУРАЦИЯ ЧЕРЕЗ "TYPESAFE DSL"

```
title = "Kotlin Script: для кого, зачем и как"

configuration {
    controls = false
    progress = false
}

slides {
    slide {
        autoanimate = true
        +title { "Kotlin Script: для кого, зачем и как" }
    }
}
```

# ПЛАГИННАЯ АРХИТЕКТУРА



# КАСТОМИЗАЦИЯ ДЕЙСТВИЙ ПОЛЬЗОВАТЕЛЕМ



# КАСТОМИЗАЦИЯ ДЕЙСТВИЙ ПОЛЬЗОВАТЕЛЕМ

## FIND USER

```
val userId = ctx["USER_ID"] as? String
    ?: throw NoSuchPropertyException("USER_ID was not found")

ctx["USER_DATA"] = ctx.withDatabaseConnectionOf("USERS_DB") { db ->
    db.select()
        .from(UsersTable)
        .where(UsersTable.ID eq userId)
        .fetchOne { r ->
            User(
                id = r[UsersTable.ID],
                name = r[UsersTable.NAME]
             )
        }
}
```

# ПРАКТИКА

# GITLAB-CI.KTS

```
val build = job("build") {
    stage = Stage("build")
    image = Image.of("alpine:latest")
    script("echo 'Hello World from build'")
}
listOf("dev-0", "dev-1", "dev-2").forEach {
    job("deploy-nullit") {
        stage = Stage("deploy")
        needs(build)
        image = Image.of("alpine:latest")
        script("echo Deployed nullit")
    }
}
```

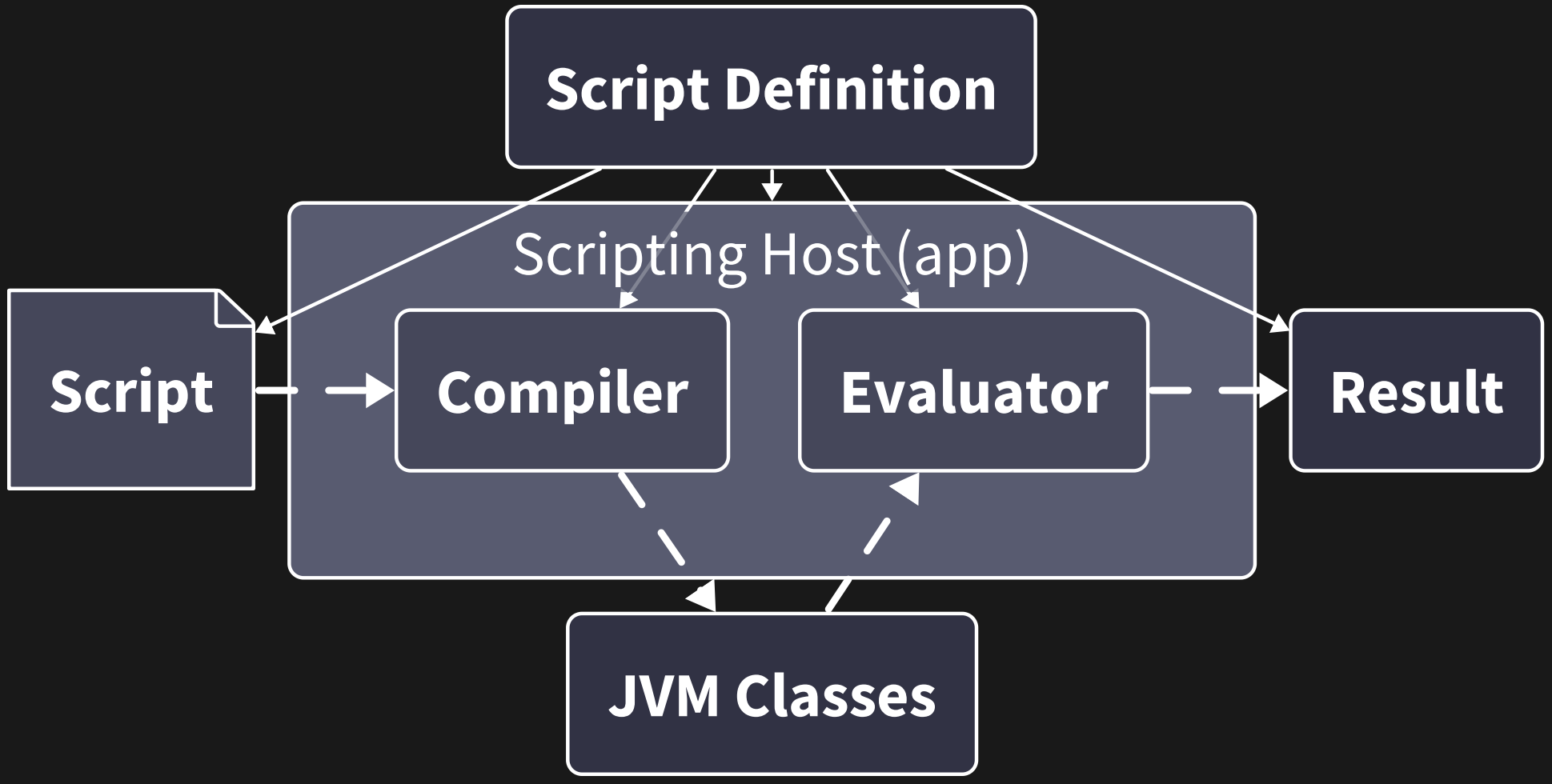
```
> java -jar gitlab-cli.jar ./main.gitlab-ci.kts > .gitlab-ci.yaml
```

# .gitlab-ci.yaml

```yaml
stages:
  - build
  - deploy
build:
  stage: build
  image:
    name: alpine:latest
  script:
    - echo 'Hello World from build'
deploy-dev-0:
  stage: deploy
  image:
    name: alpine:latest
  needs:
    - build
  script:
    - echo Deployed dev-0
deploy-dev-1:
```

# ОСНОВНЫЕ КОМПОНЕНТЫ СКРИПТИНГА

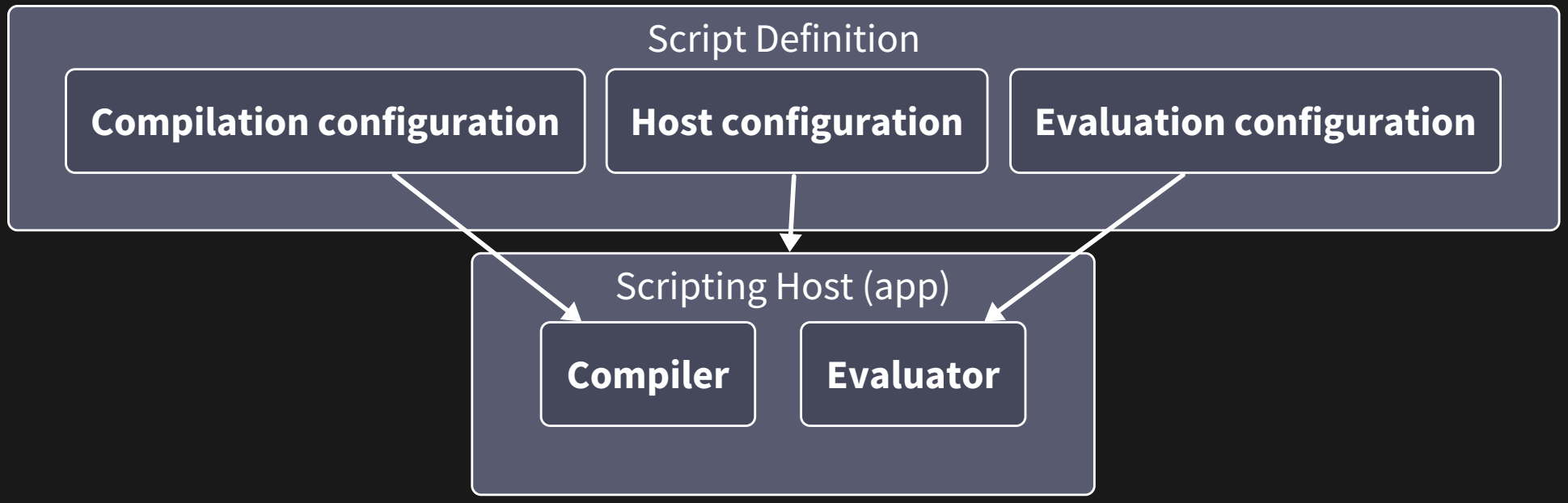
# ОСНОВНЫЕ КОМПОНЕНТЫ СКРИПТИНГА



# ОСНОВНЫЕ КОМПОНЕНТЫ СКРИПТИНГА

- Описание скрипта - Script Definition
- Исполнение скрипта - Scripting Host

# SCRIPT DEFINITION



# БАЗОВЫЙ ПРИМЕР

```kotlin
@KotlinScript(
    fileExtension = "gitlab-ci.kts",
    displayName = "Gitlab CI Kotlin configuration",
)
abstract class GitlabCiKtScript
//Класс обязательно должен быть открытым или абстрактным
```

# БАЗОВЫЙ ПРИМЕР

```kotlin
@KotlinScript(
    fileExtension = "gitlab-ci.kts",
    displayName = "Gitlab CI Kotlin configuration",
)
abstract class GitlabCiKtScript
//Класс обязательно должен быть открытым или абстрактным
```

# БАЗОВЫЙ ПРИМЕР

```kotlin
@KotlinScript(
    fileExtension = "gitlab-ci.kts",
    displayName = "Gitlab CI Kotlin configuration",
)
abstract class GitlabCiKtScript
//Класс обязательно должен быть открытым или абстрактным
```

# БАЗОВЫЙ ПРИМЕР

```kotlin
@KotlinScript(
    fileExtension = "gitlab-ci.kts",
    displayName = "Gitlab CI Kotlin configuration",
)
abstract class GitlabCiKtScript
//Класс обязательно должен быть открытым или абстрактным
```

# БАЗОВЫЙ ПРИМЕР

```kotlin
@KotlinScript(
    fileExtension = "gitlab-ci.kts",
    displayName = "Gitlab CI Kotlin configuration",
)
abstract class GitlabCiKtScript
//Класс обязательно должен быть открытым или абстрактным
```

# БАЗОВЫЙ ПРИМЕР

```kotlin
@KotlinScript(
    fileExtension = "gitlab-ci.kts",
    displayName = "Gitlab CI Kotlin configuration",
)
abstract class GitlabCiKtScript {
    val myProperty = 123
}
```

# БАЗОВЫЙ ПРИМЕР

```
println(myProperty)
//123
```

# КОНФИГУРАЦИЯ КОМПИЛЯЦИИ

# КОНФИГУРАЦИЯ КОМПИЛЯЦИИ

- Зависимости

# КОНФИГУРАЦИЯ КОМПИЛЯЦИИ

- Зависимости
- Импорты по умолчанию

# КОНФИГУРАЦИЯ КОМПИЛЯЦИИ

- Зависимости
- Импорты по умолчанию
- Конфигурация IDE

# КОНФИГУРАЦИЯ КОМПИЛЯЦИИ

- Зависимости
- Импорты по умолчанию
- Конфигурация IDE
- Параметры компилятора Kotlin

# КОНФИГУРАЦИЯ КОМПИЛЯЦИИ

- Зависимости
- Импорты по умолчанию
- Конфигурация IDE
- Параметры компилятора Kotlin
- Доступные в скрипте свойства

# КОНФИГУРАЦИЯ КОМПИЛЯЦИИ

- Зависимости
- Импорты по умолчанию
- Конфигурация IDE
- Параметры компилятора Kotlin
- Доступные в скрипте свойства
- Определение неявных (implicit) ресиверов

# КОНФИГУРАЦИЯ КОМПИЛЯЦИИ ПРИМЕР

```kotlin
object GitlabCiKtScriptCompilationConfiguration : ScriptCompilationConfiguration({
    jvm {
        dependenciesFromClassContext(PipelineBuilder::class, wholeClasspath = true)
    }
    defaultImports(
        "dev.otbe.gitlab.ci.core.model.*",
        "dev.otbe.gitlab.ci.dsl.*",
        "dev.otbe.gitlab.ci.core.goesTo"
    )
    ide { acceptedLocations(ScriptAcceptedLocation.Everywhere) }
    compilerOptions.append("-Xcontext-receivers")
    providedProperties("propName" to String::class)
    implicitReceivers(PipelineBuilder::class)
})
```

# КОНФИГУРАЦИЯ КОМПИЛЯЦИИ ПРИМЕР

```kotlin
object GitlabCiKtScriptCompilationConfiguration : ScriptCompilationConfiguration({
    jvm {
        dependenciesFromClassContext(PipelineBuilder::class, wholeClasspath = true)
    }
    defaultImports(
        "dev.otbe.gitlab.ci.core.model.*",
        "dev.otbe.gitlab.ci.dsl.*",
        "dev.otbe.gitlab.ci.core.goesTo"
    )
    ide { acceptedLocations(ScriptAcceptedLocation.Everywhere) }
    compilerOptions.append("-Xcontext-receivers")
    providedProperties("propName" to String::class)
    implicitReceivers(PipelineBuilder::class)
})
```

# КОНФИГУРАЦИЯ КОМПИЛЯЦИИ ПРИМЕР

```kotlin
object GitlabCiKtScriptCompilationConfiguration : ScriptCompilationConfiguration({
    jvm {
        dependenciesFromClassContext(PipelineBuilder::class, wholeClasspath = true)
    }
    defaultImports(
        "dev.otbe.gitlab.ci.core.model.*",
        "dev.otbe.gitlab.ci.dsl.*",
        "dev.otbe.gitlab.ci.core.goesTo"
    )
    ide { acceptedLocations(ScriptAcceptedLocation.Everywhere) }
    compilerOptions.append("-Xcontext-receivers")
    providedProperties("propName" to String::class)
    implicitReceivers(PipelineBuilder::class)
})
```

# КОНФИГУРАЦИЯ КОМПИЛЯЦИИ ПРИМЕР

```kotlin
object GitlabCiKtScriptCompilationConfiguration : ScriptCompilationConfiguration({
    jvm {
        dependenciesFromClassContext(PipelineBuilder::class, wholeClasspath = true)
    }
    defaultImports(
        "dev.otbe.gitlab.ci.core.model.*",
        "dev.otbe.gitlab.ci.dsl.*",
        "dev.otbe.gitlab.ci.core.goesTo"
    )
    ide { acceptedLocations(ScriptAcceptedLocation.Everywhere) }
    compilerOptions.append("-Xcontext-receivers")
    providedProperties("propName" to String::class)
    implicitReceivers(PipelineBuilder::class)
})
```

# КОНФИГУРАЦИЯ КОМПИЛЯЦИИ ПРИМЕР

```kotlin
object GitlabCiKtScriptCompilationConfiguration : ScriptCompilationConfiguration({
    jvm {
        dependenciesFromClassContext(PipelineBuilder::class, wholeClasspath = true)
    }
    defaultImports(
        "dev.otbe.gitlab.ci.core.model.*",
        "dev.otbe.gitlab.ci.dsl.*",
        "dev.otbe.gitlab.ci.core.goesTo"
    )
    ide { acceptedLocations(ScriptAcceptedLocation.Everywhere) }
    compilerOptions.append("-Xcontext-receivers")
    providedProperties("propName" to String::class)
    implicitReceivers(PipelineBuilder::class)
})
```

# КОНФИГУРАЦИЯ КОМПИЛЯЦИИ ПРИМЕР

```kotlin
object GitlabCiKtScriptCompilationConfiguration : ScriptCompilationConfiguration({
    jvm {
        dependenciesFromClassContext(PipelineBuilder::class, wholeClasspath = true)
    }
    defaultImports(
        "dev.otbe.gitlab.ci.core.model.*",
        "dev.otbe.gitlab.ci.dsl.*",
        "dev.otbe.gitlab.ci.core.goesTo"
    )
    ide { acceptedLocations(ScriptAcceptedLocation.Everywhere) }
    compilerOptions.append("-Xcontext-receivers")
    providedProperties("propName" to String::class)
    implicitReceivers(PipelineBuilder::class)
})
```

# КОНФИГУРАЦИЯ КОМПИЛЯЦИИ

# ПРИМЕР

```
object GitlabCiKtScriptCompilationConfiguration : ScriptCompilationConfiguration({
    jvm {
        dependenciesFromClassContext(PipelineBuilder::class, wholeClasspath = true)
    }
    defaultImports(
        "dev.otbe.gitlab.ci.core.model.*",
        "dev.otbe.gitlab.ci.dsl.*",
        "dev.otbe.gitlab.ci.core.goesTo"
    )
    ide { acceptedLocations(ScriptAcceptedLocation.Everywhere) }
    compilerOptions.append("-Xcontext-receivers")
    providedProperties("propName" to String::class)
    implicitReceivers(PipelineBuilder::class)
})
```

# КОНФИГУРАЦИЯ КОМПИЛЯЦИИ ПРИМЕР

PipelineBuilder.kt

```
class PipelineBuilder {
    var stages: List<Stage> = mutableListOf()
    fun stages(vararg stage: String) {
        stages += stage.asList().map { Stage(it) }
    }
}
```

example.gitlab-ci.kts

```
//this: PipelineBuilder
stages("build", "deploy")
```

# КОНФИГУРАЦИЯ КОМПИЛЯЦИИ

# ПРИМЕР

```
@KotlinScript(
    fileExtension = "gitlab-ci.kts",
    displayName = "Gitlab CI Kotlin configuration",
    compilationConfiguration = GitlabCiKtScriptCompilationConfiguration::class,
)
abstract class GitlabCiKtScript
```

# КОНФИГУРАЦИЯ КОМПИЛЯЦИИ ВНЕШНИЕ ЗАВИСИМОСТИ

```
object GitlabCiKtScriptCompilationConfiguration : ScriptCompilationConfiguration({
    defaultImports(DependsOn::class, Repository::class)
    refineConfiguration {
        onAnnotations(
            DependsOn::class,
            Repository::class,
            // Обработчик аннотаций
            handler = ::configureMavenDepsOnAnnotations
        )
    }

    jvm {
        dependenciesFromClassContext(PipelineBuilder::class, wholeClasspath = true)
    }
    defaultImports(
        "dev.otbe.gitlab.ci.core.model.*",
        "dev.otbe.gitlab.ci.dsl.*",
        "dev.otbe.gitlab.ci.core.goesTo"
```

# КОНФИГУРАЦИЯ КОМПИЛЯЦИИ
# ВНЕШНИЕ ЗАВИСИМОСТИ

example.gitlab-ci.kts

```
@file:Repository("https://private-nexus.company.org/ci-templates/")
@file:DependsOn("org.company.ci.templates:jvm-jobs:1.0.0")

import org.company.ci.templates.jvm.jobs.*

val appName = "kotlin-app"
gradleJob {
    task = "build"
    artifact = "./build/libs/null{appName}.jar"
}
```

# КОНФИГУРАЦИЯ ИСПОЛНЕНИЯ

# КОНФИГУРАЦИЯ ИСПОЛНЕНИЯ

- Параметры запуска JVM

# КОНФИГУРАЦИЯ ИСПОЛНЕНИЯ

- Параметры запуска JVM
- Передача созданных экземпляров implicit ресиверов

# КОНФИГУРАЦИЯ ИСПОЛНЕНИЯ

- Параметры запуска JVM
- Передача созданных экземпляров implicit ресиверов
- Аргументы конструктора для базового класса скрипта

# КОНФИГУРАЦИЯ ИСПОЛНЕНИЯ

- Параметры запуска JVM
- Передача созданных экземпляров implicit ресиверов
- Аргументы конструктора для базового класса скрипта
- Возможность разделения инстансов скрипта

# КОНФИГУРАЦИЯ ИСПОЛНЕНИЯ

- Параметры запуска JVM
- Передача созданных экземпляров implicit ресиверов
- Аргументы конструктора для базового класса скрипта
- Возможность разделения инстансов скрипта
- Просмотр истории запусков (для REPL)

# КОНФИГУРАЦИЯ ИСПОЛНЕНИЯ

- Параметры запуска JVM
- Передача созданных экземпляров implicit ресиверов
- Аргументы конструктора для базового класса скрипта
- Возможность разделения инстансов скрипта
- Просмотр истории запусков (для REPL)
- Возможность переопределения любых частей скрипта

# КОНФИГУРАЦИЯ ИСПОЛНЕНИЯ ПРИМЕР

```kotlin
object GitlabCiKtEvaluationConfiguration : ScriptEvaluationConfiguration({
    scriptsInstancesSharing(false)
})

@KotlinScript(
    fileExtension = "gitlab-ci.kts",
    displayName = "Gitlab CI Kotlin configuration",
    compilationConfiguration = GitlabCiKtScriptCompilationConfiguration::class,
    evaluationConfiguration = GitlabCiKtEvaluationConfiguration::class,
)
abstract class GitlabCiKtScript
```

# КОНФИГУРАЦИЯ ИСПОЛНЕНИЯ ПРИМЕР

```kotlin
object GitlabCiKtEvaluationConfiguration : ScriptEvaluationConfiguration({
    scriptsInstancesSharing(false)
    constructorArgs("My Arg String")
})

@KotlinScript(
    fileExtension = "gitlab-ci.kts",
    displayName = "Gitlab CI Kotlin configuration",
    compilationConfiguration = GitlabCiKtScriptCompilationConfiguration::class,
    evaluationConfiguration = GitlabCiKtEvaluationConfiguration::class,
)
abstract class GitlabCiKtScript(val arg: String)
```

# КОНФИГУРАЦИЯ ХОСТА

# КОНФИГУРАЦИЯ ХОСТА

# ПРИМЕР

```
object GitlabCiKtHostConfiguration : ScriptingHostConfiguration({
    jvm {
        val cacheBaseDir = findCacheBaseDir()
        if (cacheBaseDir != null)
            compilationCache(
                CompiledScriptJarsCache { script, scriptCompilationConfiguration ->
                    cacheBaseDir
                        .resolve(
                            compiledScriptUniqueName(
                                script,
                                scriptCompilationConfiguration
                            )
                        )
                }
            )
    }
})
```

# КОНФИГУРАЦИЯ ХОСТА

# ПРИМЕР

```
@KotlinScript(
    displayName = "Gitlab CI Kotlin configuration",
    fileExtension = "gitlab-ci.kts",
    compilationConfiguration = GitlabCiKtScriptCompilationConfiguration::class,
    evaluationConfiguration = GitlabCiKtEvaluationConfiguration::class,
    hostConfiguration = GitlabCiKtHostConfiguration::class,
)
abstract class GitlabCiKtScript
```

# SCRIPT LOADER

# SCRIPT LOADER

```
fun BasicJvmScriptingHost.evalFile(
    scriptFile: File
): ResultWithDiagnostics<EvaluationResult> {
    val compilationConfiguration =
        createJvmCompilationConfigurationFromTemplate<GitlabCiKtScript> {}
    val evaluationConfiguration =
        createJvmEvaluationConfigurationFromTemplate<GitlabCiKtScript> {}
    return eval(
        script = scriptFile.toScriptSource(),
        compilationConfiguration = compilationConfiguration,
        evaluationConfiguration = evaluationConfiguration
    )
}

val scriptingHost = BasicJvmScriptingHost()
val result = scriptingHost.evalFile(scriptFile)
```

# ПОДСВЕТКА В IDE

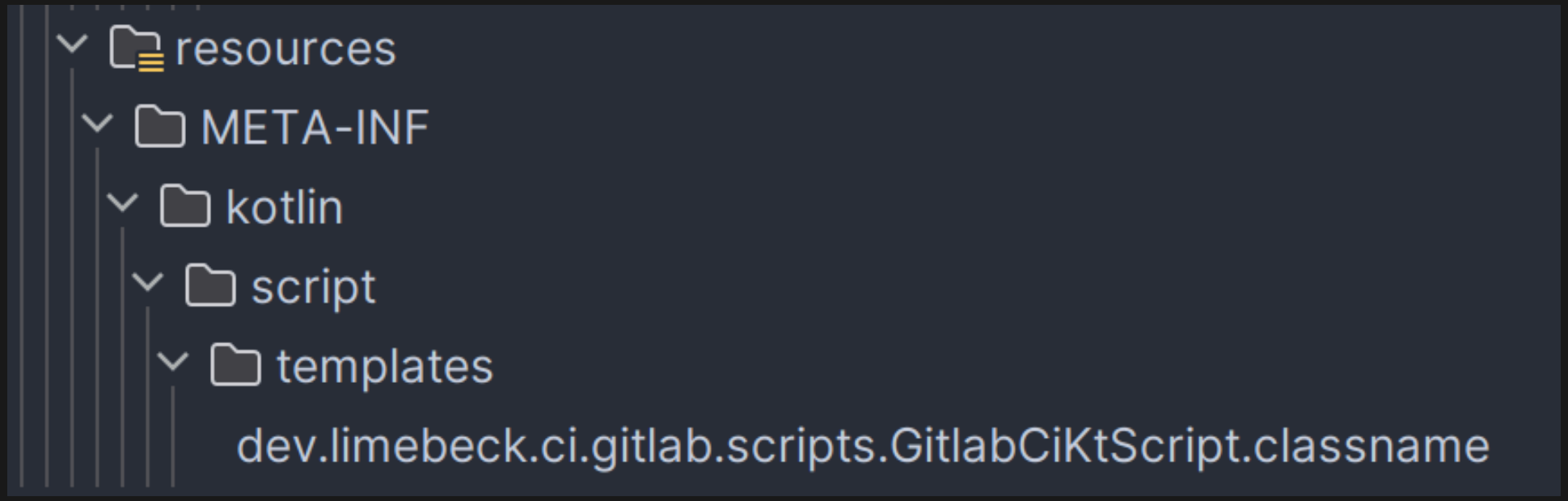
# ПОДСВЕТКА В IDE

Пустой файл в META-INF/kotlin/script/templates с полным именем класса в названии

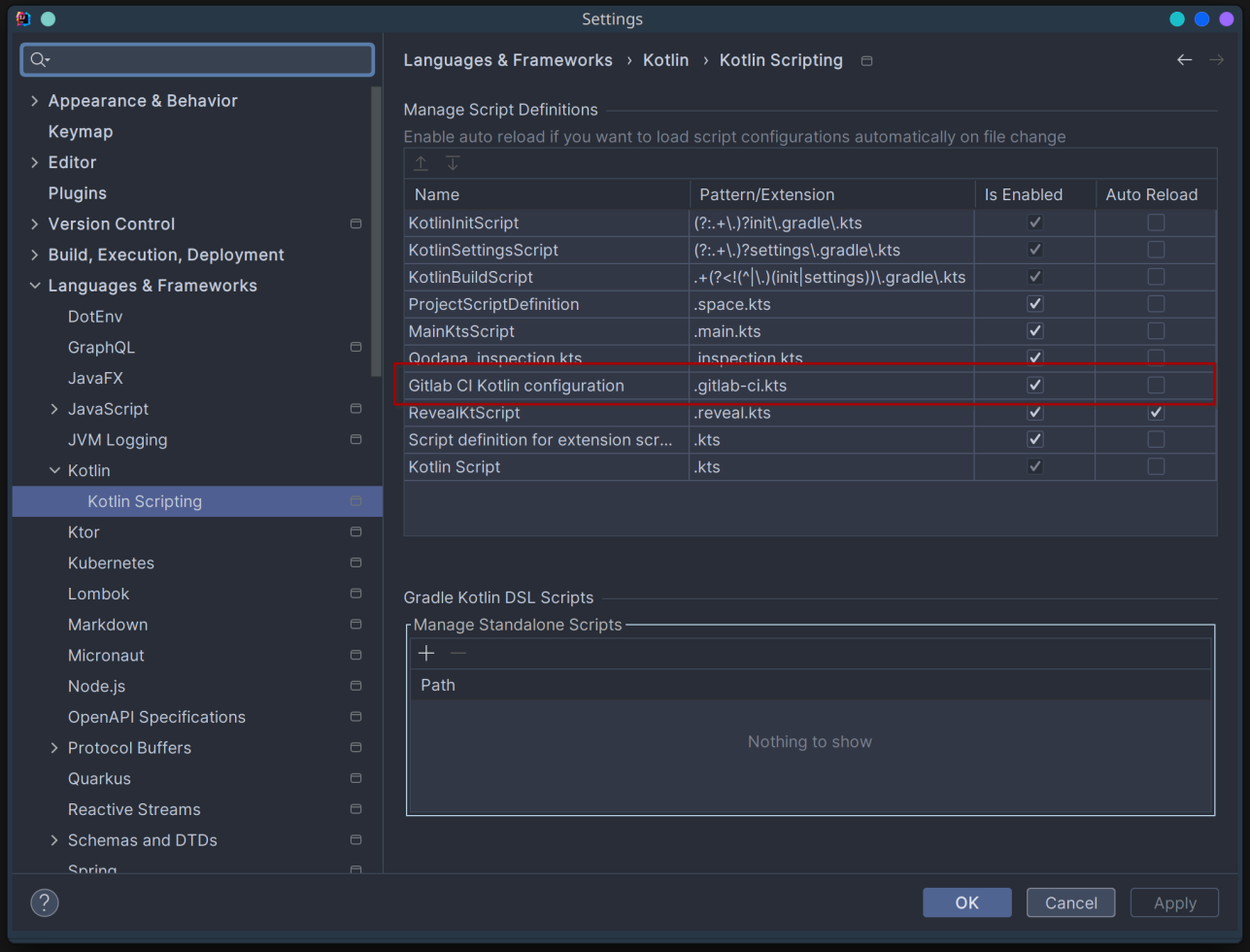
# ПОДСВЕТКА В IDE

META-INF/kotlin/script/templates/
dev.limebeck.ci.gitlab.scripts.GitlabCiKtScript.classnam

# ПОДСВЕТКА В IDE



# ПОДСВЕТКА В IDE



# ПОДСВЕТКА В IDE

Manage Script Definitions

Enable auto reload if you want to load script configurations automatically on file change

| Name | Pattern/Extension | Is Enabled | Auto Reload |
|---|---|---|---|
| KotlinInitScript | (?:.+\.)?init\.gradle\.kts | ☑ | ☐ |
| KotlinSettingsScript | (?:.+\.)?settings\.gradle\.kts | ☑ | ☐ |
| KotlinBuildScript | .+(?<!(^\|\.)(init\|settings))\.gradle\.kts | ☑ | ☐ |
| ProjectScriptDefinition | .space.kts | ☑ | ☐ |
| MainKtsScript | .main.kts | ☑ | ☐ |
| Qodana .inspection.kts | .inspection.kts | ☑ | ☐ |
| Gitlab CI Kotlin configuration | .gitlab-ci.kts | ☑ | ☐ |
| RevealKtScript | .reveal.kts | ☑ | ☑ |
| Script definition for extension script... | .kts | ☑ | ☐ |
| Kotlin Script | .kts | ☑ | ☐ |

# ПОДСВЕТКА В IDE

- Работает только в IntelliJ IDEA / Fleet
- Нужен проект Gradle/Maven для работы

# JAVA SCRIPTING API (JSR223)

# АБСТРАКЦИЯ В JVM ДЛЯ ИСПОЛНЕНИЯ СКРИПТОВ

# KOTLIN SCRIPT + JSR223

Обертка поверх BasicJvmScriptingHost

# БЕЗОПАСНОСТЬ В KOTLIN SCRIPT

# БЕЗОПАСНОСТЬ В KOTLIN SCRIPT

- Ограничение по модулям JVM

# БЕЗОПАСНОСТЬ В KOTLIN SCRIPT

- Ограничение по модулям JVM
- Ограничение по доступным классам

# НЕДОСТАТКИ ВСТРАИВАНИЯ

- Компилятор Kotlin в приложении

# НЕДОСТАТКИ ВСТРАИВАНИЯ

- Компилятор Kotlin в приложении
- Поддержка IDE - JetBrains only

# НЕДОСТАТКИ ВСТРАИВАНИЯ

- Компилятор Kotlin в приложении
- Поддержка IDE - JetBrains only
- Мало документации

# НЕДОСТАТКИ ВСТРАИВАНИЯ

- Компилятор Kotlin в приложении
- Поддержка IDE - JetBrains only
- Мало документации
- Только на JVM

# НЕДОСТАТКИ ВСТРАИВАНИЯ

- Компилятор Kotlin в приложении
- Поддержка IDE - JetBrains only
- Мало документации
- Только на JVM
- Долгий старт

# ПРЕИМУЩЕСТВА ВСТРАИВАНИЯ

- Нативно для Kotlin

# ПРЕИМУЩЕСТВА ВСТРАИВАНИЯ

- Нативно для Kotlin
- Расширяемость

# ПРЕИМУЩЕСТВА ВСТРАИВАНИЯ

- Нативно для Kotlin
- Расширяемость
- Поддержка собственных DSL

# ИТОГО ПО KOTLIN SCRIPT

- Зрелое решение (>6 лет)

# ИТОГО ПО KOTLIN SCRIPT

- Зрелое решение (>6 лет)
- Развивается

# ИТОГО ПО KOTLIN SCRIPT

- Зрелое решение (>6 лет)
- Развивается
- Лучше всего - со своим DSL

# ИТОГО ПО KOTLIN SCRIPT

- Зрелое решение (>6 лет)
- Развивается
- Лучше всего - со своим DSL
- Упрощает поддержку скриптов

# ССЫЛКА НА ПРЕЗЕНТАЦИЮ И ПОЛЕЗНЫЕ ШТУКИ