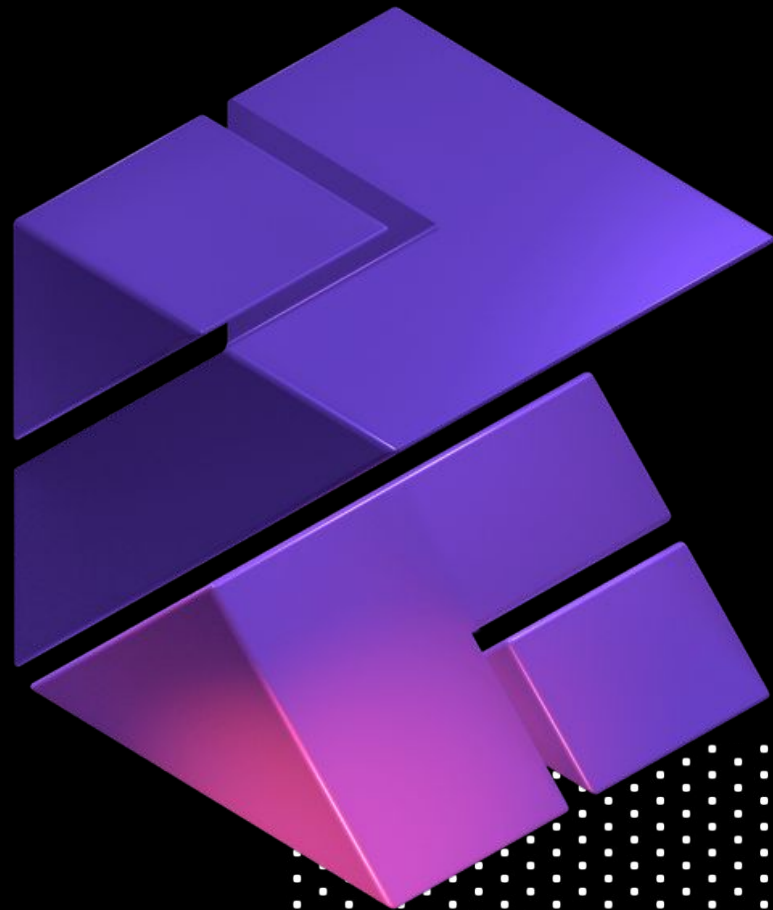
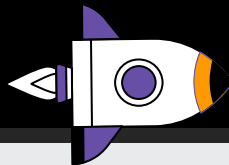


IN THE  
MEANTIME

 BOTLIN



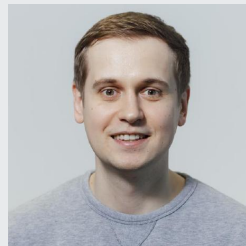
# ABOUT US:



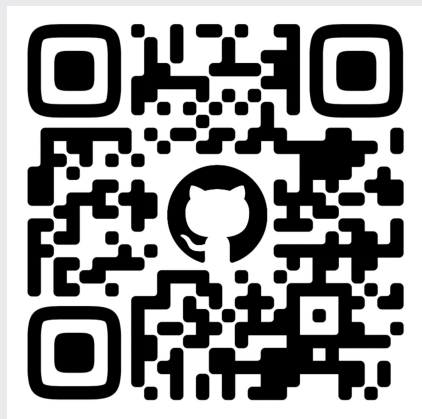
> Anzhelika Pokhodun

Kotlin developer

Developer in a major enterprise company, whose developers were among the first who took the risk of rolling out Kotlin, almost from the very first of its versions, into production.



> Andrey Kuleshov



Team Leader of DevTools team

An active open-source enthusiast & contributor to the Kotlin ecosystem. Creator of ktoml library and diktat static analyzer. Compiler and code analysis experience.

# JAVA SE



18

2022

19

2022

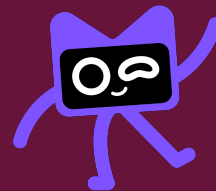
20

2023

21

2023

# KOTLIN



1.7.0 – 1.7.21

2022

1.8.0 – 1.8.20 2022 – 2023

1.9.0 - 1.9.20

2023



K2



Kotlin 2.0

PLAN

# RELEASE VERSIONS

## 1.7

- Data objects
- K2 alpha
- New operator

& other

## 1.9

now

- kotlin.time updates
- K2 beta
- Stable last features

& other

## 1.8

- Inline class updates
- K2 beta
- stdLib updates

& other

## 2.0

- K2 stable

plan



# RELEASE VERSIONS

## 1.7

- Data objects

**K2 alpha**

- New operator

& other



## 1.9

now

- kotlin.time updates

**K2 beta**

- Stable last features

& other

## 1.8

- Inline class updates

**K2 beta**

- stdLib updates

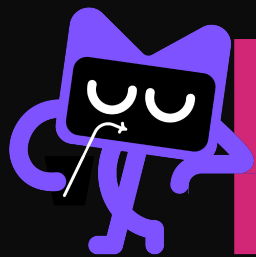
& other

## 2.0

**K2 stable**

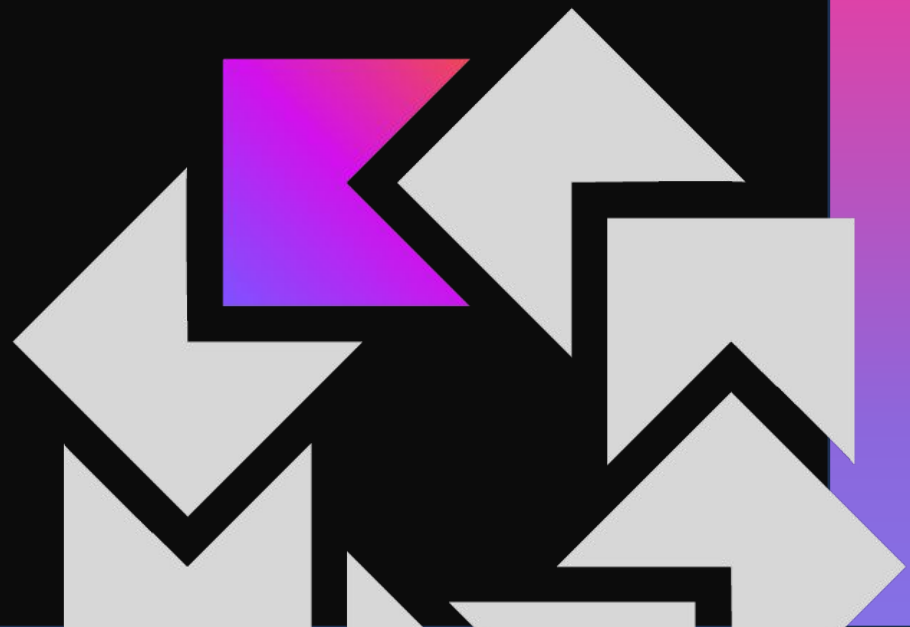
plan





# kotlin

**THE NEW  
COMPILER**



# COMPILER TERMINOLOGY



FE/BE

AST

K2

BINDING CONTEXT

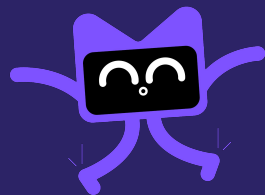
PSI

FIR

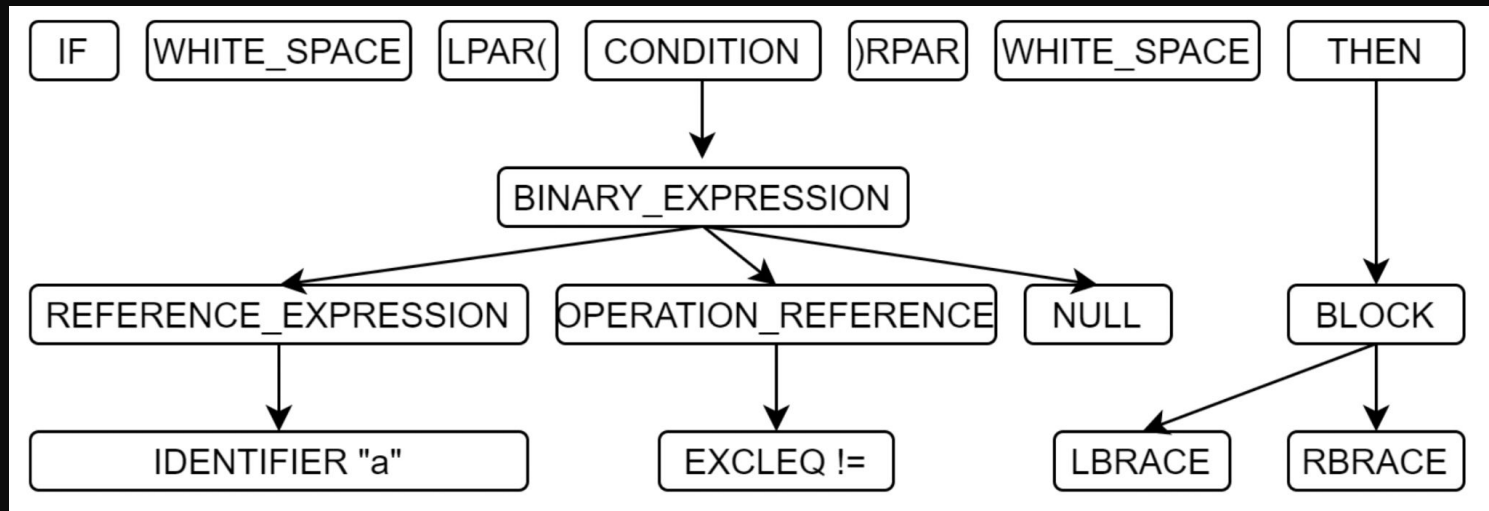
IR

LLVM

MLIR



# AST/PSI syntax



*“The Program Structure Interface, commonly referred to as just PSI, is the layer in the **IntelliJ Platform** responsible for parsing files and creating the syntactic and semantic code model that powers so many of the platform's features.”*



# PSI/AST

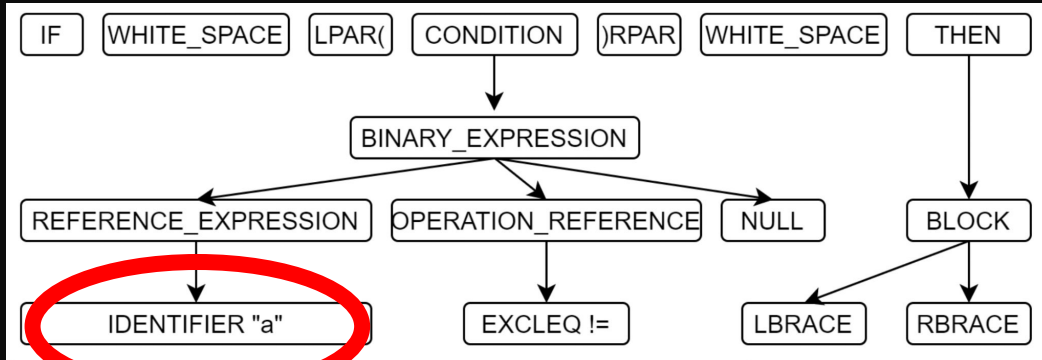


source from diktat

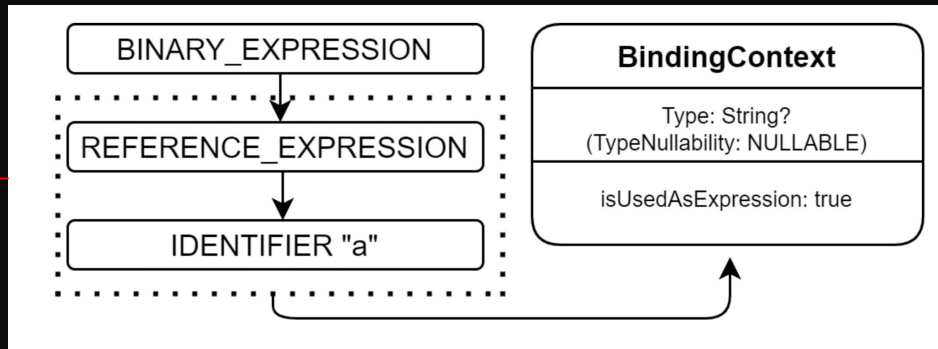
```
/**  
 * @return name of a function which is called in  
 * a [KtCallExpression] or null if it can't be found  
 */  
fun KtCallExpression.getFunctionName() =  
    (calleeExpression as? KtNameReferenceExpression)?.getReferencedName()
```

```
/**  
 * Checks whether [this] node of type PROPERTY is `var`  
 */  
fun ASTNode.isVarProperty() =  
    this.getChildren(null).any { it.elementType == KtTokens.VAR_KEYWORD }
```

# Binding Context semantics



A. Kuleshov, P. Trifanov, V. Frolov and G. Liang, "Diktat: Lightweight Static Analysis for Kotlin," 2021 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), pp. 365-370, doi:10.1109/ISSREW53611.2021.00103.

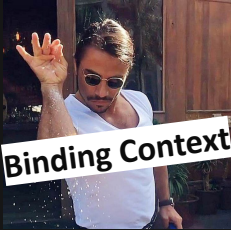
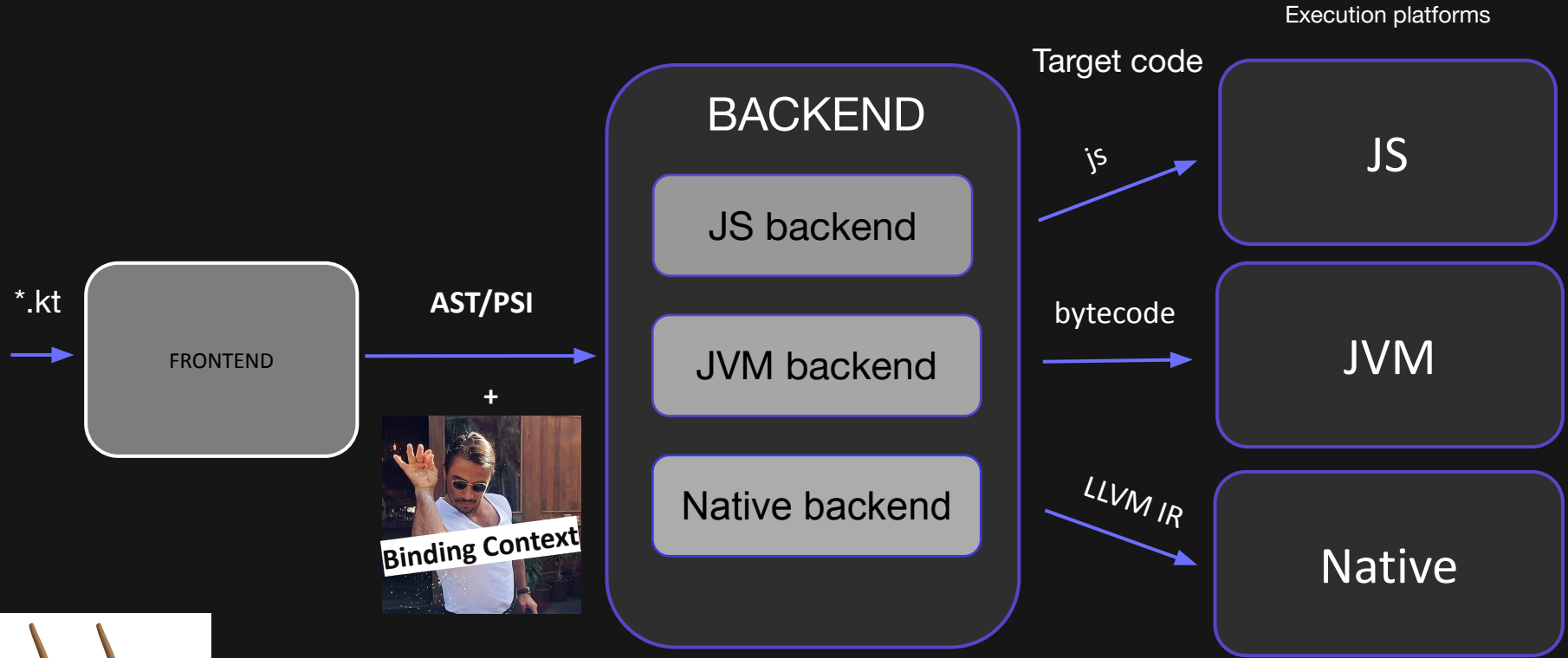


# TARGETS

build.gradle.kts

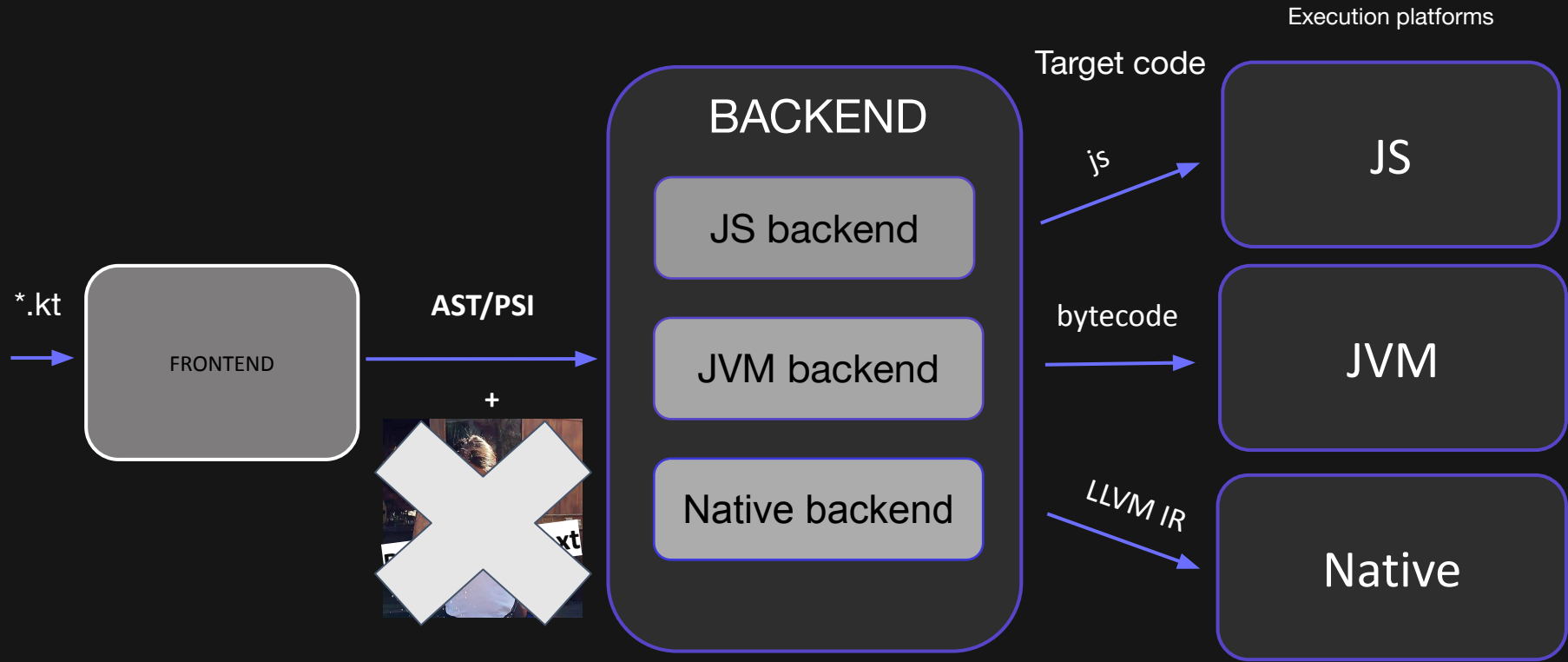
```
kotlin {  
    js(IR) { // (LEGACY)?  
        browser()  
        nodejs()  
    }  
  
    jvm()  
    mingwX64()  
    linuxX64()  
    macosX64()  
    macosArm64()  
    ios()  
    iosSimulatorArm64()  
    ...  
}
```

# EARLY COMPILER (after adding extra backends)

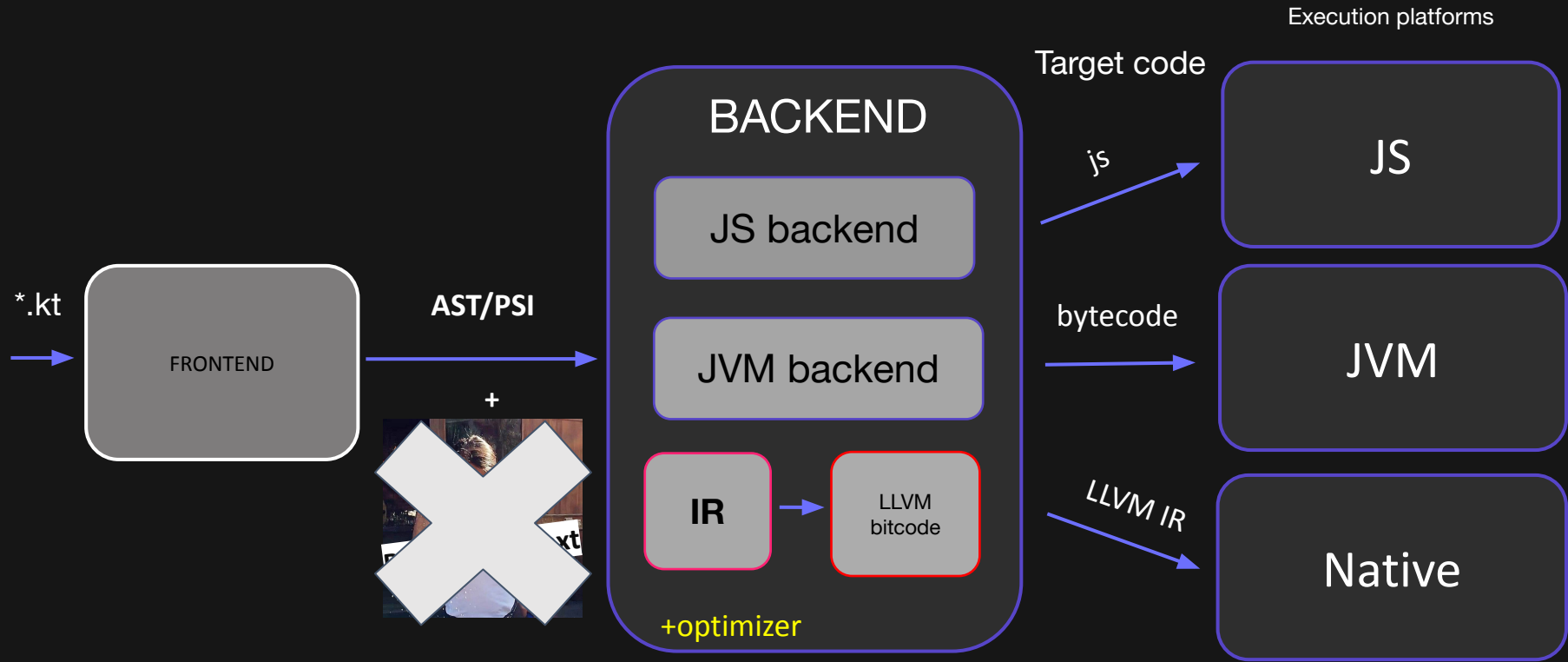


\*Clang + LLVM had same child problems = MLIR

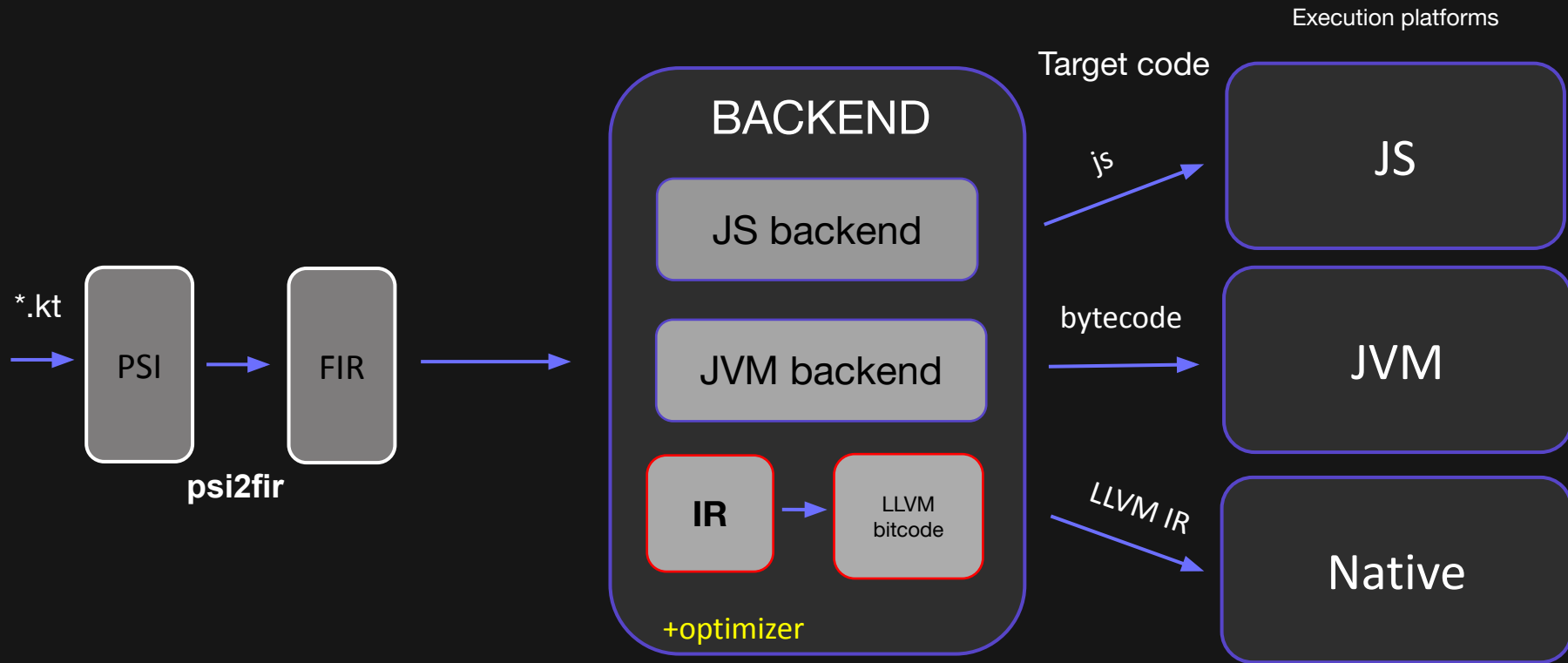
# COMPILER EVOLUTION



# COMPILER EVOLUTION



# COMPILER EVOLUTION



# K1 Frontend vs K2 Frontend CompileTime is 2x time faster

#Google

Kotlin 2.0 compiler

## Kotlin 2.0 compiler performance

Project	Kotlin 1.8 compiler	Kotlin 2.0 compiler	Improvement
Kotlin	5m 40s 2.9 KLOC/s	2m 37s 6.3 KLOC/s	~2.2x
IntelliJ IDEA Ultimate	34m 15s 2.5 KLOC/s	16m 59s 5.0 KLOC/s	~2.0x

# BUT..WHY?

KLOCs: thousands of lines of code processed per second  
Source: benchmark data by JetBrains



[Modern Android Development at Google I/O 2023](#)





K1 Frontend vs K2 Frontend CompileTime is 2x time faster

**YOU MIGHT ASK WHY**

K1 Frontend vs K2 Frontend CompileTime is 2x time faster

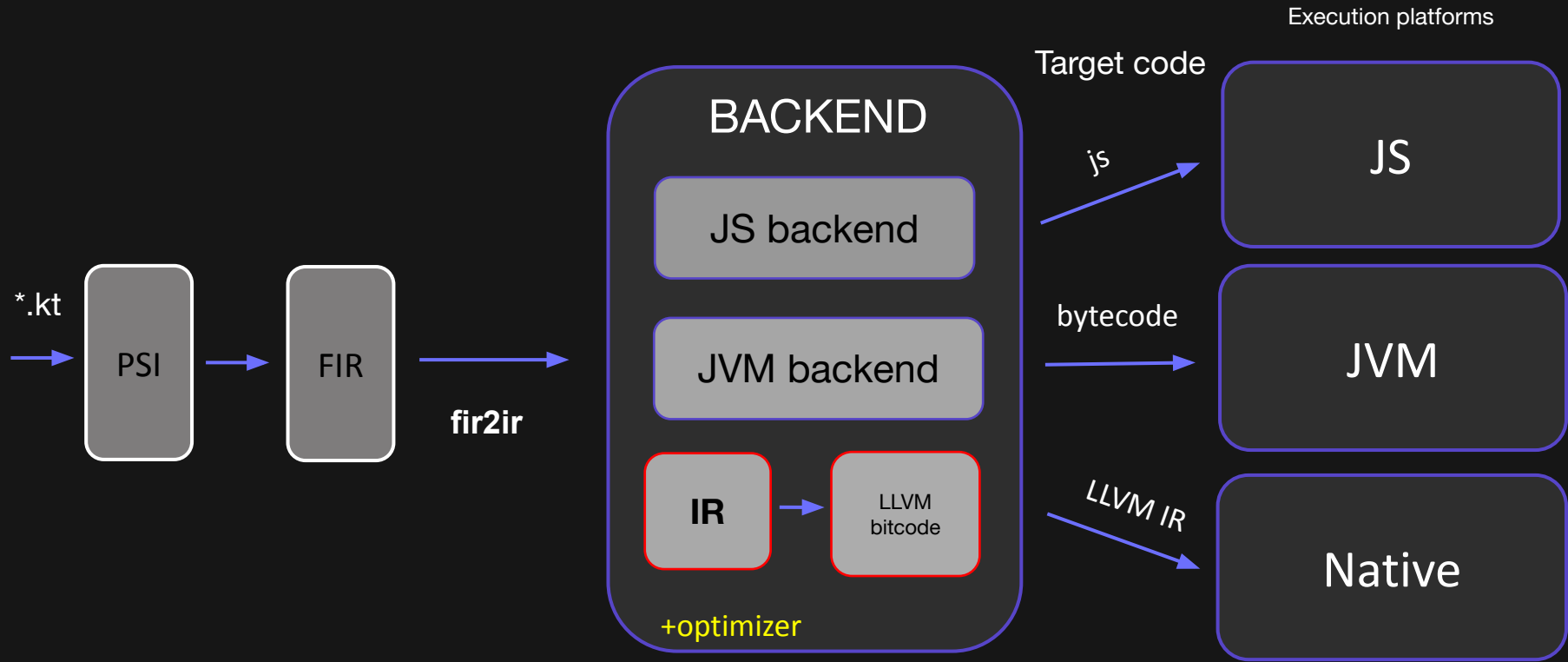
## YOU MIGHT ASK WHY



AST Structure became more “fat” (enriched)

Less calls to the map and less resolving

# COMPILER EVOLUTION



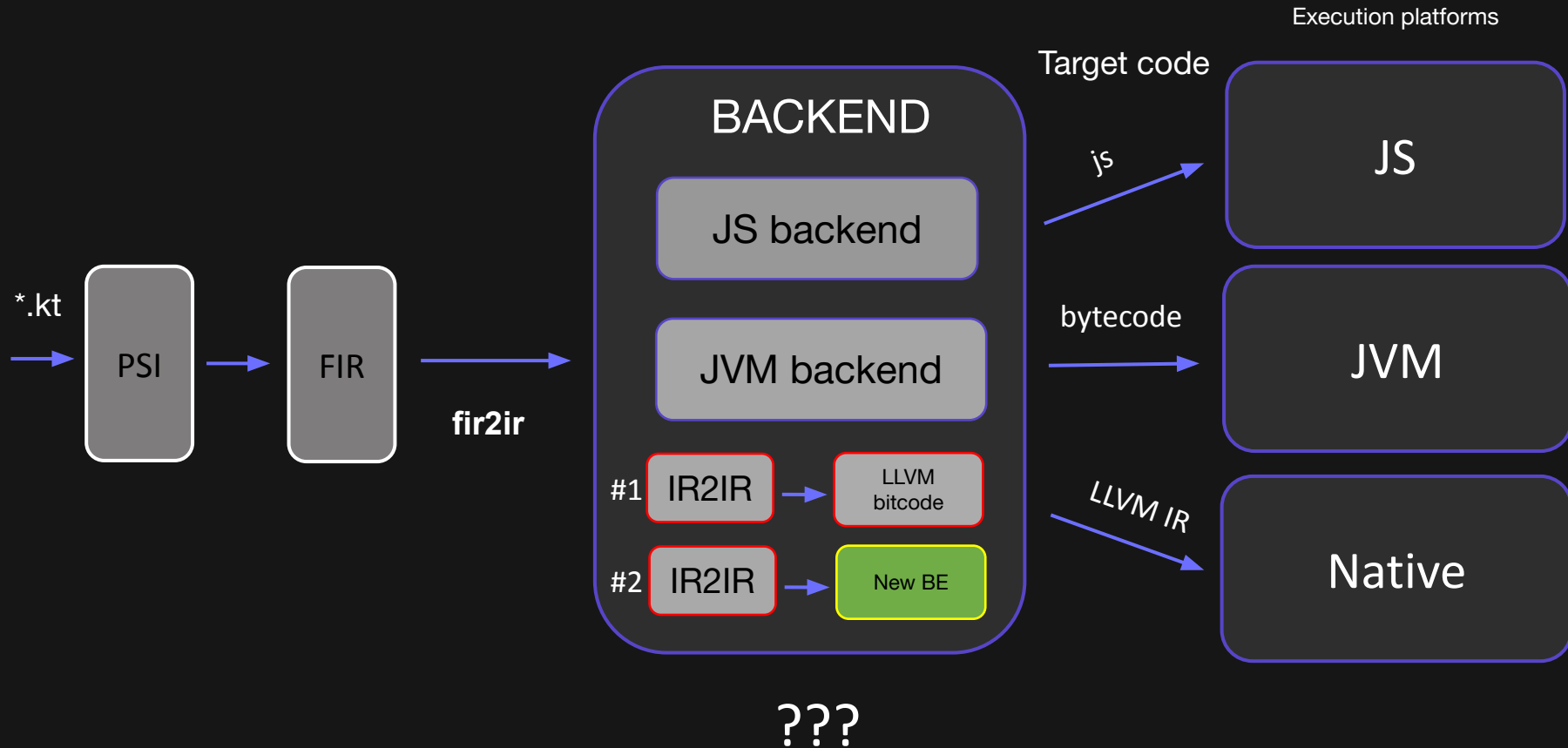
# DIFFERENCE WITH NORMAL BACKEND IRs

```
$ clang -c -emit-llvm -S
```

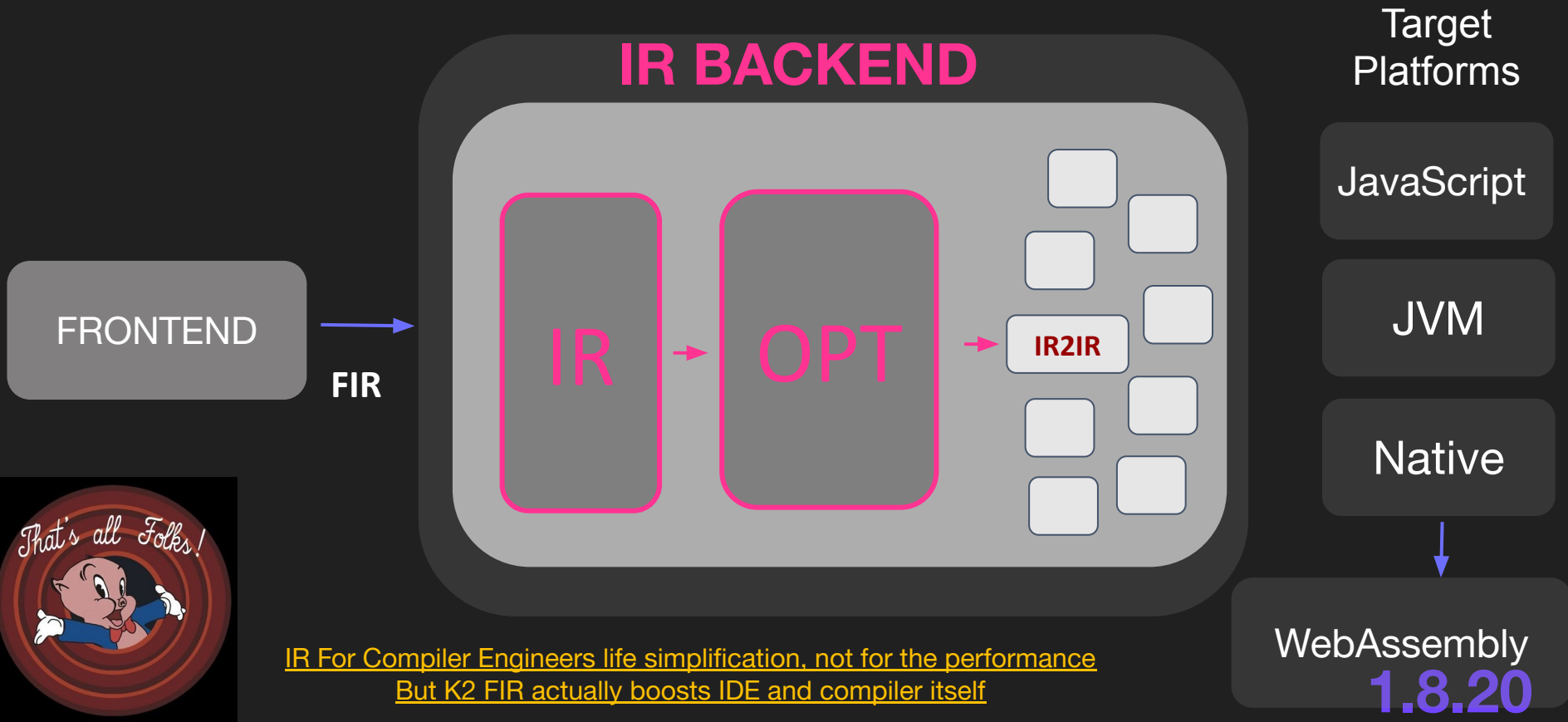
```
define i32 @main() #0 {  
    %1 = alloca i32, align 4  
    store i32 0, i32* %1, align 4  
    ret i32 0  
}
```

```
int main() {  
    return 0;  
}
```

# COMPILER EVOLUTION



# NEW COMPILER



IR For Compiler Engineers life simplification, not for the performance  
But K2 FIR actually boosts IDE and compiler itself

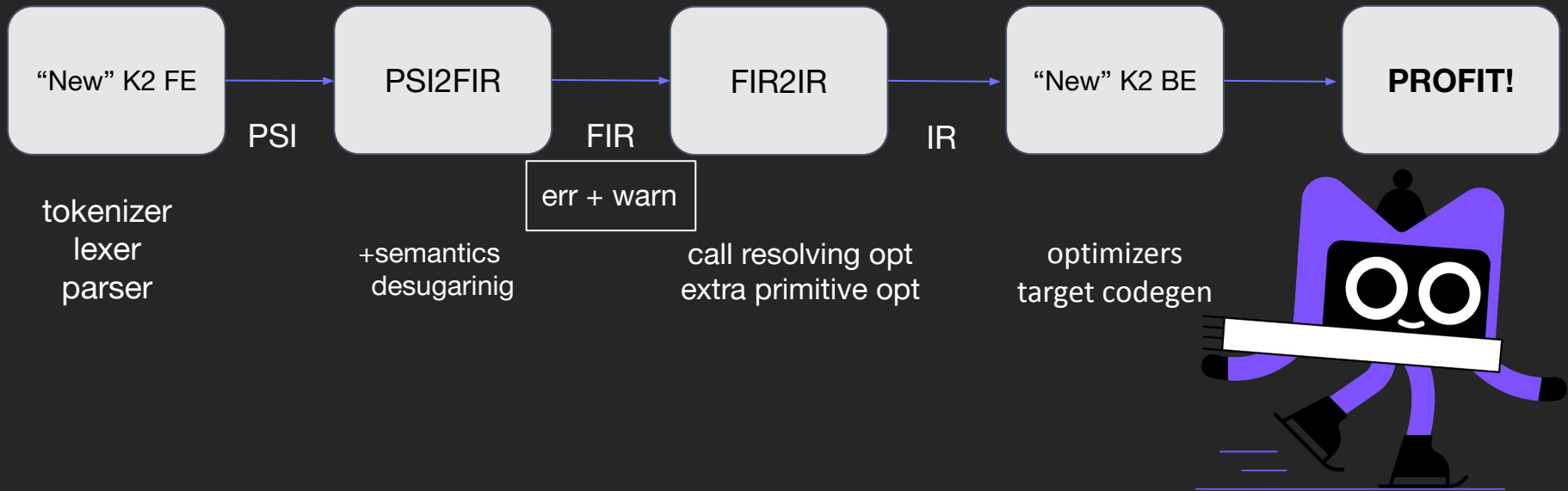


# DIFFERENCE WITH NORMAL BACKEND IRs



Whole Kotlin is just one of the frontends for LLVM (c)

# K2: our future





# COOL STORY, BRO: 1.7 + K2

```
MAP.put(  
    FIR_COMPILED_CLASS,  
    "{0} is compiled by the new Kotlin compiler frontend and cannot be loaded by the old compiler",  
    TO_STRING  
);
```

DefaultErrorMessages.java

# COOL STORY, BRO: 1.7 + K2

```
MAP.put(
```

```
    FIR_COMPILED_CLASS,
```

```
    "{0} is compiled by the new Kotlin compiler frontend and cannot be loaded by the old compiler",
```

```
    TO_STRING
```

```
);
```

DefaultErrorMessages.java

The screenshot shows a GitHub pull request page for the repository 'saveourtool / diktat'. The title of the pull request is 'Temporary removing K2 compiler Frontend #1405'. It is marked as 'Merged' and was merged by 'akuleshov7' on June 28, 2022. The pull request includes 4 commits and 5 files changed. A comment from 'akuleshov7' on June 27, 2022, lists the changes: 'removed -Xuse-k2 option'. The pull request has been approved by 'petertrr' on June 27, 2022. The navigation bar at the top shows 140 issues, 3 pull requests, 7 projects, and 1 security issue.

COOL STORY, BRO: 1.7 + K2

Directed by  
ROBERT B. WEIDE

# COOL STORY, BRO: 1.7 + K2

JetBrains / kotlin

<> Code Pull requests 162 Actions Security Insights

## Commit

**LV 2.0: drop reporting FIR\_COMPILED\_CLASS and relevant stuff**

#KT-62056 Fixed  
Related to KT-59171, KT-61951

master  
build-2.0.0-dev-4878 build-2.0.0-dev-3936

mglukhikh authored and Space Team committed 3 weeks ago Unverified

by Alexey Belkov a year ago Visible to [issue readers](#)

## new Kotlin compiler frontend and cannot be loaded by the old

tested testing the migration to K2 compiler.  
ere build with K2 cannot be loaded in projects that are built with non-K2 compiler (99.9% of user libraries).

ould be the same (and in 1.5 when you have released Kotlin IR for the JVM backend you didn't have such man-made  
ecode/LLVM IR you generate by K2 and non-K2 compilers? LLVM IR as well as Java bytecode are not so flexible

to make some pluggable functionality in compiler to work with FIR (as well as Kotlin compiler plguins work with Kotlin

Click to attach, drag files, or press ⌘V to paste an image

Activity settings

Simon Ogorodnik commented 21 Jun 2022 18:32

1. K2 Compiler is currently in Alpha state and not yet released, so we follow the same approach as with the IR Backend preview. That limitation will be removed in Beta. Currently, there can be unexpected differences in Kotlin metadata and bytecode
2. Kotlin IR plugins should be portable to K2 if it doesn't use bindingContext or descriptors. Note: @ObsoleteDescriptorBasedApi FIR has number of extension points already and intended to be pluggable. API isn't stable yet

# HOW TO TRY THE NEW COMPILER

## GRADLE

```
tasks.withType<KotlinCompile> {  
    kotlinOptions {  
        jvmTarget = "17"  
        languageVersion = "2.0"  
    }  
}
```

## PREFERRED:

```
tasks.withType<KotlinCompile> {  
    kotlinOptions.useK2 = true  
}
```

## MAVEN

```
<build>  
<plugins>  
  <plugin>  
    <groupId>org.jetbrains.kotlin</groupId>  
<artifactId>kotlin-maven-plugin</artifactId>  
    <configuration>  
      <jvmTarget>17</jvmTarget>  
      <languageVersion>2.0</languageVersion>  
    </configuration>  
  </plugin>  
</plugins>  
</build>
```

```
<configuration>  
  <args>  
    <arg>-Xuse-k2</arg>  
  </args>  
</configuration>
```

# WHAT ELSE?

## K2 COMPILER PLUGINS API

Frontend

Backend

# WHAT ELSE?

## K2 COMPILER PLUGINS

Frontend

API

Backend

Allopen

Reflekt



COMPILE-TIME РЕФЛЕКСИЯ  
В KOTLIN

Анастасия Бирилло,  
JetBrains Research



SNOWONE



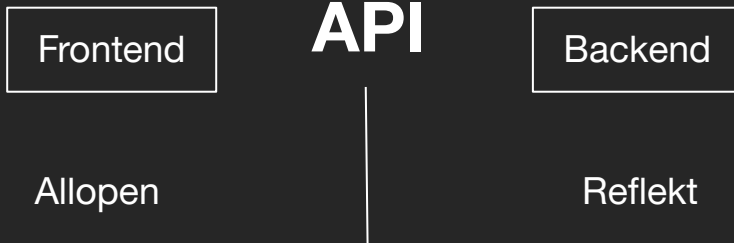
HUAWEI

ЦФТ  
ЦЕНТР  
ФИНАНСОВЫХ  
ТЕХНОЛОГИЙ

CDEK IT

# WHAT ELSE?

## K2 COMPILER PLUGINS



kotlinx.serialization



**Kotlinx.serialization:**  
ГОТОВИМ СВОЮ  
собственную библиотеку  
для сериализации



**Андрей  
Кулешов**

Huawei



# WHAT ELSE?

## K2 COMPILER PLUGINS API

Planned **new DSL** for Compiler Plugins !

**Generation** of new declarations, including  
top-level functions and properties

Transformation of modality and **visibility**

**Checkers**: for calls and expressions

# WHAT ELSE?

[compiler/fir/resolve/src/org/jetbrains/kotlin/fir/extensions/\*\*FirStatusTransformerExtension.kt\*\*](#)

```
open fun transformStatus(  
    status: FirDeclarationStatus,  
    regularClass: FirRegularClass,  
    containingClass: FirClassLikeSymbol<*>?,  
    isLocal: Boolean  
): FirDeclarationStatus {  
    // override me: status.transform(modality = OPEN)  
    return transformStatus(status, regularClass)  
}
```

# WHAT ELSE?

IDE Roadmap Item

KTIJ-23988 Created by Sergei Pecherkin 10 months ago Updated by Gavin Ray 3 months ago

Visible to issue readers

## ☆ First public release of the K2-based IntelliJ plugin

55 👍

### 📄 Summary

The creation of the new Kotlin K2 compiler is also aimed at improving the IDE experience. Performance and quality are the main focuses of the new product. By Q4'23, we are going to release the first public version of the IntelliJ plugin that will be based on the new compiler. The estimated date may change as it depends on the K2 compiler Beta release.

The general properties of this plugin include:

- Faster code completion, highlighting, and element search operations (Find Usages, Navigation, etc.).
- More stable code analysis.
- Fewer features. We also call it the "Lightweight" release, as not all IDE features are implemented. We are not trying to implement as many features as possible, but we are focused on their quality.

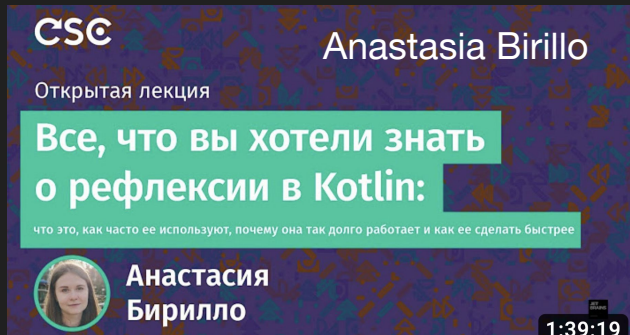
### Plans

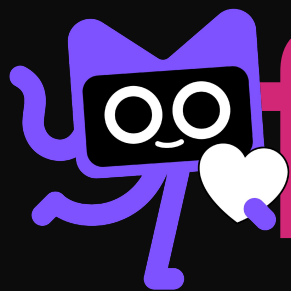
- Debugger
  - Evaluate Expression
- Refactorings
  - Change Signature
  - Copy/Paste References
  - Introduce Variable
  - Move Refactoring
- Navigation
  - Go to Type Declaration
  - Move Method Up/Down
- Code Insights
  - Auto-import
  - Breadcrumbs
  - Generate equals/hashcode, tests, and secondary constructor
  - Import Optimizer
  - Join Lines

# SOURCES & GOOD TO TAKE A LOOK



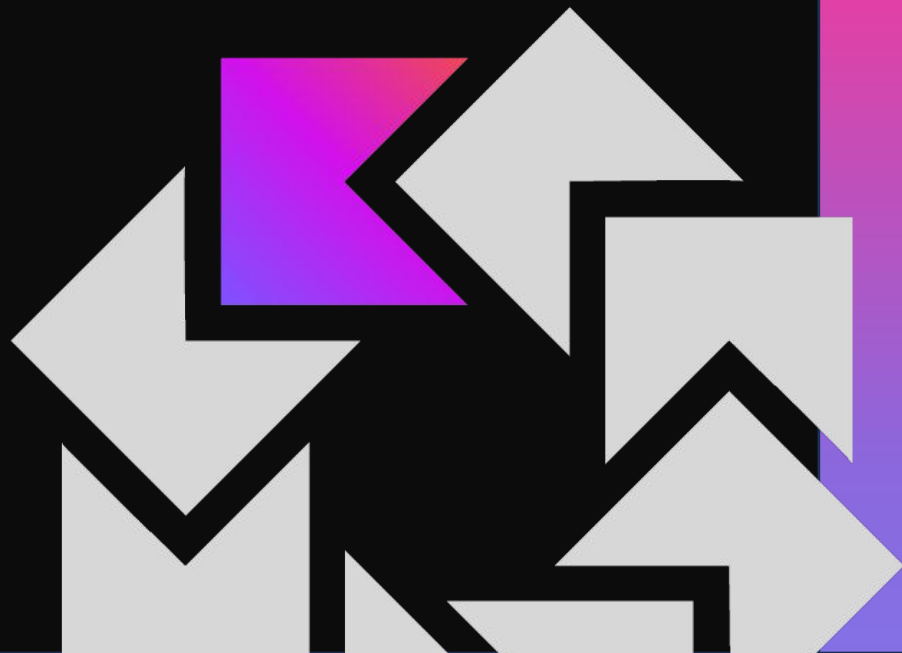
is about: to know more about updates & inner structure of compiler





# features

**LANGUAGE  
UPDATES**



# Experimental

## FEATURES



stable



experimental

Java: preview features

which are marked  
`@ExperimentalApi`  
`@SinceKotlin("x.x")`

# Experimental: is for a public review

In Kotlin 1.9, a new operator was introduced “..<>”  
How do you like it?  
I would prefer to use the the old and gold “until” more.

“The point of this “experimental” is for the user to *try it and leave the feedback* and not to wait until it'll finish to be experimental and will already be casted in stone in a new release and that's it.” © Simon Ogorodnik

So, operator ..> in 1.9 is already



stable &

DETERMINED

# Experimental

## TO TRY OUT EXPERIMENTAL FEATURES

```
graph TD; A[TO TRY OUT EXPERIMENTAL FEATURES] --> B[opt-in]; A --> C[EAP (early access preview) plugin];
```

opt-in

not really for trying features, but  
it's ok

EAP (early access preview) plugin

really for trying features

Opt-in is a mechanism for Experimental API  
to keep libraries written on kotlin up-to-date



# Opt-in usage

..and use SomeAwesomeFeature ... with

as ANNOTATION

```
@OptIn(org.foo.ExperimentalAnn::class)
```

as COMPILER ARG  
(project level)

Maven kotlin plugin:

```
<configuration>
```

```
  <arg> -opt-in=org.foo.ExperimentalAnn <arg>
```

```
</configuration>
```

Gradle:

```
kotlin {
```

```
  sourceSets {
```

```
    val commonMain by getting{
```

```
      languageSettings.optIn("org.foo.ExperimentalAnn")
```

```
    }
```

```
  }
```

```
}
```

UPDATE: from 1.7 no longer required  
special compiler arg to work

ATTENTION! Please, do not use opt-in with any experimentals in production. Just update kotlin version in a project, when release will come. Your team lead.

# Experimental: to have a quick glance

example: experimental features usage in kotlin 1.8

- use a @OptIn annotation:

```
@ExperimentalStdlibApi
@SinceKotlin(1.7)
rangeUntil()
```

```
@OptIn(ExperimentalStdlibApi::class)
fun getSimpleCount() {
    for (i in 1..<10) {
        print(i)
    }
}
```

- **OR** use a global one as a compiler arg (without annotation usage):

```
<arg>-opt-in=kotlin.ExperimentalStdlibApi</arg>
```

# Experimental: to have a quick glance

example: experimental features usage in kotlin 1.8

- use a @OptIn annotation:

```
@OptIn(ExperimentalStdLibApi::class)
fun getParameterType(title: String): ParameterType =
    ParameterType.entries.find { it.title == title }
```

```
@ExperimentalStdLibApi
@SinceKotlin(1.8)
sealed interface EnumEntries
```

- **AND** you have to use a compiler arg (without annotation usage):

```
<languageVersion>1.9</languageVersion>
```

# Experimental: to have a quick glance

example: experimental features usage in kotlin 1.8

- use a @OptIn annotation:

```
@OptIn(ExperimentalStdLibApi::class)
fun getParameterType(title: String): ParameterType =
    ParameterType.entries.find { it.title == title }
```

```
@ExperimentalStdLibApi
@SinceKotlin(1.8)
sealed interface EnumEntries
```

- **AND** use a compiler arg (without annotation usage):

```
<arg>-XXLanguage:+EnumEntries</arg>
```

you have to enable language version 1.9  
because of this flag

which is enabled in 1.9 language version, actually

# Experimental: to have a quick glance

For the features that have @SinceKotlin with the same kotlin version as yours:

SO you can USE the

```
-language-version x.x
```

but you must know  
that it's enable the corresponding  
compiler version and it will affect you

OR

read the corresponding docs and enable a flag

```
-FLAG:+FEATURENAME
```

called 'KEYS'

# Experimental: to have a quick glance

**-language-version x.x**

GRADLE:

```
tasks.withType<KotlinCompile> {
    kotlinOptions {
        jvmTarget = "17"
        languageVersion = "1.9"
    }
}
```

MAVEN:

```
<groupId>org.jetbrains.kotlin</groupId>
<artifactId>kotlin-maven-plugin</artifactId>
<configuration>
    <jvmTarget>17</jvmTarget>
    <languageVersion>1.9</languageVersion>
</configuration>
```

**-FLAG:+FEATURENAME**

GRADLE:

```
tasks.withType<KotlinCompile> {
    kotlinOptions {
        jvmTarget = "17"
        freeCompilerArgs =
            listOf("-XXLanguage:+InlineClasses")
    }
}
```

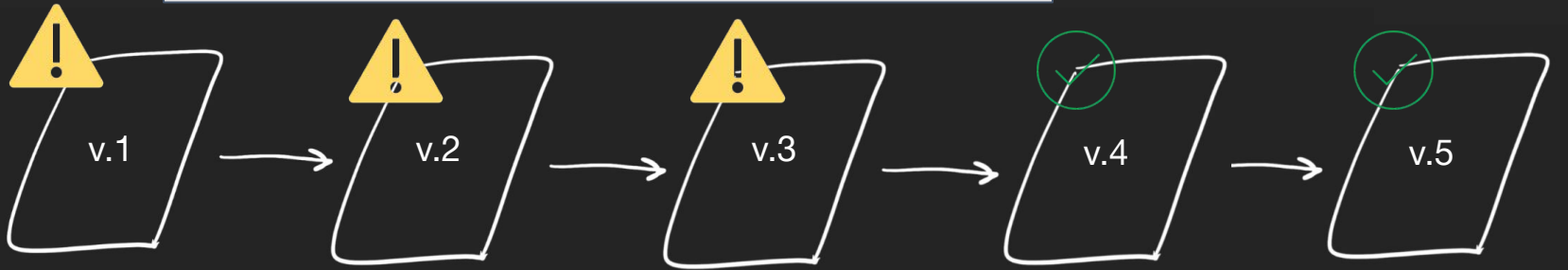
MAVEN:

```
<groupId>org.jetbrains.kotlin</groupId>
<artifactId>kotlin-maven-plugin</artifactId>
<configuration>
    <jvmTarget>17</jvmTarget>
    <args>
        <arg>-XXLanguage:+InlineClasses</arg>
    </args>
</configuration>
```

# Experimental life cycle:

~one year

`release!`



marked  
@Experimental  
@Since(1.0)

still  
@Experimental

still  
@Experimental

marked  
@WasExperimental  
@Since (4.0)

removed  
@WasExperimental

here u need and opt-in  
and a flag  
-FLAG:+FEATURENAME

here the flag will be  
enabled, so you need  
only opt-in

and the flag was  
removed

marked @Deprecated and then removed

# New features: KEEP in touch

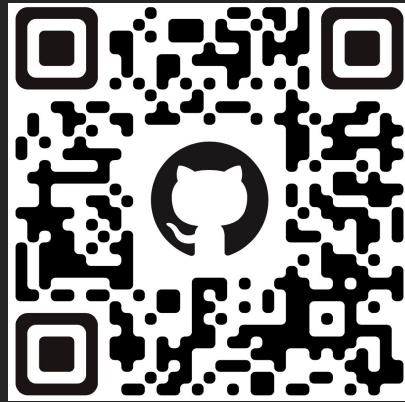
1 step:

Kotlin releases - learn about new ones



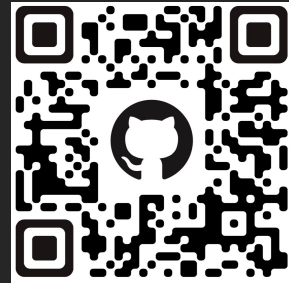
2 step:

KEEP (kotlin evolution & enhancement process) - learn for what & how to use



3 step:

left a feedback in YouTrack or KEEP issues or telegram chat





# New features: participate

Early Access Preview - last  
kotlin updates which prepared to be released



Official Kotlin Discussions



# 1.7



- Data objects
- K2 alpha
- New operator
- ...

# 1.9

- kotlin.time updates
- K2 beta
- Stable last features
- ...

# 1.8

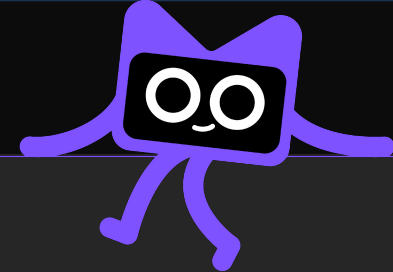
- Inline class updates
- K2 beta
- stdLib updates
- ...

# 2.0

- K2 stable
- ...



# Updates



## 1.7.0

- Using builders with builder type inference
  - Implementation by delegation to an inlined value of an inline class
  - Underscore operators for type arguments
  - Java optional extensions added
- ...

## 1.7.20

- Range Until operator "`..<<`"
  - Data objects
  - Generic inline classes
- ...

# Using custom builders with builder type inference

```
buildList(), buildMap(), .. // all collection builders  
                             // have become stable in 1.6
```

```
fun add(value: String) {  
    val result = buildList { this: MutableList<String>  
        add(value)  
    }  
}
```

result of Builder Type Inference work



```
fun add(value: String) {  
    val result : List<String> = buildList {  
        add(value)  
    }  
}
```

without its work, you have to specify the type explicitly



Since Kotlin (**1.7**)

# Using custom builders with builder type inference

```
class ItemHolder<T> {  
    ...  
}
```

Until version 1.7, when you're writing your own builder, you have to enable the opt-in or set a compiler flag: `-Xenable-builder-inference`

```
@OptIn(ExperimentalTypeInference::class)  
fun <T> itemHolderBuilder(@BuilderInference builder: ItemHolder<T>.(()) → Unit):  
ItemHolder<T> = ItemHolder<T>().apply(builder)
```

```
fun test(s: String) {  
    val itemHolder = itemHolderBuilder { this: ItemHolder<String>  
        addItem(s)  
    }  
}
```

Since Kotlin (**1.7**)

# Using custom builders with builder type inference

As for now, the compiler option `-Xenable-builder-inference` is enabled by default in 1.7, so you don't have to do anything to make it's worked

```
fun <T> itemHolderBuilder(builder: ItemHolder<T>.(()) → Unit): ItemHolder<T> =  
    ItemHolder<T>().apply(builder)
```

```
fun test(s: String) {  
    val itemHolder = itemHolderBuilder { this: ItemHolder<String>  
        addItem(s)  
    }  
}
```

Since Kotlin (**1.7**)

# Underscore operators for type arguments

Use it to **automatically infer a type** of the argument when other type is explicitly specified:

```
fun <K, T> foo(x: (K) → T): (K) → T = x

fun main() {
    val x = foo<Int, _> { it.toFloat() } (4) // Float
    val x1 = foo<Int, _> { it.toDouble() } (4) // Double
}
```

not to type foo<Int,Double> or foo<Int,Float>

Since Kotlin (**1.7**)

# Data objects



stable in 1.9

`data object` SomeObject

Intended as  
extension for object

- is an object declaration

//Characteristics:

from `object` keyword:

- creates a singleton class

from `data` keyword:

- `equals()`, `hashCode()`, `toString()`

//Limitations:

- no `copy()` and `componentN()` usage

Since Kotlin (**1.7**)



# Data objects



stable in 1.9

```
sealed interface ReadResult
data class Text(val text: String) : ReadResult

object EndOfFile : ReadResult {
    override fun toString() = "EndOfFile" // ??
}

fun main() {
    println(Text("some text")) // Text: "some text"
    println(EndOfFile) // EndOfFile
}
```

Since Kotlin (**1.7**)

# Data objects



stable in 1.9

```
sealed interface ReadResult
data class Text(val text: String) : ReadResult
data object EndOfFile : ReadResult
```

```
fun main() {
    println(EndOfFile) // EndOfFile
}
```

With 'data' keyword we have autogenerated  
`equals()`, `hashCode()` and `toString()`

Since Kotlin (**1.7**)

# Extensions for Java Optional in stdLib



stable in 1.8

kotlin-stdlib/kotlin.jvm.optionals

getOrNull()

getOrDefault()

getOrNull()

...



related  
YouTrack ticket



Spring Data JPA How to use Kotlin nulls instead of Optional

```
Optional<T> findById(ID id);  
val result = findById(id).orElse(null)
```

Since Kotlin (**1.7**)

# Extensions for Java Optional in stdLib



stable in 1.8

kotlin-stdlib/kotlin.jvm.optionals

`getOrNull()`

`getOrDefault()`

`getOrNull()`

...



related  
YouTrack ticket

```
@SinceKotlin("1.8")  
@WasExperimental(ExperimentalStdlibApi::class)  
public fun <T : Any> Optional<T>.getOrNull(): T? = orElse(null)
```

SinceKotlin(**1.7**)

# 1.7

- Data objects
- K2 alpha
- New operator
- ...

# 1.9



- kotlin.time updates
- K2 beta
- Stable last features
- ...

# 1.8

- Inline class updates
- K2 beta
- stdLib updates
- ...



# 2.0

- K2 stable
- ...

# Updates



## 1.8.0

- comparable and substructural Time marks
- extensions for `java.nio.file.path`

...

## 1.8.20

- secondary constructors for inline classes
- `entries` property for enums

...

# Inline class updates

**SINCE 1.2.30 inline classes was  
@Experimental ⚠**

```
//old version  
inline class Duration(val value: Long)
```

---

**SINCE 1.5 THEY BECAME STABLE ✓  
BUT WAS MODIFIED:**

```
inline function  
inline property  
@JvmInline  
inline value class
```



KEEP Design Notes on  
Kotlin Value Classes

# Inline class updates

```
@JvmInline  
value class Duration(val value: String)
```

underlying property

Intended as a  
wrapper over value

- is a value-based class

//Characteristics:

- `init{}`, `first`, `second` constructors
- you can define member functions and properties

...

//Limitations:

- only one immutable underlying property is allowed (for now)
- ...



# Inline class updates

OK, but..

here is the *value*

```
@JvmInline  
value class Name(val s: String)
```



and *inline* is now here!

# Inline class updates

[Java Valhalla Project](#)

## VALUE CLASSES

the main idea - performance optimization

```
//instantiate a class  
val period = new Hours(24)  
  
//& it's compiled like a primitive  
int period = 24;
```

*“Codes like a class, works like an int!”*



Project Valhalla docs



Actual JEP

# Inline class updates

## VALUE CLASSES

```
PointVal pv1 = PointVal(1, 2);  
PointVal pv2 = PointVal(1, 2);
```

```
(pv1 == pv2); // false
```

*usual identity class*

```
(pv1 == pv2); // true
```

*value class*

```
value class PointVal (Java)
```

# Inline class updates

**KOTLIN VALUE CLASS HAS NO IDENTITY SAME AS VALHALLA'S VALUE**

**Example:** two Kotlin (inline) value classes comparison

```
@JvmInline
value class SomeVal(val s: Int)
```

```
▶ fun main() {
    ... val some = SomeVal(s: 4)
    ... val some2 = SomeVal(s: 4)
    ... println(some == some2)
}
```

Identity equality for arguments of types SomeVal and SomeVal is forbidden

# Inline class updates

## Valhalla inline classes vs Kotlin inline classes:

**Inline classes in Valhalla** named like that, because instead of the class *its content is substituted*.

**Kotlin inline classes** as soon as they appeared were realized on the Valhalla inline classes concept.

Then Valhalla simply changed the name to value classes, but regardless of them Kotlin also has changed the semantics.



# Inline class updates

Optimize away boxes when possible

@\_\_TBD\_\_

```
val class Color(val rgb: Int)
```

No stable identity

Kotlin 1.4 Online Event

A Look  
Into the Future  
Roman  
Elizarov



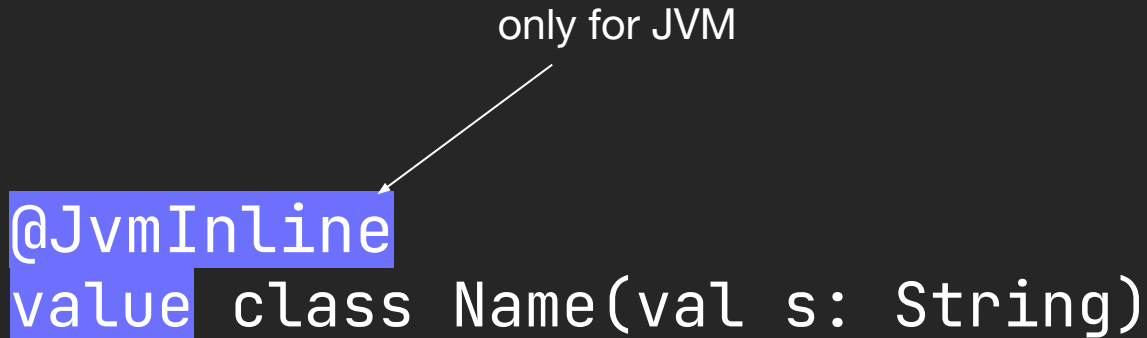
We want interoperability  
with future Valhalla  
features in Java, when it's  
JEP will be accepted

# Value class updates

So, that's why we have this:

only for JVM

```
@JvmInline  
value class Name(val s: String)
```

A diagram with two arrows. One arrow points from the text "only for JVM" to the @JvmInline annotation. The other arrow points from the text "the same as Valhalla value class, but with ONE underlying value only" to the value class definition.

the same as Valhalla value class, but with **ONE underlying value only**

# Plans & expectations

## Plan: multi-field value (inline) class

Java value classes are planned as a usual classes with keyword 'value'

```
value class PointVal {
    int x, y;
    PointVal(int x, int y){
        this.x = x;
        this.y = y;
    }
}
```

Kotlin team already has a realization of multi-field value classes, available by key:

```
@JvmInline
value class PointVal(val x: Int, val y: Int)
```

## -Xvalue-classes



Multi-field value classes  
KEEP proposal



# Inline class updates



stable in 1.9

## secondary constructors with bodies in inline classes

```
@JvmInline
value class Name(private val name: String) {

    //since kotlin 1.4.30
    init {
        check(name.isNotBlank()) {
            "shouldn't be empty"
        }
    }

    //since kotlin 1.8.20
    constructor(firstName: String, secondName: String) : this("$firstName + $secondName") {
        println(name)
    }
}
```

Since Kotlin (**1.8**)

# Inline class updates

Implementation by delegation to inlined value of inlined class is allowed with interfaces:

```
interface MyInterface {  
    fun bar()  
    fun foo() = "foo"  
}
```

```
@JvmInline  
value class MyInterfaceWrapper(val value: MyInterface) : MyInterface by value
```

```
fun main() {  
    val my = MyInterfaceWrapper(object : MyInterface {  
        override fun bar() {  
            // body  
        }  
    })  
}
```

Since Kotlin (**1.7**)

# Inline class updates

## Generic inline classes

```
value class MyWrapper<T>(val value: T)
```

```
sealed class Games  
data class BoardGame(val name: String, val playersQty: Int) : Games()  
data class Cards(val cardsInDeck: Int) : Games()
```

```
sealed class Work
```

```
@JvmInline  
value class MyDailyOccupation<T>(val occupation: T)  
  
fun getTodayOccupation(plan: MyDailyOccupation<Games>) {  
    when (val todo = plan.occupation) {  
        is BoardGame → println("BoardGame ${todo.name}! Look for friends to play: ${todo.playersQty}")  
        is Cards → println("Cards! Take along the ${todo.cardsInDeck} deck")  
    }  
}  
  
fun main() {  
    getTodayOccupation(MyDailyOccupation(BoardGame("Imaginarium", 4)))  
    getTodayOccupation(MyDailyOccupation(Work))  
}
```

Since Kotlin (1.7)

# SOURCES & GOOD TO TAKE A LOOK

Release Kotlin 1.5.0

Svetlana Isakova

From Inline to Value Classes



A good explanation on work of inline classes



Let's look at the Bytecode!

Sebastian Aigner 8:54

@JvmInline  
sealed value class



# 1.7

- Data objects
- K2 alpha
- New operator
- ...

# 1.9

- kotlin.time updates
- K2 beta
- Stable last features
- ...



# 1.8

- Inline class updates
- K2 beta
- stdLib updates
- ...

# 2.0

- K2 stable
- ...

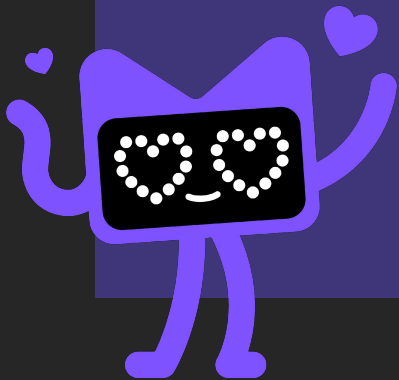


# Updates

now 1.9.10

## 1.9.0

- Full time API now is stable
- New gradle settings
- Stable ..<



## 1.9.20

- KAPT with K2
- Kotlin/WASM with K2
- GC performance
- Xcode 15 support
- Autocreation of source sets for multiplatform

Since Kotlin ( **1.9** )

# Stable `..` operator for open-ended ranges



stable in 1.9

```
number in 1..
```

Intended as  
`rangeUntil` operator

```
/**  
 * Creates an open-ended range from this [Double] value to the specified [that]  
 * value.  
 *  
 * Numbers are compared with the ends of this range according to IEEE-754.  
 */  
@SinceKotlin("1.9")  
@WasExperimental(ExperimentalStdLibApi::class)  
public operator fun Double.rangeUntil
```

```
for (number in 2..
```

SinceKotlin(**1.7**)

# Apple targets with gradle

Kotlin 1.9.0 and earlier (a standard setup)

```
kotlin {
    androidTarget()
    iosArm64()
    iosSimulatorArm64()

    sourceSets {
        val commonMain by getting

        val iosMain by creating {
            dependsOn(commonMain)
        }

        val iosArm64Main by getting {
            dependsOn(iosMain)
        }

        val iosSimulatorArm64Main by getting {
            dependsOn(iosMain)
        }
    }
}
```

Kotlin 1.9.20-RC

```
kotlin {
    androidTarget()
    iosArm64()
    iosSimulatorArm64()

    // The iosMain source set is created by default
}
```



# Updates

now 1.9.10

## 1.9.0

- Full time API now is stable
- New gradle settings
- Stable ..<

## 1.9.20

- KAPT with K2
- Kotlin/WASM with K2
- GC performance
- Xcode 15 support
- Autocreation of source sets for multiplatform

# Updates

now 1.9.10

## 1.9.0

- Full time API now is stable
- New gradle settings
- Stable

## 1.9.0

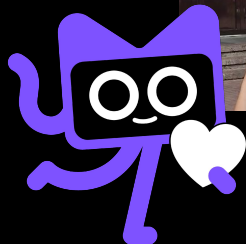
- Autocreation of source sets for multiplatform

**K2 COMPILER**

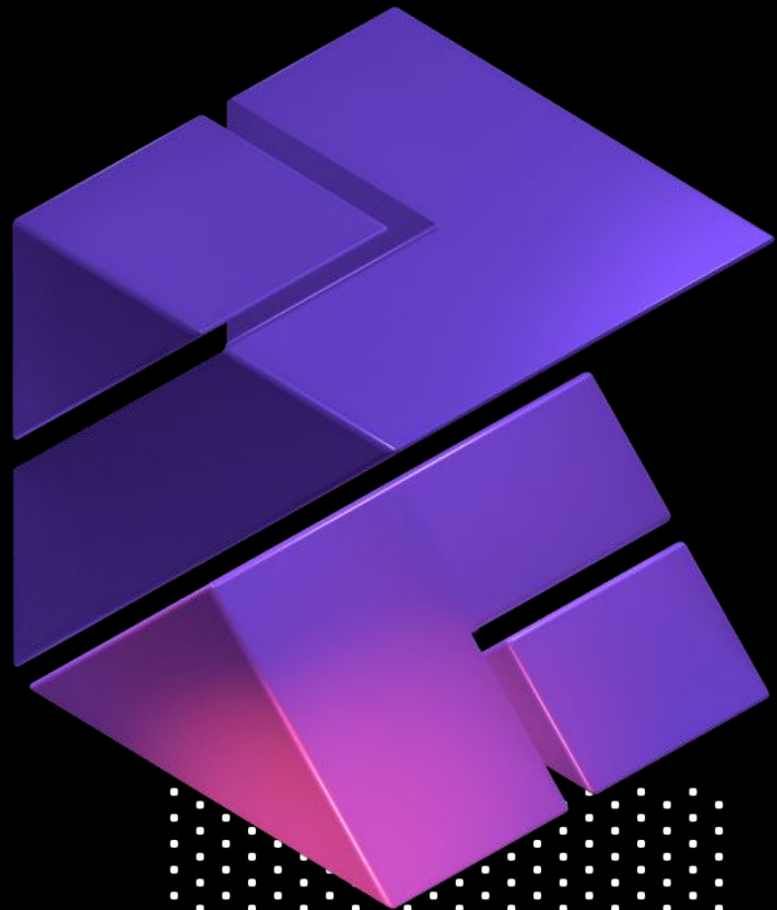
# IN THE MEANTIME

# BOTLIN

Github: anyonepaw  
Telegram: @anyonepaw



Github: akuleshov7  
Telegram: @akuleshov7



# SOURCES & GOOD TO TAKE A LOOK

What's  
New in  
Kotlin 1.7.0

Anton  
Arhipov



23:48

Kotlin 1.4 Online Event

A Look  
Into the Future

Roman  
Elizarov



Kotlin roadmap

YouTrack board



roadmap update:  
in December, 2023

So many  
features!

UPDATE

Kotlin 1.7.20

Sebastian  
Aigner



11:08

# ALL NEW KOTLIN COMPONENTS STATUSES TABLE



Component	Status	Status since version	Comment
Kotlin/JVM	Stable	1.0	
Kotlin K2 (JVM)	Alpha	1.7	
kotlin-stdlib (JVM)	Stable	1.0	
Coroutines	Stable	1.3	
kotlin-reflect (JVM)	Beta	1.0	
Kotlin/JS (Classic back-end)	Stable	1.3	Deprecated from 1.8.0, read the <a href="#">IR migration guide</a>
Kotlin/JVM (IR-based)	Stable	1.5	
Kotlin/JS (IR-based)	Stable	1.8	
Kotlin/Native Runtime	Beta	1.3	
Kotlin/Native memory manager	Beta	1.7.20	