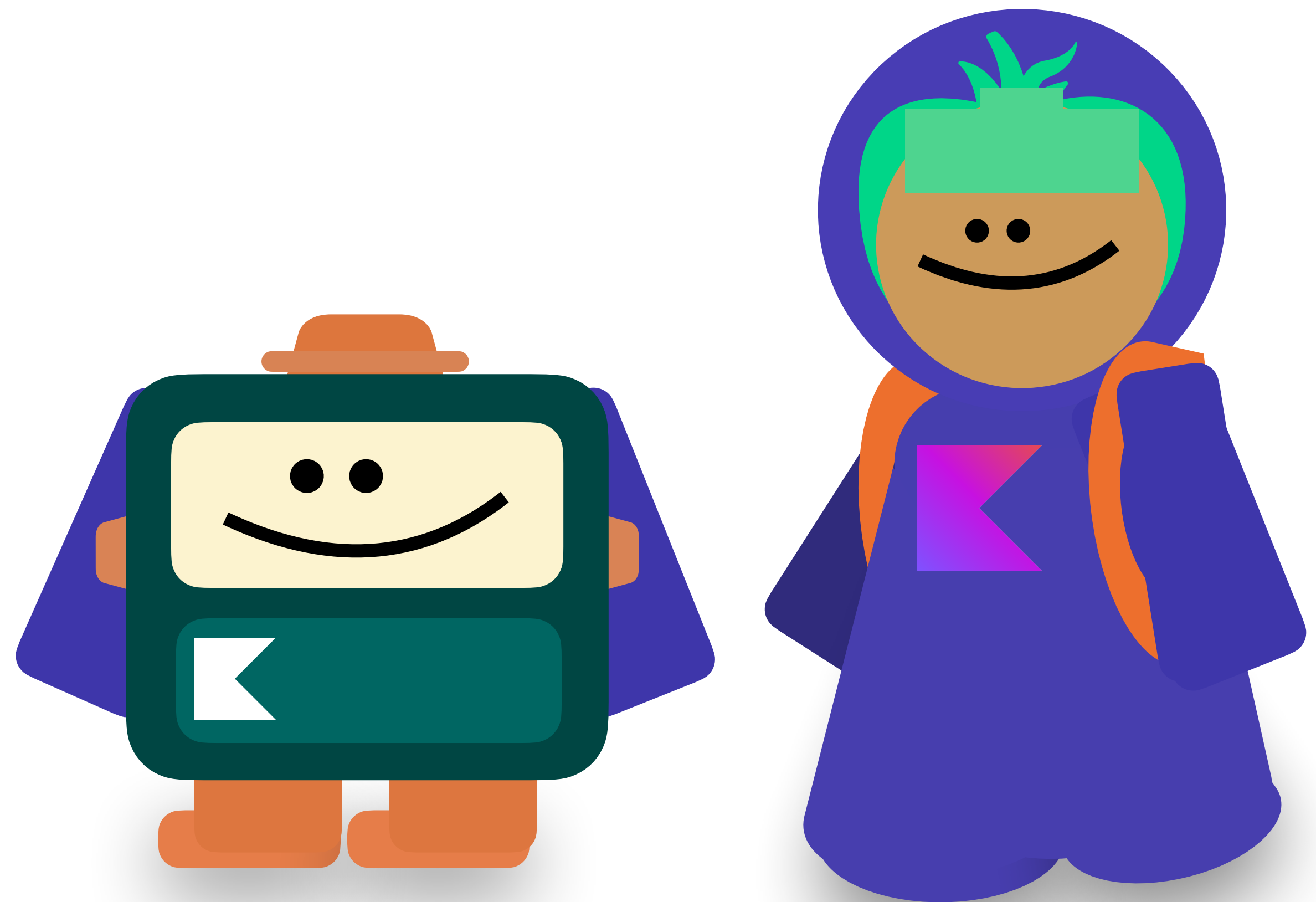


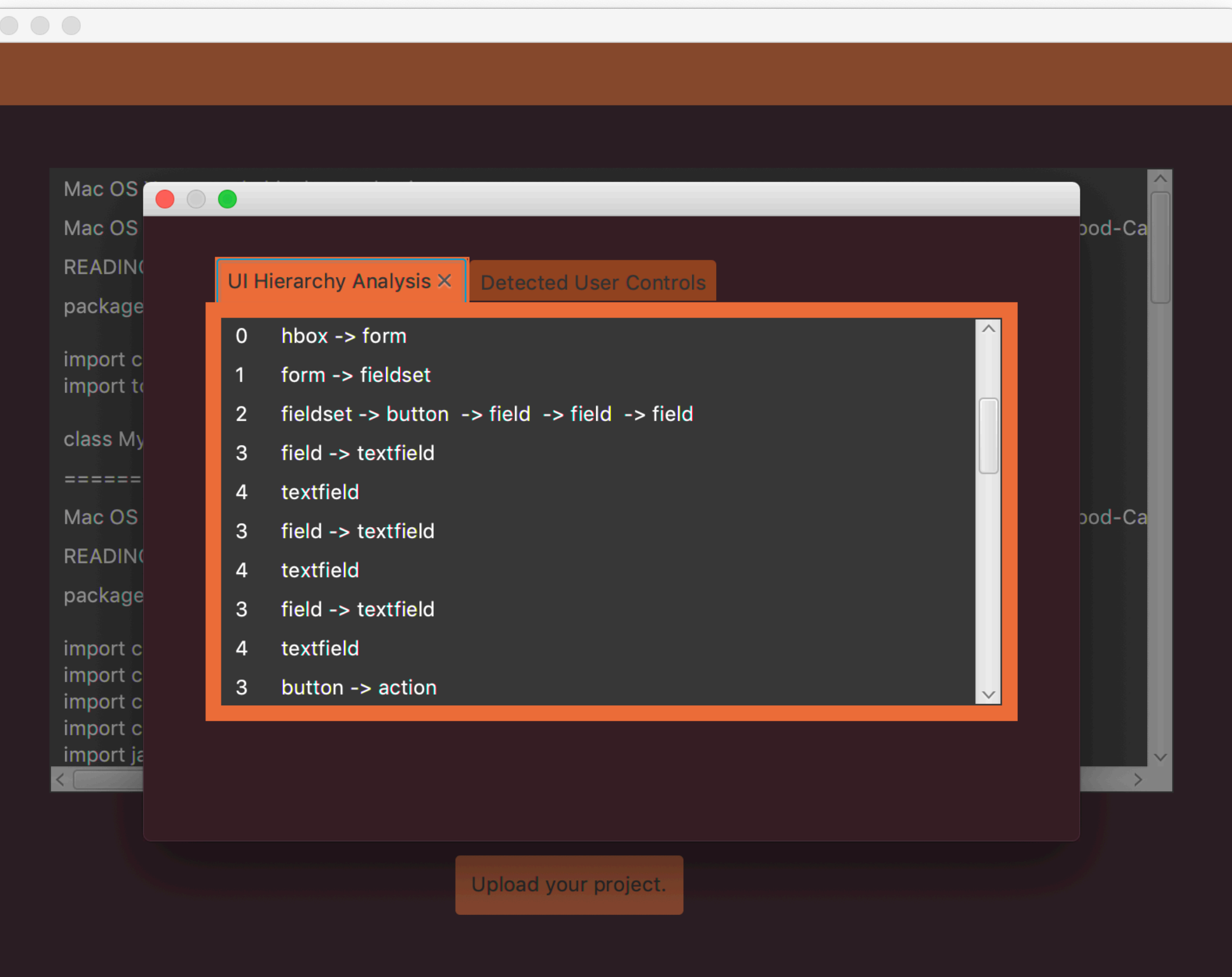
Crash Course on the Kotlin Compiler

@mvndy_hd
KotlinConf'23
Amsterdam

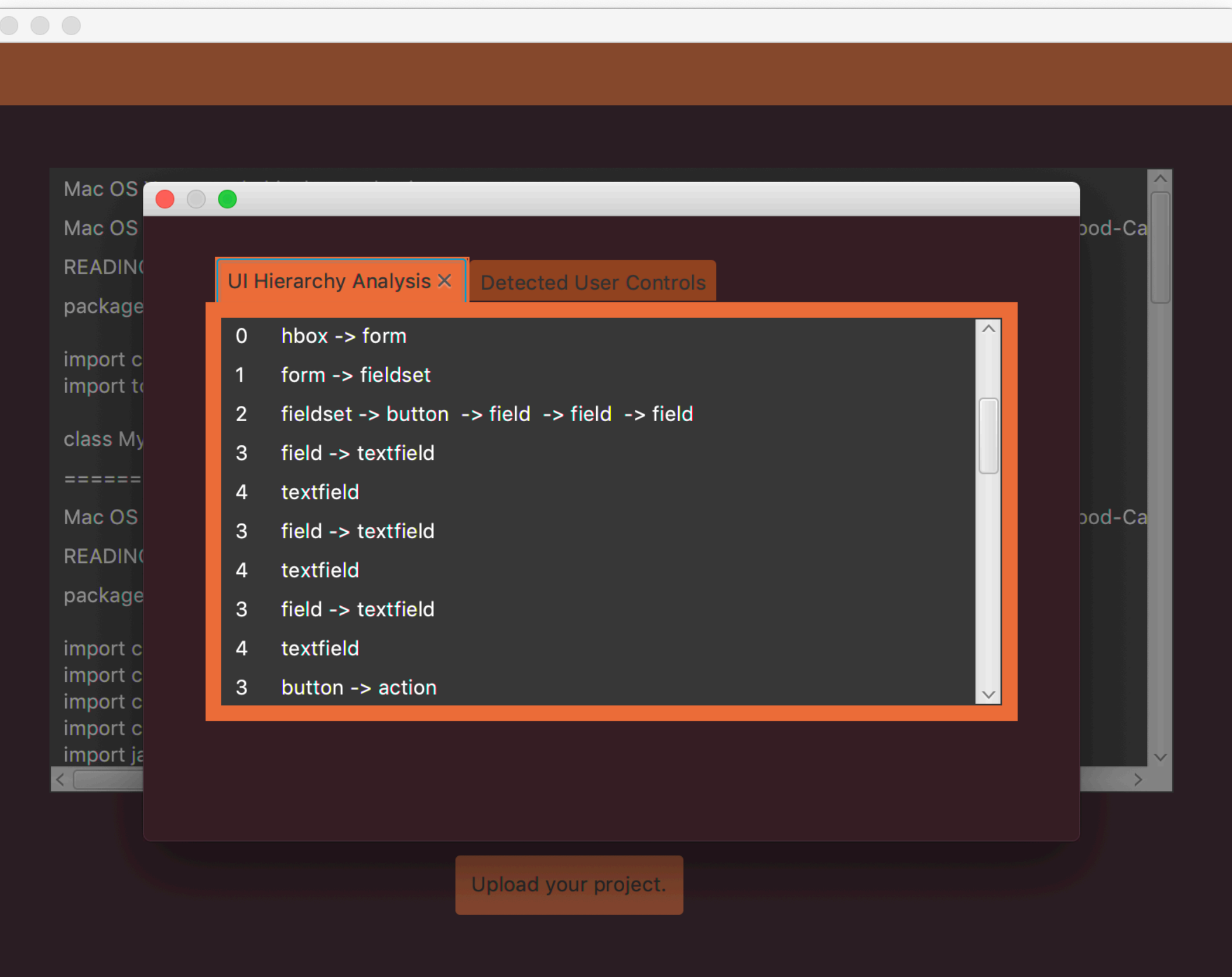


So you want to get to know the Kotlin compiler?





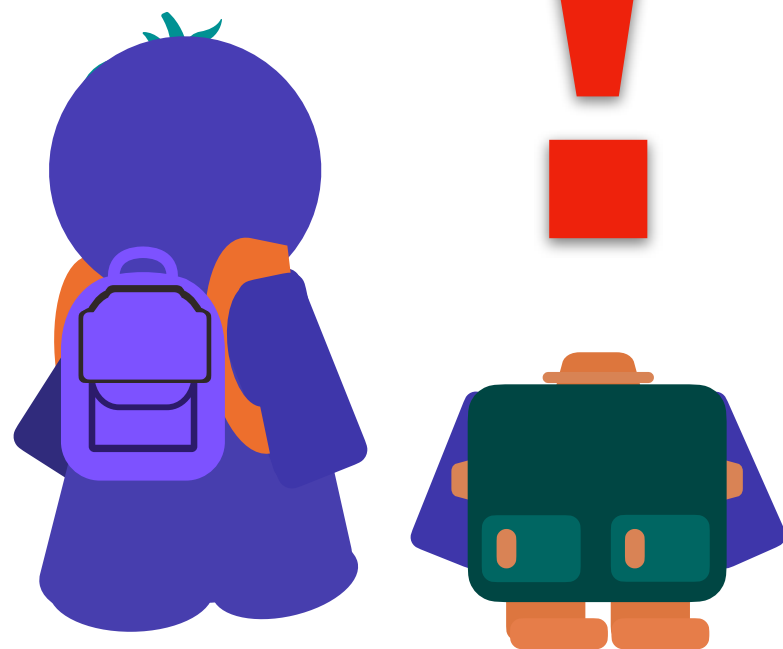
← → ↺ 🏠	github.com/pulls?page=2&q=is%3Apr+author%3Aahinchman1+archived%3Afalse+is%3Aclosed
🔴🔴	arrow-kt/arrow-meta Check to see if multiple dependencies are being used for a single test #347 by ahinchman1 was closed on Dec 17, 2019 • Draft
🔗	arrow-kt/arrow-meta Quote and template minor cleanups AST-scope #334 by ahinchman1 was merged on Dec 17, 2019 • Approved
🔗	arrow-kt/arrow-meta Fix: IfExpression testing and added validation AST-scope documentation #333 by ahinchman1 was merged on Dec 17, 2019 • Approved 4 tasks done
🔗	arrow-kt/arrow-meta TypeAlias Quote and Template Testing Validation AST-scope documentation #327 by ahinchman1 was merged on Dec 12, 2019 • Approved 4 tasks done
🔗	arrow-kt/arrow-meta ThisExpression quote-and-template scope AST-scope documentation #325 by ahinchman1 was merged on Dec 12, 2019 • Approved 4 tasks done
🔴🔴	arrow-kt/arrow-meta ThisExpression quote and template AST-scope documentation #324 by ahinchman1 was closed on Dec 11, 2019 • Approved 4 tasks done
🔗	arrow-kt/arrow-meta ContinueExpression quote and template scope AST-scope documentation #313 by ahinchman1 was merged on Dec 9, 2019 • Approved 3 tasks done
🔗	arrow-kt/arrow-meta BreakExpression quote-and-template scope AST-scope documentation #312 by ahinchman1 was merged on Dec 9, 2019 • Approved 3 tasks done
🔗	arrow-kt/arrow-meta FunctionLiteral Quote and Template Scope Testing Validation AST-scope documentation #311 by ahinchman1 was merged on Dec 12, 2019 • Approved 4 tasks done
🔗	arrow-kt/arrow-meta aAdded DotQualifiedExpression Scope and template plus testing validation AST-scope #298 by ahinchman1 was merged on Dec 4, 2019 • Approved
🔗	arrow-kt/arrow-meta Scope structure refactor & Arrow Meta Quotes Table for declarations & dokka upgrade AST-scope documentation high-priority #202 by ahinchman1 was merged on Nov 28, 2019 • Approved 22 of 24 tasks
🔗	arrow-kt/arrow-meta File quote and template scope AST-scope #184 by ahinchman1 was merged on Nov 21, 2019 • Approved 2 of 3 tasks
🔗	arrow-kt/arrow-meta Value argument quote and template scope AST-scope #183 by ahinchman1 was merged on Nov 20, 2019 • Approved 2 of 3 tasks



← → ↺ 🏠	github.com/pulls?page=2&q=is%3Apr+author%3Aahinchman1+archived%3Afalse+is%3Aclosed
🔴	arrow-kt/arrow-meta Check to see if multiple dependencies are being used for a single test #347 by ahinchman1 was closed on Dec 17, 2019 • Draft
🔗	arrow-kt/arrow-meta Quote and template minor cleanups AST-scope #334 by ahinchman1 was merged on Dec 17, 2019 • Approved
🔗	arrow-kt/arrow-meta Fix: IfExpression testing and added validation AST-scope documentation #333 by ahinchman1 was merged on Dec 17, 2019 • Approved 4 tasks done
🔗	arrow-kt/arrow-meta TypeAlias Quote and Template Testing Validation AST-scope documentation #327 by ahinchman1 was merged on Dec 12, 2019 • Approved 4 tasks done
🔗	arrow-kt/arrow-meta ThisExpression quote-and-template scope AST-scope documentation #325 by ahinchman1 was merged on Dec 12, 2019 • Approved 4 tasks done
🔴	arrow-kt/arrow-meta ThisExpression quote and template AST-scope documentation #324 by ahinchman1 was closed on Dec 11, 2019 • Approved 4 tasks done
🔗	arrow-kt/arrow-meta ContinueExpression quote and template scope AST-scope documentation #313 by ahinchman1 was merged on Dec 9, 2019 • Approved 3 tasks done
🔗	arrow-kt/arrow-meta BreakExpression quote-and-template scope AST-scope documentation #312 by ahinchman1 was merged on Dec 9, 2019 • Approved 3 tasks done
🔗	arrow-kt/arrow-meta FunctionLiteral Quote and Template Scope Testing Validation AST-scope documentation #311 by ahinchman1 was merged on Dec 12, 2019 • Approved 4 tasks done
🔗	arrow-kt/arrow-meta aAdded DotQualifiedExpression Scope and template plus testing validation AST-scope #298 by ahinchman1 was merged on Dec 4, 2019 • Approved
🔗	arrow-kt/arrow-meta Scope structure refactor & Arrow Meta Quotes Table for declarations & dokka upgrade AST-scope documentation high-priority #202 by ahinchman1 was merged on Nov 28, 2019 • Approved 22 of 24 tasks
🔗	arrow-kt/arrow-meta File quote and template scope AST-scope #184 by ahinchman1 was merged on Nov 21, 2019 • Approved 2 of 3 tasks
🔗	arrow-kt/arrow-meta Value argument quote and template scope AST-scope #183 by ahinchman1 was merged on Nov 20, 2019 • Approved 2 of 3 tasks

**Navigating the
compiler is hard**

**Documentation
feels like it's in
bits and pieces**



I put together a gear kit

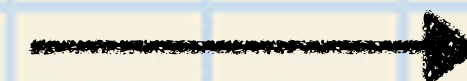
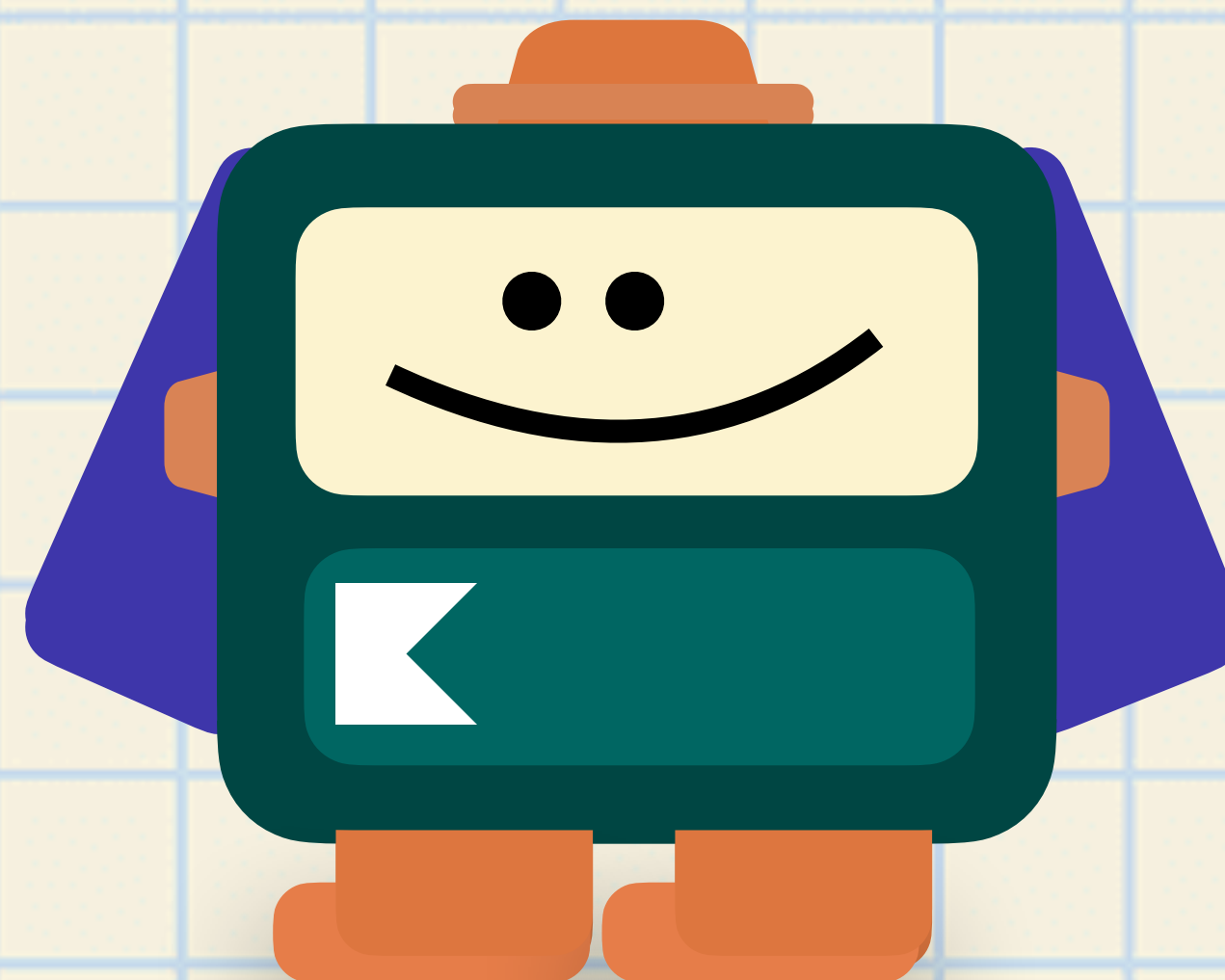
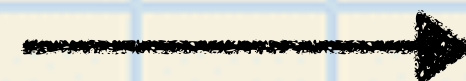


Gear kit

**Kotlin
compiler map**

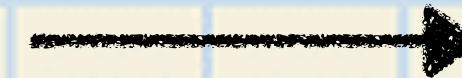
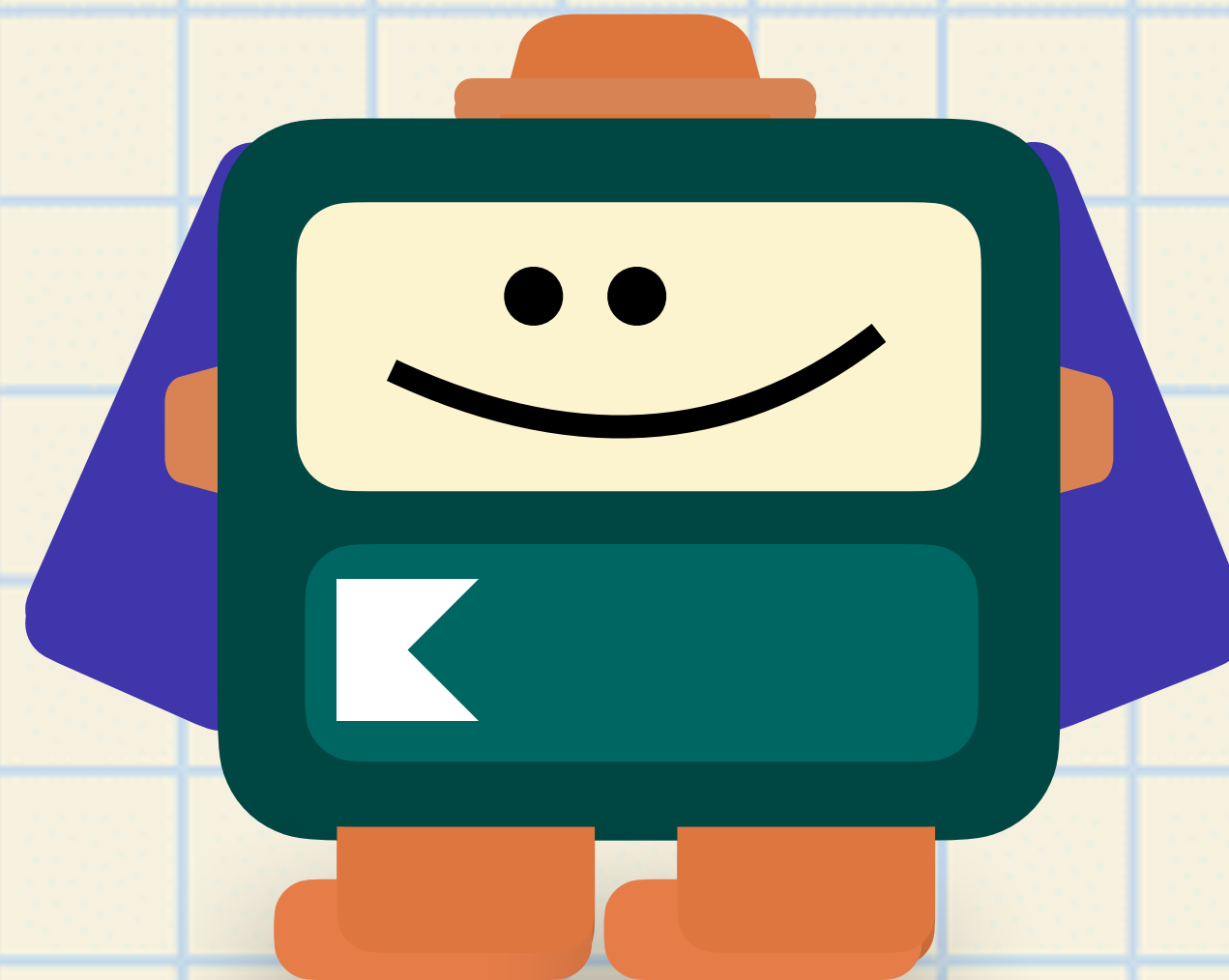
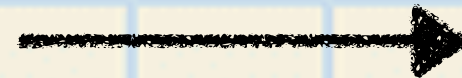
**Reverse
engineering
popular plugins**

**Exchanging
research notes**

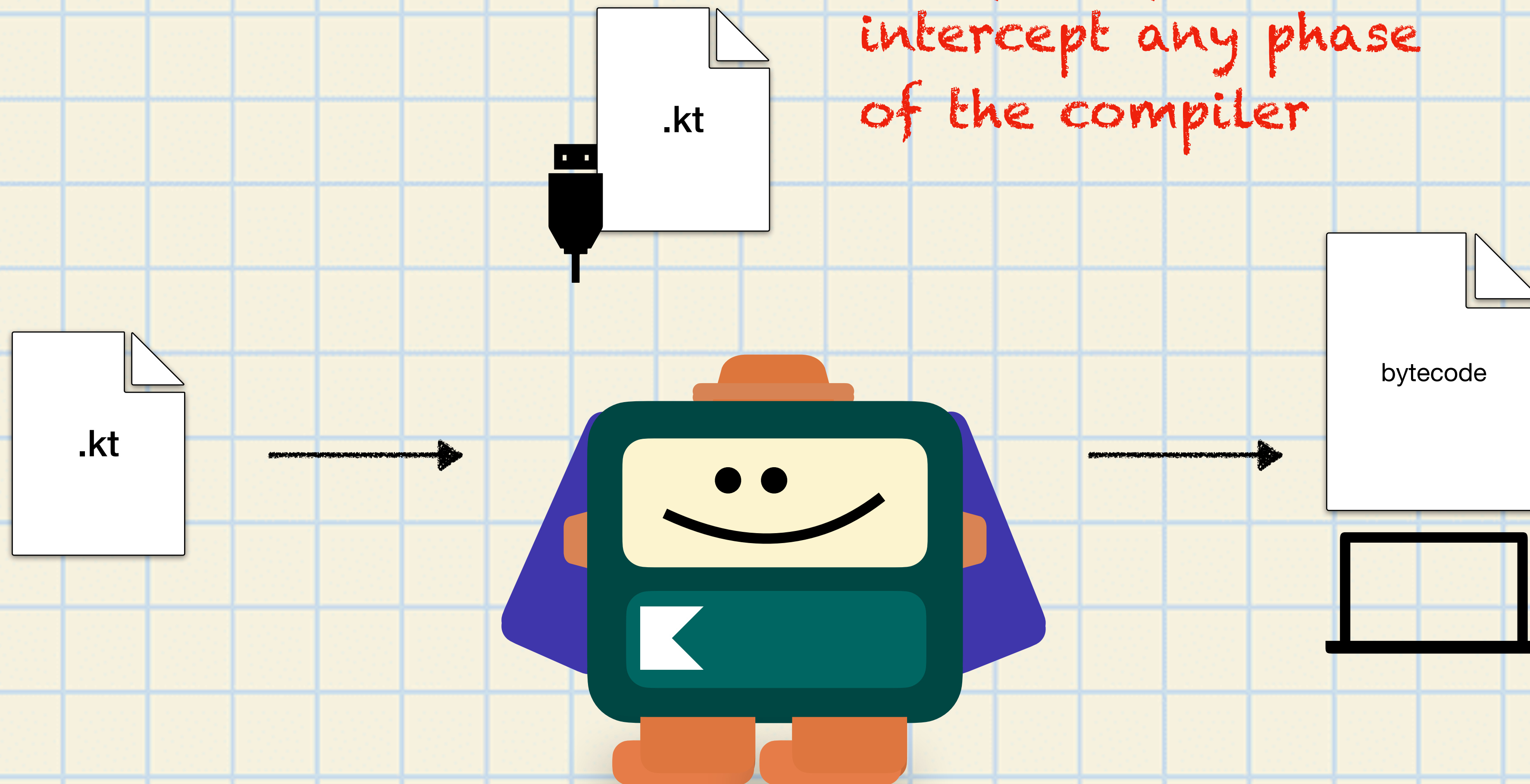


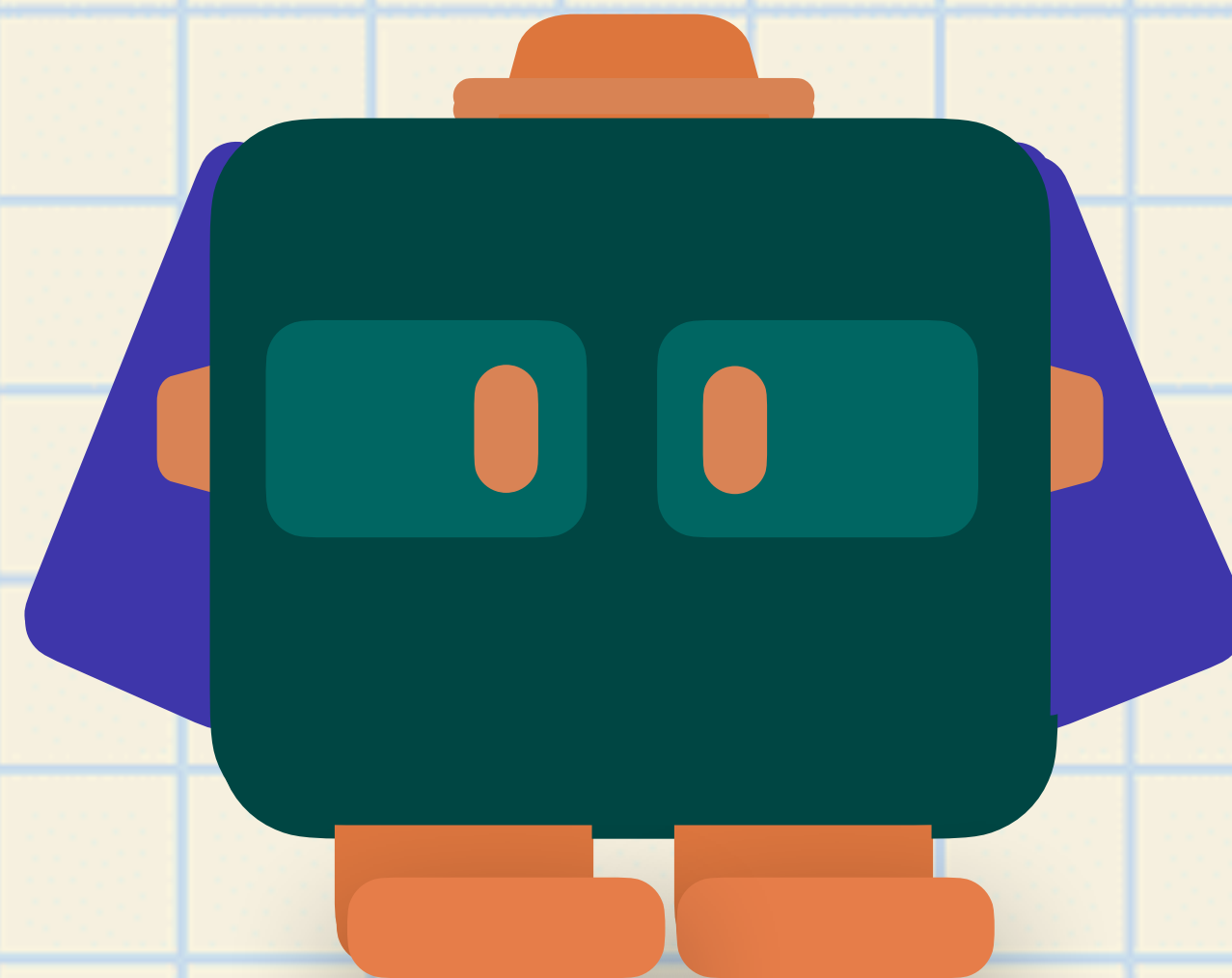
Compiling - changes
data format

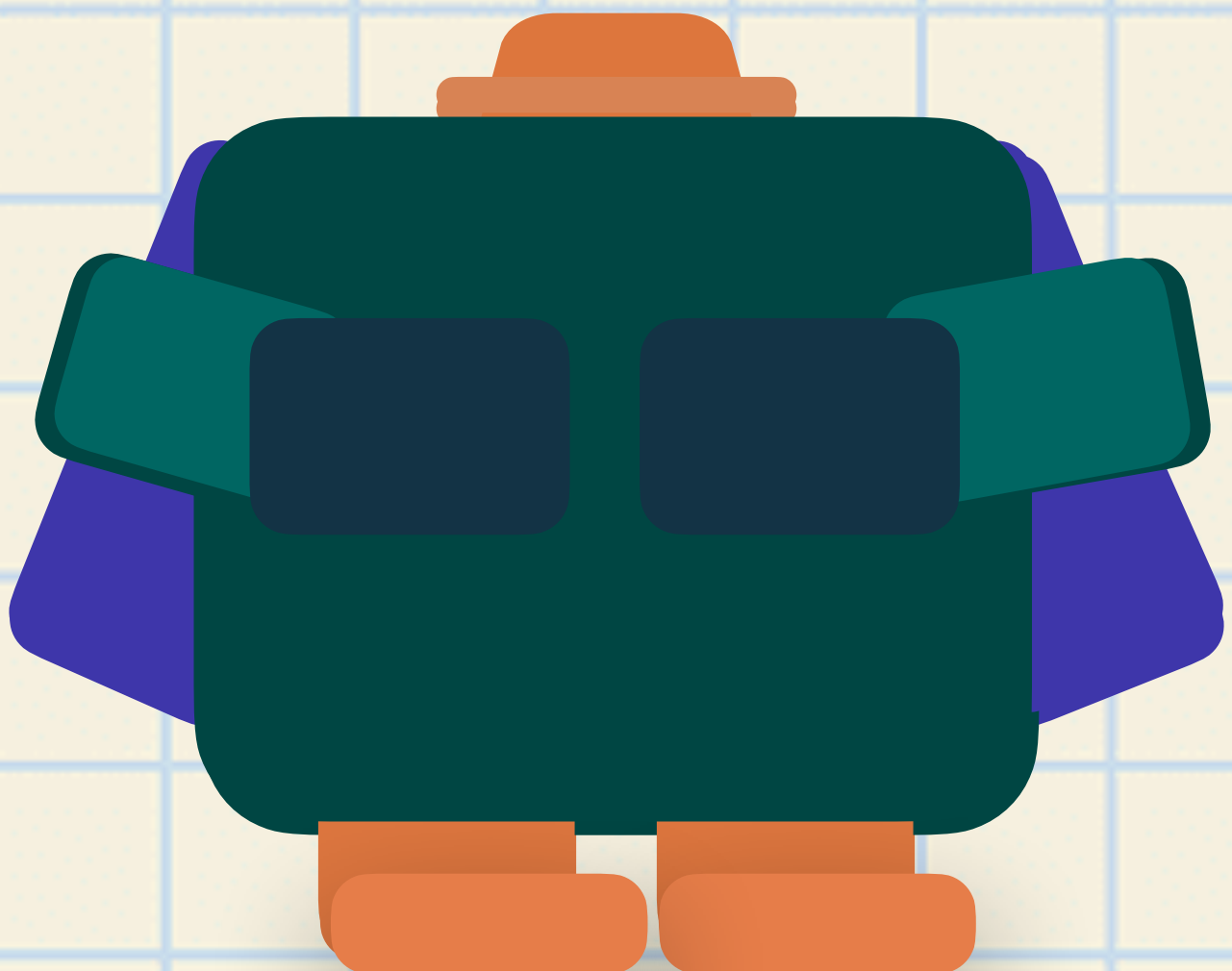
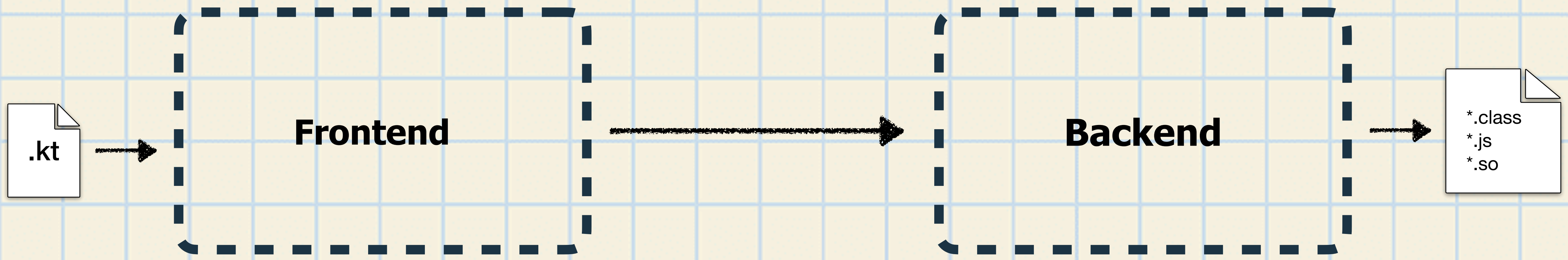
Lowering - simplifies
data format

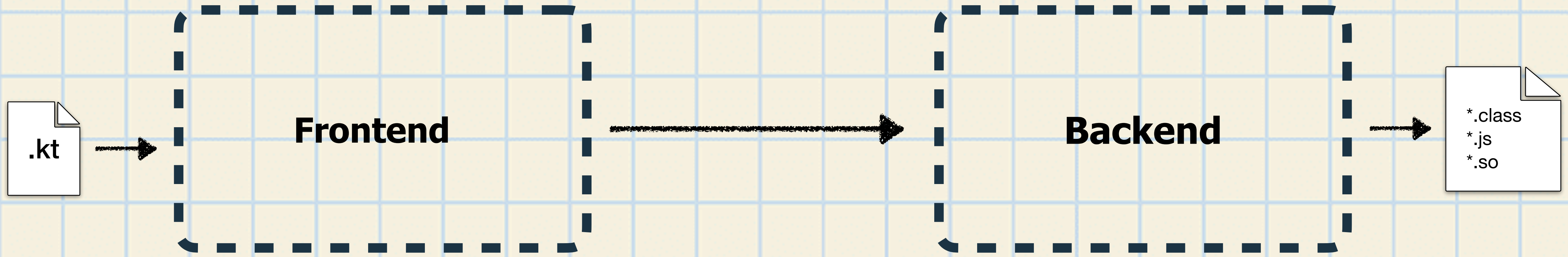


Compiler plugins can
intercept any phase
of the compiler







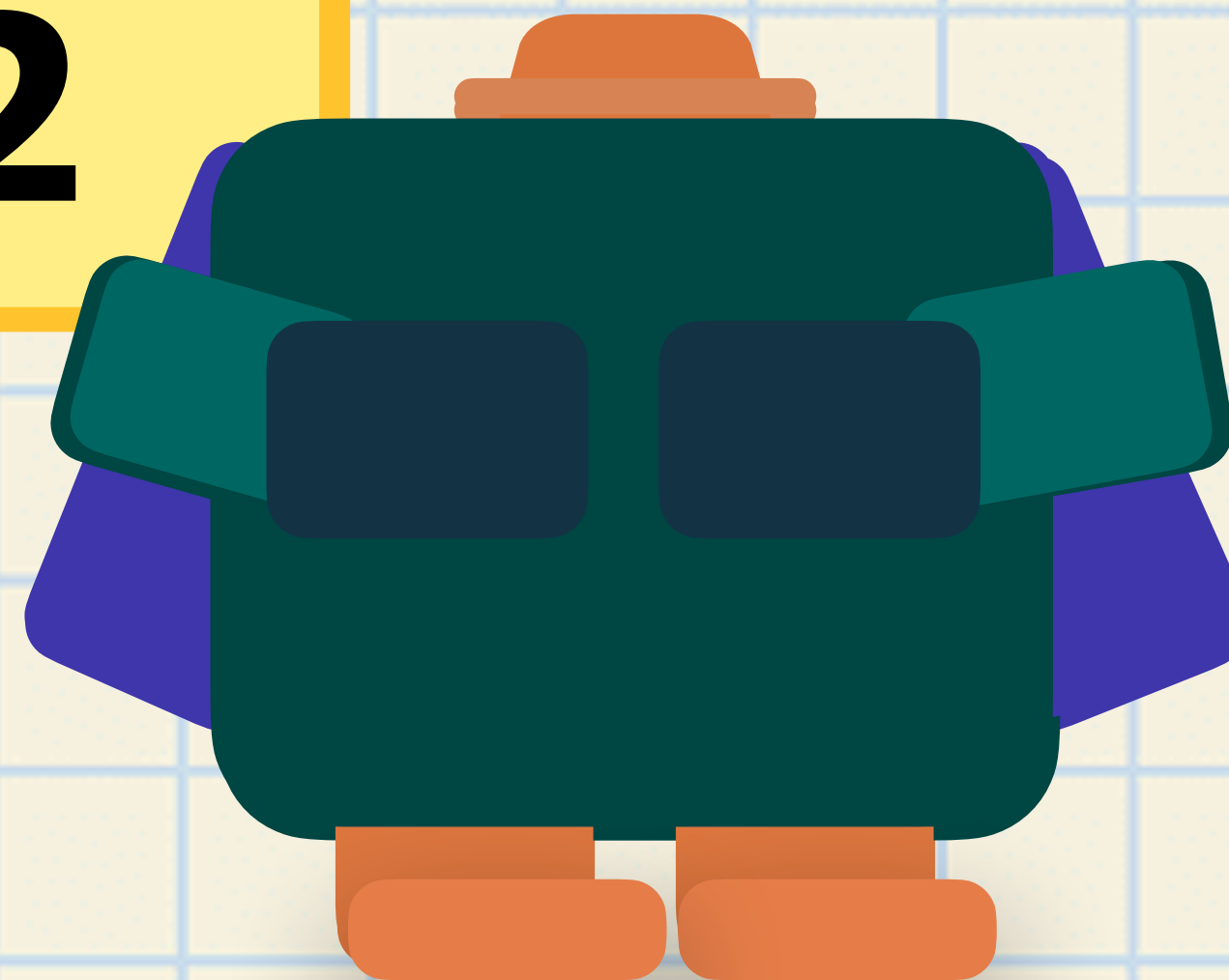


FE10-

K1

Flr-

K2



FE10-

K1

Frontend

PSI

Psi2Ir

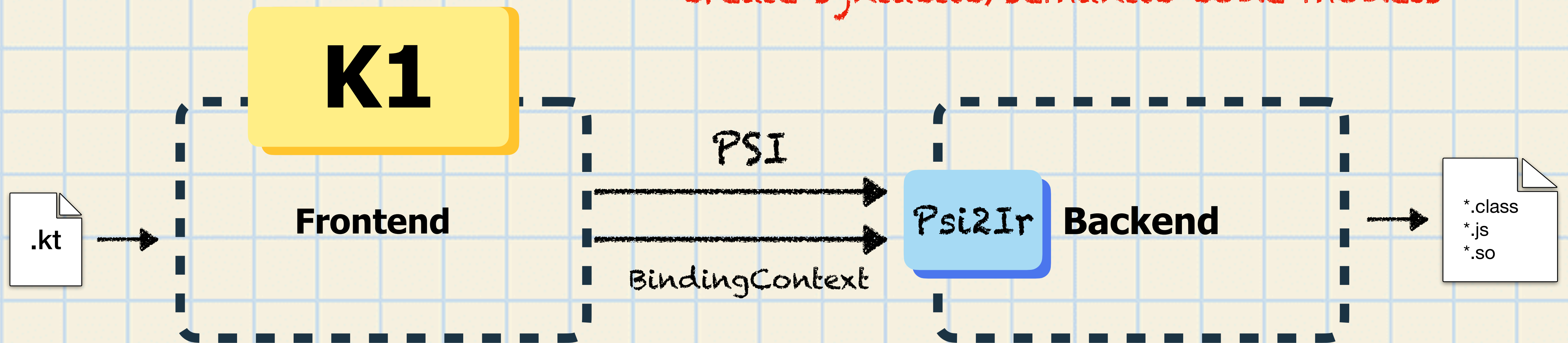
Backend

BindingContext

*.class
*.js
*.so

FE10-

Program Structure Interface (PSI) -
layer in IntelliJ to help parse files and
create syntactic/semantic code models



FE10-

Program Structure Interface (PSI) - layer
in IntelliJ to help parsing files and
creating syntactic/semantic code models

K1

Frontend

PSI

Psi2Ir

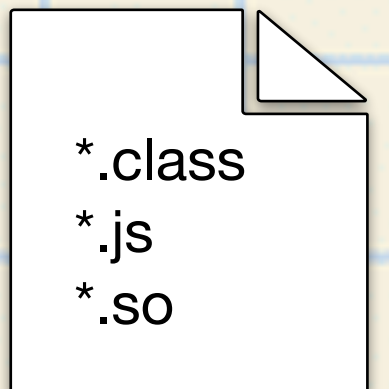
Backend

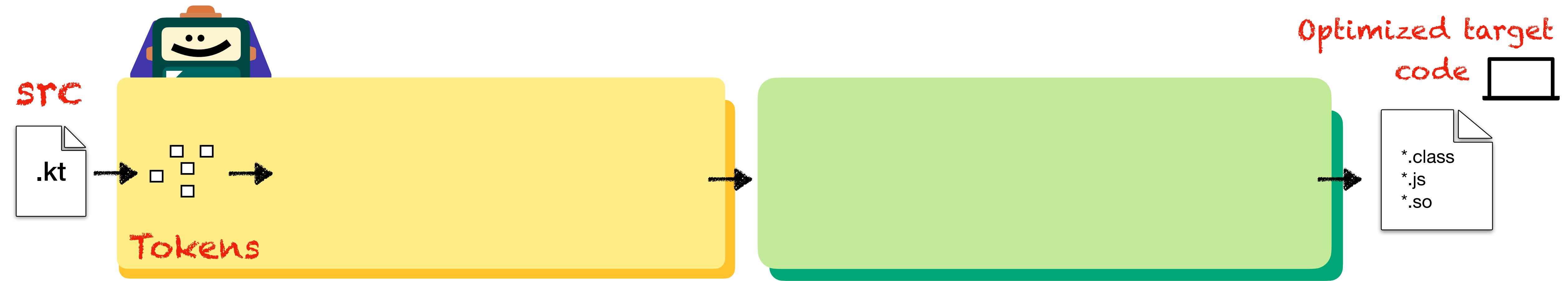
BindingContext

*.class
*.js
*.so

Binding Context - Big map which
contains information of program in
pair with PSI

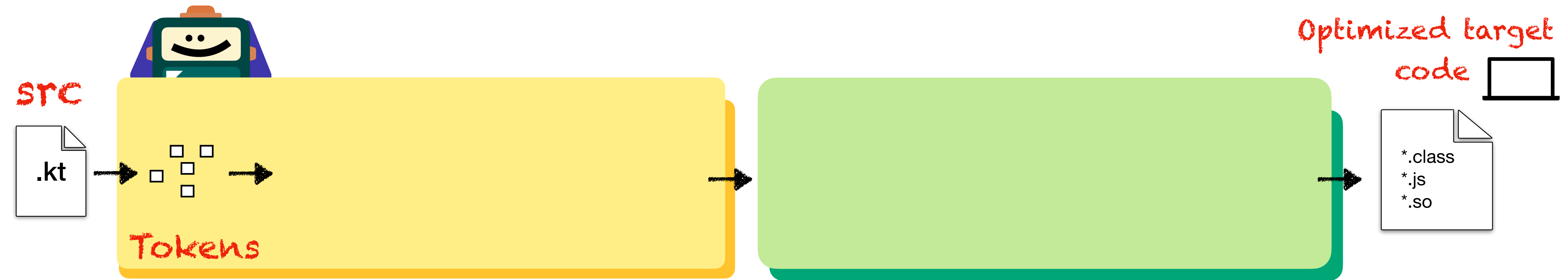
Descriptors hold context, scope,
containing info, overrides,
companion object etc.





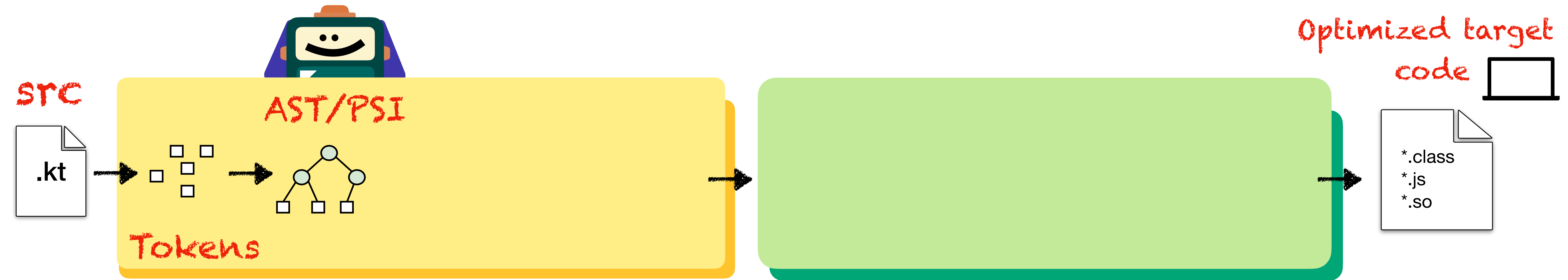
```
fun two() = 2
```

K1 Frontend: Lexical Analysis



fun	☐ KtModifierKeywordToken.FUN_KEYWORD
two	☐ KtToken.Identifier
(	☐ KtSingleValueToken.LPAR
)	☐ KtSingleValueToken.RPAR
=	☐ KtSingleValueToken.EQ
2	☐ KtToken.Identifier

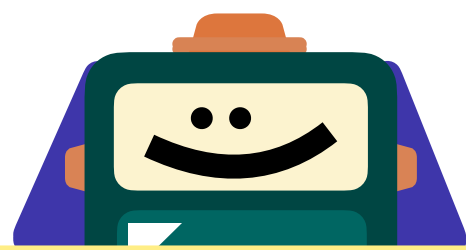
K1 Frontend: Lexical Analysis



Abstract Syntax Tree (AST)
Made of nodes that have direct mapping to the
text ranges in src

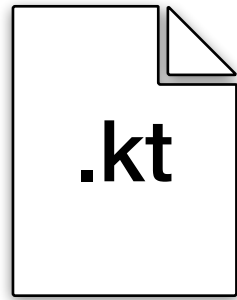
```
fun two() = 2
```

K1 Frontend: Parsing



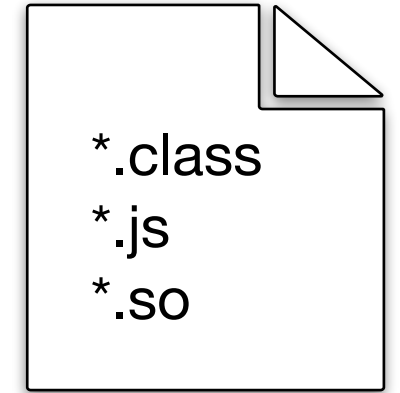
Optimized target
code 

src



AST/PSI

Tokens



Element.FUN

Token.fun

Element.IDENTIFER

Element.VALUE_ARGUMENT_LIST

Token.LPAR

Token.RPAR

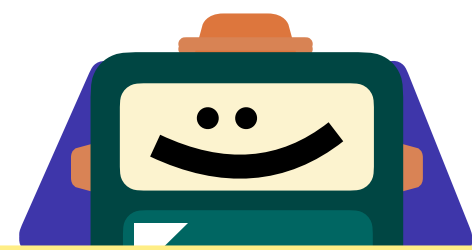
fun two() = 2

Token.EQ

Element.IDENTIFER_CONSTANT

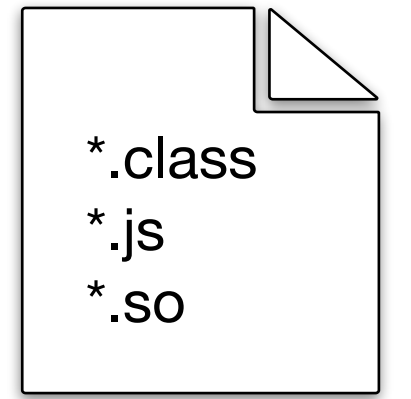
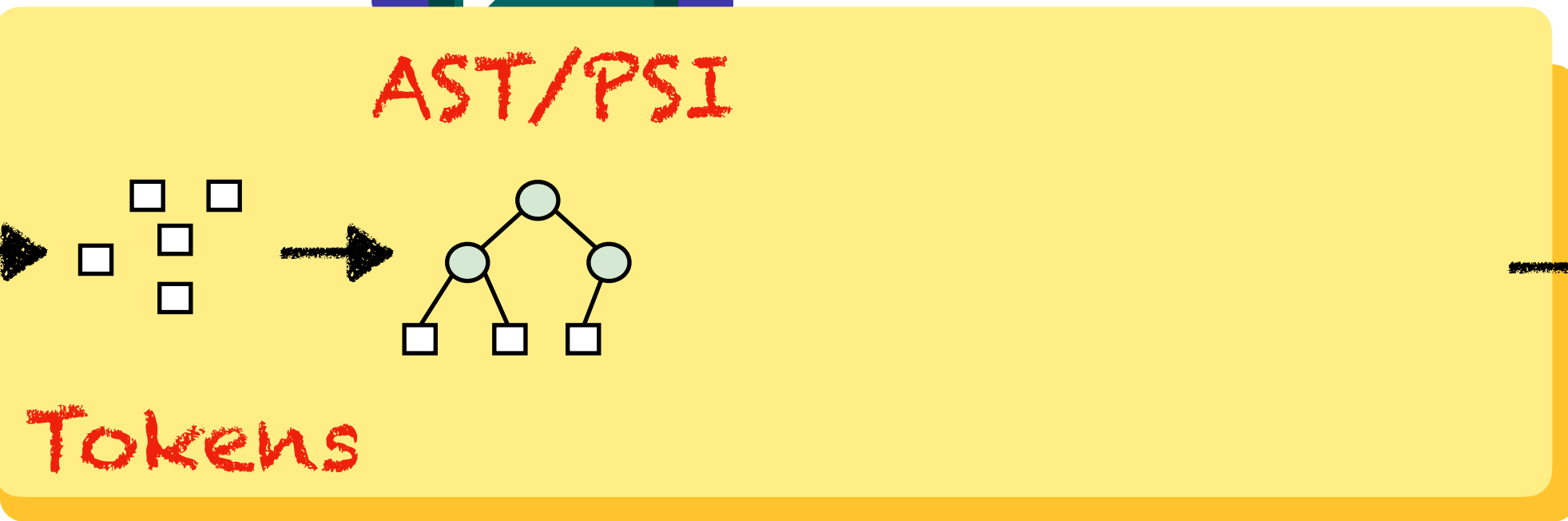
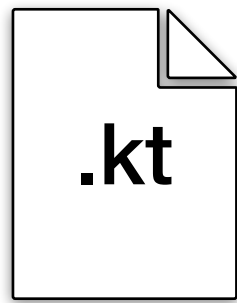
Token.INTEGER_LITERAL

K1 Frontend: Syntax Analysis



src

Optimized target
code



PSIViewer

ψ FUN

ψ PsiElement(fun)

▢ PsiWhiteSpace

ψ PsiElement(IDENTIFIER)

ψ VALUE_PARAMETER_LIST

ψ PsiElement(LPAR)

ψ PsiElement(RPAR)

▢ PsiWhiteSpace

ψ PsiElement(EQ)

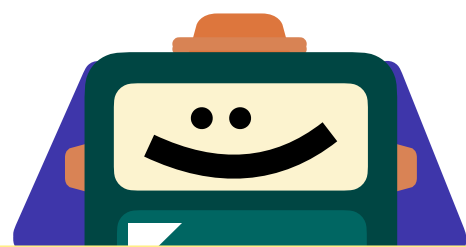
▢ PsiWhiteSpace

ψ INTEGER_CONSTANT

ψ PsiElement(INTEGER_LITERAL)

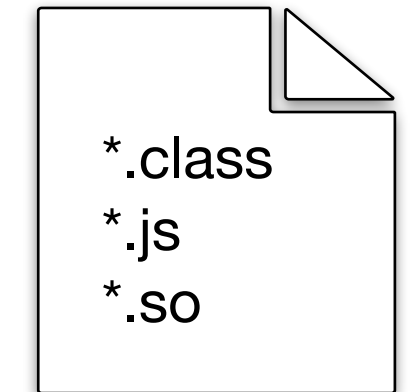
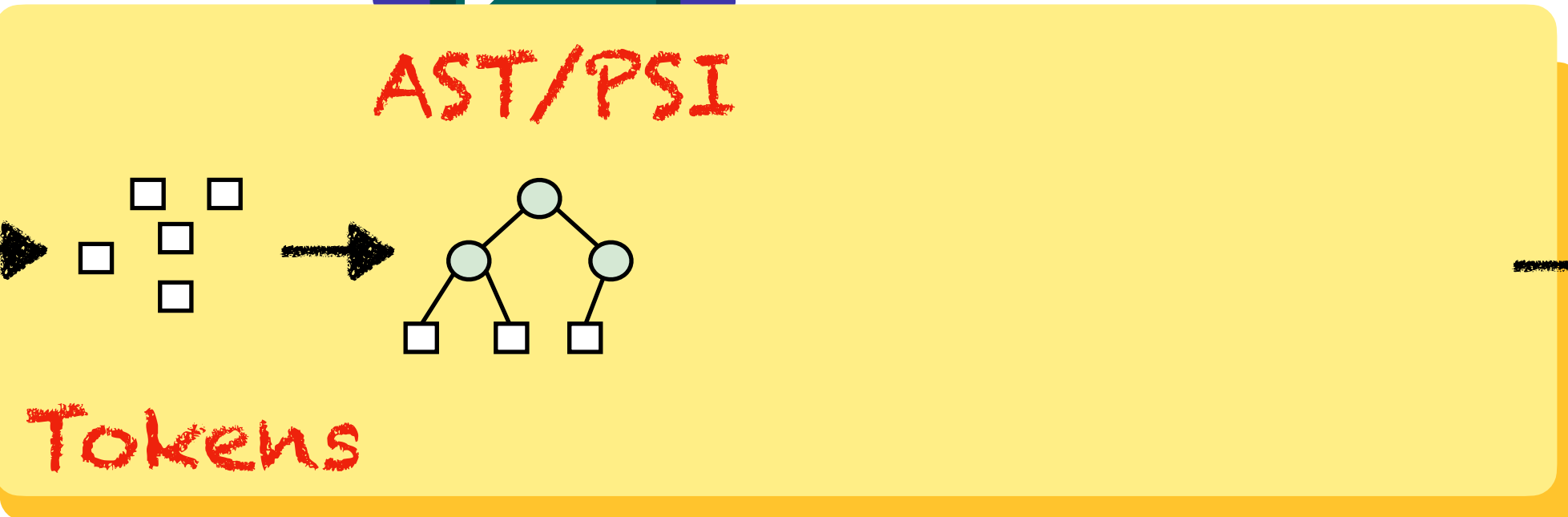
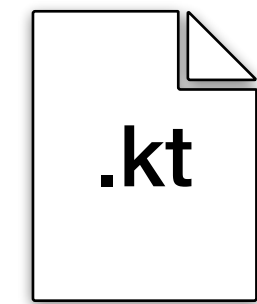
fun two() = 2

K1 Frontend: Syntax Analysis

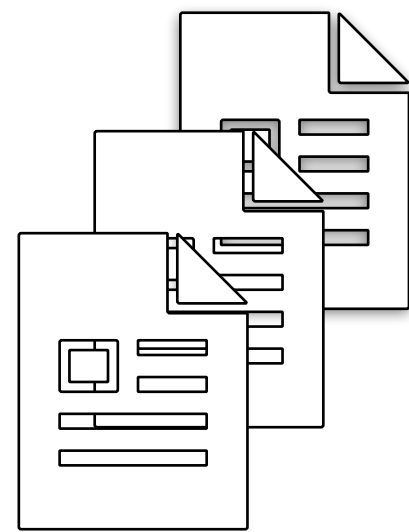


Optimized target
code 

src



Descriptors



BindingContext

fun two() = 2

K1 Frontend: Resolution

ψ FUN

ψ PsiElement(fun)

▢ PsiWhiteSpace

ψ PsiElement(IDENTIFIER)

ψ VALUE_PARAMETER_LIST

ψ PsiElement(LPAR)

ψ PsiElement(RPAR)

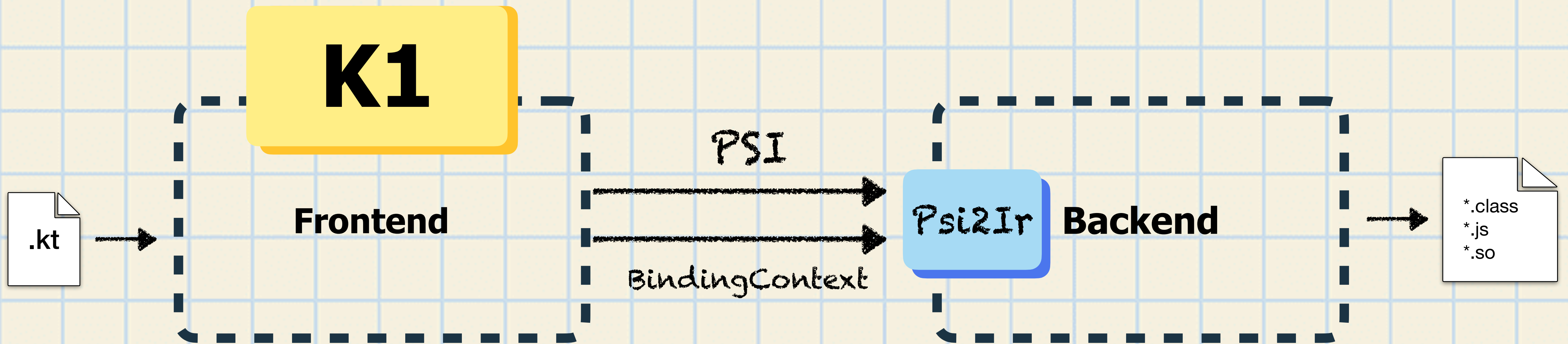
▢ PsiWhiteSpace

ψ PsiElement(EQ)

▢ PsiWhiteSpace

ψ INTEGER_CONSTANT

ψ PsiElement(INTEGER_LITERAL)



Fir-

K2

Frontend

FIR

Fir2Ir

Backend

.kt

*.class
*.js
*.so

Fir-

Frontend Intermediate
Representation (FIR)

K2

Frontend

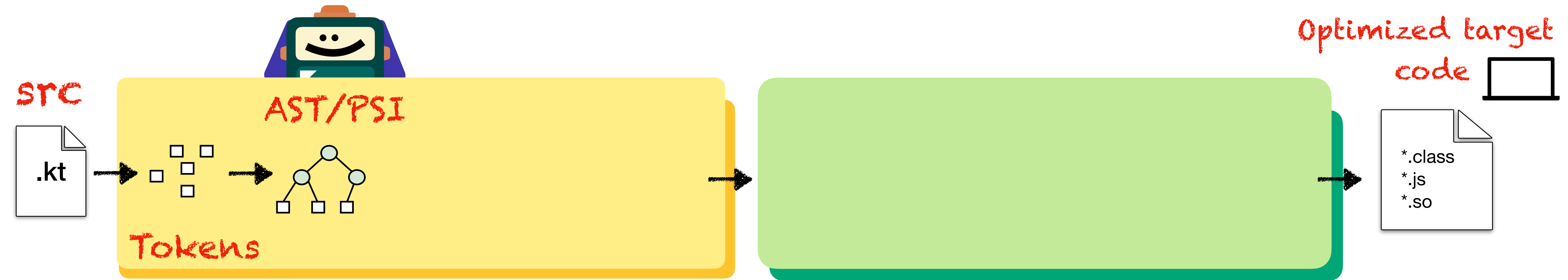
FIR

Fir2Ir

Backend

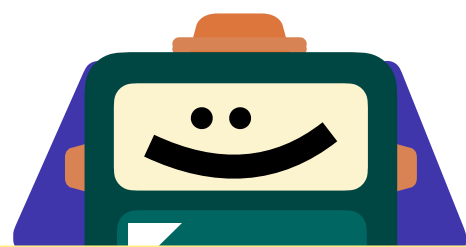
.kt

*.class
*.js
*.so

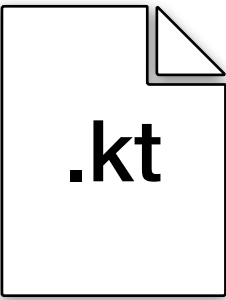


```
private val kotlinFiles = mutableListOf(1, 2, 3)
```

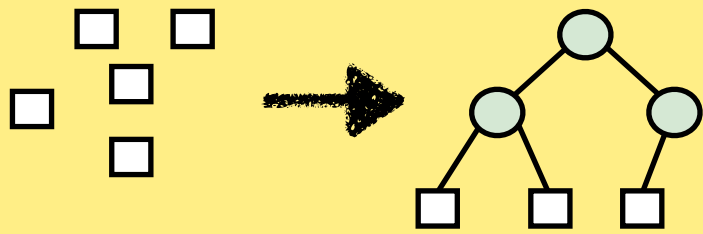
K2 Frontend: Psi2Fir



src



AST/PSI



Tokens

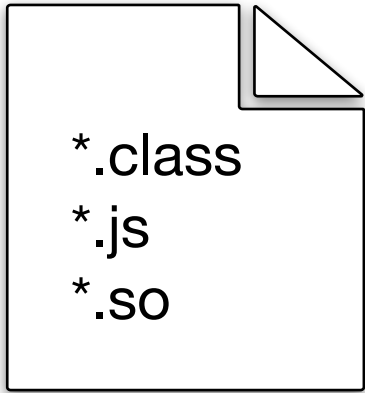


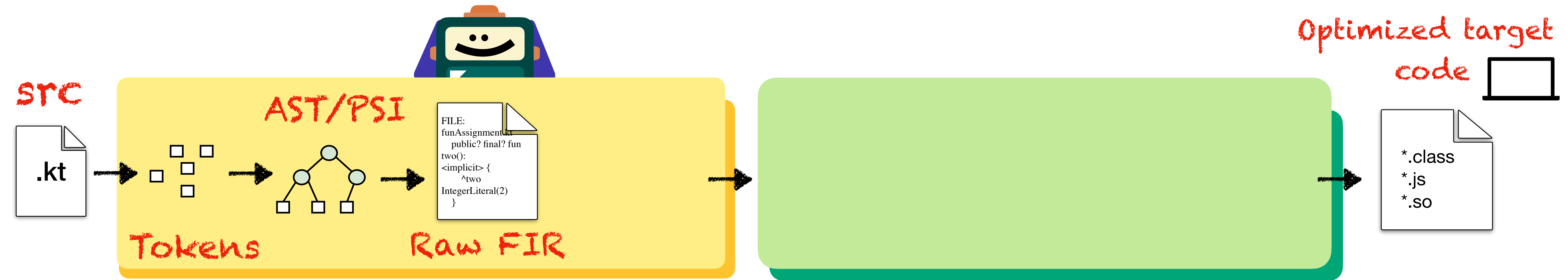
```
private val kotlinFiles =  
    mutableListOf(1, 2, 3)
```

K2 Frontend: Parsing

```
ψ PROPERTY  
  ψ MODIFIER_LIST  
    ψ PsiElement(private)  
  ψ PsiElement(val)  
  ψ PsiElement(IDENTIFIER)  
  ψ PsiElement(EQ)  
  ψ CALL_EXPRESSION  
    ψ REFERENCE_EXPRESSION  
      ψ PsiElement(IDENTIFIER)  
    ψ VALUE_ARGUMENT  
      ψ INTEGER_CONSTANT  
        ψ PsiElement(INTEGER)  
    ψ PsiElement(COMMA)  
    ψ VALUE_ARGUMENT  
      ψ INTEGER_CONSTANT  
        ψ PsiElement(INTEGER)  
    ψ PsiElement(COMMA)  
    ψ VALUE_ARGUMENT  
      ψ INTEGER_CONSTANT  
        ψ PsiElement(INTEGER)
```

Optimized target
code

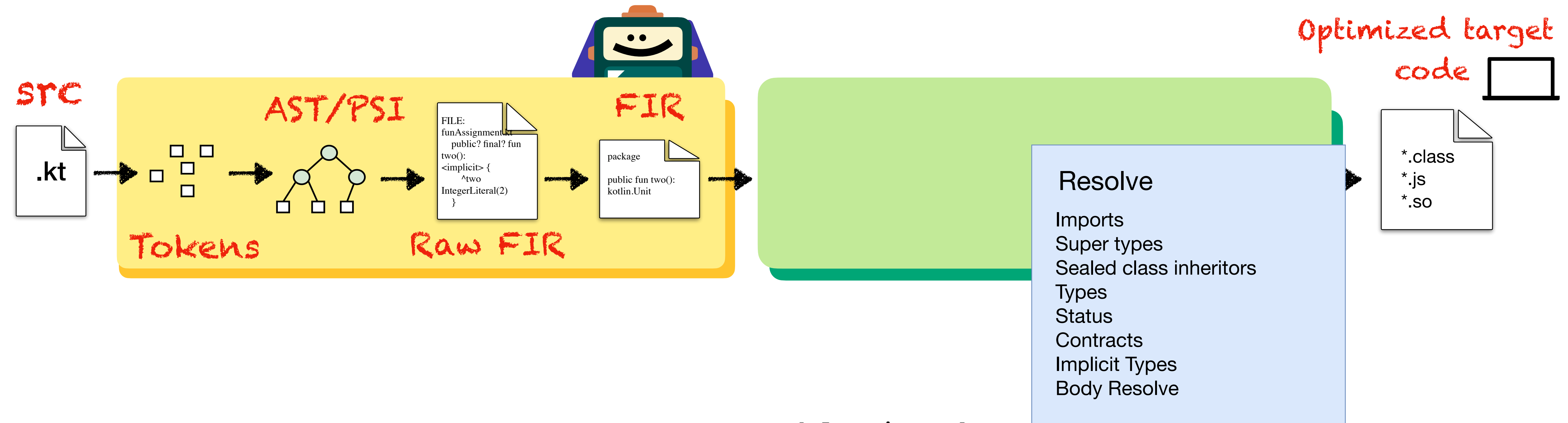




```
private val kotlinFiles =
    mutableListOf(1, 2, 3)

FILE: mutableList.kt
private? final? Val kotlinFiles: <implicit> =
    ^kotlinFiles MutableList(
        vararg(Int(1), Int(2), Int(3))
    )
```

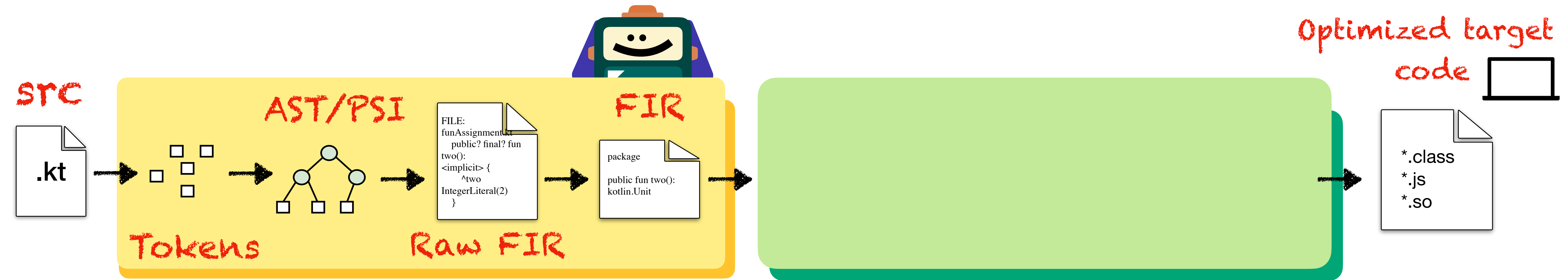
K2 Frontend: Psi2Fir



FILE: mutableList.kt

```
private? final? Val kotlinFiles: <implicit> =  
private val kotlinFiles =  
    mutableListOf(1, 2, 3)  
    ^kotlinFiles MutableList(  
        vararg(Int(1), Int(2), Int(3))  
    )
```

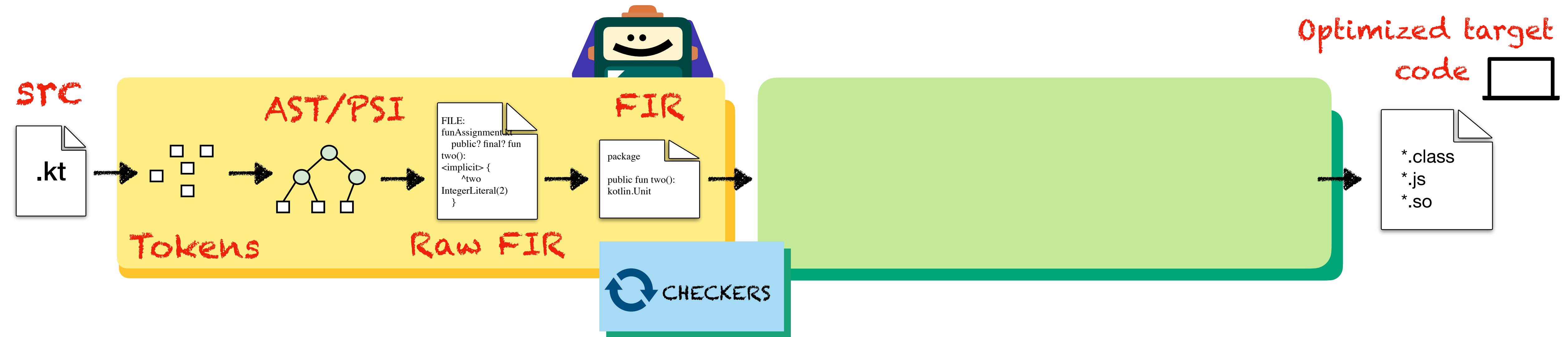
K2 Frontend: Resolution



```
private val kotlinFiles =  
    mutableListOf(1, 2, 3)
```

```
FILE: mutableList.kt  
lvar listVar:  
    R|kotlin/collections/MutableList<kotlin/Int>  
    | = R|kotlin/collections/mutableListOf  
    |<R|kotlin/Int|>(vararg(Int(1), Int(2), Int(3)))
```

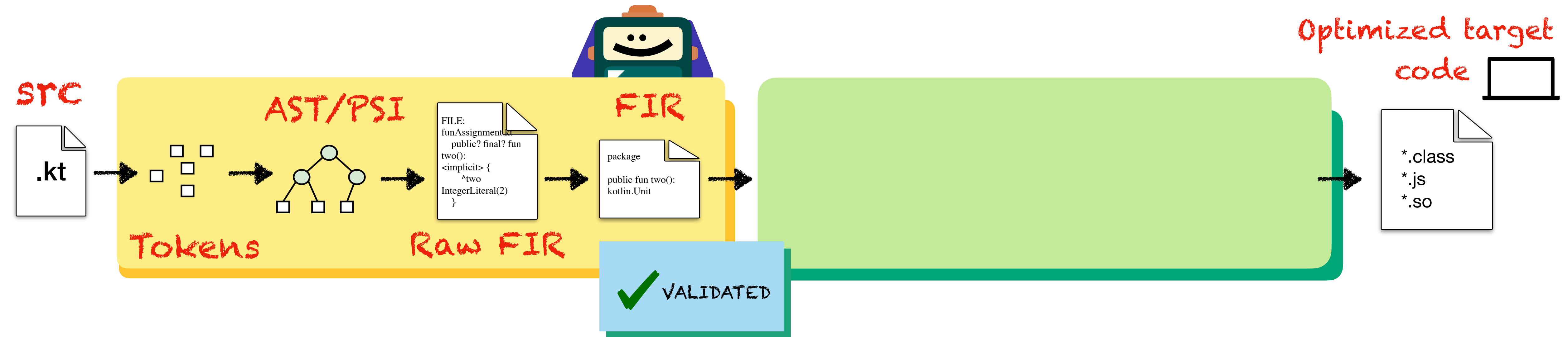
K2 Frontend: Resolution



```
private val kotlinFiles =
    mutableListOf(1, 2, 3)

FILE: mutableList.kt
lvar listVar:
    R|kotlin/collections/MutableList<kotlin/Int>
| = R|kotlin/collections/mutableListOf
|<R|kotlin/Int|>(vararg(Int(1), Int(2), Int(3)))
```

K2 Frontend: Resolution



```
private val kotlinFiles =  
    mutableListOf(1, 2, 3)  
  
FILE: mutableList.kt  
lvar listVar:  
    R|kotlin/collections/MutableList<kotlin/Int>  
| = R|kotlin/collections/mutableListOf  
|<R|kotlin/Int|>(vararg(Int(1), Int(2), Int(3)))
```

K2 Frontend: Resolution

Fir-

K2

Frontend

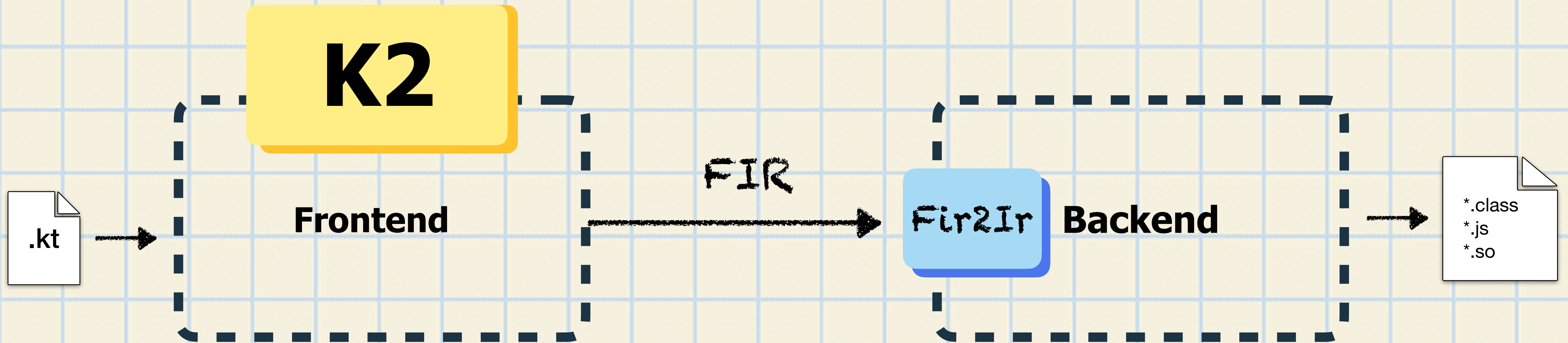
FIR

Fir2Ir

Backend

.kt

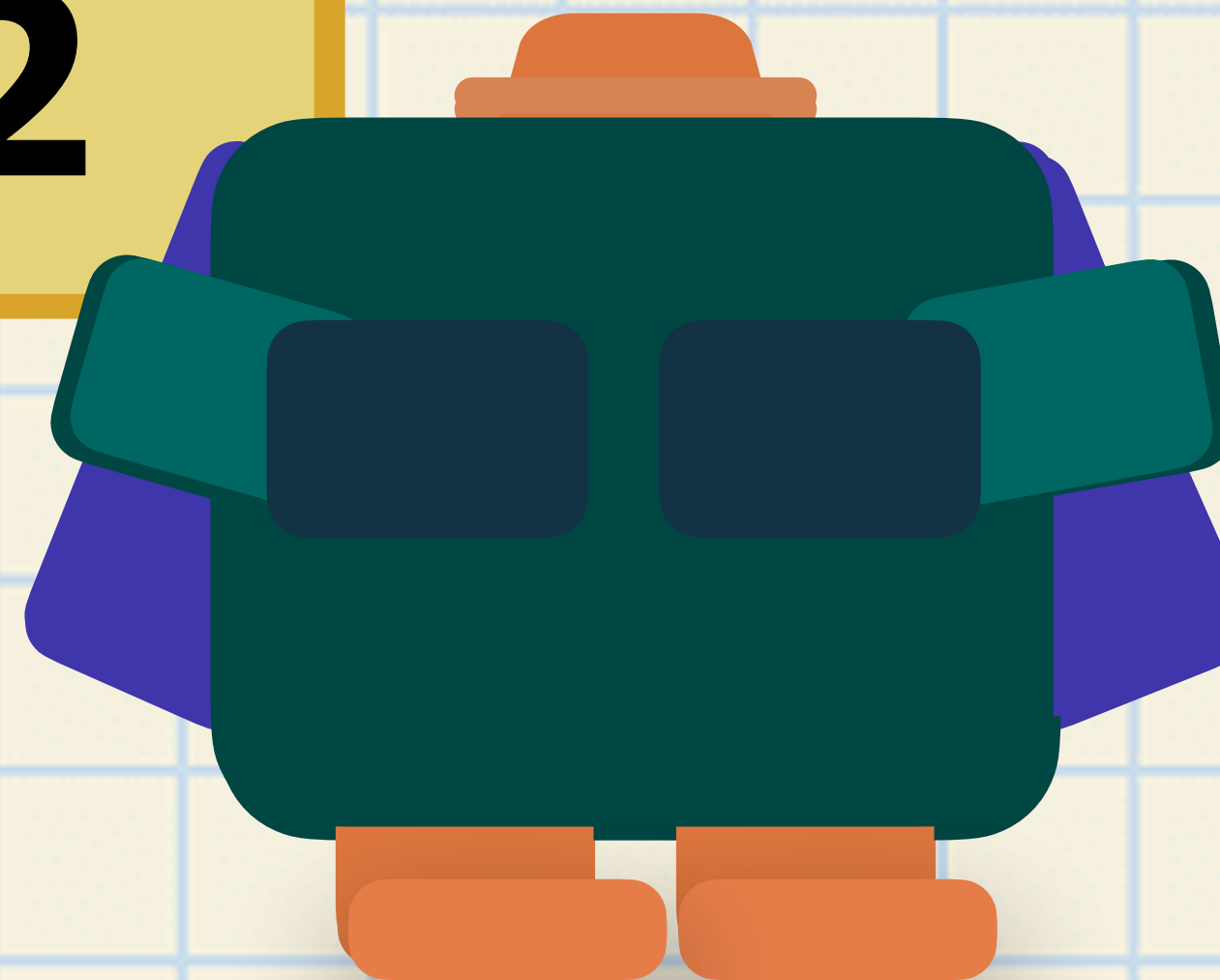
*.class
*.js
*.so





K1

K2

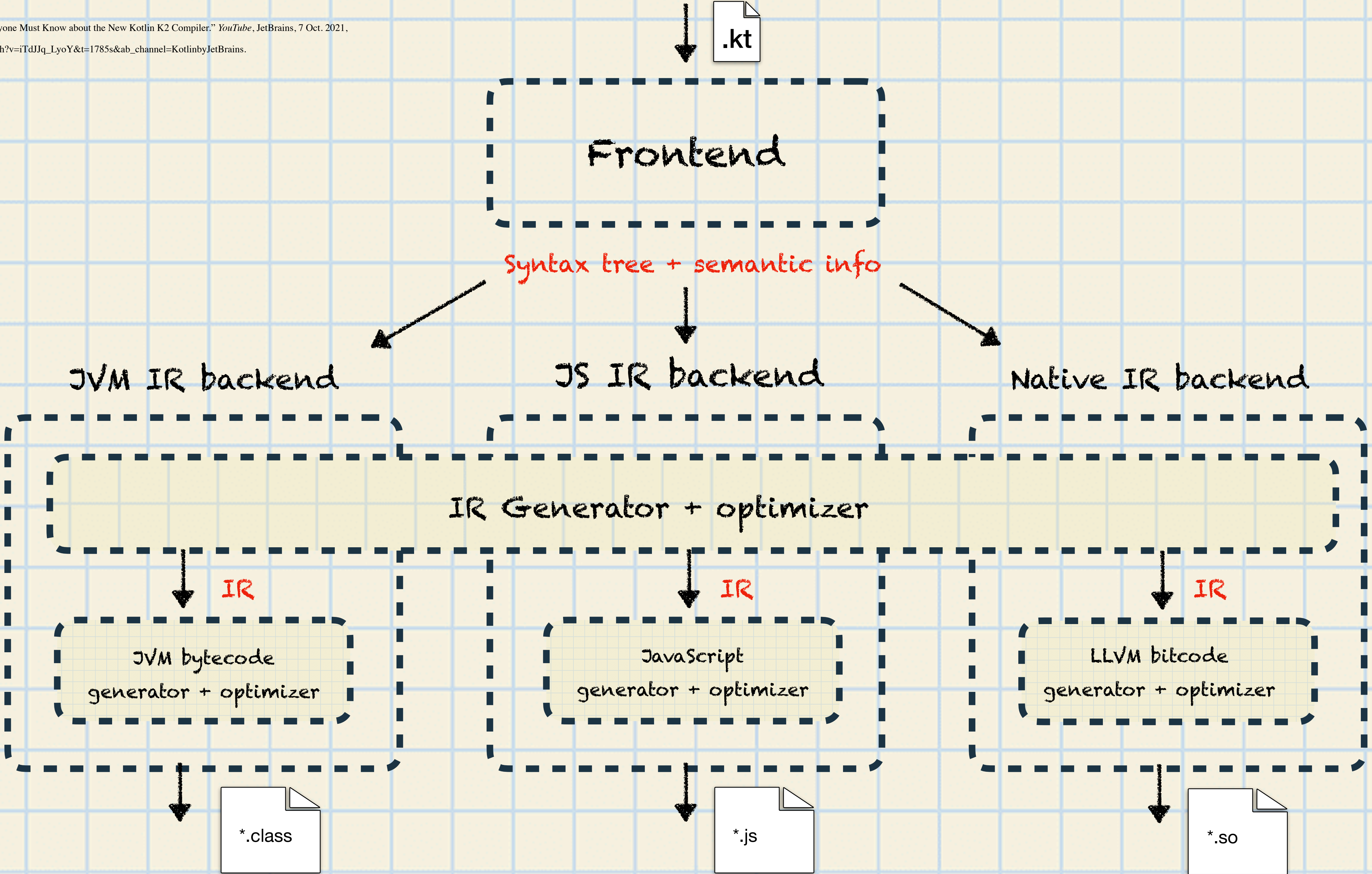


JVM

JS

Native

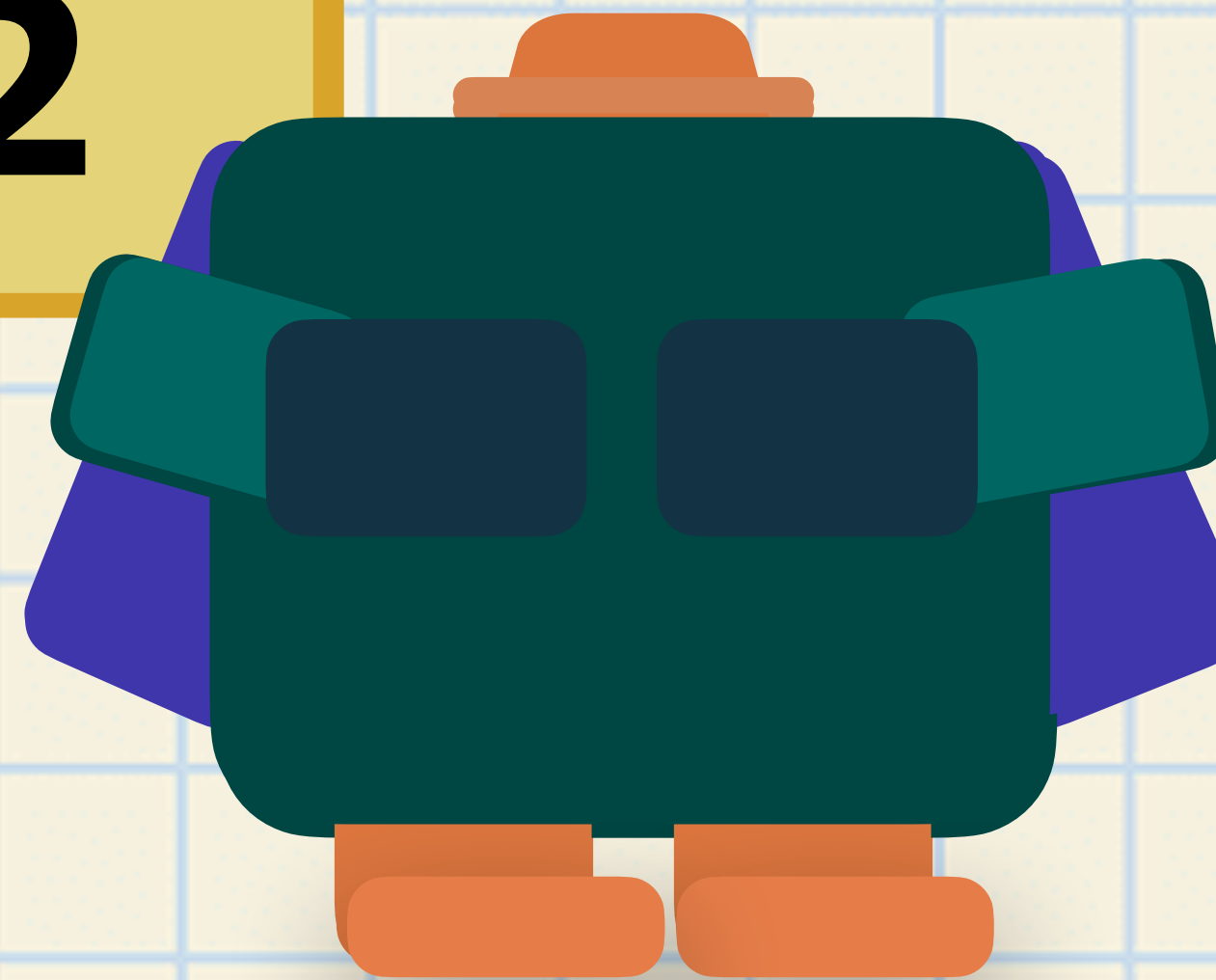
WASM





K1

K2

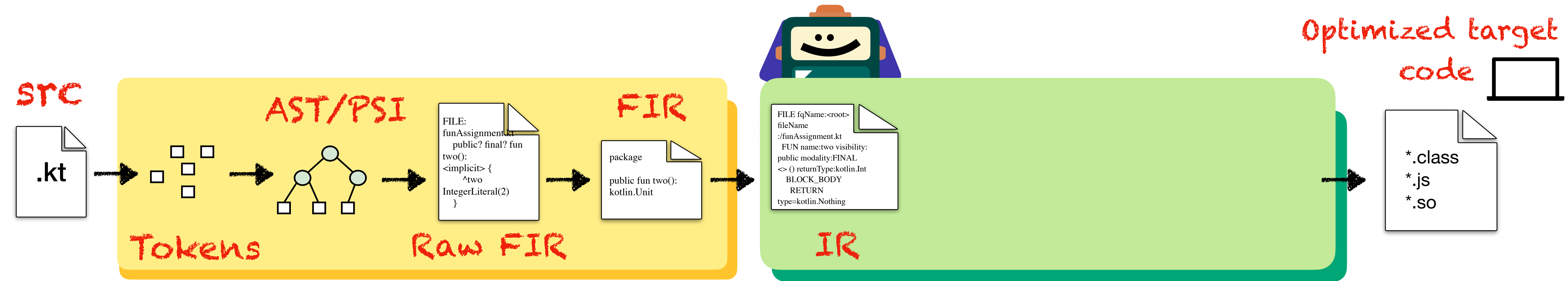


JVM

JS

Native

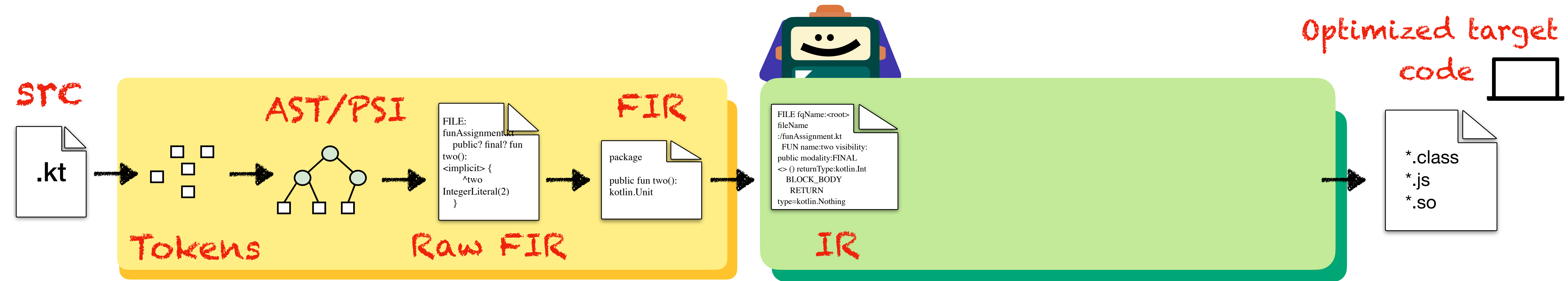
WASM



```
private val kotlinFiles =
    mutableListOf(1, 2, 3)
```

```
PROPERTY list visibility:public modality:FINAL [val]
    FIELD PROPERTY_BACKING_FIELD name:mutableListFun
type:kotlin.Function1<kotlin.collections.MutableList<kotlin.Double>,
kotlin.collections.MutableList<kotlin.Int>> visibility:private [final,static]
    EXPRESSION_BODY
        FUN_EXPR type=kotlin.Function1<kotlin.collections.MutableList<kotlin.Double>,
kotlin.collections.MutableList<kotlin.Int>> origin=ANONYMOUS_FUNCTION
            FUN LOCAL_FUNCTION name:<no name provided> visibility:local modality:FINAL <>
(1:kotlin.collections.MutableList<kotlin.Double>)
returnType:kotlin.collections.MutableList<kotlin.Int>
                VALUE_PARAMETER name:l index:0 type:kotlin.collections.MutableList<kotlin.Double>
                BLOCK_BODY
                    RETURN type=kotlin.Nothing from='local final fun <no name provided>
(1: kotlin.collections.MutableList<kotlin.Double>):
kotlin.collections.MutableList<kotlin.Int> declared in <root>.mutableListFun'
                        CALL 'public final fun CHECK_NOT_NULL <T0>
(arg0: T0 of kotlin.internal.ir.CHECK_NOT_NULL?):
{T0 of kotlin.internal.ir.CHECK_NOT_NULL & Any} declared in kotlin.internal.ir'
type=kotlin.Nothing origin=EXCLEXCL
                            <T0>: kotlin.Nothing
                            arg0: CONST Null type=kotlin.Nothing? value=null
```

JVM Backend: IR Codegen



Intermediate Representation (IR)

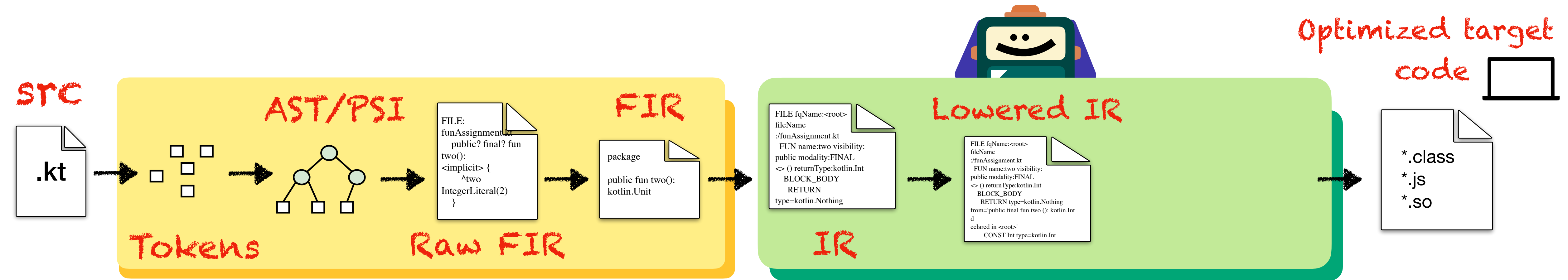
- abstracted representation for CPU-level architecture
- IR Analysis produces control flow, call stacks

JVM Backend: IR Codegen

```

PROPERTY list visibility:public modality:FINAL [val]
    FIELD PROPERTY_BACKING_FIELD name:mutableListFun
type:kotlin.Function1<kotlin.collections.MutableList<kotlin.Double>,
kotlin.collections.MutableList<kotlin.Int>> visibility:private [final,static]
    EXPRESSION_BODY
        FUN_EXPR type=kotlin.Function1<kotlin.collections.MutableList<kotlin.Double>,
kotlin.collections.MutableList<kotlin.Int>> origin=ANONYMOUS_FUNCTION
            FUN LOCAL_FUNCTION name:<no name provided> visibility:local modality:FINAL <>
(1:kotlin.collections.MutableList<kotlin.Double>)
returnType:kotlin.collections.MutableList<kotlin.Int>
            VALUE_PARAMETER name:l index:0 type:kotlin.collections.MutableList<kotlin.Double>
            BLOCK_BODY
                RETURN type=kotlin.Nothing from='local final fun <no name provided>
(1: kotlin.collections.MutableList<kotlin.Double>):
kotlin.collections.MutableList<kotlin.Int> declared in <root>.mutableListFun'
                    CALL 'public final fun CHECK_NOT_NULL <T0>
(arg0: T0 of kotlin.internal.ir.CHECK_NOT_NULL?):
{T0 of kotlin.internal.ir.CHECK_NOT_NULL & Any} declared in kotlin.internal.ir'
type=kotlin.Nothing origin=EXCLEXCL
                        <T0>: kotlin.Nothing
                        arg0: CONST Null type=kotlin.Nothing? value=null

```



Lowered IR

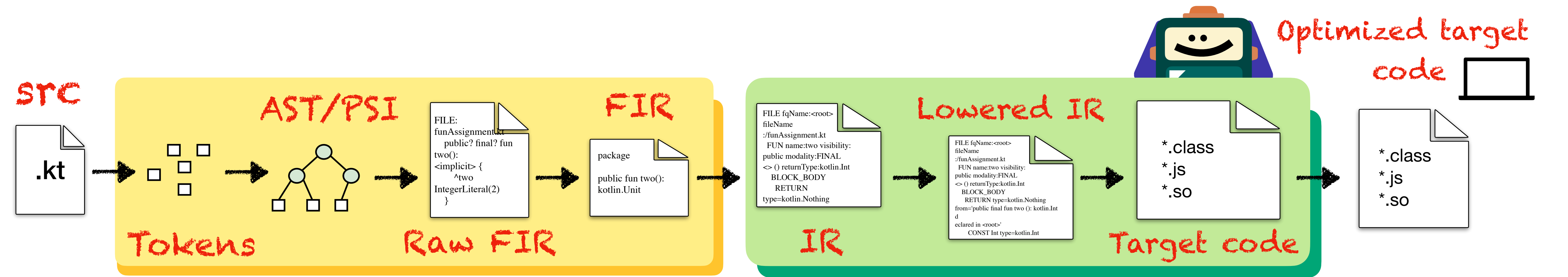
- Improve operations around performance, concurrency
- Resource + storage decisions

3^2



3×3

JVM Backend: IR Lowerings



```
private val kotlinFiles =
    mutableListOf(1, 2, 3)
```

```
// access flags 0x12
// signature Ljava/util/ArrayList<Ljava/io/File;>
// declaration: kotlinFiles extends
// java.util.ArrayList<java.io.File>
private final Ljava/util/ArrayList; kotlinFiles
```

JVM Backend: Bytecode Codegen

Reverse Engineering Plugins

**sam-with-
receiver**

**compose-
compiler-
plugin**

Reverse Engineering Plugins

**sam-with-
receiver**



`kotlin/plugins/sam-with-receiver`

**compose-
compiler-
plugin**

sam-with-receiver example

```
@Retention(RetentionPolicy.RUNTIME)
public @interface SamWithReceiver1 { }

@SamWithReceiver1
public interface Job {
    void execute(String context, int other);
}

public class Manager {
    public void doJob(Job job) {}
}
```

sam-with-receiver not configured

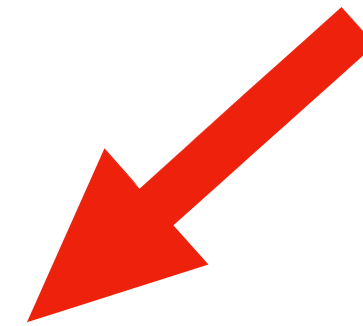
```
import lib.Manager
```

```
fun main(args: Array<String>) {  
    println("Hello World!")  
}
```

```
val manager = Manager()
```

```
manager.doJob { other ->  
    println("Context: $this | Other: $other")  
}
```

SAM interface



```
manager.doJob { context, other ->  
    println("Context: $context | Other: $other") }  
}
```

sam-with-receiver not configured

```
import lib.Manager
```

```
fun main(args: Array<String>) {  
    println("Hello World!")
```

```
    val manager = Manager()
```

```
    manager.doJob { other ->  
        println("Context: $this | Other: $other")  
    }  
}
```

```
    manager.doJob { context, other ->  
        println("Context: $context | Other: $other") }  
    }
```

```
}
```

Anonymous class



same-with-receiver not configured

```
import lib.Manager
```

```
fun main(args: Array<String>) {  
    println("Hello World!")
```

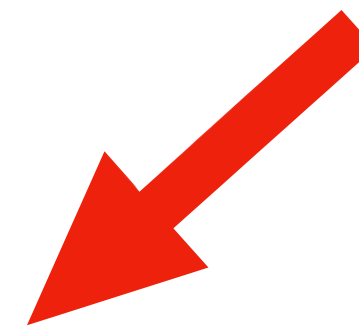
```
    val manager = Manager()
```

```
    manager.doJob { other ->  
        println("Context: $this | Other: $other")  
    }  
}
```

```
manager.doJob { context, other  
    println("Context: $context  
}
```

```
}
```

Required: (String!, Int) -> Unit
Found: (String!) -> (Int)



'this' is not defined in this context

Expected 2 parameters of types String!, Int

Type mismatch.

Required: ((String!, Int) -> Unit)!

Found: (String!) -> Unit

manager.doJob { other -> println(this) }

[Change the signature of lambda expression](#)

[More actions...](#)

```
plugins {  
    id 'org.jetbrains.kotlin.plugin.sam.with.receiver' version  
    '1.8.10'  
}  
  
samWithReceiver {  
    annotation("lib.SamWithReceiver1")  
}
```

sam-with-receiver configured

```
import lib.Manager
```

```
fun main(args: Array<String>) {  
    println("Hello World!")
```

```
    val manager = Manager()
```

```
    manager.doJob { other ->  
        println("Context: $this | Oth  
    }  
}
```

```
manager.doJob { context, other ->  
    println("Context: $context | Other: $other") }  
}
```

Required: (String!.(Int) -> Unit)!

Found: String!.(Int, Any) -> Unit

Type mismatch.

Required: (String!.(Int) -> Unit)!

Found: String!.(Int, Any?) -> Unit

Expected one parameter of type Int

Cannot infer a type for this parameter. Please specify it explicitly.

[Change the signature of lambda expression](#)

[More actions...](#)

value-parameter other: Any?

CompilerExploration.main

 `sam-with-receiver`

 `sam-with-receiver.cli`

 `sam-with-receiver.common`

 `sam-with-receiver.k1`

 `sam-with-receiver.k2`

 `FirSamWithReceiverConventionTransformer`

 `FirSamWithReceiverConventionRegistrar`

 `build.gradle.kts`

all compiler modules lays in :compiler

```
dependencies {  
    ...(project(":compiler:fir:cones"))  
    ...(project(":compiler:fir:tree"))  
    ...(project(":compiler:fir:resolve"))  
    ...(project(":compiler:fir:checkers"))  
    ...  
    ...(project(":compiler:fir:entrypoint"))  
  
    compileOnly(intelliJCore())  
    ...  
}
```

all FIR-related modules are in :compiler:fir

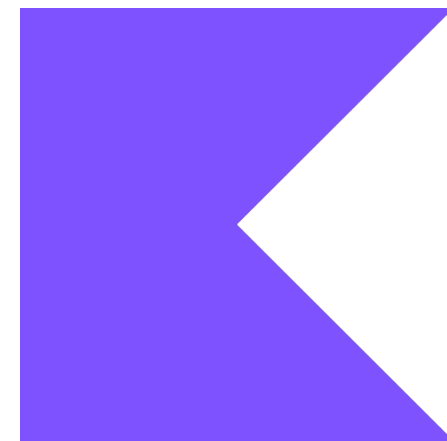
```
dependencies {  
    ...(project(":compiler:fir:cones")) - all nodes of FIR tree  
    ...(project(":compiler:fir:tree")) - classes for FIR type system  
    ...(project(":compiler:fir:resolve")) - resolution main logic  
    ...(project(":compiler:fir:checkers")) - for diagnostics  
    ...  
    ...(project(":compiler:fir:entrypoint")) - FIR compiler entry  
  
    compileOnly(intelliJCore())  
    ...  
}
```



Plugin

Subplugin

CommandLineProcessor



ComponentRegistrar

Extension

Extension

Extension

 `sam-with-receiver`

 `sam-with-receiver.cli`

 `sam-with-receiver.common`

 `sam-with-receiver.k1`

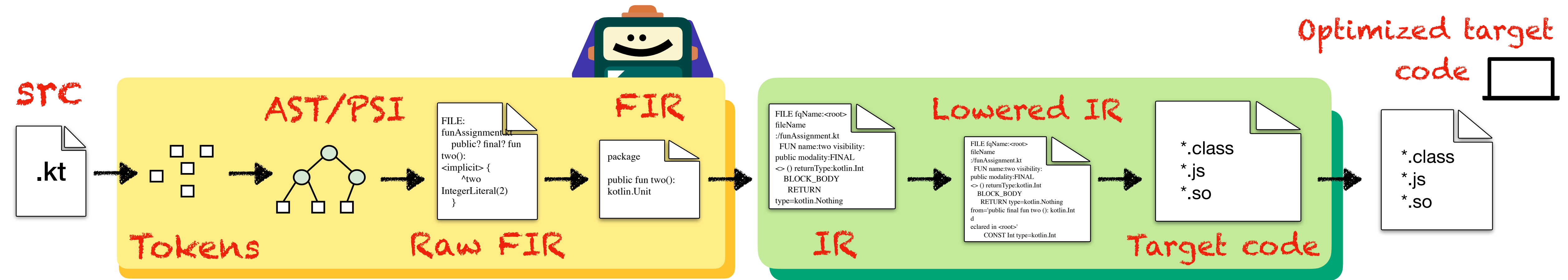
 `sam-with-receiver.k2`

 `FirSamWithReceiverConventionTransformer`

 `FirSamWithReceiverConventionRegistrar`

 `build.gradle.kts`

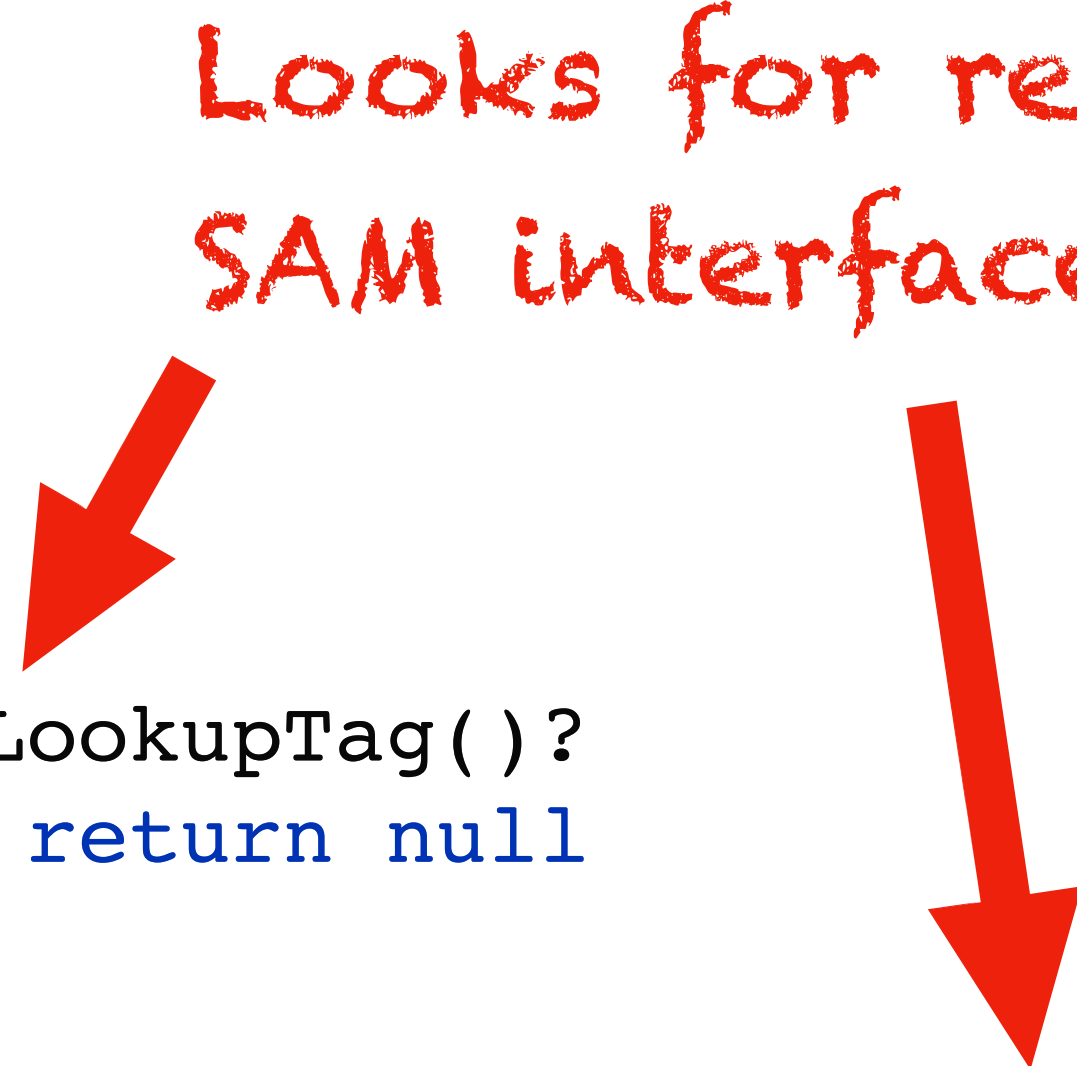
```
class FirSamWithReceiverExtensionRegistrar(  
    private val annotations: List<String>  
) : FirExtensionRegistrar() {  
    override fun ExtensionRegistrarContext.configurePlugin() {  
        +::FirSamWithReceiverConventionTransformer.bind(annotations)  
    }  
}
```



```
class FirSamWithReceiverExtensionRegistrar(
    private val annotations: List<String>
) : FirExtensionRegistrar() {
    override fun ExtensionRegistrarContext.configurePlugin() {
        +:::FirSamWithReceiverConventionTransformer.bind(annotations)
    }
}
```

```
class FirSamWithReceiverConventionTransformer(  
    private val annotations: List<String>,  
    session: FirSession  
) : FirSamConversionTransformerExtension(session) {  
    override fun getCustomFunctionalTypeForSamConversion(  
        function: FirSimpleFunction  
    ): ConeLookupTagBasedType? {  
        val containingClassSymbol = function.containingClassLookupTag()?  
            .toFirRegularClassSymbol(session) ?: return null  
        return runIf(  
            containingClassSymbol.resolvedAnnotationClassIds  
                .any { it.asSingleFqName().asString() in annotations }  
        ) {  
            val parameterTypes = function.valueParameters.map { it.returnTypeRef.coneType }  
            if (parameterTypes.isEmpty()) return null  
            createFunctionalType(  
                parameters = parameterTypes.subList(1, parameterTypes.size),  
                receiverType = parameterTypes[0],  
                rawReturnType = function.returnTypeRef.coneType,  
                isSuspend = function.isSuspend  
            )  
        }  
    }  
}
```

Looks for registered
SAM interface



```

class FirSamWithReceiverConventionTransformer(
    private val annotations: List<String>,
    session: FirSession
) : FirSamConversionTransformerExtension(session) {
    override fun getCustomFunctionalTypeForSamConversion(
        function: FirSimpleFunction
    ): ConeLookupTagBasedType? {
        val containingClassSymbol = function.containingClassLookupTag()?
            .toFirRegularClassSymbol(session) ?: return null
        return runIf(
            containingClassSymbol.resolvedAnnotationClassIds
                .any { it.asSingleFqName().asString() in annotations }
        ) {
            val parameterTypes = function.valueParameters.map { it.returnTypeRef.coneType }
            if (parameterTypes.isEmpty()) return null
            createFunctionalType(
                parameters = parameterTypes.subList(1, parameterTypes.size),
                receiverType = parameterTypes[0],
                rawReturnType = function.returnTypeRef.coneType,
                isSuspend = function.isSuspend
            )
        }
    }
}

```

Changes args so first
param is receiver



sam-with-receiver configured

```
import lib.Manager
```

```
fun main(args: Array<String>) {  
    println("Hello World!")
```

```
    val manager = Manager()
```

```
    manager.doJob { other ->  
        println("Context: $this | Other: $other")  
    }
```

```
    manager.doJob { context, other ->  
        println("Context: $context | Other: $other") }  
    }  
}
```

Reverse Engineering Plugins

androidx/compose/compiler

**sam-with-
receiver**

**compose-
compiler-
plugin**



```
@Composable  
fun HelloWorld(greeting: String) {  
    Text("Hello, $greeting!")  
}
```

```
@Composable
fun HelloWorld(greeting: String) {
    Text("Hello, $greeting!")
}

HelloWorld("KotlinConf")
```



Group(5)
EMPTY
EMPTY
EMPTY
EMPTY
EMPTY
EMPTY

```
@Composable
fun HelloWorld(greeting: String) {
    Text("Hello, $greeting!")
}

HelloWorld("KotlinConf")
```



Group(5)
Group(6)
Group(7)
Group(8)
"Hello KotlinConf!"
EMPTY
EMPTY

```
@Composable
fun HelloWorld(greeting: String) {
    Text("Hello, $greeting!")
}
```

@Composable

```
fun HelloWorld(greeting: String, %composer: Composer?, %changed: Int) {
    %composer = %composer.startRestartGroup(<>)
    val %dirty = %changed
    if (%changed and 0b1110 === 0) {
        %dirty = %dirty or if (%composer.changed(greeting)) 0b0100
        else 0b0010
    }
    if (%dirty and 0b1011 xor 0b0010 !== 0 || !%composer.skipping) {
        Text("Hello, $greeting!")
    } else {
        %composer.skipToGroupEnd()
    }
    %composer.endRestartGroup()
    ?.updateScope { %composer: Composer?, %force: Int ->
        HelloWorld(greeting, %composer, %changed or 0b0001)
    }
}
```

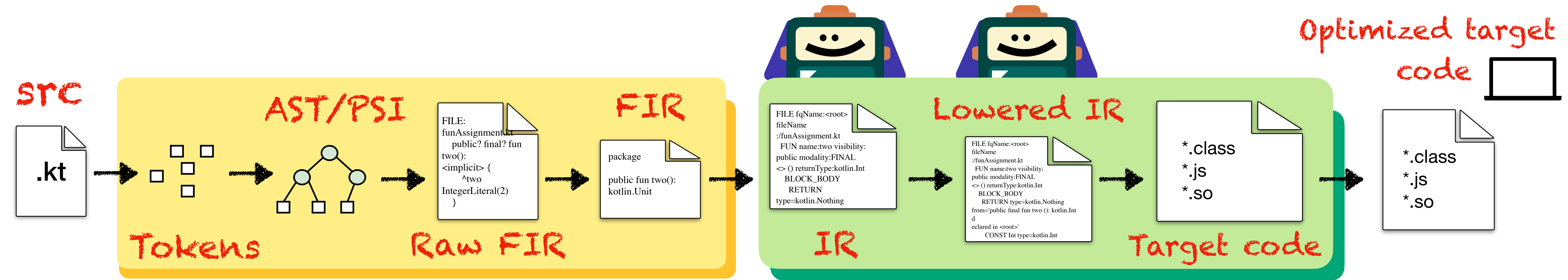
```
@Composable
fun HelloWorld(greeting: String, %composer: Composer?, %changed: Int) {
    %composer = %composer.startRestartGroup(<>)
    val %dirty = %changed
    if (%changed and 0b1110 === 0) {
        %dirty = %dirty or if (%composer.changed(greeting)) 0b0100
        else 0b0010
    }
    if (%dirty and 0b1011 xor 0b0010 !== 0 || !%composer.skipping) {
        Text("Hello, $greeting!")
    } else {
        %composer.skipToGroupEnd()
    }
    %composer.endRestartGroup()
    ?.updateScope { %composer: Composer?, %force: Int ->
        HelloWorld(greeting, %composer, %changed or 0b0001)
    }
}
```

```
@Composable
fun HelloWorld(greeting: String, %composer: Composer?, %changed: Int) {
    %composer = %composer.startRestartGroup(<>)
    val %dirty = %changed
    if (%changed and 0b1110 === 0) {
        %dirty = %dirty or if (%composer.changed(greeting)) 0b0100
        else 0b0010
    }
    if (%dirty and 0b1011 xor 0b0010 !== 0 || !%composer.skipping) {
        Text("Hello, $greeting!")
    } else {
        %composer.skipToGroupEnd()
    }
    %composer.endRestartGroup()
    ?.updateScope { %composer: Composer?, %force: Int ->
        HelloWorld(greeting, %composer, %changed or 0b0001)
    }
}
```

```
@Composable
fun HelloWorld(greeting: String, %composer: Composer?, %changed: Int) {
    %composer = %composer.startRestartGroup(<>)
    val %dirty = %changed
    if (%changed and 0b1110 === 0) {
        %dirty = %dirty or if (%composer.changed(greeting)) 0b0100
        else 0b0010
    }
    if (%dirty and 0b1011 xor 0b0010 !== 0 || !%composer.skipping) {
        Text("Hello, $greeting!")
    } else {
        %composer.skipToGroupEnd()
    }
    %composer.endRestartGroup()
    ?.updateScope { %composer: Composer?, %force: Int ->
        HelloWorld(greeting, %composer, %changed or 0b0001)
    }
}
```

@Composable

```
fun HelloWorld(greeting: String, %composer: Composer?, %changed: Int) {  
    %composer = %composer.startRestartGroup(<>)  
    val %dirty = %changed  
    if (%changed and 0b1110 === 0) {  
        %dirty = %dirty or if (%composer.changed(greeting)) 0b0100  
        else 0b0010  
    }  
    if (%dirty and 0b1011 xor 0b0010 !== 0 || !%composer.skipping) {  
        Text("Hello, $greeting!")  
    } else {  
        %composer.skipToGroupEnd()  
    }  
    %composer.endRestartGroup()  
    ?.updateScope { %composer: Composer?, %force: Int ->  
        HelloWorld(greeting, %composer, %changed or 0b0001)  
    }  
}
```



```
class ComposeComponentRegistrar : @Suppress("DEPRECATION")
ComponentRegistrar {
    override fun registerProjectComponents(
        project: MockProject,
        configuration: CompilerConfiguration
    ) {
        if (checkCompilerVersion(configuration)) {
            registerCommonExtensions(project)
            IrGenerationExtension.registerExtension(...)
        }
    }
}
```

```
class ComposeIrGenerationExtension(...) : IrGenerationExtension {  
    ...  
    override fun generate(...) {  
        ...  
        // Verify that our transformations didn't break something  
        if (validateIr)  
            validateIr(moduleFragment, pluginContext.irBuiltIns)  
    }  
}
```

```
class ComposeIrGenerationExtension(...) : IrGenerationExtension {  
    ...  
    override fun generate(...) {  
        ...  
        // Input check. This should always pass...  
        if (validateIr)  
            validateIr(moduleFragment, pluginContext.irBuiltIns)  
  
        // create a symbol remapper to be used across all transforms  
        val symbolRemapper = ComposableSymbolRemapper()  
        ...  
        ClassStabilityTransformer(...).lower(moduleFragment)  
    }  
}
```

fun A(@Composable f() -> Unit)



fun A(Composer<>, Int -> Unit)*

ComposeIrGenerationExtension.kt

```
// transform all composable functions to have an extra synthetic composer  
// parameter.
```

```
ComposerParamTransformer(  
    pluginContext,  
    symbolRemapper,  
    decoysEnabled,  
    metrics,  
).lower(moduleFragment)
```

```
...
```

```
// transform calls to the currentComposer to just use the local parameter from  
// previous transform  
ComposerIntrinsicTransformer(pluginContext, decoysEnabled).lower(moduleFragment)
```

ComposerParamTransformer.kt

```
// $changed[n]
for (i in 0 until changedParamCount(realValueParams, ownerFn.thisParamCount)) {
    if (argIndex < ownerFn.valueParameters.size) {
        it.putValueArgument(
            argIndex++,
            irConst(0)
        )
    } else {
        error("1. expected value parameter count to be higher: ${this.dumpSrc()}")
    }
}
```

ComposerParamTransformer.kt

```
// $changed[n]
for (i in 0 until changedParamCount(realValueParams, ownerFn.thisParamCount)) {
    if (argIndex < ownerFn.valueParameters.size) {
        it.putValueArgument(
            argIndex++,
            irConst(0)
        )
    } else {
        error("1. expected value parameter count to be higher: ${this.dumpSrc()}")
    }
}
```

Calculates the number of 'changed' params needed



```
val IrFunction.thisParamCount
    get() = contextReceiverParametersCount +
        (if (dispatchReceiverParameter != null) 1 else 0) +
        (if (extensionReceiverParameter != null) 1 else 0)

fun changedParamCount(realValueParams: Int, thisParams: Int): Int {
    val totalParams = realValueParams + thisParams
    if (totalParams == 0) return 1 // There is always at least 1 changed param
    return ceil(
        totalParams.toDouble() / SLOTS_PER_INT.toDouble()
    ).toInt()
}
```

ComposableFunctionBodyTransformer.kt

```
@Composable
fun HelloWorld(greeting: String) {
    Text("Hello, $greeting!")
}
```

```
1 MODULE_FRAGMENT name: <test-module>
2   FILE fqName: <root> fileName:/Test.kt
3   FUN name:HelloWorld visibility:public modality:FINAL <>
  (greeting:kotlin.String) returnType: Kotlin.unit
4     annotations:
5       Composable
6     VALUE_PARAMETER name: greeting index:0 type:kotlin.String
7
8     BLOCK_BODY
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
```

```
1 MODULE_FRAGMENT name: <test-module>
2   FILE fqName: <root> fileName:/Test.kt
3   FUN name:HelloWorld visibility:public modality:FINAL <> (greeting:kotlin.String,
  $composer:androidx.compose.runtime.Composer?, $changed:kotlin.Int) returnType: Kotlin.uni
4     annotations:
5       Composable
6     VALUE_PARAMETER name:greeting index:0 type:kotlin.String
7     VALUE_PARAMETER name:$composer index:1 type:androidx.compose.runtime.Composer?,
  [assignable]
8     VALUE_PARAMETER name:$changed index:2 type:kotlin.Int
9     BLOCK_BODY
10     BLOCK type=kotlin.Unit origin=null
11     SET_VAR '$composer: android.compose.runtime.Composer? [assignable] declared in
  <root>.HelloWorld' type=kotlin.Unit origin=null
12     CALL `public abstract fun startRestartGroup (key: Kotlin:Int):
  androids.compose.runtime.Composer declared in
  androidx.compose.runtime.Composer' type=androidx.compose.runtime.Composer? origin=null
13     $this: GET_VAR '$composer: androidx.compose.runtime.Composer? [assignment]
  declared in <root>.Hello-World' type=androidx.compose.runtime.Composer? origin=null
14     key: CONST Int type=kotlin.Int value=73430592
15     CALL 'public final fun sourceInformation(composer:
  androidx.compose.runtime.ComposerKt' type=kotlin.Unit origin=null
16     composer: GET_VAR '$composer: androidx.compose.runtime.Composer? [assignable]
  declared in <root>.HelloWorld' type=kotlin.Unit origin=null
17     sourceInformation: CONST String type=kotlin.String
  value="((HelloWorld):Test.kt"
18     WHEN type=kotlin.Boolean origin=IF
19     BRANCH
20     if: WHEN type=kotlin.Boolean origin=OROR
21     BRANCH
22     if: CALL 'public final fun not (): kotlin.Boolean [operator] declared in
  kotlin.Boolean' type=kotlin.Boolean origin=null
23     $this: CALL 'public final fun EQEQEQ (arg0: kotlin.Any?, arg1:
  kotlin.Any?): kotlinBoolean declared in kotlin.internal.ir' type=kotlinBoolean \.
  origin=null
24     arg0: CALL 'public final fun and (other: kotlin.Int): kotlin.Int
  [infix] declared in kotlin.Int type=kotlin.Int origin=null
25     $this: GET_VAR '$changed: kotlin.Int declared in <root>.HelloWorld'
  type=kotlin.Int origin=null
26     other: CONST Int type=kotlin.Int value=1
27     arg1: CONST Int type=kotlin.Int value=0
28     then: CONST Boolean type=kotlin.Boolean value=true
29     BRANCH
30     if: CONST Boolean type=kotlin.Boolean value=true
31     $this: CALL 'public final fun not(): kotlin.Boolean [operator]
```

 @mvndy_hd  @amanda@androiddev.social

What now?



Exchange notes!



mvndy

Mar 28, 2022 · 6 min read ·  Listen



Crash course on the Kotlin compiler | 1. Frontend: Parsing phase

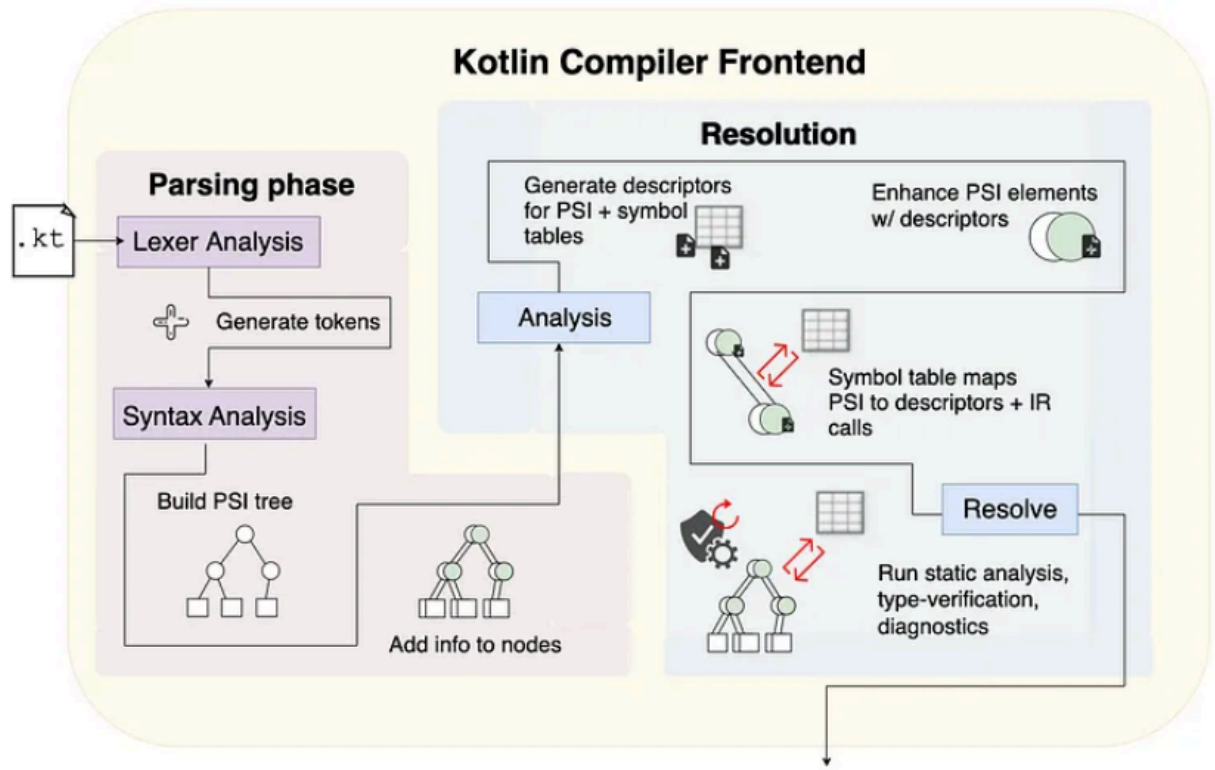
Lexers, parsers, PSI, AST v. CST, and other fundamental concepts to help understand writing IDE/compiler plugins

As programmers, we like to talk about tech things being a “black box”. For better or worse, the Kotlin compiler feels a bit like one. There is documentation around Kotlin PSI use for IDE plugins, but aside from whatever comments are left in the source code, there’s not too much else out there. It’s my hope that I can help others learn just enough information to get their own elbows dirty (and even encourage others to share what they find).

This mini-series will be released in two parts:

- 1. Frontend: Parsing phase
- 2. Frontend: Resolution phase (*coming soon*)

The Kotlin compiler is unique in that its frontend is built on top of it, making it easy to share the frontend with both compiler plugins and IDE plugins. For Kotlin, the goal of the frontend is to parse the written code and analyze its interpreted structure so it can generate *intermediate representation (IR)*. This IR, along with additional generated information is then fed to the backend of the compiler¹, which further analyzes, augments, and optimizes the IR before it eventually becomes machine code.



Get unlimited access



mvndy

1.2K Followers

software engineer and crocheting enthusiast

[Edit profile](#)

More from Medium



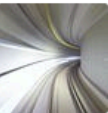
The PyCoach in Artificial Corner

You're Using ChatGPT Wrong! Here's How to Be Ahead of 99% of ChatGPT Users



Cihat Gün... in Better Program...

Migrating to The Composable Architecture (TCA) 1.0



Gabriel Shana... in The Kotlin Pri...

Programming with Result: kotlin.Result



The Coding D... in The Coding D...

Why Experienced Programmers Fail Coding Interviews



Resolution is where call resolution and type inference occurs, along with additional checks for semantic verification.: ~~inline parameters, experimental marker annotations, infix modifiers, so on and so forth.~~ Resolution also generates *descriptors*, which contains useful information like type resolution, modifiers, and data flow and can be used for more complicated PSI evaluations that might require more information about its relationship to other PSI (other files, imports, declarations, etc).

~~This blurb will be admittedly dense, as we end up covering a split of information sent off both to the backend of the Kotlin compiler as well as generated FIR and its analysis that may take place for IDE plugins, which only needs results produced in the frontend of the compiler.~~

Resolution, at a high level

The Resolution phase represents the heavier work the Kotlin compiler does in the frontend. We can think of Resolution as iterations of analysis and resolving over PSI nodes again and again with more information by drawing inference upon newly generated information and resolving components accordingly.

More specifically, the Resolution phase:

- generates symbol tables
- runs Kotlin type checking
- generates context and descriptor for PSI nodes
- map symbols to associated Kotlin types, and later on, IR in code generation
- generate synthetic descriptors for IDE plugins
- diagnostics—the logic needed to indicate to the developer whether the code is written correctly
- code verification—whether the code compiles or not
- generates FIR (*frontend intermediate representation*) used for frontend plugins and IR (*intermediate representation*) for the backend plugins

FIR allows for verification capabilities not needed by the rest of the compiler, such as

Parser is responsible for checking of grammar. Frontend performs semantic resolution



Amanda Hinchman

8:32 PM Mar 10

I had the semantic analysis bit in here, it must have been removed at one point, will definitely put it back in.

[Show more](#)



Dmitriy Novozhilov

1:42 AM Mar 13

Checking of grammar is just a check that the program is syntactically correct. For example, a program of

[Show more](#)



Dmitriy Novozhilov

1:43 AM Mar 13

`withValidityAssertion` checks that corresponding PSI was not changed (e.g. by user) during analysis

I suggest not referring to AA if you want to understand the compiler



Dmitriy Novozhilov

9:40 AM Mar 7

Absolutely wrong. FIR frontend has



 ahinchman1 Update README.md de61865 5 days ago 19 commits

 Kotlin Compiler Modified.png	Add files via upload	4 years ago
 Kotlin_compiler.png	Add files via upload	2 years ago
 README.md	Update README.md	5 days ago
 Updated_kotlin_compiler.png	Updated Kotlin Compiler rendering	2 weeks ago
 link-repo.md	Create link-repo.md	last week

A repository of helpful sources to figure out what the Kotlin compiler really is

-  Readme
-  325 stars
-  13 watching
-  15 forks

Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

Contributors 2

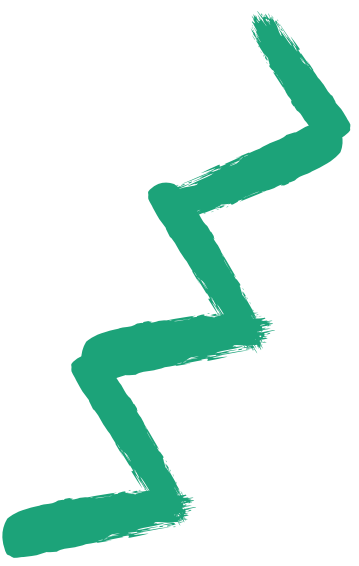
-  ahinchman1 Amanda Hinchman-Dom...
-  JeneaVranceanu Jenea Vranceanu

Octotree

 README.md 

Kotlin-Compiler-Crash-Course

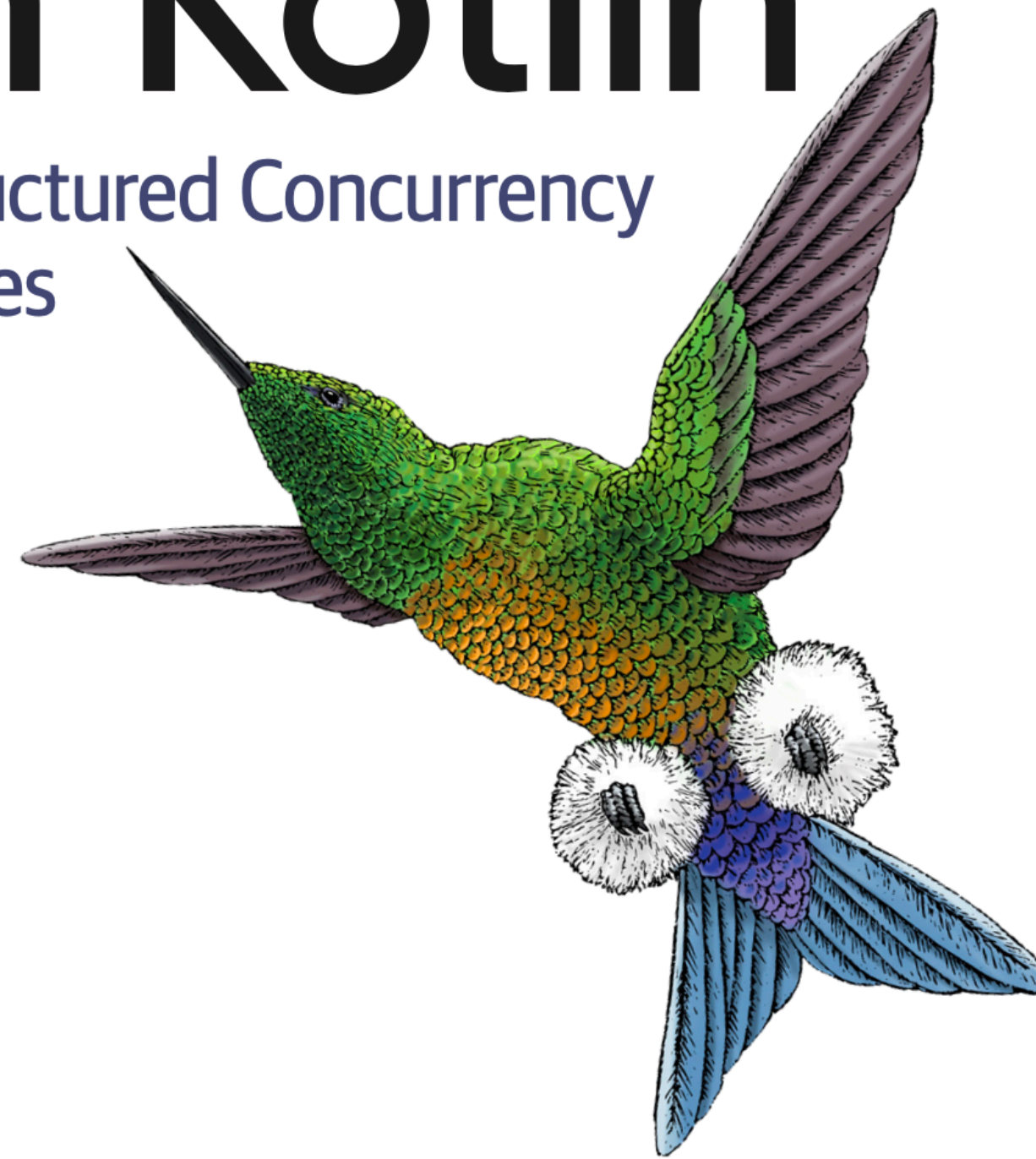
A repository of helpful sources to figure out what the Kotlin compiler really is. Consider this as more of scratchnotes hopefully useful to your Kotlin research.



O'REILLY®

Programming Android with Kotlin

Achieving Structured Concurrency
with Coroutines



Pierre-Olivier Laurence &
Amanda Hinchman-Dominguez
with G. Blake Meike & Mike Dunn

Thank you, and don't forget to vote

@mvndy_hd
KotlinConf'23
Amsterdam

