JET BRAINS Kotlin

另類的Kotlin開發者: 從零入門Kotlin編譯器插件

黃惠勤

前情提要

本次介紹將圍繞 K1 技術內容為主

講者介紹

- 黃惠勤
- FAANG ~2yrs
- 前身iOS軟體工程師 (6yrs+)

故事是這樣的...

- 大公司罕見的工作機會
- 5-10年的未來展望
- 欣賞新興語言的便利，想嘗試貢獻

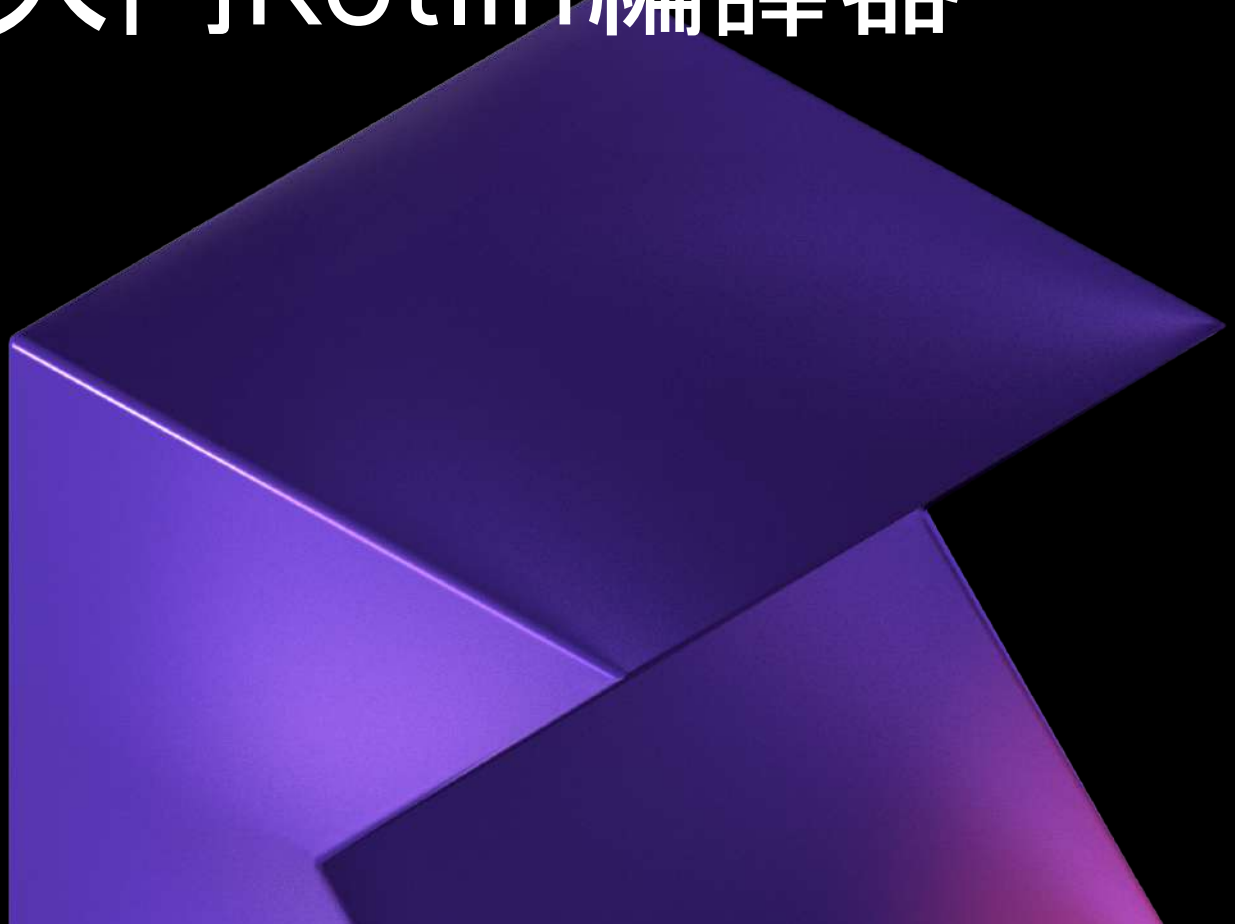
為什麼會有這樣的角色？

- Monorepo
- 上千萬行的程式碼
- 分散式運算處理
- 支援內部系統與敝司的Android開發者

攻擊與守備範圍

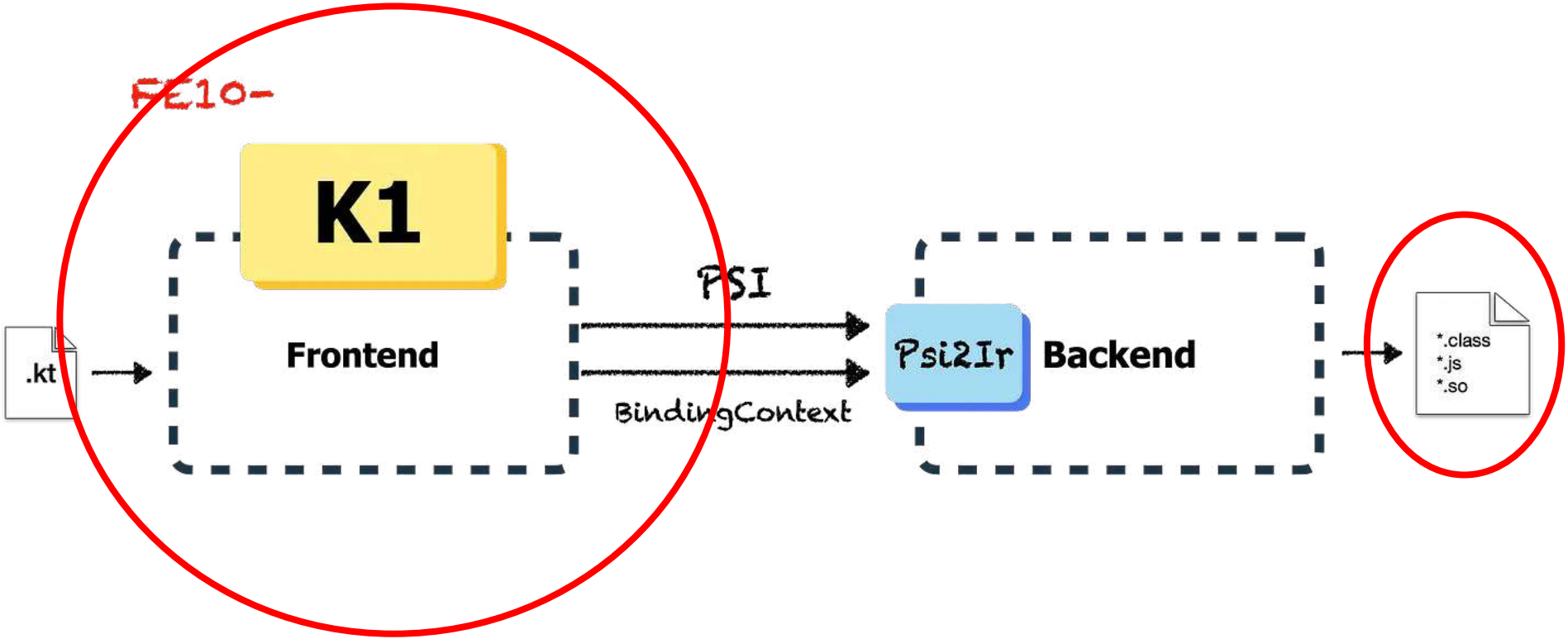
- Build Speed
- APK Size
- Null Safe
- Code Modification (大量並且自動化)
- Android Lint
- Mixed compilation

入門Kotlin編譯器

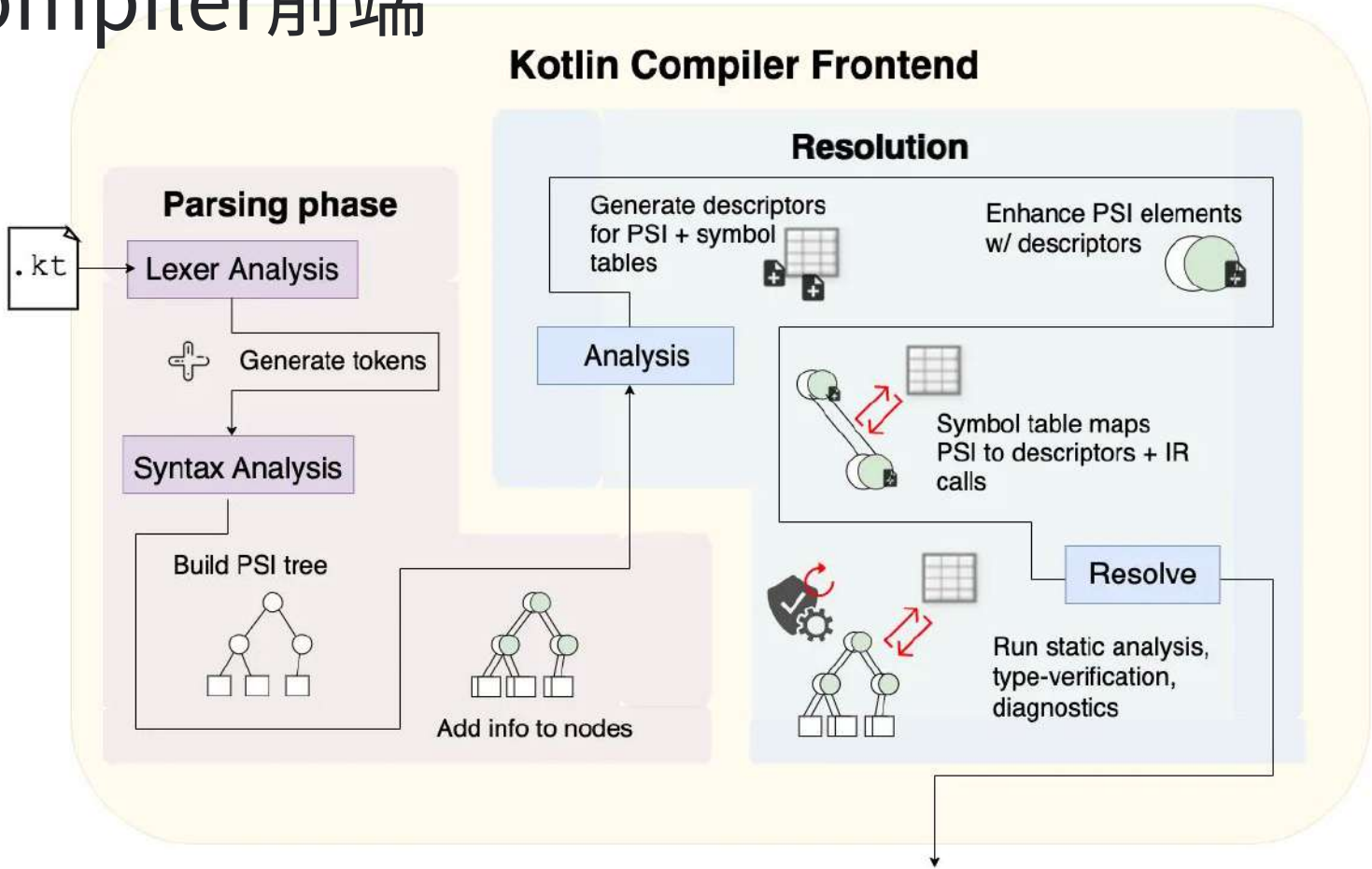


一般Android開發者視角

Compiler界也分前後端



窺看Compiler前端



反向解碼

oo.properties

```
class Person {
    var name: String = "Dave"
    fun sayHello() = println("Hello $name")
}
fun main(args: Array<String>) {
    val p = Person()
    println(p.name)
    p.sayHello()
    p.name = "Jane"
    println(p.name)
    p.sayHello()
}
```

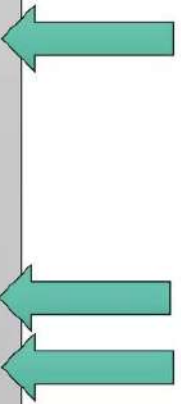
Note Kotlin does not require 'new'

```
Dave
Hello Dave
Jane
Hello Jane
```

```
javap -p oo.properties.Person

Compiled from "Person.kt"
public final class oo.properties.Person {
  private java.lang.String name;
  public final java.lang.String getName();
  public final void setName(java.lang.String);
  public final void sayHello();
  public oo.properties.regular.Person();
}
```

反向解碼

```
$ javap -p -c oo.properties.regular.PersonKt
Compiled from "Person.kt"
public final class oo.properties.regular.PersonKt {
  public static final void main(java.lang.String[]);
    Code:
       0: aload_0
       1: ldc           #9          // String args
       3: invokestatic  #15         // Method …/Intrinsics.checkParameterIsNotNull:(…)V
       6: new           #17         // class oo/properties/regular/Person
       9: dup
      10: invokespecial #21         // Method oo/properties/regular/Person."<init>":()V
      13: astore_1
      14: aload_1
      15: invokevirtual #25         // Method oo/properties/regular/Person.getName:()Ljava/lang/String;
      18: astore_2
      19: getstatic     #31         // Field java/lang/System.out:Ljava/io/PrintStream;
      22: aload_2
      23: invokevirtual #37         // Method java/io/PrintStream.println:(Ljava/lang/Object;)V
      26: aload_1
      27: invokevirtual #40         // Method oo/properties/regular/Person.sayHello:()V
      30: aload_1
      31: ldc           #42         // String Jane
      33: invokevirtual #46         // Method oo/properties/regular/Person.setName:(Ljava/lang/String;)V
      36: aload_1
      37: invokevirtual #25         // Method oo/properties/regular/Person.getName:()Ljava/lang/String;
      40: astore_2
      41: getstatic     #31         // Field java/lang/System.out:Ljava/io/PrintStream;
      44: aload_2
      45: invokevirtual #37         // Method java/io/PrintStream.println:(Ljava/lang/Object;)V
      48: aload_1
      49: invokevirtual #40         // Method oo/properties/regular/Person.sayHello:()V
      52: return
}
```



技能樹該怎麼點

- Kotlin 語法特性 / Java
- 爬Code - 開源程式碼
- JVM - Class file format
- ABI (Application Binary Interface) - Decompile
- Psi Elements (K1)
- ASM
- Annotation Processors - KAPT/KSP
- Bytecode / DEX

常用工具與參考文獻

- Kotlin playground
- JetBrains/Kotlin
- https://godbolt.org/
- PsiViewer
- Javap
- Kotlin-compile-testing
- Kotlin poet

如何不用Gradle開發

Buck / Bazel

內部原理使用 kotlinc

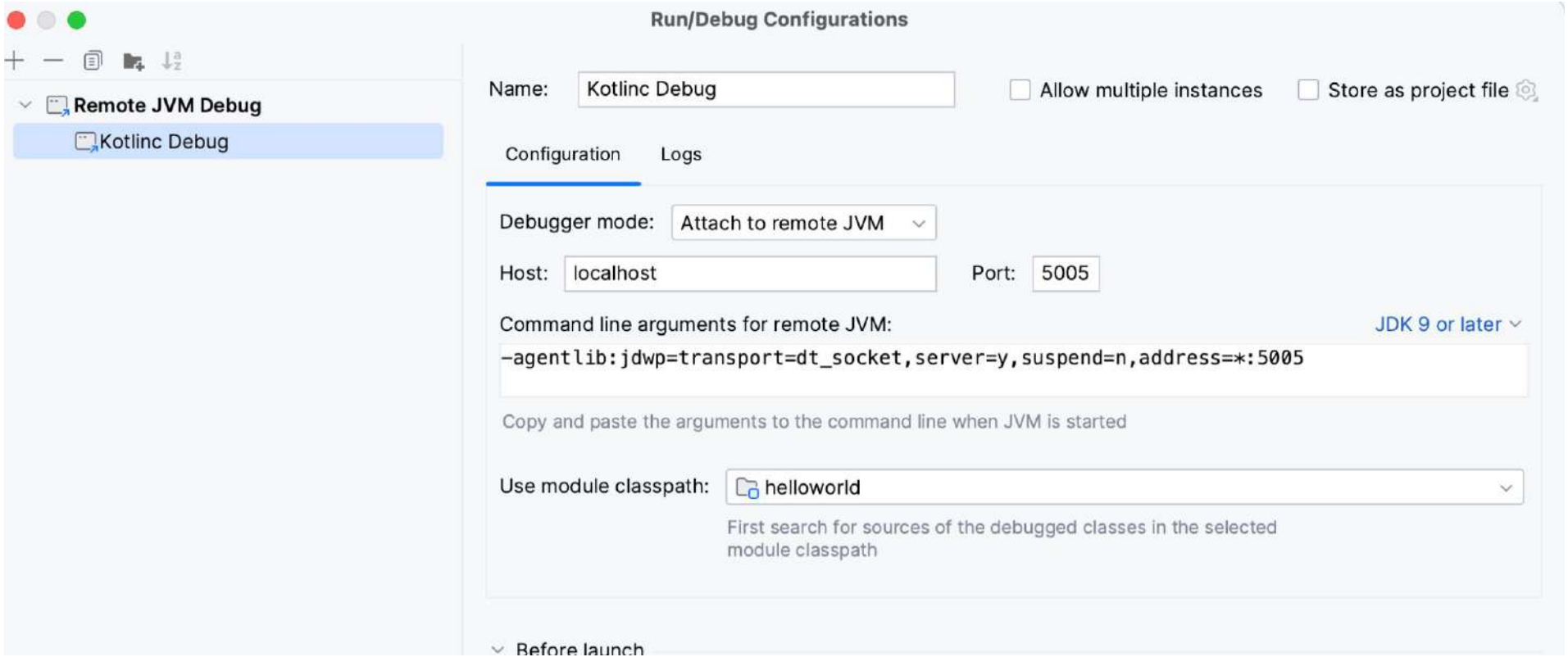
```
kotlinc hello.kt -include-runtime -d hello.jar

java -jar hello.jar
```

https://kotlinlang.org/docs/command-line.html

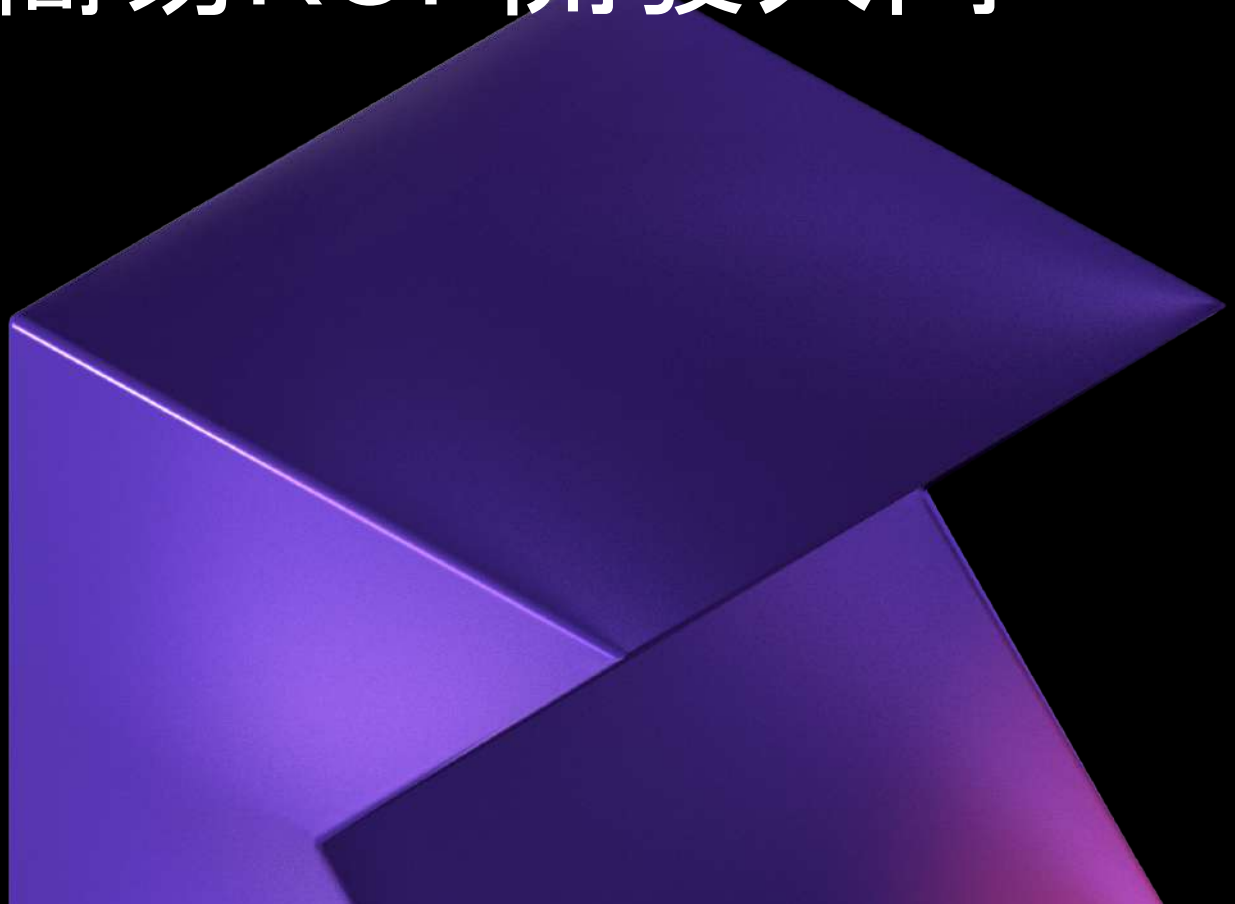
Debugging

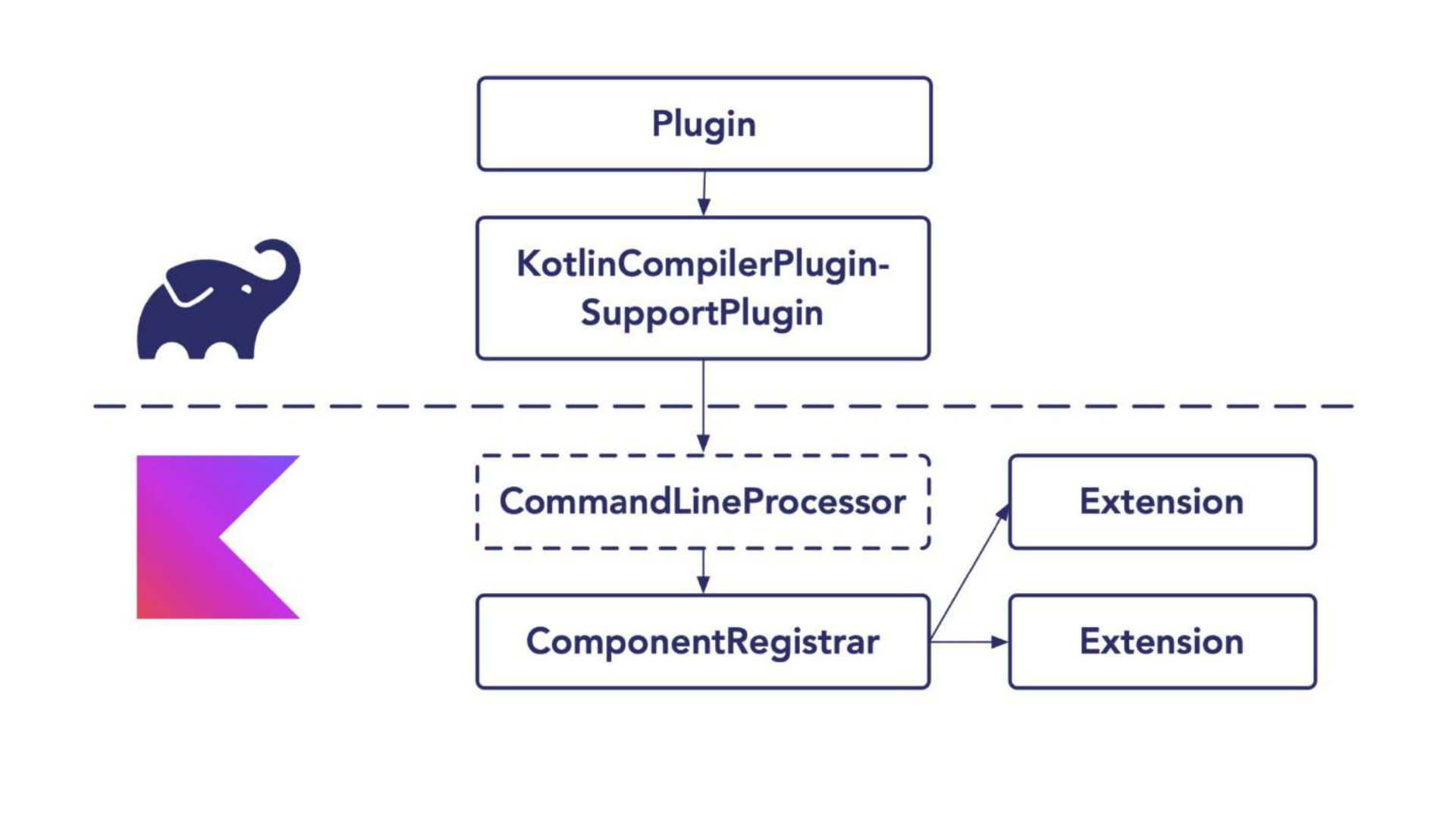
```
java
-agentlib:jdwp=transport=dt_socket,server=y,suspend=y,address=
5005
-jar hello.jar
```

Debugging


Run/Debug Configurations
Remote JVM Debug
Kotlinc Debug
Name:
Kotlinc Debug
Allow multiple instances
Store as project file
Configuration
Logs
Debugger mode:
Attach to remote JVM
Host:
localhost
Port:
5005
Command line arguments for remote JVM:
JDK 9 or later
-agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=*:5005
Copy and paste the arguments to the command line when JVM is started
Use module classpath:
helloworld
First search for sources of the debugged classes in the selected module classpath
Before launch

簡易KCP開發入門




Plugin
KotlinCompilerPlugin-
SupportPlugin
CommandLineProcessor
ComponentRegistrar
Extension
Extension

Extensions

SyntheticResolveExtension
SyntheticJavaResolveExtension
ClassFileFactoryFinalizerExtension
PackageFragmentProviderExtension
DeclarationAttributeAltererExtension
JsSyntheticTranslateExtension
CollectAdditionalSourcesExtension
ExpressionCodegenExtension
ShellExtension
CandidateInterceptor
FirExtensionRegistrarAdapter

IrGenerationExtension
ClassBuilderInterceptorExtension
AnalysisHandlerExtension
StorageComponentContainerContributor
PreprocessedVirtualFileFactoryExtension
CompilerConfigurationExtension
ExtraImportsProviderExtension
ScriptEvaluationExtension
TypeResolutionInterceptor
DescriptorSerializerPlugin
TypeAttributeTranslatorExtension

Compile and run your plugin library

```
kotlinc -cp kotlin-compiler-embeddable.jar
plugin.kt
-include-runtime -d plugin.jar

kotlinc -Xplugin=plugin.jar sources.kt …
```

Unit testing

kotlin-compile-testing

```
val result = KotlinCompilation().apply {
    sources = listOf(kotlinSource, javaSource)

    // pass your own instance of a compiler plugin
    compilerPlugins = listOf(MyComponentRegistrar())
    commandLineProcessors = listOf(MyCommandlineProcessor())

    inheritClassPath = true
    messageOutputStream = System.out
}.compile()
```

Demo

Kotlin v1.9.21

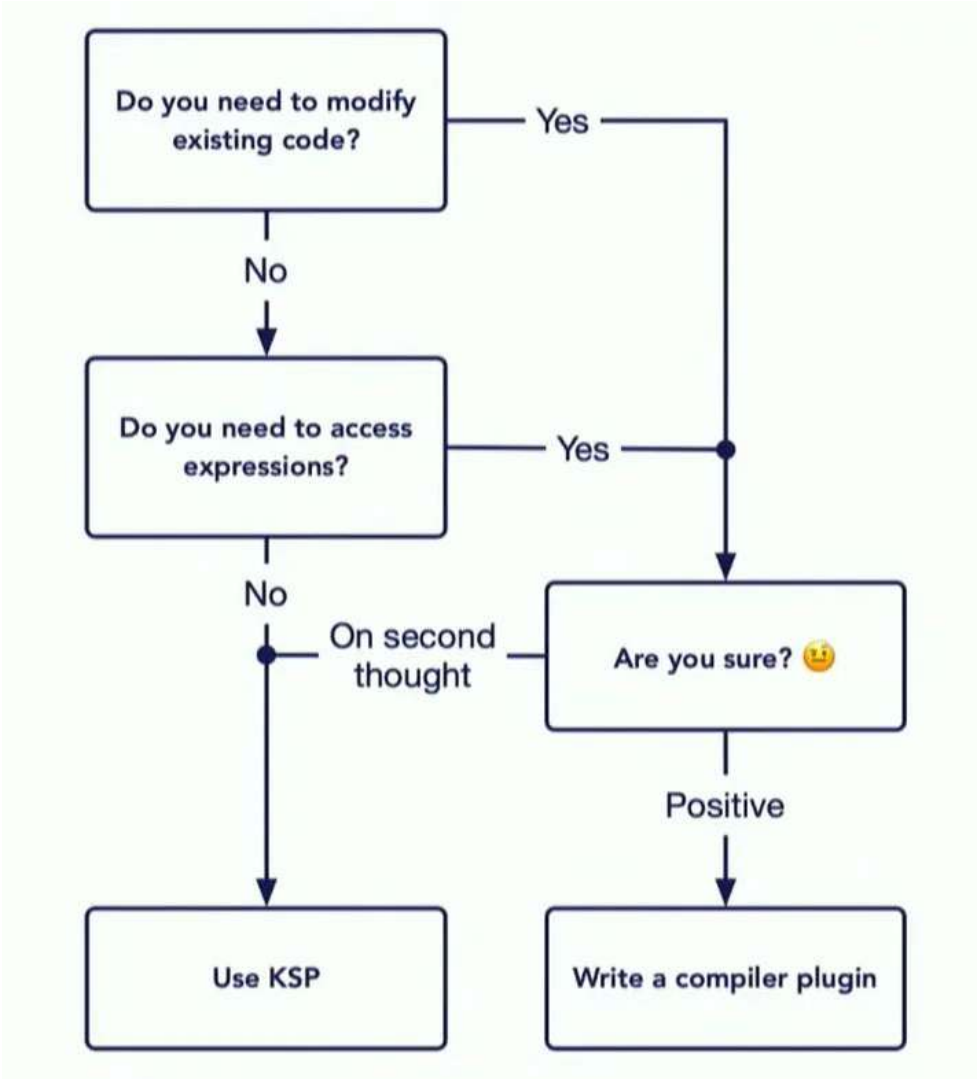
kotlinc

kotlin-compiler-embeddable

開發編譯器插件的挑戰

- 非官方正式 API
- 維護困難：每次Kotlin版本升級都會是一個挑戰
- 與其他插件相容性
- 必要時需客製化添加補丁並重新打包 Kotlin compiler 原始碼
- K2將會是一個重構後的架構，許多K1 plugin將不復存在

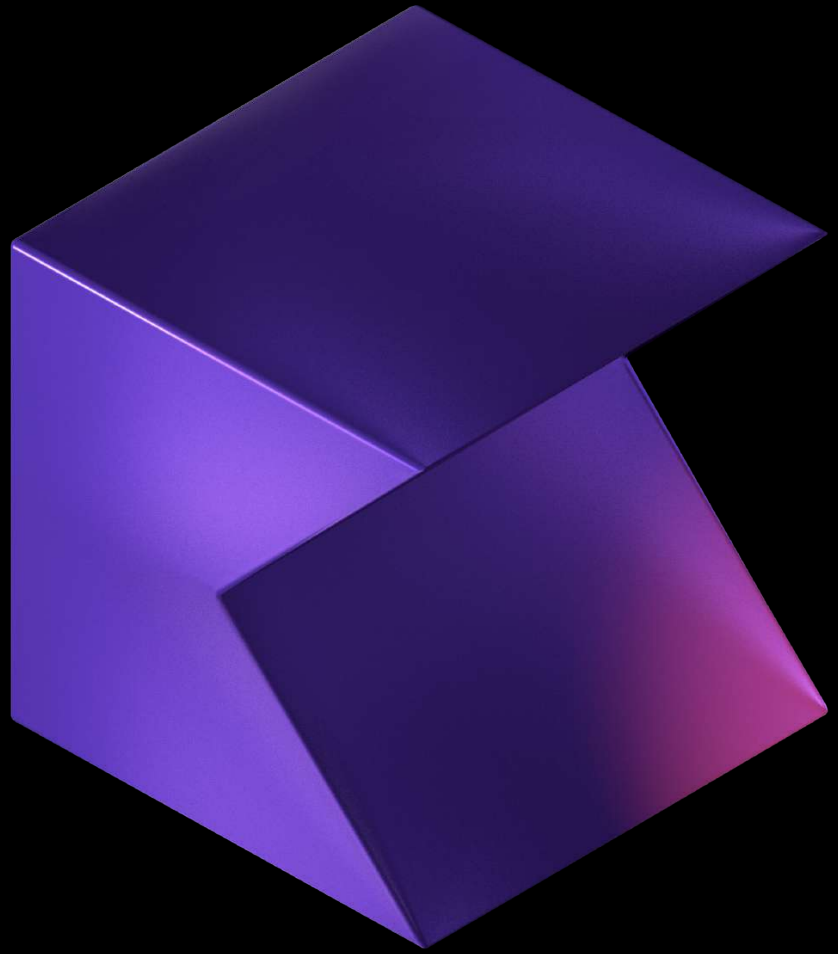
開發前請三思


Do you need to modify existing code?
Yes
No
Do you need to access expressions?
Yes
No
On second thought
Are you sure? 😐
Positive
Use KSP
Write a compiler plugin

推薦閱讀

- Kotlin compiler crash course: Part 1, K1/K2, Talk
- Meta programming with KSP and Kotlin Compiler Plugin: Talk

Q&A



Thanks!
Have a nice Kotlin

@kotlin | Developed by JetBrains