

# KOTLIN

One language to rule them all

# OUTLINE

- Kotlin
  - History
  - Multiplatform
- Demo
- Conclusion



# HISTORY

2011/07 unveiled by JetBrains

2012/02 open sourced (Apache License 2.0)

2016/02 Kotlin v1.0 

2017/05 first-class support for Kotlin on Android @GoogleI/O

2017/11 Kotlin v1.2 with Kotlin/JS

2018/10 Kotlin v1.3 with Kotlin/Native

2019/05 Preferred language for Android development @GoogleI/O



# KOTLIN

- Statically typed
- Type inference
- OOP + Functional
- Coroutines (async programming)
- Cross-platform

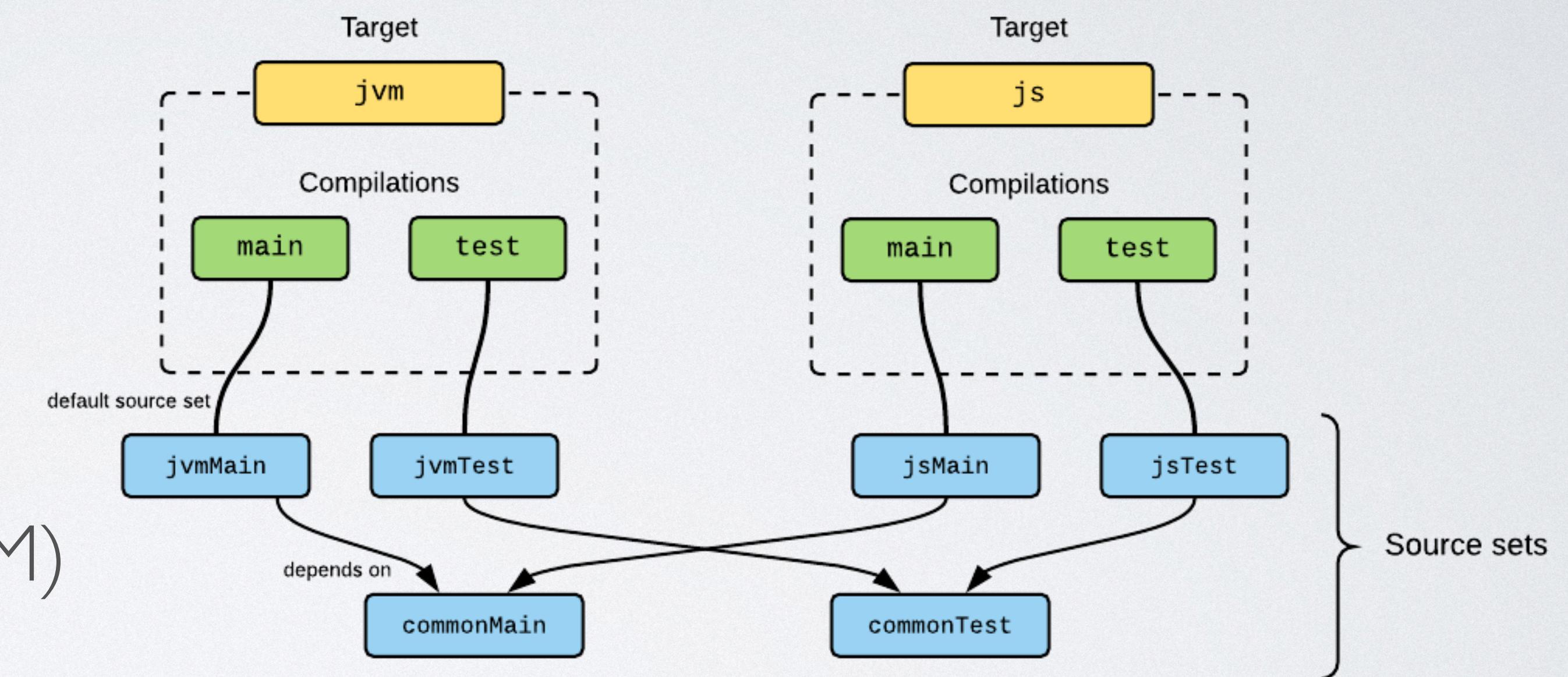


# MULTIPLATFORM

Common code (es. Kotlin stdlib)

+

Platform specific code (es. Kotlin/JVM)



# MULTIPLATFORM

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

```
public final class TestKt {  
  
    // access flags 0x19  
    public final static main([Ljava/lang/String;)V  
        @Lorg/jetbrains/annotations/NotNull;() // in  
        L0  
        ALOAD 0  
        LDC "args"  
        INVOKESTATIC kotlin/jvm/internal/Intrinsics.  
        L1  
        LINENUMBER 5 L1  
        LDC "Hello, World"  
        ASTORE 1  
        L2  
        GETSTATIC java/lang/System.out : Ljava/io/Pr  
        ALOAD 1  
        INVOKEVIRTUAL java/io/PrintStream.println (L  
        L3  
        LINENUMBER 6 L4  
        RETURN  
        L5  
        LOCALVARIABLE args [Ljava/lang/String; L0 L5  
        MAXSTACK = 2  
        MAXLOCALS = 2
```

Kotlin/JVM

```
kotlin.kotlin.io.output.flush();  
if (typeof kotlin === 'undefined'  
    throw new Error("Error loading  
}  
var moduleId = function (_ , Kotlin  
    'use strict';  
    var println = Kotlin.kotlin.io  
    function main(args) {  
        println('Hello, World');  
    }  
    _main_kand9s$ = main;  
    main([]);  
    Kotlin.defineModule('moduleId'  
        return _;  
    }(typeof moduleId === 'undefined'  
        kotlin.kotlin.io.output.buffer;
```

Kotlin/JS

```
<kfun:main(kotlin.Array<kotlin.String>):>  
push  %rbp  
mov   %rsp,%rbp  
mov   $0x4491b0,%edi  
callq 429210 <Kotlin_io_Console_println>  
pop   %rbp  
retq  
  
<EntryPointSelector:>  
push  %rax  
callq 409ea0 <kfun:main(kotlin.Array<kotlin  
pop   %rax  
retq  
nopl  0x0(%rax,%rax,1)  
  
<kfun:kotlin.Unit.toString()Reference:>  
push  %rbp  
mov   %rsp,%rbp  
mov   %rsi,%rax  
mov   $0x4491f0,%esi  
mov   %rax,%rdi  
callq 4252f0 <UpdateReturnRef>  
mov   $0x4491f0,%eax  
pop   %rbp  
retq  
nopl  0x0(%rax,%rax,1)
```

Kotlin/Native



# DEMO TIME



# CONCLUSION

- Pros
  - Different approach vs full native
  - Faster development & more robust/maintainable code
  - Consistency between platforms
- Cons
  - Different approach vs full native
  - IDE
  - Immature (but fast improving)



# REFERENCES

## Kotlin / Kotlin MP

- [kotlinlang](#)
- [Kotlin Koans](#)
- [Building Multiplatform Projects with Gradle](#)

## Kotlin MP libraries

- [khttp](#)
- [ktor](#)
- [kotlinx.serialization](#)
- [klock](#)

