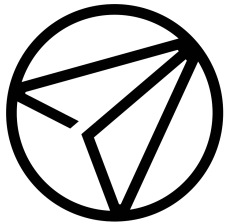
Snake, Kotlin & Multiplatform

What is this?

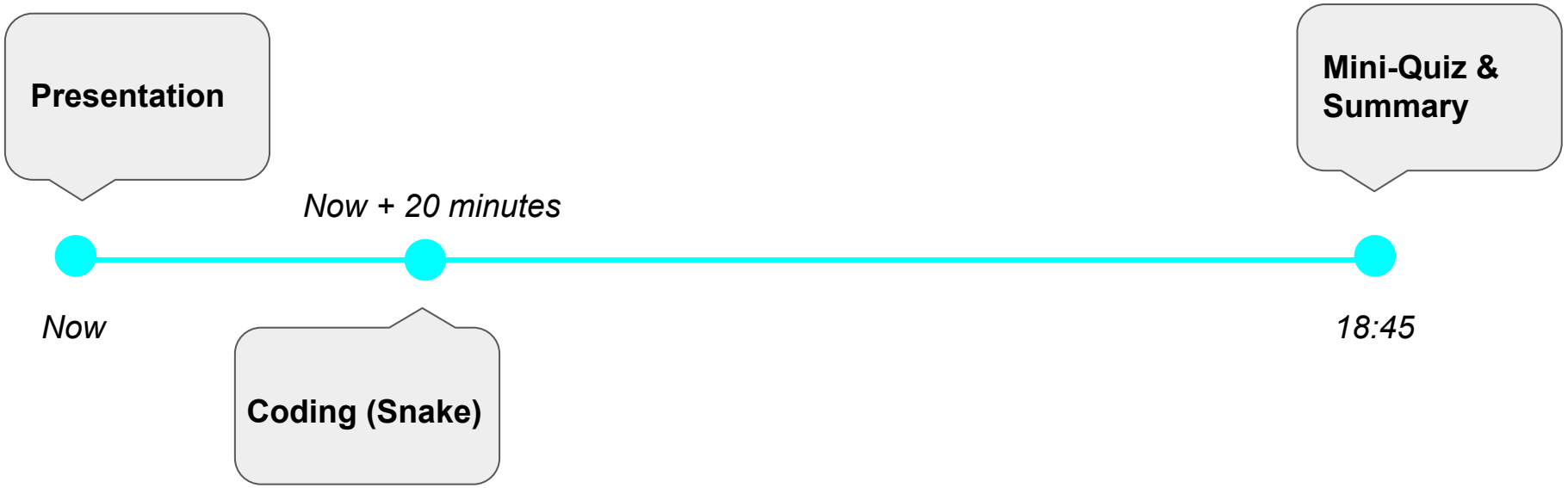


AFRY

ÅF PÖYRY

github.com/Lundez

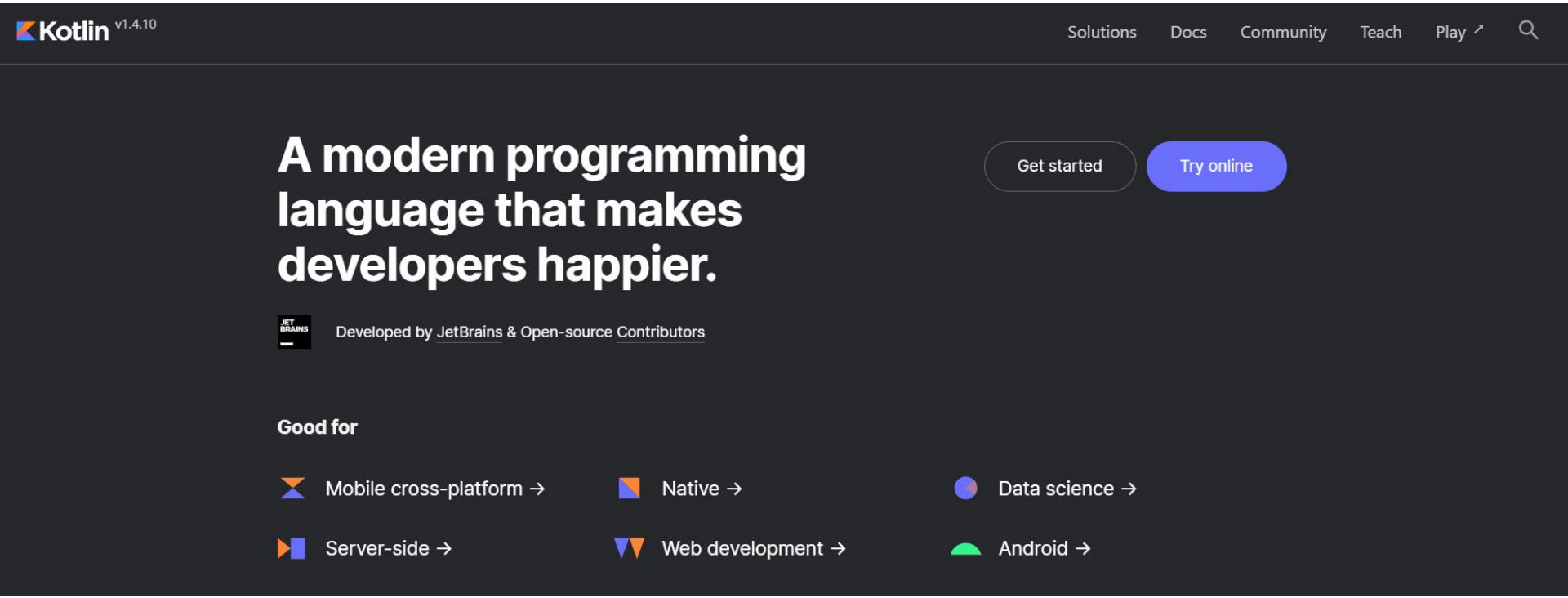
github.com/londogard



Approximate Timeline

Lightning Talk

Disclaimer
Partly inspired by a talk from Öredev by @dkandalov


Kotlin v1.4.10
Solutions
Docs
Community
Teach
Play ↗
A modern programming language that makes developers happier.
Get started
Try online
Developed by JetBrains & Open-source Contributors
Good for
Mobile cross-platform →
Native →
Data science →
Server-side →
Web development →
Android →

Good for

 Mobile cross-platform →

 Native →

 Data science →

 Server-side →

 Web development →

 Android →

Kotlin/Native for Native

Edit Page

Kotlin/Native is a technology for compiling Kotlin code to native binaries, which can run without a virtual machine. It is an LLVM based backend for the Kotlin compiler and native implementation of the Kotlin standard library.

Kotlin/Native

- Compile to native binaries

Kotlin/Native

- Compile to native binaries
- LLVM backend for Kotlin Compiler

Kotlin/Native

- Compile to native binaries
- LLVM backend for Kotlin Compiler
- Runtime implementation

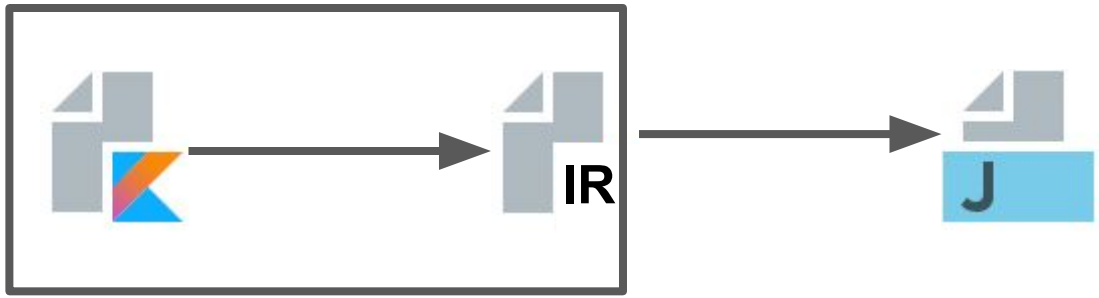
Kotlin/Native

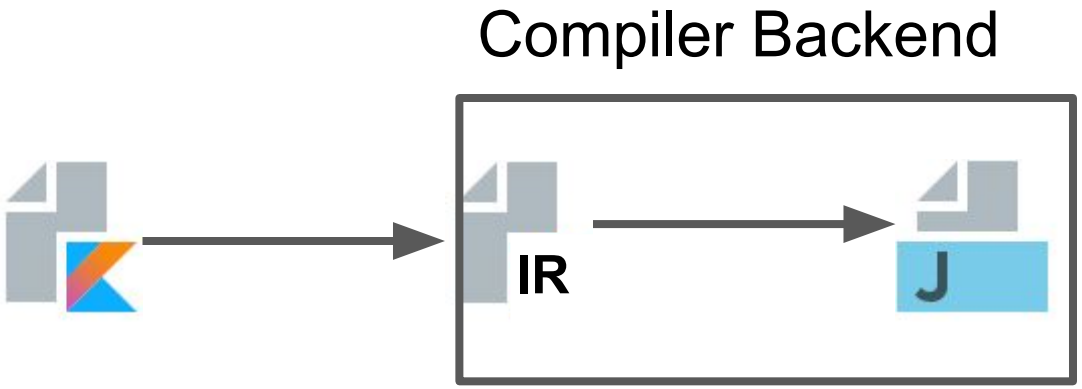
- Compile to native binaries
- LLVM **backend** for Kotlin Compiler
- Runtime implementation

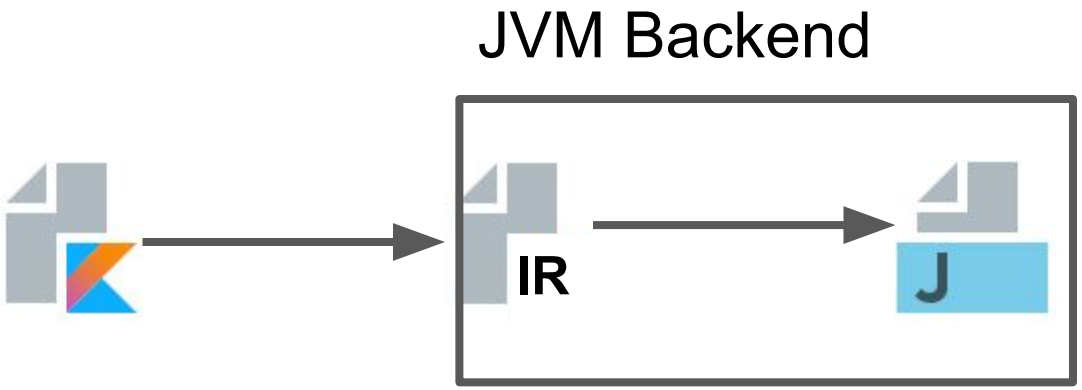

IR

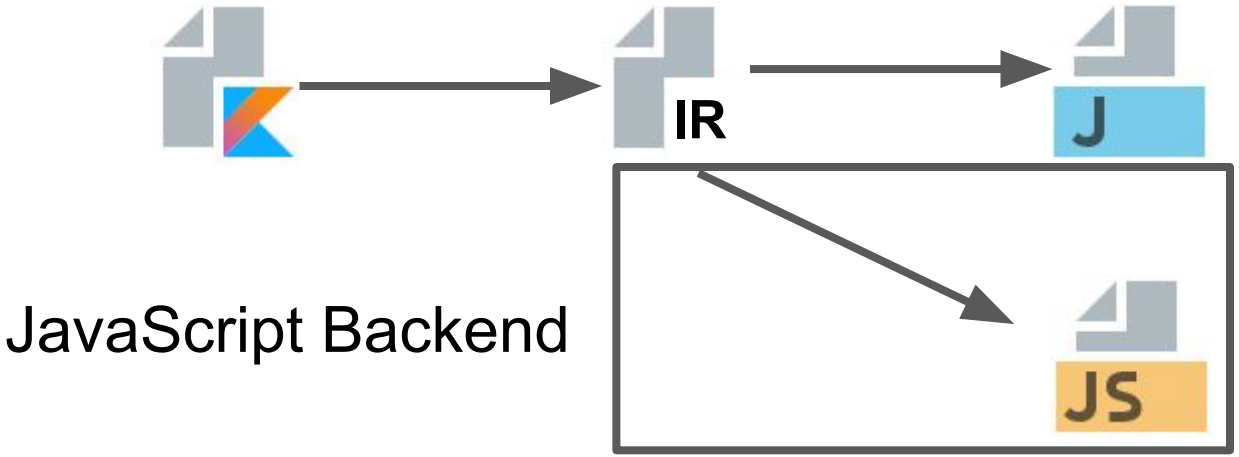

IR
J

Compiler Frontend

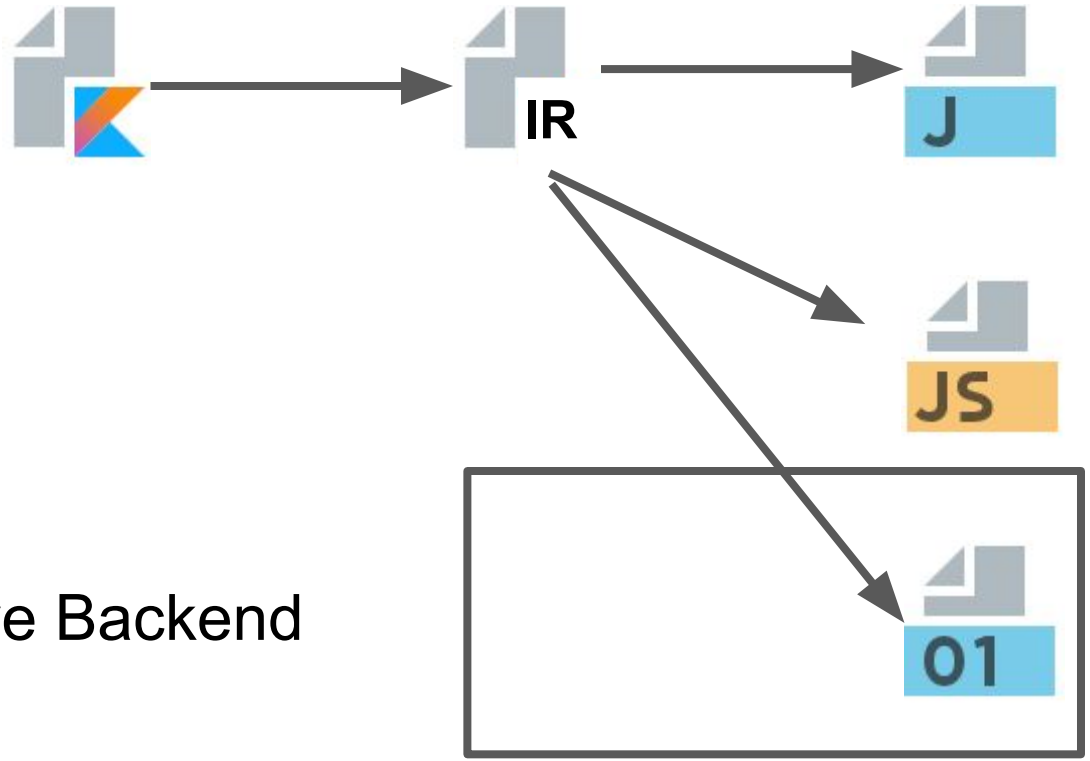



Compiler Backend
IR
J

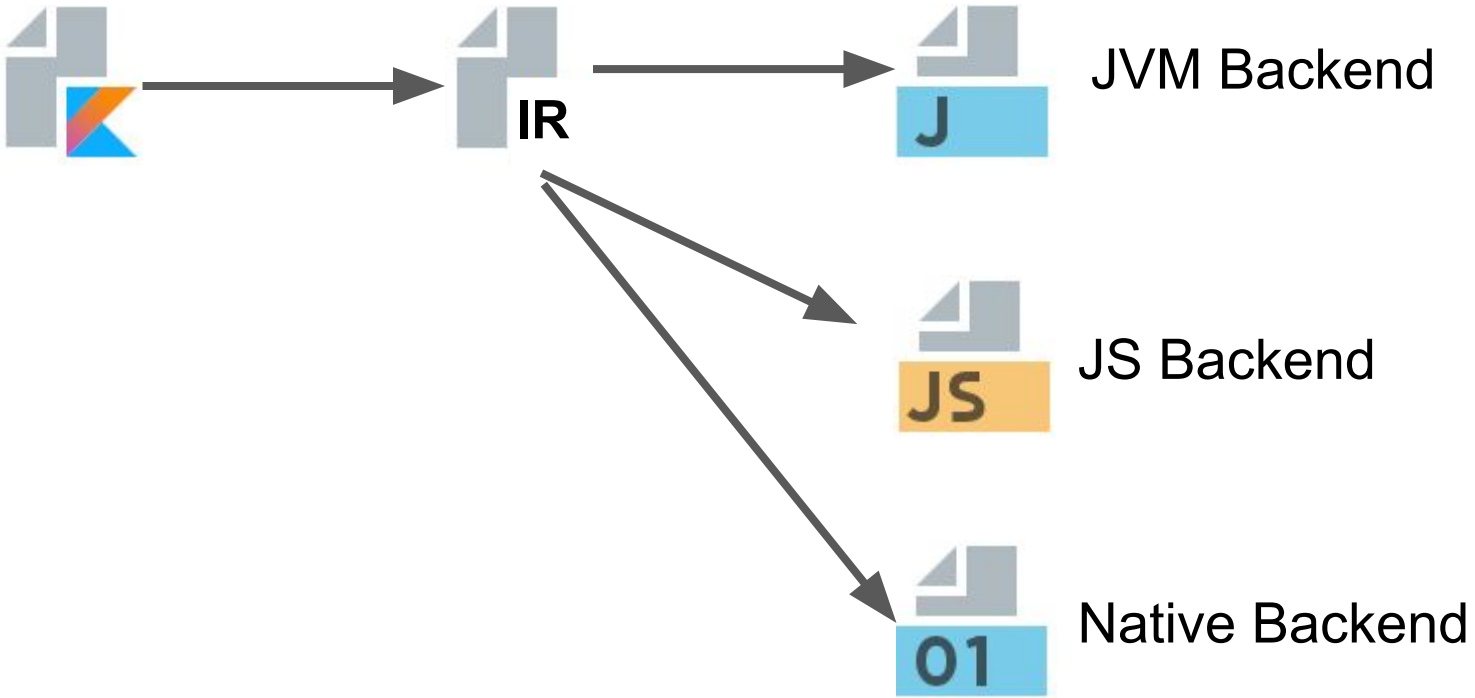

JVM Backend
IR
J



JavaScript Backend



Native Backend


IR
J
JVM Backend
JS
JS Backend
01
Native Backend

Kotlin/Native

- Compile to native binaries
- **LLVM** backend for Kotlin Compiler
- Runtime implementation

LLVM

The **LLVM** Compiler Infrastructure

Site Map:

Download!

Download now:
LLVM 10.0.1
All Releases
APT Packages
Win Installer
Pre-releases

View the open-source **license**

LLVM Overview

The LLVM Project is a collection of modular and reusable compiler and toolchain technologies. Despite its name, LLVM has little to do with traditional virtual machines. The name "LLVM" itself is not an acronym; it is the full name of the project.

LLVM began as a research project at the University of Illinois, with the goal of providing a modern, SSA-based compilation strategy capable of supporting both static and dynamic compilation of arbitrary programming languages. Since then, LLVM has grown to be an umbrella project consisting of a number of subprojects, many of which are being used in production by a wide variety of commercial and open source projects as well as being widely used in academic research. Code in the LLVM project is licensed under the "Apache 2.0 License with LLVM exceptions"

The primary sub-projects of LLVM are:

1. The **LLVM Core** libraries provide a modern source- and target-independent optimizer, along with code generation support for many popular CPUs (as well as some less common ones!) These libraries are built around a well specified code representation known as the LLVM intermediate representation ("LLVM IR"). The LLVM Core libraries are well documented, and it is particularly easy to invent your own language (or port an existing compiler) to use LLVM as an optimizer and code generator.

2. **Clang** is an "LLVM native" C/C++/Objective-C compiler, which aims to deliver amazingly fast compiles, extremely useful error and warning messages and to provide a platform for building great source level tools. The Clang Static Analyzer and clang-tidy are tools that automatically find bugs in your code, and are great examples of the sort of tools that can be built using the Clang frontend as a library to parse C/C++ code.

3. The **LLDB** project builds on libraries provided by LLVM and Clang to provide a great native debugger. It uses the Clang ASTs and expression parser, LLVM JIT, LLVM disassembler, etc so that it provides an experience that "just works". It is also blazing fast and much more memory efficient than GDB at loading symbols.

Latest LLVM Release!

6 August 2020: LLVM 10.0.1 is now **available for download**! LLVM is publicly available under an open source License. Also, you might want to check out **the new features** in Git that will appear in the next LLVM release. If you want them early, download LLVM through anonymous Git.

ACM Software System Award!

LLVM has been awarded the **2012 ACM Software System Award**! This award is given by ACM to *one* software system worldwide every year. LLVM is in highly distinguished company! Click on any of the individual recipients' names on that page for the detailed citation describing the award.

GitHub Migration

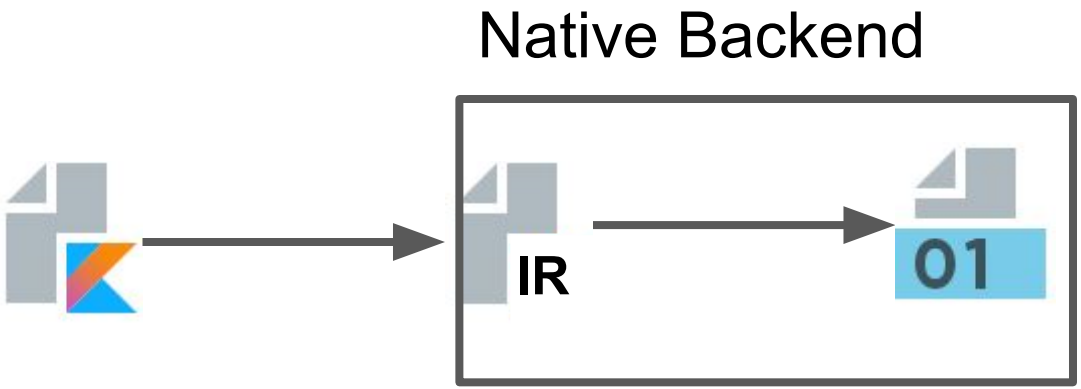
Completed! Many thanks to all the participants

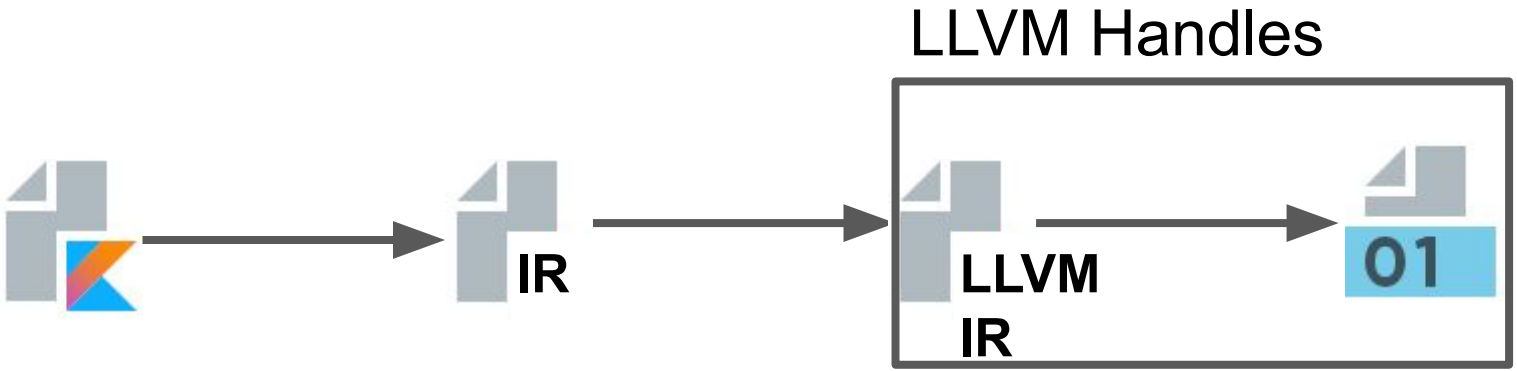
Upcoming Releases

LLVM Release Schedule:

- 10.0.1:
  - May 18: 10.0.1-rc1

Rust
C/C++ (clang)
Swift
Julia
& more

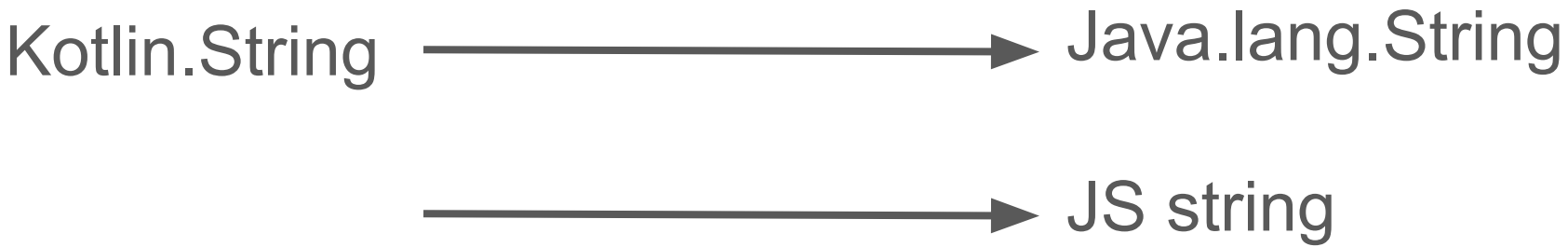

Native Backend
IR
01

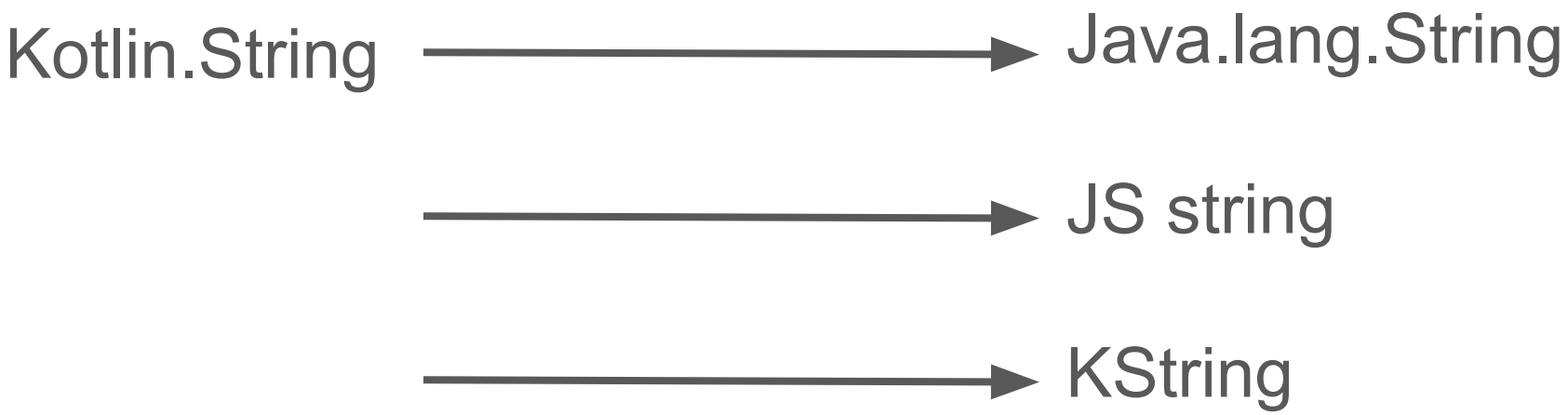

LLVM Handles
IR
LLVM
IR
01

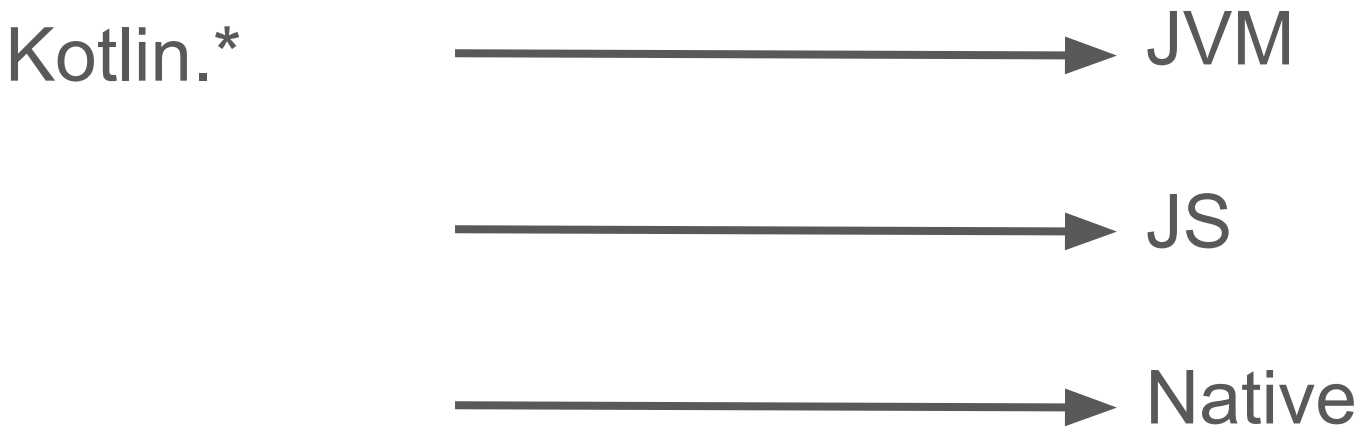
Kotlin/Native

- Compile to native binaries
- LLVM backend for Kotlin Compiler
- **Runtime implementation**


Kotlin.String
Java.lang.String


Kotlin.String
Java.lang.String
JS string


Kotlin.String
Java.lang.String
JS string
KString


Kotlin.*
JVM
JS
Native

Reference Counting Native Specific Code

Why Multiplatform?

- Time

- Time
- Resources

- Time
- Resources
- Delightful

What now?

Two groups

Experienced

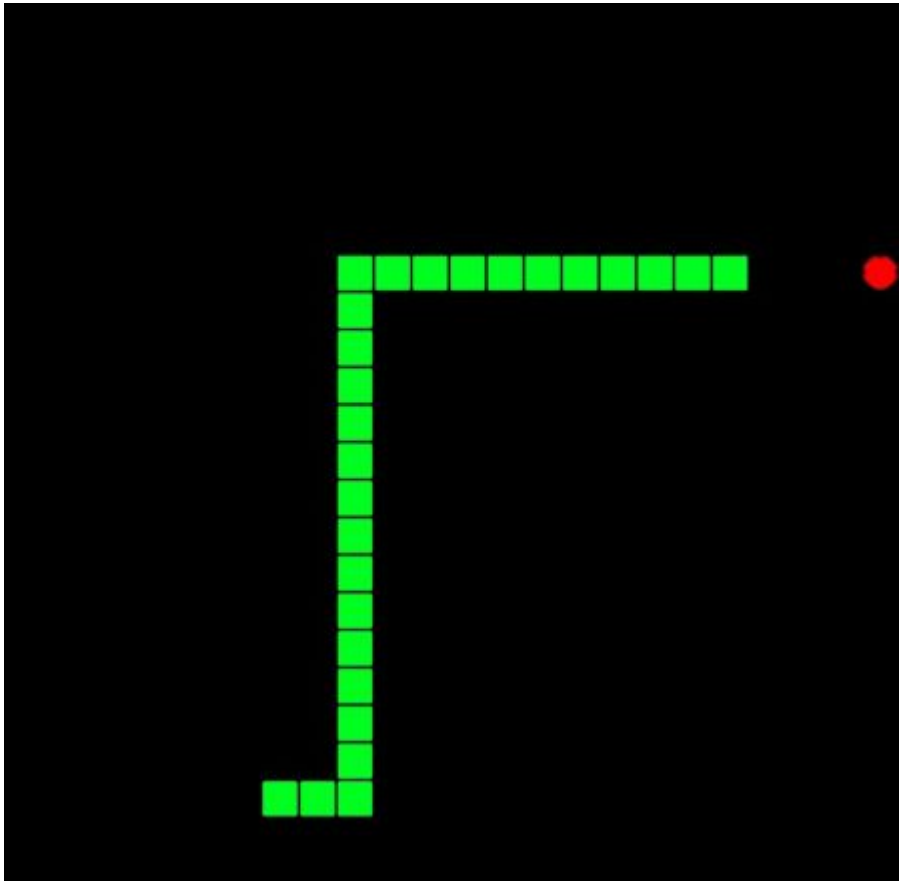
- Work in your own pace
- Together, or with friends
- **Dennis will help when you ask**

Unexperienced

- Live-coding together
- Slower pace / more listening
- **Hampus will help and live code**

Editor of choice

- **User Experience Ranking**
- Reverse for ease of installation



https://github.com/londogard/snake-js-jvm-native

Diskussionspunkter … Men även för nybörjare

- Compiler (KString etc)
- Varför inte alltid Native
- Val, Var & Mutability
- inline (?)
- Hur kan vi göra det bättre?
- `object : KeyAdapter()`
- Sequence, lazy vs non-lazy -- When to Use Sequences - Dave Leeds on Kotlin (typealias.com)