# Kotlin JVM language and compiler

## CONCEPTS

Types are non-null by default: String cannot hold null, String? can.

**hello.kt** source
↓ kotlinc
**hello.jar** bytecode
↓ -include-runtime
**JVM** java -jar
↓
**Output** stdout

## COMPILER AND RUNNER

**kotlinc** *<file>*.**kt -d** *<dir>*
Compile to class files

**kotlinc** *<file>*.**kt -include-runtime -d** *<out>*.**jar**
Self-contained jar

**java -jar** *<out>*.**jar**
Run a self-contained jar

**kotlinc -script** *<file>*.**kts**
Run a Kotlin script

**kotlinc -cp** *<path>* *<file>*.**kt**
Set the classpath

**kotlinc -version**
Print compiler version

## VARIABLES AND TYPES

**val** *<n>*: *<T>* = *<v>*
Read-only value

**var** *<n>* = *<v>*
Mutable variable, inferred type

**"Hi $**<name>** ${**<expr>**}"**
String template

*<x>* **is** *<T>*
Type check with smart cast

*<a>*..*<b>* / *<a>* **until** *<b>*
Inclusive / exclusive range

## NULL SAFETY

*<T>***?**
Nullable type

*<x>***?.***<member>*
Safe call

*<x>* **?:** *<default>*
Elvis: fallback if null

## NULL SAFETY (cont.)

*<x>***?.let {** *<block>* **}**
Run only if not null

## FUNCTIONS AND LAMBDAS

**fun** *<n>*(*<p>*: *<T>* = *<d>*): *<R>* **{ }**
Function with default value

**fun** *<n>*(*<p>*: *<T>*) = *<expr>*
Expression body

**{** *<x>* → *<expr>* **}**
Lambda

**it**
Implicit single lambda parameter

**fun** *<T>*.*<n>*(): *<R>*
Extension function

## CLASSES AND OBJECTS

**class** *<N>*(**val** *<p>*: *<T>*)
Class with primary constructor

**data class** *<N>*(**val** *<p>*: *<T>*)
equals, copy, toString for free

## CONTROL FLOW

**if** (*<c>*) *<a>* **else** *<b>*
Conditional expression

**when** (*<x>*) **{** *<v>* → *<r>*; **else** → *<d>* **}**
Pattern-style switch

**for** (*<i>* **in** *<range>*) **{ }**
Loop over range or iterable

## COLLECTIONS

**listOf**(*<a>*, *<b>*)
Read-only list

**mutableListOf**(*<a>*)
Mutable list

**mapOf**(*<k>* **to** *<v>*)
Read-only map

*<c>***.map {** *<expr>* **}**
Transform each element

*<c>***.filter {** *<cond>* **}**
Keep matching elements

## COROUTINES

**suspend fun** *<n>*()
Function that can suspend