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

Most Kotlin projects build through Gradle; kotlinc is for single files and scripts.

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

kotlin -classpath *<out>*.jar *<Name>*Kt
Run a main class

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

kotlinc -Xrepl
Interactive shell

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

<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

<obj>.copy(*<p>* = *<v>*)
Copy with one field changed

object *<N>* { }
Singleton

companion object { }
Class-level members

sealed class *<N>* / **sealed interface** *<N>*
Closed type hierarchy

CONTROL FLOW

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

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

CONTROL FLOW (cont.)

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

<c>.first { *<cond>* } / *<c>*.firstOrNull { *<cond>* }
Find element

SCOPE FUNCTIONS

<x>.let { *<it>* → *<r>* }
Transform, returns lambda result

<x>.apply { *<this>* }
Configure, returns object

COROUTINES

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

runBlocking { *<block>* }
Bridge into blocking code

launch { *<block>* }
Start a coroutine, no result

async { *<expr>* }.await()
Start and await a result

RECIPES

Hello world as a jar

```
# hello.kt
fun main() = println("Hello")

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

Quick script

```
# list.kts
java.io.File(".").listFiles()?.
forEach { println(it.name) }

kotlinc -script list.kts
```