Kotlin for Android

Writing Android Apps in Kotlin
Contact Info

Ken Kousen
Kousen IT, Inc.
ken.kousen@kousenit.com
http://www.kousenit.com
http://kousenit.org (blog)
@kenkousen
Videos

O'Reilly video courses: See Safari Books Online for details

Groovy Programming Fundamentals
Practical Groovy Programming
Mastering Groovy Programming
Learning Android
Practical Android
Gradle Fundamentals
Gradle for Android
Spring Framework Essentials
Advanced Java Development
Kotlin

JetBrains created and maintains the language

Provides null safety at the compiler level

Statically typed and statically bound by default

Runs on the JVM → Clean interoperability with Java
Kotlin

Home page is https://kotlinlang.org

Many code simplifications borrowed from other languages

  Closures similar to Groovy

  Typing similar to Scala

  Co-routines similar to .Net (and others)
Kotlin

Officially endorsed by Google as an Android development language

Android Studio is the official IDE for Android

Kotlin is a plugin for both Android Studio and IntelliJ IDEA

JetBrains supports an Eclipse plugin as well
Learning Kotlin

http://try.kotlinlang.org/ → online script engine

Kotlin Koans → https://kotlinlang.org/docs/tutorials/koans.html

Get complex fairly quickly (don't be discouraged :)

Kotlin reference → https://kotlinlang.org/docs/reference/

Kotlin idioms → https://kotlinlang.org/docs/reference/idioms.html

Demonstrates good practices and usage patterns
Kotlin for Android

Book: [Kotlin for Android Developers](https://github.com/antoniolg/Kotlin-for-Android-Developers)

LeanPub, Antonio Leiva

GitHub repo:

[https://github.com/antoniolg/Kotlin-for-Android-Developers](https://github.com/antoniolg/Kotlin-for-Android-Developers)
Udacity Course

Kotlin for Android Developers

https://www.udacity.com/course/kotlin-for-android-developers--ud888
Basic Syntax

Types declared after the variable, separated by a colon

    var s : String

var and val define types

    var is a variable (mutable)

    val is a value (immutable, i.e., final)
Basic Syntax

Variables are non-null by default

Must declare nullable types using "?"

```kotlin
val s : String?
```

Implies "s" can be assigned null; not true otherwise
Data Classes

Classes defined using the keyword "data"

```kotlin
data class Customer(val name: String, val email: String)
```

(That's the entire class)

Data classes have:

- generated getters and setters
- `toString`, `equals`, `hashCode`
- `copy()` method
Functions

Functions defined with the "fun" keyword

```kotlin
fun main(args: Array<String>) { ... }
```

If function consists of one statement, can use assignment

```kotlin
fun sayHello(name: String) = println("Hello, $name!")
```

(note: semicolons not needed)
Functions

Return type shown after signature

```kotlin
fun sum(a: Int, b: Int): Int {
    return a + b
}
```

Simpler:

```kotlin
fun sum(a: Int, b: Int) = a + b
```

Return type inferred

(Use "Unit" return type for Java "void")
Functions

Support default parameters

```kotlin
fun read(b: Array<Byte>, off: Int = 0, len: Int = b.size) {
    ...
}
```

Override defaults by supplying actual values
Functions

Can use named parameters

```kotlin
fun reformat(str: String, normalizeCase: Boolean = true,
             upperCaseFirstLetter: Boolean = true,
             divideByCamelHumps: Boolean = false,
             wordSeparator: Char = ' ') {
    ...
}
```

reformat(str, normalizeCase = true,
          upperCaseFirstLetter = true,
          divideByCamelHumps = false, wordSeparator = '_')
"if" clause returns value automatically

val max = if (a > b) a else b

Acts like Java ternary operator (which isn't supported)
Like a Java switch statement with a return

```java
when (x) {
    1 -> print("x == 1")
    2 -> print("x == 2")
    else -> {
        print("x is neither 1 nor 2")
    }
}
```
when

Works with many options, including ranges

when (x) {
  in 1..10 -> print("x is in the range")
  in validNumbers -> print("x is valid")
  !in 10..20 -> print("x is outside the range")
  else -> print("none of the above")
}
when

when expressions also return a value

If you return a value, conditionals must be exhaustive

Either cover all cases, or include an else clause

fun modulo3(arg: Int) = when (arg % 3) {
    0 -> "$arg divisible by 3"
    1 -> "$arg % 3 == 1"
    2 -> "$arg % 3 == 2"
    else -> "Houston, we have a problem..."
}
for

Traditional Java for loop not supported

Use for-in loop

for (item in collection) print(item)

for (item: Int in ints) {
    // ...
}

Looping over arrays, using indices

```kotlin
for (i in array.indices) {
    print(array[i])
}
```

Looping over maps, use "destructuring"

```kotlin
for ((index, value) in array.withIndex()) {
    println("the element at $index is $value")
}
```
Elvis operator

Can use ?: as in Groovy

If value is not null, use it, otherwise default

val s = person.name ?: "World"
Lambdas

Kotlin supports lambda expressions

```kotlin
max(strings, { a, b -> a.length < b.length })
```

Lambda contained within {}

```kotlin
max(strings) { a, b -> a.length < b.length }
```

Can place lambda after parentheses in method call
Lambdas

Basic syntax:

```kotlin
val sum = { x: Int, y: Int -> x + y }
```

Can declare return type (optional here)

```kotlin
val sum: (Int, Int) -> Int = { x, y -> x + y }
```

If single argument, default is "it"

```kotlin
ints.filter { it > 0 }
```
Lambdas

Like Java, lambdas can access variables in scope

Unlike Java (but like Groovy), it can modify them

```java
var sum = 0
ints.filter { it > 0 }.forEach {
    sum += it
}
print(sum)
```

Note: don't do it this way → use `sum()` instead
Classes and Objects

Classes are defined as usual

Don't need "new" to instantiate

val customer = Customer("Fred", "flintstone@slatequarry.com")
Classes and Objects

To extend, class must be declared "open"

Functions must also have "open" or you can't override them

```kotlin
open class Base {
    open fun v() {}  // open function
    fun nv() {}      // regular function
}
class Derived() : Base() {
    override fun v() {}  // override the open function
}
```
Classes and Objects

Kotlin does not support static members

Use "object" and companion objects instead

```kotlin
object DataProviderManager {
    fun registerDataProvider(provider: DataProvider) {
        // ...
    }
}
```

Result is a singleton
Companion objects are singletons inside classes → home for statics

class MyClass {
    companion object {
        fun create(): MyClass = MyClass()
    }
}

val instance = MyClass.create()
Classes and Objects

Note default access for everything is public

Also can put functions inside a file without a class

Become part of the generated class
Extension functions

Can add methods to existing classes

Good for optional methods

```kotlin
fun MutableList<Int>.swap(index1: Int, index2: Int) {
    val tmp = this[index1]
    this[index1] = this[index2]
    this[index2] = tmp
}
```

"MutableList" is class, "swap" is added method; "this" is instance
Sequences

Methods like "map", "filter" are added to collections

The "asSequence()" method converts collection to sequence

Like Java streams

Evaluated element at a time

No data processed unless there is a terminal expression
Anko Library

Extension library for Android

https://github.com/Kotlin/anko

Wiki has usage info
KTX

Kotlin extensions provided by Google

https://github.com/android/android-ktx

Blog post:

For more information

See reference at kotlinlang.org, but also:

https://github.com/JetBrains/kotlin-workshop

Two-day workshop

Presentations are on slideshare.net (linked in GitHub repo)

e.g., https://speakerdeck.com/svtk/1-intro-kotlin-workshop
GitHub Repository

https://github.com/kousen/HelloKotlinAndroid

App consumes RESTful web service

Works with Sqlite database

Converts JSON data to Kotlin data classes

Operates asynchronously using Anko extension library