KOTLIN

FOR ANDROID DEVELOPERS

Learn Kotlin the easy way while developing an Android App

6th edition
Updated to Kotlin 1.2.31

Antonio Leiva

Kotlin for Android Developers

Learn Kotlin the easy way while developing an Android App

Antonio Leiva

This book is for sale at http://leanpub.com/kotlin-for-android-developers

This version was published on 2018-04-05

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Tweet This Book!

Please help Antonio Leiva by spreading the word about this book on Twitter!

The suggested hashtag for this book is #kotlinandroiddev.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

#kotlinandroiddev

This book is dedicated to all the loyal readers of antonioleiva.com, who made me believe that writing about Android development was a powerful tool to help others learn about it. I felt that this book was a necessary step forward.

Special mention goes to Luis Herrero, who designed the excellent cover of this book, and to Gautier Mechling for helping me so much by reviewing this book. It is thanks to him that these pages are not full of typos and mistakes.

And, of course, this is specially dedicated to you. With your support and your help, this book is growing and becoming a reference. So any suggestions to improve the quality of this book will be welcomed. Feel free to write anytime to contact@antonioleiva.com.

Contents

I. About this book

Hey! Thanks so much for your interest in this book, I am thrilled that you decided to become a Kotlin for Android expert.

Many things have happened since I started writing "Kotlin for Android Developers" in mid-2015. At that point, Kotlin was still in an early access edition. However, I felt so much power in the language, that I decided to explain everything that I was learning in a book.

Since then, the language has kept growing to the point that Google announced that they would start supporting Kotlin as an official language to develop Android apps.

These were amazing news. But it was even more amazing that both Google and JetBrains suggested this book to Android developers who want to learn the language.

As you may know, this is a lean publication. This book grew and progressed thanks to the readers' comments. So I can only thank you for helping me bring this dream to life.

What is "Kotlin for Android Developers" about

Throughout this book, I create an Android app from the ground up using Kotlin as the primary language. The idea is to learn the language by example, instead of following a regular reference book structure. I will be stopping to explain the most useful concepts and ideas about Kotlin, comparing them to Java 6. This way, you can see what the differences are and which parts of the language can help you speed up your work.

This book is not meant to be a language reference, but a tool for Android developers to learn Kotlin and be able to continue with their projects by themselves. The examples are meant to overcome the most recurring problems we have to face in our daily lives as app developers, by making use of Kotlin's expressiveness and some other exciting

tools and libraries. However, this text covers most of the features of Kotlin, so by the end of the reading, you will have an in-depth knowledge of the language.

The content is very practical, so I recommend that you follow the examples and the code in front of a computer and try everything it suggests. You could, however, take one first read to get a broad idea and then dive into practice.

Even though this book now finished, I will review it from time to time to keep it up to date with new Kotlin versions. So feel free to write and tell me what you think about the book, or what could be improved. I want this book to be the perfect tool for Android developers, and as such, help and ideas will be welcomed.

If you are reading a printed copy and want to receive the latest updates, please feel free to email me at *contact@antonioleiva.com* with a proof of purchase, and I will give you access to the digital copy. That way, you will keep receiving the updates. Do the same if you got it from any bookstores (Kindle, Kobo…) and you want to get access to the PDF version.

Thanks for becoming part of this exciting project!

II. Is this book for you?

This book is written to be useful to Android developers who are interested in learning the Kotlin language.

This book is for you if you are in some of the following situations:

- You have some basic knowledge of Android Development and the Android SDK, as well as the Java language.
- You want to learn how to develop Android apps using Kotlin by following an example.
- You need a guide on how to solve many of the common challenges Android developers face every day, by using a cleaner and more expressive language.

On the other hand, this book may not be for you. These are the topics that you will not find in it:

- The content of these pages is not a Kotlin Bible. I shall explain all language basics, and even more complicated ideas when they come out during the process, just when we need them. So you will learn by example and not the other way round.
- I will not explain how to develop an Android app. You do not need a deep understanding of the platform, but at least some basics, such as some knowledge of Android Studio, Gradle, Java programming and Android SDK. You may even learn some new Android things in the process!
- The book is not a guide to functional programming. Of course, I am showing what you need, as Java 6 is not functional at all, but I will not dive deep into the programming paradigm.

III. About the author

Antonio Leiva is an Android Engineer who spends time learning about new ways to get the most out of Android and then writes about it. He writes a blog at antonioleiva.com[1], focused on helping other Android developers learn Kotlin.

He also leads intensive live workshops, where the attendants put all the content from this book into practice. In 10 hours, people steps from no Kotlin knowledge to being able to create their apps from scratch.

Antonio started as a consultant in CRM technologies, but after some time, looking for his real passion, he discovered the Android platform. After getting some experience, he started a new adventure at a mobile company, where he led several projects for well-known companies in Spain.

He currently works as an Android Engineer at Plex[2], where he also plays an essential role in the design and UX of the Android applications.

You can find Antonio on Twitter as @lime_cl[3] or Google+ as +AntonioLeivaGordillo[4].

[1] http://antonioleiva.com

[2] http://plex.tv

[3] https://twitter.com/lime_cl

[4] http://plus.google.com/+AntonioLeivaGordillo‘

1 Introduction

Things are changing for good for Android Developers. In Google I/O 2017, the Android team announced that Kotlin was becoming an official language to develop Android apps.

This means that, while it is still possible to develop Android apps using Java, from now on Kotlin is fully supported, and Google is making sure that every new Android feature, the framework, the IDE and all their libraries work seamlessly with the new language.

Google listened to the community, who was asking for years that Kotlin became a first-party language. So you can now take advantage of all the features of a modern language while developing for Android.

Throughout this book, I will show you how, so I hope that I can help you understand the various ways that Kotlin can take you one step ahead and make your code much better.

However, before diving into the features of the language, let me tell you just a little bit of background.

1.1 What is Kotlin?

Kotlin is language developed by JetBrains[5], a company known for building the IntelliJ IDEA, a powerful IDE for Java development. Android Studio, the official Android IDE, is based on IntelliJ. It was initially implemented to run on the Java Virtual Machine.

JetBrains designed Kotlin with Java developers in mind, and with IntelliJ as its primary development IDE. These two factors are breaking points that made Android developers quickly adopt the language:

[5] https://www.jetbrains.com/

- **Kotlin is very intuitive and easy to learn for Java developers**. Most parts of the language are very similar to what we already know, and the differences can be mastered in no time.
- **We have total integration with our daily IDE for free**. Android Studio can understand, compile and run Kotlin code. Moreover, the support for this language comes from the company who develops the IDE, so we Android developers are first-class citizens.

However, this is only related to how the language integrates with our tools. What are the advantages of the language when compared to Java 6?

- ** It is more expressive**: this is one of its main points. You can write more with much less code.
- ** It is safer**: Kotlin is null safe, which means that we deal with possible null situations at compile time, to prevent execution time exceptions. We need to specify that an object can be null explicitly, and then check its nullity before using it. You can save much time debugging null pointer exceptions and fixing nullity bugs.
- ** It is functional**: Kotlin is fundamentally an object-oriented language, not a pure functional language. However, like many other modern languages, it uses many concepts from functional programming, such as lambda expressions, to solve some problems more naturally. Another nice feature is the way it deals with collections.
- **It makes use of extension functions**: This means we can extend any class with new features even if we do not have access to the source code.
- ** It is highly interoperable**: You can continue using most libraries and code written in Java because the interoperability between both languages is excellent. It is even possible to create mixed projects, with both Kotlin and Java files coexisting.

However, this is only the tip of the iceberg: - Since Kotlin 1.1, the final version of Kotlin JS[6] was released. This new variant allows you to develop web apps using

[6] https://kotlinlang.org/docs/reference/js-overview.html

Kotlin. - Since Kotlin 1.2, you can also create multiplatform projects[7]. With it, you can share code between JVM and Javascript. - As an experimental feature, the JetBrains team has also released Kotlin/Native[8], a project that finally takes Kotlin out of the JVM. In the future, we will be able to implement the server, the web and the Android and iOS Apps using Kotlin for most of the code base. Also, you can try it today. - Gradle is adding support to use Kotlin Script (a simplified version of Kotlin) to write Gradle files instead of Groovy. The project is quickly approaching its 1.0 release.

So you can see that the future of Kotlin is pretty promising. Learning Kotlin can become the language of reference in many other platforms, and sharing code among all them is undoubtedly a high selling point.

1.2 What do we get with Kotlin?

Without diving too deep into the Kotlin language (we will learn everything about it throughout this book), these are some interesting features we miss in Java:

Expressiveness

With Kotlin, it is much easier to avoid boilerplate because the language covers the most common patterns by default. For instance, in Java, if we want to create a data class, we need to write (or at least generate) this code:

```
1  public class Artist {
2      private long id;
3      private String name;
4      private String url;
5      private String mbid;
6
7      public long getId() {
8          return id;
9      }
10
11     public void setId(long id) {
```

[7] https://kotlinlang.org/docs/reference/whatsnew12.html#multiplatform-projects-experimental
[8] https://kotlinlang.org/docs/reference/native-overview.html

```
12          this.id = id;
13      }
14
15      public String getName() {
16          return name;
17      }
18
19      public void setName(String name) {
20          this.name = name;
21      }
22
23      public String getUrl() {
24          return url;
25      }
26
27      public void setUrl(String url) {
28          this.url = url;
29      }
30
31      public String getMbid() {
32          return mbid;
33      }
34
35      public void setMbid(String mbid) {
36          this.mbid = mbid;
37      }
38
39      @Override public String toString() {
40          return "Artist{" +
41                  "id=" + id +
42                  ", name='" + name + '\'' +
43                  ", url='" + url + '\'' +
44                  ", mbid='" + mbid + '\'' +
45                  '}';
46      }
47  }
```

With Kotlin, you just need to make use of a data class:

```
data class Artist(
    var id: Long,
    var name: String,
    var url: String,
    var mbid: String)
```

This data class auto-generates all the fields and property accessors, as well as some useful methods such as `toString()`. You also get `equals()` and `hashCode()` for free, which are very verbose and can be dangerous if they are incorrectly implemented.

Null Safety

When we use Java, a significant amount of our code is defensive. We need to check once and another whether something is null before using it to prevent unexpected *NullPointerException*. Kotlin, like many other modern languages, is null-safe because the type explicitly defines whether an object can be null by using the safe call operator (written `?`).

We can do things like this:

```
// This does not compile. Artist cannot be null
var notNullArtist: Artist = null

// Artist can be null
var artist: Artist? = null

// Will not compile, artist could be null and we need to deal with that
artist.print()

// Will print only if artist != null
artist?.print()

// Smart cast. We don not need to use safe call operator if we previously
// checked nullity
if (artist != null) {
    artist.print()
}

// Only use it when we are sure it is not null. It throws an exception otherwise.
```

```
20 artist!!.print()
21
22 // Use Elvis operator to give an alternative in case the object is null.
23 val name = artist?.name ?: "empty"
```

Extension functions

Thanks to extension functions, you can add new functions to any class. It is a cleaner substitute for the common utility classes we all have in our projects. You could, for instance, add a new method to fragments to show a toast:

```
1 fun Fragment.toast(message: CharSequence, duration: Int = Toast.LENGTH_SHORT) {
2     Toast.makeText(getActivity(), message, duration).show()
3 }
```

And then use it like this:

```
1 fragment.toast("Hello world!")
```

Functional support (Lambdas)

What if, instead of having to declare an anonymous class every time we need to implement a click listener, we could just define what we want to do? We can indeed do it. This (and many other interesting things) is what we get thanks to lambdas:

```
1 view.setOnClickListener { toast("Hello world!") }
```

This set of features is only a small selection of what Kotlin can do to simplify your code. Now that you know some of the many great features of the language, you may decide that this is not for you. If you continue, we will start writing some code right away in the next chapter.