KOTLIN

FOR ANDROID DEVELOPERS

Learn Kotlin the easy way while developing an Android App.

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 2017-09-01

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 this book as a necessary step forward.

I also want to do a special mention to Luis Herrero Jiménez, who has designed the awesome cover of this book, and to Gautier Mechling for helping me so much by reviewing this book. It's thanks to him that this 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 I hope it will become a reference. So any claim or suggestion you think it will 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 on this book, I'm really glad 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. But I felt so much power in the language, that I decided to explain all what 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 that want to learn the language.

As you may know, this is a lean publication. This means that the 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

In this book, I'll be creating an Android app from ground up using Kotlin as the main language. The idea is to learn the language by example, instead of following a regular reference book structure. I'll be stopping to explain the most interesting 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 will 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 own projects by themselves. I'll be solving many of the most recurring problems we have to face in our daily lives as App developers, by making use of Kotlin expressiveness and some other really

interesting tools and libraries. However, this text covers most Kotlin features, so by the end of the reading you will have a deep knowledge about the language.

The content is very practical, so I recommend you to follow the examples and the code in front of a computer and try everything it's suggested. You could, however, take a 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 write me back to *contact@antonioleiva.com* with a proof of purchase, and I'll give you access to the digital copy. That way, you'll keep receiving the updates.

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 Kotlin language.

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

- You have some basic knowledge about Android Development and the Android SDK, as well as 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 find every day, by using a cleaner and more expressive language.

On the other hand, this book may not be for you. This is what you won't find in it:

- This is not a Kotlin Bible. I'll explain all language basics, and even more complex 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 won't 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!
- This is not a guide to learn functional programming. Of course, I'll explain what you need, as Java 6 is not functional at all, but I won't dive deep into functional topics.

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 in helping other Android developers learn Kotlin.

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

Antonio started as a consultant in CRM technologies, but after some time, looking for his real passion, he discovered the Android world. After getting some experience on such an awesome platform, he started a new adventure at a mobile company, where he led several projects for important Spanish companies.

He now works as an Android Engineer at Plex[2], where he also plays an important 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. The Android team announced during Google I/O 2017 that Kotlin is now an official language to develop Android Apps.

This means that, while it's still possible to develop Android Apps using Java, from now on Kotlin is fully supported and Google will make sure that all new Android features, 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 from all the features of a modern language while developing Android.

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

But 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 a JVM based language developed by JetBrains[5], a company known for creating IntelliJ IDEA, a powerful IDE for Java development. Android Studio, the official Android IDE, is based on IntelliJ.

Kotlin was created with Java developers in mind, and with IntelliJ as its main development IDE. And these are two very interesting features for Android developers:

- **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 in basic concepts can be learnt in no time.

---

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

- **We have total integration with our daily IDE for free**. Android Studio can understand, compile and run Kotlin code. And the support for this language comes from the company who develops the IDE, so we Android developers are first-class citizens.

But 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's more expressive**: this is one of its most important qualities. You can write more with much less code.
- **It's safer**: Kotlin is null safe, which means that we deal with possible null situations in compile time, to prevent execution time exceptions. We need to explicitly specify that an object can be null, and then check its nullity before using it. You will save a lot of time debugging null pointer exceptions and fixing nullity bugs.
- **It's functional**: Kotlin is basically an object oriented language, not a pure functional language. However, as many other modern languages, it uses many concepts from functional programming, such as lambda expressions, to solve some problems in a much easier way. 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 don't have access to the source code.
- **It's highly interoperable**: You can continue using most libraries and code written in Java, because the interoperability between both languages is excellent. It's even possible to create mixed projects, with both Kotlin and Java files coexisting.

1.2 What do we get with Kotlin?

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

Expressiveness

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

```java
public class Artist {
    private long id;
    private String name;
    private String url;
    private String mbid;

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getUrl() {
        return url;
    }

    public void setUrl(String url) {
        this.url = url;
    }

    public String getMbid() {
        return mbid;
    }

    public void setMbid(String mbid) {
        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:

```
1    data class Artist(
2        var id: Long,
3        var name: String,
4        var url: String,
5        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 big amount of our code is defensive. We need to check once and another whether something is null before using it if we don't want to find unexpected *NullPointerException*. Kotlin, as 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:

```kotlin
1   // This won't compile. Artist can't be null
2   var notNullArtist: Artist = null
3
4   // Artist can be null
5   var artist: Artist? = null
6
7   // Won't compile, artist could be null and we need to deal with that
8   artist.print()
9
10  // Will print only if artist != null
11  artist?.print()
12
13  // Smart cast. We don't need to use safe call operator if we previously
14  // checked nullity
15  if (artist != null) {
16      artist.print()
17  }
18
19  // Only use it when we are sure it's not null. Will throw 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

You can add new functions to any class. It's a much more readable substitute to the usual utility classes we all have in our projects. You could, for instance, add a new method to fragments to show a toast:

```kotlin
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:

```kotlin
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. This (and many more interesting things) is what we get thanks to lambdas:

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

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