Extracted from:

# From Objects to Functions

## Build Your Software Faster and Safer with Functional Programming and Kotlin



The Pragmatic Bookshelf

Dallas, Texas

The Pragmatic Programmers

# From Objects to Functions

## Build Your Software Faster and Safer with Functional Programming and Kotlin

Uberto Barbini

*edited by Adaobi Obi Tulton*

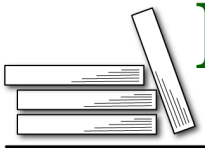
# From Objects to Functions

## Build Your Software Faster and Safer with Functional Programming and Kotlin

Uberto Barbini

The Pragmatic Bookshelf

Dallas, Texas





For our complete catalog of hands-on, practical, and Pragmatic content for software developers, please visit *https://pragprog.com*.

The team that produced this book includes:

CEO: Dave Rankin
COO: Janet Furlow
Managing Editor: Tammy Coron
Development Editor: Adaobi Obi Tulton
Copy Editor: Karen Galle
Indexing: Potomac Indexing, LLC
Layout: Gilson Graphics
Founders: Andy Hunt and Dave Thomas

For sales, volume licensing, and support, please contact *support@pragprog.com*.

For international rights, please contact *rights@pragprog.com*.



ISBN-13: 978-1-68050-845-1
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—September 2023

# Preface

Functional programming makes my job more pleasant every day, even if it drives me crazy sometimes.

Learning about the functional style has profoundly changed how I design my applications; not only that, but it also made programming fun again. So my aim here is to write the book that I wished I'd read when I started learning functional programming.

I ended up as a developer by chance; I learned programming by writing video games in BASIC, and then I started writing small programs as a part-time job when studying philosophy at my university. At that time, I was writing a lot of (horrible) code and having a lot of fun. I didn't know anything about object-oriented principles, design patterns, and other such things. Still, my code worked, which was magic enough for me.

Around the year 2000, I started working on bigger projects and discovered automatic tests. This was a revelation for me, and it pushed me to study computer science more seriously. Once I learned more, my programs became more maintainable but also more complicated. I struggled more with writing code. This somehow removed the fun out of my job, but I told myself that I was a more "professional" programmer.

Then at some point, I decided to pursue simplicity and productivity over abstract principles, and started having fun again. With this book, I want to show how it's possible to use a functional design style and still have fun writing applications. In other words, this is a book written by a developer writing code every day for the benefit of other people like him.

As an industry, I think we need to improve the quality of the software we write. Functional programming isn't a silver bullet,[1] but it can help reduce the amount of work needed to deliver value to the business.

1. https://en.wikipedia.org/wiki/No_Silver_Bullet

Still, quality alone can't give us good velocity, but it can provide us with speed. To transform it into velocity, we also need direction. Without direction, we may build the perfect software only to find that it was not what our business needed.

## About This Book

The main goal isn't to present original ideas or novel techniques but to show an easy way to learn and use functional thinking in your day-to-day work. I hope that the diagrams and explanations in this book will help you understand how it's possible to effectively eliminate mutable state and side effects by relying on functional programming constructs.

However, I don't think that functional programming itself is the ultimate goal. Rather, the goal is to quickly construct robust and adaptable software, and I believe functional programming, together with tests and domain exploration, is the best tool to accomplish this. Nevertheless, I'm not interested in functional programming just for the sake of it. If a partially functional solution performs better than a purely functional one, I wouldn't hesitate to suggest the former.

That being said, this book isn't exclusively about functional programming. As one of the reviewers accurately put it, this is a book on how to write better software using functional programming and other best practices, as per my personal experience in coding for over two decades. You may have different ideas about some of the code presented here; I think this is perfectly fine. I hope to pass on the idea of how to design software in a functional way, not to dictate the exact patterns that you should follow.

To me, this book is like a journey, and much like any adventure, there are many side quests that the hero (that will be you) needs to complete before advancing with the main mission.

For this reason, in this book, we'll also cover how to transform user stories into high-level tests to verify them. We'll look at other good practices for delivering a successful project. Even if they may not seem related to functional programming at first, they are part of the whole functional approach to writing applications, of which writing code is only a part.

I am aware of the possibility of covering too many topics without diving deeply enough into each one, making everybody unhappy. Nonetheless, I believe it's important to explore all the aspects of software development to show how to build software in a functional way. While I may not succeed in every area, it won't be for a lack of effort.

## Who Should Read This Book

This is a book aimed at intermediate and expert programmers. I'll try to introduce each new concept, but this book takes many things for granted. But it should be possible to read this book without knowing Kotlin—it should be enough to know Java, C#, or a similar language.

When writing this book, I kept three categories of readers in mind:

- Seasoned developers with a solid background in object-oriented programming that are interested in knowing more about functional programming and its benefits
- Beginners to functional programming who struggle to translate the classic textbook example into a full application
- Programmers proficient in functional programming in other languages, such as Scala or F#, who are interested in Kotlin

However, a word of caution: I don't expect that you can fully understand the concepts behind this book without studying the code and undertaking at least some of the exercises. You might manage it, but personally, I know I wouldn't be able to.

## How to Read This Book

In an effort to keep this book light and concise while maintaining clarity, some detailed explanations and interesting topics had to be left out. If you find yourself unsure about how something works, please refer to the included code.

In any case, for feedback or clarifications, you can always contact me through social media:

Twitter: https://twitter.com/ramtop

Medium: https://medium.com/@ramtop

LinkedIn: https://www.linkedin.com/in/uberto/

Mastodon: https://mastodon.online/@ramtop

Here are some suggestions to get the most out of this book.

The introduction aims to address the question of why using functional programming can be a good idea. However, if you are new to functional programming, it's recommended that you start by reading Appendix 1, What Is Functional Programming?, on page ?, first. This appendix provides a brief history of functional programming and presents practical examples of its

principles to give you a clearer understanding of what functional programming is all about.

If you don't know (much) about Kotlin, you can refer at any moment to Appendix 2, About Functional Kotlin, on page ?, to better understand Kotlin syntax, comparing it with Java. It concentrates on Kotlin's functional idioms used in the rest of the book. Please remember that this book isn't a complete tutorial on Kotlin or the JVM. If you want a more complete—and fun-to-read—introduction to the Kotlin language from the ground up, I suggest *Programming Kotlin* by Venkat Subramaniam.

Chapters 1 through 12 are the core of the book, and they'll show how we can progressively build an application following functional design. They start gradually, but as our application progresses, we'll cover advanced concepts like monads, applicatives, and profunctors.

The idea is that reading the chapters in order will help you understand the process of building a complete application. For this reason you'll find the code from the book repository split into different steps that align with the progression of the chapters.

Steps 1 to 5 contain the code of the application as it appears every two chapters, from Chapter 2 to Chapter 10. Step 6 corresponds to Chapter 11, and step 7 corresponds to Chapter 12.

Still, if you aren't interested in some parts, it should be possible to skip to the next chapter. Each chapter starts with an explanation of what we'll focus on and ends with a recap of what we've discussed.

Each chapter in this book concludes with exercises designed to help you understand functional principles and verify your functional insight. Based on my experiences as both a teacher and a student, I believe the mastery of functional programming truly comes with a lot of practice. These exercises are especially beneficial when you find something hard to grasp. Writing and playing with code until it fully clicks is the best way to master any new concept. In the book's repository, you'll also find solutions for all exercises. You can compare your solutions with mine, keeping in mind that my approach isn't necessarily the best one.

The final chapter, Chapter 13, Designing a Functional Architecture, on page ?, is about how to design and build a complete software architecture spanning multiple services using a functional approach. It can be read independently from the previous chapters, and it has a higher-level perspective than they do.

Finally, if you are interested in understanding more about the theory behind functional programming, Appendix 3, A Pinch of Theory, on page ?, has an introduction to category theory. Appendix 4, Additional Resources, on page ?, contains some suggestions on where to find material to study this fascinating field further and help you better understand it.

Dear reader, if reading this book makes you smile and learn new things, just like I did while writing it, then I'll consider it a success.