O'REILLY®

Learning Scala

学习Scala（影印版）

東南大學出版社

Jason Swartz 著

学习Scala（影印版）

Learning Scala

Jason Swartz 著

Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo O'REILLY®

O'Reilly Media, Inc.授权东南大学出版社出版

南京 东南大学出版社

图书在版编目(CIP)数据

学习 Scala:英文/(美)斯沃茨(Swartz, J.)著. —影印本. —南京:东南大学出版社,2015.8

书名原文:Learning Scala

ISBN 978-7-5641-5920-7

Ⅰ.①学… Ⅱ.①斯… Ⅲ.①JAVA 语言-程序设计-英文 Ⅳ.①TP312

中国版本图书馆 CIP 数据核字(2015)第 165496 号

江苏省版权局著作权合同登记

图字:10-2015-157 号

学习 Scala(影印版)

出版发行:东南大学出版社

地　　址:南京四牌楼 2 号　　邮编:210096

出 版 人:江建中

网　　址:http://www.seupress.com

电子邮件:press@seupress.com

印　　刷:常州市武进第三印刷有限公司

开　　本:787 毫米×980 毫米　　16 开本

印　　张:16

字　　数:313 千字

版　　次:2015 年 8 月第 1 版

印　　次:2015 年 8 月第 1 次印刷

书　　号:ISBN 978-7-5641-5920-7

定　　价:52.00 元

本社图书若有印装质量问题,请直接与营销部联系。电话(传真):025-83791830

For my loving wife, who foresees great prospects; and for my loving daughter, who also foresees the first printed copy coming her way.

Preface

Welcome to *Learning Scala*. In this book I will provide you with a comprehensive yet approachable introduction to the Scala programming language.

Who This Book Is For

This book is meant for developers who have worked in object-oriented languages such as Java, Ruby, or Python and are interested in improving their craft by learning Scala. Java developers will recognize the core object-oriented, static typing and generic collections in Scala. However, they may be challenged to switch to Scala's more expressive and flexible syntax, and the use of immutable data and function literals to solve problems. Ruby and Python developers will be familiar with the use of function literals (aka closures or blocks) to work with collections, but may be challenged with its static, generic-supporting type system.

For these and any other developers who want to learn how to develop in the Scala programming language, this book provides an organized and examples-based guide that follows a gradual learning curve.

Why Write "Learning Scala"?

When I picked up Scala in early 2012, I found the process of learning the language was longer and more challenging than it ought to be. The available books on Scala did cover the core features of the language. However, I found it difficult to switch from Java to Scala's unfamiliar syntax, its preference for immutable data structures, and its sheer extensibility. It took me several weeks to become comfortable writing new code, several months to fully understand other developers' code, and up to a year to figure out the more advanced features of the language.

I chose to write this book so that future developers will have an easier time learning the language. Now, even using this book the process of learning Scala won't be *easy*; picking

up new skills is always going to be challenging, and learning a new language with an unfamiliar syntax and new methodologies is going to take dedication and lots of work. However, this book at least should make the process easier. Hopefully it will ensure that more developers than before will pick up Scala, and also become capable enough to work with it as their main language.

Why Learn Scala (or, Why Should You Read "Learning Scala")?

I enjoy developing with Scala and highly recommend it to anyone writing server applications and other types of programs suitable for Java-like languages. If you are working in domains suitable for running the Java Virtual Machine such as web applications, services, jobs, or data processing, then I'll certainly recommend that you try using Scala.

Here's *why* you should take this advice and learn to develop in Scala.

Reason 1—Your Code Will Be Better

You will be able to start using functional programming techniques to stabilize your applications and reduce issues that arise from unintended side effects. By switching from mutable data structures to immutable data structures and from regular methods to pure functions that have no effect on their environment, your code will be safer, more stable, and much easier to comprehend.

Your code will also be simpler and more expressive. If you currently work in a dynamic language such as Python, Ruby, or JavaScript, you already are familiar with the benefits of using a short, expressive syntax, avoiding unnecessary punctuation, and condensing map, filter, and reduce operations to simple one-liners. If you are more familiar with statically typed languages like Java, C#, or C++, you'll be able to shed explicit types, punctuation, and boilerplate code. You will also be able to pick up an expressive syntax rarely seen in other compiled languages.

Finally, your code will be strongly typed (even without specifying explicit types) and support both multiple inheritance and mixin capabilities. Also, any type incompatibilities will be caught before your code ever runs. Developers in statically typed languages will be familiar with the type safety and performance available in Scala. Those using dynamic languages will be able to drastically increase safety and performance while staying with an expressive language.

Reason 2—You'll Be a Better Engineer

An engineer who can write short and expressive code (as one expects in Ruby or Python) while also delivering a type-safe and high-performance application (as one expects from Java or C++) would be considered both impressive and valuable. I am assuming that if

you read this book and take up Scala programming you will be writing programs that have all of these benefits. You'll be able to take full advantage of Scala's functional programming features, deliver type-safe and expressive code, and be more productive than you have ever been.

Learning any new programming language is a worthwhile endeavor, because you'll pick up new and different ways to approach problem solving and algorithm and data structure design, along with ways to express these new techniques in a foreign syntax. On top of this, taking up a functional programming language like Scala will help to shape how you view the concepts of data mutability, higher-order functions, and side effects, not only as new ideas but how they apply to your current coding work and designs. You may find that working with inline functions and static types are unnecessary for your current needs, but you'll have some experience with their benefits and drawbacks. Plus, if it becomes possible to apply these features in a partial manner to your current language, such as the new lambda expression support in Java 8, you'll be ready to handle them appropriately.

Reason 3—You'll Be a Happier Engineer

This is admittedly a bold statement from someone you haven't met and who shouldn't presume to know what effect Scala development will have on your brain. I'll only state that if your code proficiency improves to the point that you are easily writing code that works better, reads better, debugs better, and runs faster than before, and on top of all this takes less time to write, you're going to be happier doing so.

Not that life is all about coding, of course. Nor does the work schedule of average software engineers involve more than half of their time spent actually writing code.

But that time spent writing code will be more fun, and you'll be able to take more pride in your work. That should be reason enough to learn something new.

Why Learning Scala May Not Be for You

You should know that Scala has a reputation for being difficult to learn. The language combines two apparently conflicting software engineering paradigms: object-oriented programming and functional programming. This synergy will be surprising to newcomers and the resulting syntax takes some practice to pick up. Scala also has a sophisticated type system that enables custom typing declarations at a level rarely seen outside of academic languages. Ascertaining the syntax and utility of this type system will be challenging, especially if you do not have academic experience with abstract algebra or type theory.

If you do not have enough time to spend on reading this book and going through its exercises, or alternately prefer more challenging or theoretical routes to learning the language, then this book may not be suitable for you.

About the Syntax Notation in This Book

Here is an example of the syntax notation you'll encounter in this book:

```
val <identifier>[: <type>] = <data>
```

This specific example is the definition of a *value*, a type of variable in Scala that cannot be reassigned. It uses my own informal notation for defining the Scala language's syntax, one that can be easier to read than the traditional notations used to define languages but that comes at the cost of being less formal and precise.

Here is how this notation works:

- Keywords and punctuation are printed normally as they would appear in source code.
- Variable items, such as values, types, and literals, are surrounded by angular brackets ("<" and ">").
- Optional segments are surrounded by square brackets ("[" and "]").

For example, in the preceding example "val" is a keyword, "identifier" and "data" are variable items that change with the context, and "type" is an optional item that (if specified) must be separated from the identifier by a colon (":").

I do suggest reading the formal Scala language specification in addition to this book. Although it uses a traditional syntax notation that may be difficult to learn, it is still invaluable for determining the exact syntax requirements of any given feature. The official title is *The Scala Language Specification* (Odersky, 2011), and you can find it either on the official Scala site or with a quick web search.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic
: Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`
: Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

`Constant width bold`
: Shows commands or other text that should be typed literally by the user.

`Constant width italic`
: Shows text that should be replaced with user-supplied values or by values determined by context.

This element signifies a tip or suggestion.

This element signifies a general note.

This element indicates a warning or caution.

Using Code Examples

Supplemental material (code examples, exercises, etc.) is available for download at *http://bit.ly/Learning-Scala-materials*.

This book is here to help you get your job done. In general, if example code is offered with this book, you may use it in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Learning Scala* by Jason Swartz (O'Reilly). Copyright 2015 Jason Swartz, 978-1-449-36793-0."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

Safari® Books Online

Safari Books Online is an on-demand digital library that delivers expert content in both book and video form from the world's leading authors in technology and business.

Technology professionals, software developers, web designers, and business and creative professionals use Safari Books Online as their primary resource for research, problem solving, learning, and certification training.

Safari Books Online offers a range of plans and pricing for enterprise, government, education, and individuals.

Members have access to thousands of books, training videos, and prepublication manuscripts in one fully searchable database from publishers like O'Reilly Media, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, Course Technology, and hundreds more. For more information about Safari Books Online, please visit us online.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at *http://bit.ly/learning-scala*.

To comment or ask technical questions about this book, send email to *bookquestions@oreilly.com*.

For more information about our books, courses, conferences, and news, see our website at *http://www.oreilly.com*.

Find us on Facebook: *http://facebook.com/oreilly*

Follow us on Twitter: *http://twitter.com/oreillymedia*

Watch us on YouTube: *http://www.youtube.com/oreillymedia*

Acknowledgments

I would like to thank my editor, Meghan Blanchette, for all her efforts to improve the quality of the book and to make its delivery possible. I would also like to thank Simon St. Laurent for his help and encouragement in proposing the book and launching the entire process.

This book would also not have been possible without the many excellent reviewers who spent their own time reading and reviewing its many revisions. Thank you so much, Edward Yue Shung Wong, Shannon "JJ" Behrens, Manish Pandit, Devendra Jaisinghani, Art Peel, Ryan Delucchi, Semmy Purewal, Luc Perkins, Robert Geist, and Alexander Trauzzi! I've learned so much from you and really appreciate everything you have done.

I would like to thank Professor Martin Odersky, the fine folks at EPFL and Typesafe, and the members of the Scala community for creating and improving such an amazing language.

I'd also like to thank my wife, Jeanne, and daughter, Oona, for making their sacrifices and providing moral support so I could write this book.

Finally, I'd like to thank my brother, Joshua, for suggesting that I just go ahead and write a book. Josh, I don't know what you were expecting when you said that, but here it is.

Table of Contents