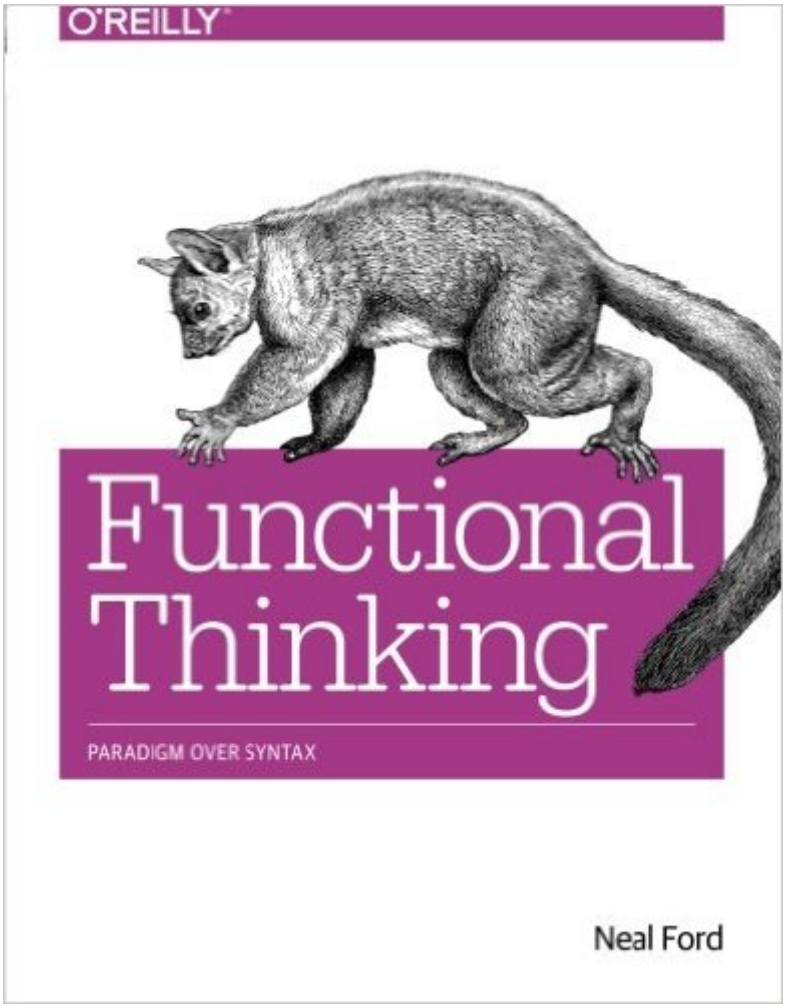
Functional Thinking: Paradigm Over Syntax



Synopsis

If youâ ™re familiar with functional programming basics and want to gain a much deeper understanding, this in-depth guide takes you beyond syntax and demonstrates how you need to think in a new way. Software architect Neal Ford shows intermediate to advanced developers how functional coding allows you to step back a level of abstraction so you can see your programming problem with greater clarity.Each chapter shows you various examples of functional thinking, using numerous code examples from Java 8 and other JVM languages that include functional capabilities. This book may bend your mind, but youâ ™ll come away with a much better grasp of functional programming concepts.Understand why many imperative languages are adding functional capabilitiesCompare functional and imperative solutions to common problemsExamine ways to cede control of routine chores to the runtimeLearn how memoization and laziness eliminate hand-crafted solutionsExplore functional approaches to design patterns and code reuseView real-world examples of functional thinking with Java 8, and in functional architectures and web frameworksLearn the pros and cons of living in a paradigmatically richer worldIf youâ ™re new to functional programming, check out Josh Backfieldâ ™s book Becoming Functional.

Book Information

Paperback: 180 pages
Publisher: O'Reilly Media; 1 edition (July 20, 2014)
Language: English
ISBN-10: 1449365515
ISBN-13: 978-1449365516
Product Dimensions: 7 x 0.4 x 9 inches
Shipping Weight: 12 ounces (View shipping rates and policies)
Average Customer Review: 3.6 out of 5 starsÂ Â See all reviewsÂ (10 customer reviews)
Best Sellers Rank: #415,848 in Books (See Top 100 in Books) #42 inÂ Books > Computers & Technology > Programming > Functional #155 inÂ Books > Textbooks > Computer Science > Object-Oriented Software Design #178 inÂ Books > Computers & Technology > Programming > Languages & Tools > C#

Customer Reviews

I think this may be my first review, but I had to respond after seeing two very negative reviews about this book. First, I can understand that this book can be confusing, as it dives right in to functional programming (FP) without a primer or appendix to help beginners. (I strongly recommend

[defmacro - Functional Programming For The Rest of Us](http://www.defmacro.org/ramblings/fp.html) for that purpose.) However, as an intermediate developer with an (apparently) above-average exposure to FP, I found this book to be extremely illuminating and incredibly useful for me to advance to the next level of FP understanding.Sidenote: One reviewer asserts that the author is confused and perhaps lacks knowledge. I don't know Neal Ford, but I am very familiar with the output of the company at which he works - ThoughtWorks. I can say for certain that anyone employed by them for years is certainly knowledgeable about software engineering, particularly when it comes to real-world usage.Why did I find the book so valuable?* As you can see from the [hosted code](https://github.com/oreillymedia/functional_thinking), Ford accompanies all of his code examples with unit tests, which I find essential for understanding and trust.* Most examples are done in Clojure (a LISP variant for the JVM), Groovy (a dynamic JVM language), **and** Java 8 (sometimes using the Functional Java library), as well as a number in Scala. I find that comparisons between languages improve my learning and retention, in addition to giving extra perspective.* Ford guides the reader through the mix of terminology for the essential FP functions and how they differ by language: map (when it is called 'collect' and why; a.k.

of Heidegger: A Paradigm Shift (New Heidegger Research) The Sustainability Revolution: Portrait of a Paradigm Shift Eye to Eye: The Quest for the New Paradigm