

---

## The Productive Programmer Neal Ford

As recognized, adventure as without difficulty as experience practically lesson, amusement, as skillfully as understanding can be gotten by just checking out a books **The Productive Programmer Neal Ford** also it is not directly done, you could receive even more going on for this life, on the order of the world.

We allow you this proper as competently as simple habit to get those all. We manage to pay for The Productive Programmer Neal Ford and numerous ebook collections from fictions to scientific research in any way. along with them is this The Productive Programmer Neal Ford that can be your partner.



The Productive Programmer O'Reilly Media

A guide to the skills required for state-of-the-art web development, this book covers a variety of web development frameworks. The uses of the standard web API to create applications with increasingly sophisticated architectures are highlighted, and a discussion of the development of industry-accepted best practices for architecture is included. The history and evolution toward this architecture and the reasons it is superior to previous efforts are described, and an overview of the most popular web application frameworks, their architecture, and use is provided. The same application is built in six different frameworks, allowing developers to conduct an

informed comparison. An evaluation of the pros and cons of each framework is provided to assist developers in making decisions or evaluating frameworks on their own. Best practices covered include sophisticated user interface techniques, intelligent caching and resource management, performance tuning, debugging, testing, and web services. The Everything Store Taylor & Francis

Suggests ways for software developers and programmers to work " better, faster, and cheaper." Offers advice on tools for and the mechanics of productivity - "how to work smarter, spurn interruptions, get the most out of your computer, and avoid repetition. ... [Recommends] practices that will help you elude common traps, improve your code, and become more valuable to your team."--Publisher

The Signal and the Noise Addison-Wesley

With more than 200 practical recipes, this book helps you perform data analysis with R quickly and efficiently. The R language provides everything you need to do statistical work, but its structure can be difficult to master. This collection of concise, task-oriented recipes makes you productive with R immediately, with solutions ranging

---

from basic tasks to input and output, general statistics, graphics, and linear regression. Each recipe addresses a specific problem, with a discussion that explains the solution and offers insight into how it works. If you're a beginner, R Cookbook will help get you started. If you're an experienced data programmer, it will jog your memory and expand your horizons. You'll get the job done faster and learn more about R in the process. Create vectors, handle variables, and perform other basic functions Input and output data Tackle data structures such as matrices, lists, factors, and data frames Work with probability, probability distributions, and random variables Calculate statistics and confidence intervals, and perform statistical tests Create a variety of graphic displays Build statistical models with linear regressions and analysis of variance (ANOVA) Explore advanced statistical techniques, such as finding clusters in your data

"Wonderfully readable, R Cookbook serves not only as a solutions manual of sorts, but as a truly enjoyable way to explore the R language—one practical example at a time."—Jeffrey Ryan, software consultant and R package author

*The Productive Programmer* Simon and Schuster  
How do the experts solve difficult problems in software development? In this unique and insightful book, leading computer scientists offer case studies that reveal how they found unusual, carefully designed solutions to high-profile projects. You will be able to look over the shoulder of major coding and design experts to see problems through their eyes. This is not simply another design patterns book, or another software engineering treatise on the right and wrong way to do things. The authors think aloud as they work

through their project's architecture, the tradeoffs made in its construction, and when it was important to break rules. This book contains 33 chapters contributed by Brian Kernighan, Karl Fogel, Jon Bentley, Tim Bray, Eliotte Rusty Harold, Michael Feathers, Alberto Savoia, Charles Petzold, Douglas Crockford, Henry S. Warren, Jr., Ashish Gulhati, Lincoln Stein, Jim Kent, Jack Dongarra and Piotr Luszczek, Adam Kolawa, Greg Kroah-Hartman, Diomidis Spinellis, Andrew Kuchling, Travis E. Oliphant, Ronald Mak, Rogerio Atem de Carvalho and Rafael Monnerat, Bryan Cantrill, Jeff Dean and Sanjay Ghemawat, Simon Peyton Jones, Kent Dybvig, William Otte and Douglas C. Schmidt, Andrew Patzer, Andreas Zeller, Yukihiro Matsumoto, Arun Mehta, TV Raman, Laura Wingerd and Christopher Seiwald, and Brian Hayes. Beautiful Code is an opportunity for master coders to tell their story. All author royalties will be donated to Amnesty International.

**Pragmatic Thinking and Learning** "O'Reilly Media, Inc."  
Anyone who develops software for a living needs a proven way to produce it better, faster, and cheaper. The Productive Programmer offers critical timesaving and productivity tools that you can adopt right away, no matter what platform you use. Master developer Neal Ford not only offers advice on the mechanics of productivity-how to work smarter, spurn interruptions, get the most out your computer, and avoid repetition-he also details valuable practices that will help you elude common traps, improve your code, and become more valuable to your team. You'll learn to: Write the test before you write the code Manage

---

the lifecycle of your objects fastidiously Build only what you need now, not what you might need later Apply ancient philosophies to software development Question authority, rather than blindly adhere to standards Make hard things easier and impossible things possible through meta-programming Be sure all code within a method is at the same level of abstraction Pick the right editor and assemble the best tools for the job This isn't theory, but the fruits of Ford's real-world experience as an Application Architect at the global IT consultancy ThoughtWorks. Whether you're a beginner or a pro with years of experience, you'll improve your work and your career with the simple and straightforward principles in *The Productive Programmer*. [Practical Statecharts in C/C++](#) MIT Press

'Downright revolutionary... the title is a major understatement... 'Quantum Programming' may ultimately change the way embedded software is designed.' -- Michael Barr, Editor-in-Chief, *Embedded Systems Programming* magazine ([Click here](#) **The ThoughtWorks Anthology, Volume 2** "O'Reilly Media, Inc." MVC and CRUD make software easier to write, but harder to change. Microservice-based architectures can help even the smallest of projects remain agile in the long term, but most tutorials meander in theory or completely miss the point of what it means to be microservice-based. Roll up your sleeves with real projects and learn the most important concepts of evented architectures. You'll have your own deployable, testable project and a direction for where to go next. Much ink has been spilled on the topic of microservices, but all of this writing fails to accurately identify what makes a system a monolith, define what microservices are, or give complete, practical examples, so you're probably left thinking they have nothing to offer

you. You don't have to be at Google or Facebook scale to benefit from a microservice-based architecture. Microservices will keep even small and medium teams productive by keeping the pieces of your system focused and decoupled. Discover the basics of message-based architectures, render the same state in different shapes to fit the task at hand, and learn what it is that makes something a monolith (it has nothing to do with how many machines you deploy to). Conserve resources by performing background jobs with microservices. Deploy specialized microservices for registration, authentication, payment processing, e-mail, and more. Tune your services by defining appropriate service boundaries. Deploy your services effectively for continuous integration. Master debugging techniques that work across different services. You'll finish with a deployable system and skills you can apply to your current project. Add the responsiveness and flexibility of microservices to your project, no matter what the size or complexity. What You Need: While the principles of this book transcend programming language, the code examples are in Node.js because JavaScript, for better or worse, is widely read. You'll use PostgreSQL for data storage, so familiarity with it is a plus. The book does provide Docker images to make working with PostgreSQL a bit easier, but extensive Docker knowledge is not required.

### **Programming Rust** Simon and Schuster

Who are computer hackers? What is free software? And what does the emergence of a community dedicated to the production of free and open source software--and to hacking as a technical, aesthetic, and moral project--reveal about the values of contemporary liberalism? Exploring the rise and political significance of the free and open source software (F/OSS) movement in the United States and Europe, *Coding Freedom* details the ethics behind hackers' devotion to F/OSS, the social codes that guide its

---

production, and the political struggles through which hackers question the scope and direction of copyright and patent law. In telling the story of the F/OSS movement, the book unfolds a broader narrative involving computing, the politics of access, and intellectual property. E. Gabriella Coleman tracks the ways in which hackers collaborate and examines passionate manifestos, hacker humor, free software project governance, and festive hacker conferences. Looking at the ways that hackers sustain their productive freedom, Coleman shows that these activists, driven by a commitment to their work, reformulate key ideals including free speech, transparency, and meritocracy, and refuse restrictive intellectual protections. Coleman demonstrates how hacking, so often marginalized or misunderstood, sheds light on the continuing relevance of liberalism in online collaboration.

### **How I Became a Quant** "O'Reilly Media, Inc."

The software development ecosystem is constantly changing, providing a constant stream of new tools, frameworks, techniques, and paradigms. Over the past few years, incremental developments in core engineering practices for software development have created the foundations for rethinking how architecture changes over time, along with ways to protect important architectural characteristics as it evolves. This practical guide ties those parts together with a new way to think about architecture and time.

*The Productive Programmer* Princeton University Press

Tap into the wisdom of experts to learn what every programmer should know, no matter what language you use. With the 97 short and

extremely useful tips for programmers in this book, you'll expand your skills by adopting new approaches to old problems, learning appropriate best practices, and honing your craft through sound advice. With contributions from some of the most experienced and respected practitioners in the industry—including Michael Feathers, Pete Goodliffe, Diomidis Spinellis, Cay Horstmann, Verity Stob, and many more--this book contains practical knowledge and principles that you can apply to all kinds of projects. A few of the 97 things you should know: "Code in the Language of the Domain" by Dan North "Write Tests for People" by Gerard Meszaros "Convenience Is Not an -ility" by Gregor Hohpe "Know Your IDE" by Heinz Kabutz "A Message to the Future" by Linda Rising "The Boy Scout Rule" by Robert C. Martin (Uncle Bob) "Beware the Share" by Udi Dahan

### **Functional Thinking** Pragmatic Bookshelf

UPDATED FOR 2020 WITH A NEW PREFACE BY NATE

SILVER "One of the more momentous books of the decade."

—The New York Times Book Review Nate Silver built an innovative system for predicting baseball performance, predicted the 2008 election within a hair's breadth, and became a national sensation as a blogger—all by the time he was thirty. He solidified his standing as the nation's foremost political forecaster with his near perfect prediction of the 2012 election. Silver is the founder and editor in chief of the website FiveThirtyEight. Drawing on his own groundbreaking work, Silver examines the world of prediction, investigating how we can distinguish a true signal from a universe of noisy data. Most predictions fail, often at great cost to society, because most of us have a poor understanding of probability and uncertainty. Both experts and laypeople mistake more confident predictions for more accurate ones. But overconfidence is often the reason for failure. If our appreciation of uncertainty improves, our predictions can get

---

better too. This is the “prediction paradox”: The more humility we have about our ability to make predictions, the more successful we can be in planning for the future. In keeping with his own aim to seek truth from data, Silver visits the most successful forecasters in a range of areas, from hurricanes to baseball to global pandemics, from the poker table to the stock market, from Capitol Hill to the NBA. He explains and evaluates how these forecasters think and what bonds they share. What lies behind their success? Are they good—or just lucky? What patterns have they unraveled? And are their forecasts really right? He explores unanticipated commonalities and exposes unexpected juxtapositions. And sometimes, it is not so much how good a prediction is in an absolute sense that matters but how good it is relative to the competition. In other cases, prediction is still a very rudimentary—and dangerous—science. Silver observes that the most accurate forecasters tend to have a superior command of probability, and they tend to be both humble and hardworking. They distinguish the predictable from the unpredictable, and they notice a thousand little details that lead them closer to the truth. Because of their appreciation of probability, they can distinguish the signal from the noise. With everything from the health of the global economy to our ability to fight terrorism dependent on the quality of our predictions, Nate Silver’s insights are an essential read.

### **Building Evolutionary Architectures** CRC Press

If you’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 you’ll come away with a much better grasp of functional programming concepts. Understand why many imperative languages are adding functional capabilities Compare functional and imperative solutions to common problems Examine ways to cede control of routine chores to the runtime Learn how memoization and laziness eliminate hand-crafted solutions Explore functional approaches to design patterns and code reuse View real-world examples of functional thinking with Java 8, and in functional architectures and web frameworks Learn the pros and cons of living in a paradigmatically richer world If you’re new to functional programming, check out Josh Backfield’s book *Becoming Functional*.

**Windows PowerShell for Developers** "O'Reilly Media, Inc."

Threads are a fundamental part of the Java platform. As multicore processors become the norm, using concurrency effectively becomes essential for building high-performance applications. Java SE 5 and 6 are a huge step forward for the development of concurrent applications, with improvements to the Java Virtual Machine to support high-performance, highly scalable concurrent classes and a rich set of new concurrency building blocks. In Java

---

Concurrency in Practice, the creators of these new facilities explain not only how they work and how to use them, but also the motivation and design patterns behind them. However, developing, testing, and debugging multithreaded programs can still be very difficult; it is all too easy to create concurrent programs that appear to work, but fail when it matters most: in production, under heavy load. Java Concurrency in Practice arms readers with both the theoretical underpinnings and concrete techniques for building reliable, scalable, maintainable concurrent applications. Rather than simply offering an inventory of concurrency APIs and mechanisms, it provides design rules, patterns, and mental models that make it easier to build concurrent programs that are both correct and performant. This book covers: Basic concepts of concurrency and thread safety Techniques for building and composing thread-safe classes Using the concurrency building blocks in `java.util.concurrent` Performance optimization dos and don'ts Testing concurrent programs Advanced topics such as atomic variables, nonblocking algorithms, and the Java Memory Model

### Beautiful Code Microsoft Press

In this truly unique technical book, today's leading software architects present valuable principles on key development issues that go way beyond technology. More than four dozen architects -- including Neal Ford, Michael Nygard, and Bill de hOra -- offer advice for communicating with stakeholders, eliminating complexity, empowering

developers, and many more practical lessons they've learned from years of experience. Among the 97 principles in this book, you'll find useful advice such as: Don't Put Your Resume Ahead of the Requirements (Nitin Borwankar) Chances Are, Your Biggest Problem Isn't Technical (Mark Ramm) Communication Is King; Clarity and Leadership, Its Humble Servants (Mark Richards) Simplicity Before Generality, Use Before Reuse (Kevlin Henney) For the End User, the Interface Is the System (Vinayak Hegde) It's Never Too Early to Think About Performance (Rebecca Parsons) To be successful as a software architect, you need to master both business and technology. This book tells you what top software architects think is important and how they approach a project. If you want to enhance your career, 97 Things Every Software Architect Should Know is essential reading.

### Java Concurrency in Practice Pragmatic Bookshelf

Presentation Patterns is the first book on presentations that categorizes and organizes the building blocks (or patterns) that you'll need to communicate effectively using presentation tools like Keynote and PowerPoint. Patterns are like the lower-level steps found inside recipes; they are the techniques you must master to be considered a master chef or master presenter. You can use the patterns in this book to construct your own recipes for different contexts, such as business meetings, technical demonstrations, scientific expositions, and keynotes, just to name a few. Although there are no such things as antirecipes, this book shows you lots of antipatterns—things you should avoid doing in presentations. Modern presentation tools often encourage ineffective presentation techniques, but this book shows you how to avoid them. Each pattern is introduced with a memorable

---

name, a definition, and a brief explanation of motivation. Readers learn where the pattern applies, the consequences of applying it, and how to apply it. The authors also identify critical antipatterns: clichés, fallacies, and design mistakes that cause presentations to disappoint. These problems are easy to avoid—once you know how. *Presentation Patterns* will help you Plan what you'll say, who you'll say it to, how long you'll talk, and where you'll present Perfectly calibrate your presentation to your audience Use the storyteller's "narrative arc" to full advantage Strengthen your credibility—and avoid mistakes that hurt it Hone your message before you ever touch presentation software Incorporate visuals that support your message instead of hindering it Create highly effective "infodecks" that work when you're not able to deliver a talk in person Construct slides that really communicate and avoid "Ant Fonts," "Floodmarks," "Alienating Artifacts," and other errors Master 13 powerful techniques for delivering your presentation with power, authority, and clarity Whether you use this book as a handy reference or read it from start to finish, it will be a revelation: an entirely new language for systematically planning, creating, and delivering more powerful presentations. You'll quickly find it indispensable—no matter what you're presenting, who your audiences are, or what message you're driving home.

**97 Things Every Programmer Should Know** "O'Reilly Media, Inc."

"For coders early in their careers who are familiar with an object-oriented language, such as Java or C#"--Back cover.

Lean Architecture John Wiley & Sons

"Thirteen illuminating and insightful essays on modern software development practices"--T.p. verso.

*Presentation Patterns* Pearson Education

Today, software engineers need to know not only how to program effectively but also how to develop proper engineering practices to make their codebase sustainable and healthy. This book emphasizes this difference between programming and software engineering. How can software engineers manage a living codebase that evolves and

responds to changing requirements and demands over the length of its life? Based on their experience at Google, software engineers Titus Winters and Hyrum Wright, along with technical writer Tom Manshreck, present a candid and insightful look at how some of the world's leading practitioners construct and maintain software. This book covers Google's unique engineering culture, processes, and tools and how these aspects contribute to the effectiveness of an engineering organization. You'll explore three fundamental principles that software organizations should keep in mind when designing, architecting, writing, and maintaining code: How time affects the sustainability of software and how to make your code resilient over time How scale affects the viability of software practices within an engineering organization What trade-offs a typical engineer needs to make when evaluating design and development decisions

The Programmer's Brain Little, Brown

'NoSQL Distilled' is designed to provide you with enough background on how NoSQL databases work, so that you can choose the right data store without having to trawl the whole web to do it. It won't answer your questions definitively, but it should narrow down the range of options you have to consider.

*R Cookbook* Manning Publications

Praise for *How I Became a Quant* "Led by two top-notch quants, Richard R. Lindsey and Barry Schachter, *How I Became a Quant* details the quirky world of quantitative analysis through stories told by some of today's most successful quants. For anyone who might have thought otherwise, there are engaging personalities behind all that number crunching!" --Ira Kawaller, Kawaller & Co. and the Kawaller Fund "A fun and fascinating read. This book tells the story of how academics, physicists, mathematicians, and other scientists became professional investors managing billions." --David A. Krell, President

---

and CEO, International Securities Exchange "How I Became a Quant should be must reading for all students with a quantitative aptitude. It provides fascinating examples of the dynamic career opportunities potentially open to anyone with the skills and passion for quantitative analysis." --Roy D. Henriksson, Chief Investment Officer, Advanced Portfolio Management "Quants"--those who design and implement mathematical models for the pricing of derivatives, assessment of risk, or prediction of market movements--are the backbone of today's investment industry. As the greater volatility of current financial markets has driven investors to seek shelter from increasing uncertainty, the quant revolution has given people the opportunity to avoid unwanted financial risk by literally trading it away, or more specifically, paying someone else to take on the unwanted risk. How I Became a Quant reveals the faces behind the quant revolution, offering you the chance to learn firsthand what it's like to be a quant today. In this fascinating collection of Wall Street war stories, more than two dozen quants detail their roots, roles, and contributions, explaining what they do and how they do it, as well as outlining the sometimes unexpected paths they have followed from the halls of academia to the front lines of an investment revolution.