

---

# The Productive Programmer Neal Ford

Getting the books *The Productive Programmer* Neal Ford now is not type of challenging means. You could not abandoned going taking into consideration books heap or library or borrowing from your links to open them. This is an no question easy means to specifically acquire lead by on-line. This online broadcast *The Productive Programmer* Neal Ford can be one of the options to accompany you later than having additional time.

It will not waste your time. consent me, the e-book will extremely heavens you supplementary situation to read. Just invest tiny period to entry this on-line statement *The Productive Programmer* Neal Ford as with ease as review them wherever you are now.



*No Fluff, Just Stuff*  
*Anthology* Addison-  
Wesley Professional  
Refactoring has proven its value in a wide range of development projects—helping software professionals improve system designs, maintainability, extensibility, and performance. Now, for the first time, leading agile methodologist Scott Ambler and renowned consultant Pramodkumar Sadalage introduce powerful refactoring techniques specifically designed for database systems. Ambler and

Sadalage demonstrate how small changes to table structures, data, stored procedures, and triggers can significantly enhance virtually any database design—without changing semantics. You'll learn how to evolve database schemas in step with source code—and become far more effective in projects relying on iterative, agile methodologies. This comprehensive guide and reference helps you overcome the practical obstacles to refactoring real-world databases by covering every fundamental concept underlying database refactoring. Using start-to-finish examples, the authors walk you through refactoring simple standalone database applications as well as

sophisticated multi-application scenarios. You'll master every task involved in refactoring database schemas, and discover best practices for deploying refactorings in even the most complex production environments. The second half of this book systematically covers five major categories of database refactorings. You'll learn how to use refactoring to enhance database structure, data quality, and referential integrity; and how to refactor both architectures and methods. This book provides an extensive set of examples built with Oracle and Java and easily adaptable for other languages, such as C#, C++, or VB.NET, and other databases, such as DB2, SQL Server, MySQL, and Sybase. Using this

---

book's techniques and examples, you can reduce waste, rework, risk, and cost—and build database systems capable of evolving smoothly, far into the future.

### *The Programmer's Brain*

Pragmatic Bookshelf

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.

### Building Evolutionary

Architectures 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.

### The Thoughtworks

Anthology "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

*Rails for Java*

*Developers Pragmatic Bookshelf*

The Java Developer's Reference provides definitive solutions to your Java needs. Written for serious Java programmers, this comprehensive volume is actually three books in one, combining all the latest information on Java programming to make this the most timely and enduring Java reference book on the market.

## Coding Freedom

"O'Reilly Media, Inc."

Why This Book? You can learn the most popular frameworks, use the best programming languages, and work at the biggest tech companies, but if you cultivate bad habits, it will be hard for you to become a top developer. This book doesn't offer a straight path or pre-defined formula of success. This book is a result of a quest. A quest to uncover what habits

can be cultivated to become a better software engineer.

"I wish I had access to this book while I was starting in the software industry. The information presented is not only logical, not only personal, but very well backed up by many expert opinions throughout the book. A must-read, for both beginners and experts alike." - Zachary Sohovich, Software Engineer at Nike

What Will You Read? How to keep up with all the new technologies What should you focus? Being a specialist or generalist? How to stay productive and not feel overwhelmed The importance of estimating tasks correctly How to approach new side project ideas And much more Who Should Read This Book? It doesn't matter if you're a Junior or Senior

developer. It doesn't matter how experienced you are. This book can help you cultivate new habits or rethink existing behaviors. What's Inside? This is not a traditional book. You won't find the same format or structure that a regular book has. In fact, this book was designed to be as simple and objective as possible. You can follow the order of chapters, or you can read them individually. Everything is standalone and doesn't depend on previous knowledge. At the end of each chapter, you'll find a section marked as "Questions & Answers", where I interview senior developers and tech leads from various companies to understand how they got there. I went after tech giants such as Google, Amazon, Microsoft,

---

and Adobe. Powerful startups such as GitHub, Spotify, Elastic, Segment, GoDaddy, and Shopify. All the way to established organizations such as Citibank, BlackBerry, and The New York Times. These people come from all over the world and have a pretty diverse background. From San Francisco to New York. From São Paulo to Montreal. From London to Stockholm. The idea is to present you not a one man's point of view, but a collection of insights on how to navigate your career. Who's The Author? Zeno Rocha is a Brazilian creator and programmer. He currently lives in Los Angeles, California, where he's the Chief Product Officer at Liferay Cloud. His lifelong appreciation for building software and sharing

knowledge led him to speak in over 110 conferences worldwide. His passion for open source put him on the top 20 most active users on GitHub at age 22. Before moving to the US, Zeno developed multiple applications, mentored startups, and worked at major companies in Latin America, such as Globo and Petrobras. An Introduction to General Systems Thinking John Wiley & Sons 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 14 Habits of Highly Productive Developers "O'Reilly Media, Inc." Shows how to write, debug, and run a Perl

---

program, describes CGI scripting and data manipulation, and describes scalar values, basic operators, and associative arrays. *Cloud Security and Privacy* "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*. [The Quest for Artificial Intelligence](#) Manning Publications Can a system be considered truly reliable if it isn't

fundamentally secure? Or can it be considered secure if it's unreliable? Security is crucial to the design and operation of scalable systems in production, as it plays an important part in product quality, performance, and availability. In this book, experts from Google share best practices to help your organization design scalable and reliable systems that are fundamentally secure. Two previous O'Reilly books from Google—*Site Reliability Engineering* and *The Site Reliability Workbook*—demonstrated how and why a commitment to the entire service lifecycle enables organizations to successfully build, deploy, monitor, and maintain software systems. In this latest guide, the authors offer insights into system design, implementation, and maintenance from practitioners who specialize in

---

security and reliability. They also discuss how building and adopting their recommended best practices requires a culture that's supportive of such change. You'll learn about secure and reliable systems through: Design strategies Recommendations for coding, testing, and debugging practices Strategies to prepare for, respond to, and recover from incidents Cultural best practices that help teams across your organization collaborate effectively

### **Software Project Survival Guide**

Addison-Wesley  
For more than twenty-five years, An Introduction to General Systems Thinking has been hailed as an innovative introduction to systems theory, with applications in computer science and beyond. Used in university courses and professional seminars all over the world, the text

has proven its ability to open minds and sharpen thinking. Originally published in 1975 and reprinted more than twenty times over a quarter century-and now available for the first time from Dorset House Publishing-the text uses clear writing and basic algebraic principles to explore new approaches to projects, products, organizations, and virtually any kind of system. Scientists, engineers, organization leaders, managers, doctors, students, and thinkers of all disciplines can use this book to dispel the mental fog that clouds problem-solving. As author Gerald M. Weinberg writes in the new Preface to the Silver Anniversary Edition, "I haven't changed my conviction that most people don't think nearly as

well as they could had they been taught some principles of thinking." Now an award-winning author of nearly forty books spanning the entire software development life cycle-including The Psychology of Computer Programming: Silver Anniversary Edition and Exploring Requirements (with Donald C. Gause)-Weinberg had already acquired extensive experience as a programmer, manager, university professor, and consultant when this book was originally published. With helpful illustrations, numerous end-of-chapter exercises, and an appendix on a mathematical notation used in problem-solving, An Introduction to General Systems Thinking may be your most powerful

---

tool in working with problems, systems, and solutions.

*Art of Java Web Development* The Productive Programmer

More and more Agile projects are seeking architectural roots as they struggle with complexity and scale - and they're seeking lightweight ways to do it Still seeking? In this book the authors help you to find your own path Taking cues from Lean development, they can help steer your project toward practices with longstanding track records Up-front architecture? Sure. You can deliver an architecture as code that compiles and that concretely guides development without bogging it down in a mass of documents and guesses about the implementation Documentation? Even a whiteboard diagram, or a CRC card, is documentation: the goal isn't to avoid documentation, but to document just the right things in just the right amount Process? This all works within the frameworks of Scrum, XP, and other Agile approaches

Become an Effective

Software Engineering  
Manager O'Reilly  
Media  
ThoughtWorks is a well-known global consulting firm; ThoughtWorkers are leaders in areas of design, architecture, SOA, testing, and agile methodologies. This collection of essays brings together contributions from well-known ThoughtWorkers such as Martin Fowler, along with other authors you may not know yet. While ThoughtWorks is perhaps best known for their work in the Agile community, this anthology confronts issues throughout the software development life cycle. From technology issues that transcend methodology, to issues of realizing business value from applications, you'll find it here.

Functional Thinking  
Sams  
The Productive Programmer  
O'Reilly  
Media  
*Idea Man* Zeno Rocha  
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.

Software Architecture: The Hard Parts  
Microsoft Press

---

You may regard cloud computing as an ideal way for your company to control IT costs, but do you know how private and secure this service really is? Not many people do. With Cloud Security and Privacy, you'll learn what's at stake when you trust your data to the cloud, and what you can do to keep your virtual infrastructure and web applications secure. Ideal for IT staffers, information security and privacy practitioners, business managers, service providers, and investors alike, this book offers you sound advice from three well-known authorities in the tech security world. You'll learn detailed information on cloud computing security that-until now- has been sorely lacking. Review the current state of data security and storage in the cloud, including confidentiality, integrity, and availability. Learn about the identity and access management (IAM) practice for authentication, authorization, and auditing of the users accessing cloud services. Discover which security management frameworks and

standards are relevant for the cloud. Understand the privacy aspects you need to consider in the cloud, including how they compare with traditional computing models. Learn the importance of audit and compliance functions within the cloud, and the various standards and frameworks to consider. Examine security delivered as a service-a different facet of cloud security.

**Pragmatic Thinking and Learning** O'Reilly Media  
Shows you, using detailed comparisons and commentary, how to translate your hard-earned Java knowledge and skills into the world of Ruby and Rails.

**Adaptive Code via C#** "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.

**Good Code, Bad Code** Pearson Education  
Printed in full color. Software development happens in your head. Not in an editor, IDE, or design tool. You're well educated on how to work with software

and hardware, but what about wetware--our own brains? Learning new skills and new technology is critical to your career, and it's all in your head. In this book by Andy Hunt, you'll learn how our brains are wired, and how to take advantage of your brain's architecture. You'll learn new tricks and tipsto learn more, faster, and retain more of what you learn. You need a pragmatic approach to thinking and learning. You need to Refactor Your Wetware. Programmers have to learn constantly; not just the stereotypical new technologies, but also the problem domain of the application, the whims of the user community, the quirks of your teammates, the shifting sands of the industry, and the evolving characteristics of the project itself as it is built. We'll journey together through bits of cognitive and neuroscience,



---

learning and behavioral theory. You'll see some surprising aspects of how our brains work, and how you can take advantage of the system to improve your own learning and thinking skills. In this book you'll learn how to: Use the Dreyfus Model of Skill Acquisition to become more expert Leverage the architecture of the brain to strengthen different thinking modes Avoid common "known bugs" in your mind Learn more deliberately and more effectively Manage knowledge more efficiently

### **Programming Scala**

Pragmatic Bookshelf

"A great book with deep insights into the bridge between programming and the human mind." - Mike Taylor, CGI Your brain responds in a predictable way when it encounters new or difficult tasks. This unique book teaches you concrete techniques rooted in cognitive science that will improve the way you learn and think about code. In *The Programmer's Brain: What every programmer*

needs to know about cognition you will learn: Fast and effective ways to master new programming languages Speed reading skills to quickly comprehend new code Techniques to unravel the meaning of complex code Ways to learn new syntax and keep it memorized Writing code that is easy for others to read Picking the right names for your variables Making your codebase more understandable to newcomers Onboarding new developers to your team Learn how to optimize your brain's natural cognitive processes to read code more easily, write code faster, and pick up new languages in much less time. This book will help you through the confusion you feel when faced with strange and complex code, and explain a codebase in ways that can make a new team member productive in days! Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About the technology Take advantage of your brain's natural processes to be a better programmer. Techniques based in cognitive science make

it possible to learn new languages faster, improve productivity, reduce the need for code rewrites, and more. This unique book will help you achieve these gains. About the book *The Programmer's Brain* unlocks the way we think about code. It offers scientifically sound techniques that can radically improve the way you master new technology, comprehend code, and memorize syntax. You'll learn how to benefit from productive struggle and turn confusion into a learning tool. Along the way, you'll discover how to create study resources as you become an expert at teaching yourself and bringing new colleagues up to speed. What's inside Understand how your brain sees code Speed reading skills to learn code quickly Techniques to unravel complex code Tips for making codebases understandable About the reader For programmers who have experience working in more than one language. About the author Dr. Felienne Hermans is an associate professor at Leiden University in the Netherlands. She has spent the last decade researching programming, how to

---

learn and how to teach  
it. Table of Contents  
PART 1 ON READING CODE  
BETTER 1 Decoding your  
confusion while coding  
2 Speed reading for  
code 3 How to learn  
programming syntax  
quickly 4 How to read  
complex code PART 2 ON  
THINKING ABOUT CODE 5  
Reaching a deeper  
understanding of code 6  
Getting better at  
solving programming  
problems 7  
Misconceptions: Bugs in  
thinking PART 3 ON  
WRITING BETTER CODE 8  
How to get better at  
naming things 9  
Avoiding bad code and  
cognitive load: Two  
frameworks 10 Getting  
better at solving  
complex problems PART 4  
ON COLLABORATING ON  
CODE 11 The act of  
writing code 12  
Designing and improving  
larger systems 13 How  
to onboard new  
developers