

The Productive Programmer Neal Ford

As recognized, adventure as skillfully as experience just about lesson, amusement, as well as concord can be gotten by just checking out a books **The Productive Programmer Neal Ford** as well as it is not directly done, you could allow even more approaching this life, on the subject of the world.

We manage to pay for you this proper as with ease as easy way to acquire those all. We allow The Productive Programmer Neal Ford and numerous books collections from fictions to scientific research in any way. accompanied by them is this The Productive Programmer Neal Ford that can be your partner.



Refactoring Databases "O'Reilly Media, Inc."

Software startups make global headlines every day. As technology companies succeed and grow, so do their engineering departments. In your career, you'll may suddenly get the opportunity to lead teams: to become a manager. But this is often uncharted territory. How can you decide whether this career move is right for you? And if you do, what do you need to learn to succeed? Where do you start? How do you know that you're doing it right? What does "it" even mean? And isn't management a dirty word? This book will share the secrets you need to know to manage engineers successfully. Going from engineer to manager doesn't have to be intimidating. Engineers can be managers, and fantastic ones at that. Cast aside the rhetoric and focus on practical, hands-on techniques and tools. You'll become an effective and supportive team leader that your staff will look up to. Start with your transition to being a manager and see how that compares to being an engineer. Learn how to better organize information, feel productive, and delegate, but not micromanage. Discover how to manage your own boss, hire and fire, do performance and salary reviews, and build a great team. You'll also learn the psychology: how to ship while keeping staff happy, coach and mentor, deal with deadline pressure, handle sensitive information, and navigate workplace politics. Consider your whole department. How can you work with other teams to ensure best practice? How do you help form guilds and committees and communicate effectively? How can you create career tracks for individual contributors and managers? How can you support flexible and remote working? How can you improve diversity in the industry through your own actions? This book will show you how. Great managers can make the world a better place. Join us.

Programming Concurrency on the JVM Pragmatic Bookshelf

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

Learning Perl Lulu.com

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*.

Cloud Security and Privacy Manning Publications

Essay Collection covering the point where software, law and social justice meet.

An Introduction to General Systems Thinking "O'Reilly Media, Inc."

Clojure is a practical, general-purpose language that offers expressivity rivaling other dynamic languages like Ruby and Python, while seamlessly taking advantage of Java libraries, services, and all of the resources of the JVM ecosystem. This book helps you learn the fundamentals of Clojure with examples relating it to the languages you know already, in the domains and topics you work with every day. See how this JVM language can help

eliminate unnecessary complexity from your programming practice and open up new options for solving the most challenging problems. Clojure Programming demonstrates the language's flexibility by showing how it can be used for common tasks like web programming and working with databases, up through more demanding applications that require safe, effective concurrency and parallelism, data analysis, and more. This in-depth look helps tie together the full Clojure development experience, from how to organize your project and an introduction to Clojure build tooling, to a tutorial on how to make the most of Clojure's REPL during development, and how to deploy your finished application in a cloud environment. Learn how to use Clojure while leveraging your investment in the Java platform Understand the advantages of Clojure as an efficient Lisp for the JVM See how Clojure is used today in several practical domains Discover how Clojure eliminates the need for many verbose and complicated design patterns Deploy large or small web applications to the cloud with Clojure Free Software, Free Society "O'Reilly Media, Inc."

Artificial intelligence (AI) is a field within computer science that is attempting to build enhanced intelligence into computer systems. This book traces the history of the subject, from the early dreams of eighteenth-century (and earlier) pioneers to the more successful work of today's AI engineers. AI is becoming more and more a part of everyone's life. The technology is already embedded in face-recognizing cameras, speech-recognition software, Internet search engines, and health-care robots, among other applications. The book's many diagrams and easy-to-understand descriptions of AI programs will help the casual reader gain an understanding of how these and other AI systems actually work. Its thorough (but unobtrusive) end-of-chapter notes containing citations to important source materials will be of great use to AI scholars and researchers. This book promises to be the definitive history of a field that has captivated the imaginations of scientists, philosophers, and writers for centuries.

Professional Git "O'Reilly Media, Inc."

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 O'Reilly Media

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.

Become an Effective Software Engineering Manager "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.

Pragmatic Thinking and Learning Penguin

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 97 Things Every Programmer Should Know Sams

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.

The Developer's Code O'Reilly Media

By his early thirties, Paul Allen was a world-famous billionaire-and that was just the beginning. In 2007 and 2008, Time named Paul Allen, the cofounder of Microsoft, one of the hundred most influential people in the world.

Since he made his fortune, his impact has been felt in science, technology, business, medicine, sports, music, and philanthropy. His passion, curiosity, and intellectual rigor-combined with the resources to launch and support new initiatives-have literally changed the world. In 2009 Allen discovered that he had lymphoma, lending urgency to his desire to share his story for the first time. In this classic memoir, Allen explains how he has solved problems, what he's learned from his many endeavors-both the triumphs and the failures-and his compelling vision for the future. He reflects candidly on an extraordinary life. The book also features previously untold stories about everything from the true origins of Microsoft to Allen's role in the dawn of private space travel (with SpaceShipOne) and in discoveries at the frontiers of brain science. With honesty, humor, and insight, Allen tells the story of a life of ideas made real.

Functional Thinking Addison-Wesley Professional

You're already a great coder, but awesome coding chops aren't always enough to get you through your toughest projects. You need these 50+ nuggets of wisdom. Veteran programmers: reinvigorate your passion for developing web applications. New programmers: here's the guidance you need to get started. With this book, you'll think about your job in new and enlightened ways. The Developer's Code isn't about the code you write, it's about the code you live by. There are no trite superlatives here. Packed with lessons learned from more than a decade of software development experience, author Ka Wai Cheung takes you through the programming profession from nearly every angle to uncover ways of sustaining a healthy connection with your work. You'll see how to stay productive even on the longest projects. You'll create a workflow that works with you, not against you. And you'll learn how to deal with clients whose goals don't align with your own. If you don't handle them just right, issues such as these can crush even the most seasoned, motivated developer. But with the right approach, you can transcend these common problems and become the professional developer you want to be. In more than 50 nuggets of wisdom, you'll learn: Why many traditional approaches to process and development roles in this industry are wrong - and how to sniff them out. Why you must always say "no" to the software pet project and open-ended timelines. How to incorporate code generation into your development process, and why its benefits go far beyond just faster code output. What to do when your client or end user disagrees with an approach you believe in. How to pay your knowledge forward to future generations of programmers through teaching and evangelism. If you're in this industry for the long run, you'll be coming back to this book again and again.

Software Architecture: The Hard Parts Princeton University Press

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

Building Evolutionary Architectures Zeno Rocha

"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

The Productive Programmer

Agile coding with design patterns and SOLID principles As every developer knows, requirements are subject to change. But when you build adaptability into your code, you can respond to change more easily and avoid disruptive rework. Focusing on Agile programming, this book describes the best practices, principles, and patterns that enable you to create flexible, adaptive

code--and deliver better business value. Expert guidance to bridge the gap between theory and practice Get grounded in Scrum: artifacts, roles, metrics, phases Organize and manage architectural dependencies Review best practices for patterns and anti-patterns Master SOLID principles: single-responsibility, open/closed, Liskov substitution Manage the versatility of interfaces for adaptive code Perform unit testing and refactoring in tandem See how delegation and abstraction impact code adaptability Learn best ways to implement dependency interjection Apply what you learn to a pragmatic, agile coding project Get code samples at:

<http://github.com/garymclean/AdaptiveCode>

The Pragmatic Programmer Pragmatic Bookshelf

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.

Coding Freedom Dorset House Publishing Company, Incorporated

Presents guidelines on the art of coding with Perl, covering such topics as naming conventions, data and control structures, program decomposition, interface design, and error handling.

The Productive Programmer "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.

Programming Scala Microsoft Press

There are no easy decisions in software architecture. Instead, there are many hard parts--difficult problems or issues with no best practices--that force you to choose among various compromises. With this book, you'll learn how to think critically about the trade-offs involved with distributed architectures. Architecture veterans and practicing consultants Neal Ford, Mark Richards, Pramod Sadalage, and Zhamak Dehghani discuss strategies for choosing an appropriate architecture. By interweaving a story about a fictional group of technology professionals--the Sysops Squad--they examine everything from how to determine service granularity, manage workflows and orchestration, manage and decouple contracts, and manage distributed transactions to how to optimize operational characteristics, such as scalability, elasticity, and performance. By focusing on commonly asked questions, this book provides techniques to help you discover and weigh the trade-offs as you confront the issues you face as an architect. Analyze trade-offs and effectively document your decisions Make better decisions regarding service granularity Understand the complexities of breaking apart monolithic applications Manage and decouple contracts between services Handle data in a highly distributed architecture Learn patterns to manage workflow and transactions when breaking apart applications