

Effective Programming More Than Writing Code Hfwebs

Success in today's IT environment requires you to view your career as a business endeavor. In this book, you'll learn how to become an entrepreneur, driving your career in the direction of your choosing. You'll learn how to build your software development career step by step, following the same path that you would follow if you were building, marketing, and selling a product. After all, your skills themselves are a product. The choices you make about which technologies to focus on and which business domains to master have at least as much impact on your success as your technical knowledge itself--don't let those choices be accidental. We'll walk through all aspects of the decision-making process, so you can ensure that you're investing your time and energy in the right areas. You'll develop a structured plan for keeping your mind engaged and your skills fresh. You'll learn how to assess your skills in terms of where they fit on the value chain, driving you away from commodity skills and toward those that are in high demand. Through a mix of high-level, thought-provoking essays and tactical "Act on It" sections, you will come away with concrete plans you can put into action immediately. You'll also get a chance to read the perspectives of several highly successful members of our industry from a variety of career paths. As with any product or service, if nobody knows what you're selling, nobody will buy. We'll walk through the often-neglected world of marketing, and you'll create a plan to market yourself both inside your company and to the industry in general. Above all, you'll see how you can set the direction of your career, leading to a more fulfilling and remarkable professional life.

What others in the trenches say about *The Pragmatic Programmer*... “The cool thing about this book is that it’s great for keeping the programming process fresh. The book helps you to continue to grow and clearly comes from people who have been there.” —Kent Beck, author of *Extreme Programming Explained: Embrace Change* “I found this book to be a great mix of solid advice and wonderful analogies!” —Martin Fowler, author of *Refactoring* and *UML Distilled* “I would buy a copy, read it twice, then tell all my colleagues to run out and grab a copy. This is a book I would never loan because I would worry about it being lost.” —Kevin Ruland, Management Science, MSG-Logistics “The wisdom and practical experience of the authors is obvious. The topics presented are relevant and useful.... By far its greatest strength for me has been the outstanding analogies—tracer bullets, broken windows, and the fabulous helicopter-based explanation of the need for orthogonality, especially in a crisis situation. I have little doubt that this book will eventually become an excellent source of useful information for journeymen programmers and expert mentors alike.” —John Lakos, author of *Large-Scale C++ Software Design* “This is the sort of book I will buy a dozen copies of when it comes out so I can give it to my clients.” —Eric Vought, Software Engineer “Most modern books on software development fail to cover the basics of what makes a great software developer, instead spending their time on syntax or technology where in reality the greatest leverage possible for any software team is in having talented developers who really know their craft well. An excellent book.” —Pete McBreen, Independent Consultant “Since reading this book, I have implemented many of the practical suggestions and tips it contains. Across the board, they have saved my company time and money while helping me get my job

done quicker! This should be a desktop reference for everyone who works with code for a living.” —Jared Richardson, Senior Software Developer, iRenaissance, Inc. “I would like to see this issued to every new employee at my company....” —Chris Cleeland, Senior Software Engineer, Object Computing, Inc. “If I’m putting together a project, it’s the authors of this book that I want. . . . And failing that I’d settle for people who’ve read their book.” —Ward Cunningham Straight from the programming trenches, *The Pragmatic Programmer* cuts through the increasing specialization and technicalities of modern software development to examine the core process--taking a requirement and producing working, maintainable code that delights its users. It covers topics ranging from personal responsibility and career development to architectural techniques for keeping your code flexible and easy to adapt and reuse. Read this book, and you'll learn how to Fight software rot; Avoid the trap of duplicating knowledge; Write flexible, dynamic, and adaptable code; Avoid programming by coincidence; Bullet-proof your code with contracts, assertions, and exceptions; Capture real requirements; Test ruthlessly and effectively; Delight your users; Build teams of pragmatic programmers; and Make your developments more precise with automation. Written as a series of self-contained sections and filled with entertaining anecdotes, thoughtful examples, and interesting analogies, *The Pragmatic Programmer* illustrates the best practices and major pitfalls of many different aspects of software development. Whether you're a new coder, an experienced programmer, or a manager responsible for software projects, use these lessons daily, and you'll quickly see improvements in personal productivity, accuracy, and job satisfaction. You'll learn skills and develop habits and attitudes that form the foundation for long-term success in your career. You'll become a Pragmatic Programmer.

Users can dramatically improve the design, performance, and manageability of object-oriented code without altering its interfaces or behavior. "Refactoring" shows users exactly how to spot the best opportunities for refactoring and exactly how to do it, step by step.

There are many excellent R resources for visualization, data science, and package development. Hundreds of scattered vignettes, web pages, and forums explain how to use R in particular domains. But little has been written on how to simply make R work effectively—until now. This hands-on book teaches novices and experienced R users how to write efficient R code. Drawing on years of experience teaching R courses, authors Colin Gillespie and Robin Lovelace provide practical advice on a range of topics—from optimizing the set-up of RStudio to leveraging C++—that make this book a useful addition to any R user’s bookshelf. Academics, business users, and programmers from a wide range of backgrounds stand to benefit from the guidance in *Efficient R Programming*. Get advice for setting up an R programming environment Explore general programming concepts and R coding techniques Understand the ingredients of an efficient R workflow Learn how to efficiently read and write data in R Dive into data carpentry—the vital skill for cleaning raw data Optimize your code with profiling, standard tricks, and other methods Determine your hardware capabilities for handling R computation Maximize the benefits of collaborative R programming Accelerate your transition from R hacker to R programmer

Get to grips with various performance improvement techniques such as concurrency, lock-free programming, atomic operations,

parallelism, and memory management

Key Features

Understand the limitations of modern CPUs and their performance impact
Find out how you can avoid writing inefficient code and get the best optimizations from the compiler
Learn the tradeoffs and costs of writing high-performance programs

Book Description

The great free lunch of "performance taking care of itself" is over. Until recently, programs got faster by themselves as CPUs were upgraded, but that doesn't happen anymore. The clock frequency of new processors has almost peaked. New architectures provide small improvements to existing programs, but this only helps slightly. Processors do get larger and more powerful, but most of this new power is consumed by the increased number of processing cores and other "extra" computing units. To write efficient software, you now have to know how to program by making good use of the available computing resources, and this book will teach you how to do that. The book covers all the major aspects of writing efficient programs, such as using CPU resources and memory efficiently, avoiding unnecessary computations, measuring performance, and how to put concurrency and multithreading to good use. You'll also learn about compiler optimizations and how to use the programming language (C++) more efficiently. Finally, you'll understand how design decisions impact performance. By the end of this book, you'll not only have enough knowledge of processors and compilers to write efficient programs, but you'll also be able to understand which techniques to use and what to measure while improving performance. At its core, this book is about learning how to learn. What you will learn

- Discover how to use the hardware computing resources in your programs effectively
- Understand the relationship between memory order and memory barriers
- Familiarize yourself with the performance implications of different data structures and organizations
- Assess the performance impact of concurrent memory accessed and how to minimize it
- Discover when to use and when not to use lock-free programming techniques
- Explore different ways to improve the effectiveness of compiler optimizations
- Design APIs for concurrent data structures and high-performance data structures to avoid inefficiencies

Who this book is for
This book is for experienced developers and programmers who work on performance-critical projects and want to learn different techniques to improve the performance of their code. Programmers who belong to algorithmic trading, gaming, bioinformatics, computational genomics, or computational fluid dynamics communities can learn various techniques from this book and apply them in their domain of work. Although this book uses the C++ language, the concepts demonstrated in the book can be easily transferred or applied to other compiled languages such as C, Java, Rust, Go, and more.

This guide will help readers learn how to employ the significant power of use cases to their software development efforts. It provides a practical methodology, presenting key use case concepts.

Programming in TCP/IP can seem deceptively simple. Nonetheless, many network programmers recognize that their applications could be much more robust. *Effective TCP/IP Programming* is designed to boost programmers to a higher level of competence by focusing on the protocol suite's more subtle features and techniques. It gives you the know-how you need to produce highly effective TCP/IP programs. In forty-four concise, self-contained lessons, this book offers experience-based tips, practices, and rules of thumb for learning high-performance TCP/IP programming techniques. Moreover, it shows you how to avoid many of

TCP/IP's most common trouble spots. Effective TCP/IP Programming offers valuable advice on such topics as: Exploring IP addressing, subnets, and CIDR Preferring the sockets interface over XTI/TLI Using two TCP connections Making your applications event-driven Using one large write instead of multiple small writes Avoiding data copying Understanding what TCP reliability really means Recognizing the effects of buffer sizes Using tcpdump, traceroute, netstat, and ping effectively Numerous examples demonstrate essential ideas and concepts. Skeleton code and a library of common functions allow you to write applications without having to worry about routine chores. Through individual tips and explanations, you will acquire an overall understanding of TCP/IP's inner workings and the practical knowledge needed to put it to work. Using Effective TCP/IP Programming, you'll speed through the learning process and quickly achieve the programming capabilities of a seasoned pro.

"I don't even feel like I've scratched the surface of what I can do with Python" With Python Tricks: The Book you'll discover Python's best practices and the power of beautiful & Pythonic code with simple examples and a step-by-step narrative. You'll get one step closer to mastering Python, so you can write beautiful and idiomatic code that comes to you naturally. Learning the ins and outs of Python is difficult-and with this book you'll be able to focus on the practical skills that really matter. Discover the "hidden gold" in Python's standard library and start writing clean and Pythonic code today. Who Should Read This Book: If you're wondering which lesser known parts in Python you should know about, you'll get a roadmap with this book. Discover cool (yet practical!) Python tricks and blow your coworkers' minds in your next code review. If you've got experience with legacy versions of Python, the book will get you up to speed with modern patterns and features introduced in Python 3 and backported to Python 2. If you've worked with other programming languages and you want to get up to speed with Python, you'll pick up the idioms and practical tips you need to become a confident and effective Pythonista. If you want to make Python your own and learn how to write clean and Pythonic code, you'll discover best practices and little-known tricks to round out your knowledge. What Python Developers Say About The Book: "I kept thinking that I wished I had access to a book like this when I started learning Python many years ago." - Mariatta Wijaya, Python Core Developer "This book makes you write better Python code!" - Bob Belderbos, Software Developer at Oracle "Far from being just a shallow collection of snippets, this book will leave the attentive reader with a deeper understanding of the inner workings of Python as well as an appreciation for its beauty." - Ben Felder, Pythonista "It's like having a seasoned tutor explaining, well, tricks!" - Daniel Meyer, Sr. Desktop Administrator at Tesla Inc.

A detailed introduction to the C programming language for experienced programmers. The world runs on code written in the C programming language, yet most schools begin the curriculum with Python or Java. Effective C bridges this gap and brings C into the modern era--covering the modern C17 Standard as well as potential C2x features. With the aid of this instant classic, you'll soon be writing professional, portable, and secure C programs to power robust systems and solve real-world problems. Robert C. Seacord introduces C and the C Standard Library while addressing best practices, common errors, and open debates in the C community. Developed together with other C Standards committee experts,

Effective C will teach you how to debug, test, and analyze C programs. You'll benefit from Seacord's concise explanations of C language constructs and behaviors, and from his 40 years of coding experience. You'll learn:

- How to identify and handle undefined behavior in a C program
- The range and representations of integers and floating-point values
- How dynamic memory allocation works and how to use nonstandard functions
- How to use character encodings and types
- How to perform I/O with terminals and filesystems using C Standard streams and POSIX file descriptors
- How to understand the C compiler's translation phases and the role of the preprocessor
- How to test, debug, and analyze C programs

Effective C will teach you how to write professional, secure, and portable C code that will stand the test of time and help strengthen the foundation of the computing world.

The Go Programming Language is the authoritative resource for any programmer who wants to learn Go. It shows how to write clear and idiomatic Go to solve real-world problems. The book does not assume prior knowledge of Go nor experience with any specific language, so you'll find it accessible whether you're most comfortable with JavaScript, Ruby, Python, Java, or C++. The first chapter is a tutorial on the basic concepts of Go, introduced through programs for file I/O and text processing, simple graphics, and web clients and servers. Early chapters cover the structural elements of Go programs: syntax, control flow, data types, and the organization of a program into packages, files, and functions. The examples illustrate many packages from the standard library and show how to create new ones of your own. Later chapters explain the package mechanism in more detail, and how to build, test, and maintain projects using the go tool. The chapters on methods and interfaces introduce Go's unconventional approach to object-oriented programming, in which methods can be declared on any type and interfaces are implicitly satisfied. They explain the key principles of encapsulation, composition, and substitutability using realistic examples. Two chapters on concurrency present in-depth approaches to this increasingly important topic. The first, which covers the basic mechanisms of goroutines and channels, illustrates the style known as communicating sequential processes for which Go is renowned. The second covers more traditional aspects of concurrency with shared variables. These chapters provide a solid foundation for programmers encountering concurrency for the first time. The final two chapters explore lower-level features of Go. One covers the art of metaprogramming using reflection. The other shows how to use the unsafe package to step outside the type system for special situations, and how to use the cgo tool to create Go bindings for C libraries. The book features hundreds of interesting and practical examples of well-written Go code that cover the whole language, its most important packages, and a wide range of applications. Each chapter has exercises to test your understanding and explore extensions and alternatives. Source code is freely available for download from <http://gopl.io/> and may be conveniently fetched, built, and installed using the go get command.

This easy-to-follow and classroom-tested textbook guides the reader through the fundamentals of programming with Python, an accessible language which can be learned incrementally. Features: includes numerous examples and practice exercises throughout the text, with additional exercises, solutions and review questions at the end of each chapter; highlights the patterns which frequently appear when writing programs, reinforcing the application of these patterns for problem-solving through practice exercises; introduces the use of a debugger tool to inspect a program, enabling students to discover for themselves how programs work and enhance their understanding; presents the Tkinter framework for building graphical user interface applications and event-driven programs; provides instructional videos and additional information for students, as well as support materials for instructors, at an associated website.

More than 150,000 copies in print! Praise for Scott Meyers' first book, *Effective C++*: "I heartily recommend *Effective C++* to anyone who aspires to mastery of C++ at the intermediate level or above." – *The C/C++ User's Journal* From the author of the indispensable *Effective C++*, here are 35 new ways to improve your programs and designs. Drawing on years of experience, Meyers explains how to write software that is more effective: more efficient, more robust, more consistent, more portable, and more reusable. In short, how to write C++ software that's just plain better. *More Effective C++* includes: Proven methods for improving program efficiency, including incisive examinations of the time/space costs of C++ language features Comprehensive descriptions of advanced techniques used by C++ experts, including placement new, virtual constructors, smart pointers, reference counting, proxy classes, and double-dispatching Examples of the profound impact of exception handling on the structure and behavior of C++ classes and functions Practical treatments of new language features, including `bool`, `mutable`, `explicit`, namespaces, member templates, the Standard Template Library, and more. If your compilers don't yet support these features, Meyers shows you how to get the job done without them. *More Effective C++* is filled with pragmatic, down-to-earth advice you'll use every day. Like *Effective C++* before it, *More Effective C++* is essential reading for anyone working with C++.

An industry insider explains why there is so much bad software—and why academia doesn't teach programmers what industry wants them to know. Why is software so prone to bugs? So vulnerable to viruses? Why are software products so often delayed, or even canceled? Is software development really hard, or are software developers just not that good at it? In *The Problem with Software*, Adam Barr examines the proliferation of bad software, explains what causes it, and offers some suggestions on how to improve the situation. For one thing, Barr points out, academia doesn't teach programmers what they actually need to know to do their jobs: how to work in a team to create code that works reliably and can be maintained by somebody other than the original authors. As the size and complexity of commercial software have grown, the gap between academic computer science and industry has widened. It's an open secret that there is little engineering

in software engineering, which continues to rely not on codified scientific knowledge but on intuition and experience. Barr, who worked as a programmer for more than twenty years, describes how the industry has evolved, from the era of mainframes and Fortran to today's embrace of the cloud. He explains bugs and why software has so many of them, and why today's interconnected computers offer fertile ground for viruses and worms. The difference between good and bad software can be a single line of code, and Barr includes code to illustrate the consequences of seemingly inconsequential choices by programmers. Looking to the future, Barr writes that the best prospect for improving software engineering is the move to the cloud. When software is a service and not a product, companies will have more incentive to make it good rather than "good enough to ship."

Now in paperback, here's a guide to writing a full-length transformational nonfiction book, from an editor with two decades' experience working in publishing. "I know I have a book in me." "I've always wanted to be an author." "People always ask me when I'm going to write my book." "I have a story to tell, but I never seem to make time to write." Are you a thought leader, healer, or change-agent stuck at the starting line of book publication? Life coach and publishing industry insider Kelly Notaras offers a clear, step-by-step path for turning your transformational idea or story into a finished book as quickly as possible. With humor, encouragement, and common sense, she demystifies the publishing process so you can get started, keep writing, and successfully get your wisdom out into the world. Notaras guides you through: • Getting clear on your motivation for writing a book, • Crafting a powerful, compelling hook and strong internal book structure, • Overcoming resistance and writer's block, and • Getting your finished manuscript onto the printed page, whether through traditional publishing or self-publishing. Publishing a book has never been as simple, accessible, and affordable as it is today, and in our tumultuous world, readers need your healing voice. Be brave, be bold, and take the steps you need to share your message with those who need to hear it most.

If you want to push your Java skills to the next level, this book provides expert advice from Java leaders and practitioners. You'll be encouraged to look at problems in new ways, take broader responsibility for your work, stretch yourself by learning new techniques, and become as good at the entire craft of development as you possibly can. Edited by Kevlin Henney and Trisha Gee, *97 Things Every Java Programmer Should Know* reflects lifetimes of experience writing Java software and living with the process of software development. Great programmers share their collected wisdom to help you rethink Java practices, whether working with legacy code or incorporating changes since Java 8. A few of the 97 things you should know: "Behavior Is Easy, State Is Hard"—Edson Yanaga "Learn Java Idioms and Cache in Your Brain"—Jeanne Boyarsky "Java Programming from a JVM Performance Perspective"—Monica Beckwith "Garbage Collection Is Your Friend"—Holly K Cummins "Java's Unspeakable Types"—Ben Evans "The Rebirth of Java"—Sander

Mak "Do You Know What Time It Is?"—Christin Gorman

ABOUT THE BOOK Jeff Atwood began the Coding Horror blog in 2004, and is convinced that it changed his life. He needed a way to keep track of software development over time - whatever he was thinking about or working on. He researched subjects he found interesting, then documented his research with a public blog post, which he could easily find and refer to later. Over time, increasing numbers of blog visitors found the posts helpful, relevant and interesting. Now, approximately 100,000 readers visit the blog per day and nearly as many comment and interact on the site. Effective Programming: More Than Writing Code is your one-stop shop for all things programming. Jeff writes with humor and understanding, allowing for both seasoned programmers and newbies to appreciate the depth of his research. From such posts as "The Programmer's Bill of Rights" and "Why Cant Programmers... Program?" to "Working With the Chaos Monkey," this book introduces the importance of writing responsible code, the logistics involved, and how people should view it more as a lifestyle than a career. TABLE OF CONTENTS - Introduction - The Art of Getting Shit Done - Principles of Good Programming - Hiring Programmers the Right Way - Getting Your Team to Work Together - The Batcave: Effective Workspaces for Programmers - Designing With the User in Mind - Security Basics: Protecting Your Users' Data - Testing Your Code, So it Doesn't Suck More Than it Has To - Building, Managing and Benefiting from a Community - Marketing Weasels and How Not to Be One - Keeping Your Priorities Straight EXCERPT FROM THE BOOK As a software developer, you are your own worst enemy. The sooner you realize that, the better off you'll be. I know you have the best of intentions. We all do. We're software developers; we love writing code. It's what we do. We never met a problem we couldn't solve with some duct tape, a jury-rigged coat hanger and a pinch of code. But Wil Shipley argues that we should rein in our natural tendencies to write lots of code: The fundamental nature of coding is that our task, as programmers, is to recognize that every decision we make is a trade-off. To be a master programmer is to understand the nature of these trade-offs, and be conscious of them in everything we write. In coding, you have many dimensions in which you can rate code: Brevity of code Featurefulness Speed of execution Time spent coding Robustness Flexibility Now, remember, these dimensions are all in opposition to one another. You can spend three days writing a routine which is really beautiful and fast, so you've gotten two of your dimensions up, but you've spent three days, so the "time spent coding" dimension is way down. So, when is this worth it? How do we make these decisions? The answer turns out to be very sane, very simple, and also the one nobody, ever, listens to: Start with brevity. Increase the other dimensions as required by testing. I couldn't agree more. I've given similar advice when I exhorted developers to Code Smaller. And I'm not talking about a reductio ad absurdum contest where we use up all the clever tricks in our books to make the code fit into less physical space. I'm talking about practical, sensible strategies to reduce the volume of code an individual

programmer has to read to understand how a program works. Here's a trivial little example of what I'm talking about: `if (s == String.Empty)if (s == "")` It seems obvious to me that the latter case is... ..buy the book to read more!

The "Writing Idiomatic Python" book is finally here! Chock full of code samples, you'll learn the "Pythonic" way to accomplish common tasks. Each idiom comes with a detailed description, example code showing the "wrong" way to do it, and code for the idiomatic, "Pythonic" alternative. *This version of the book is for Python 3.3+. There is also a Python 2.7.3+ version available.* "Writing Idiomatic Python" contains the most common and important Python idioms in a format that maximizes identification and understanding. Each idiom is presented as a recommendation to write some commonly used piece of code. It is followed by an explanation of why the idiom is important. It also contains two code samples: the "Harmful" way to write it and the "Idiomatic" way. * The "Harmful" way helps you identify the idiom in your own code. * The "Idiomatic" way shows you how to easily translate that code into idiomatic Python. This book is perfect for you: * If you're coming to Python from another programming language * If you're learning Python as a first programming language * If you're looking to increase the readability, maintainability, and correctness of your Python code What is "Idiomatic" Python? Every programming language has its own idioms. Programming language idioms are nothing more than the generally accepted way of writing a certain piece of code. Consistently writing idiomatic code has a number of important benefits: * Others can read and understand your code easily * Others can maintain and enhance your code with minimal effort * Your code will contain fewer bugs * Your code will teach others to write correct code without any effort on your part

A guide to applying software design principles and coding practices to VHDL to improve the readability, maintainability, and quality of VHDL code. This book addresses an often-neglected aspect of the creation of VHDL designs. A VHDL description is also source code, and VHDL designers can use the best practices of software development to write high-quality code and to organize it in a design. This book presents this unique set of skills, teaching VHDL designers of all experience levels how to apply the best design principles and coding practices from the software world to the world of hardware. The concepts introduced here will help readers write code that is easier to understand and more likely to be correct, with improved readability, maintainability, and overall quality. After a brief review of VHDL, the book presents fundamental design principles for writing code, discussing such topics as design, quality, architecture, modularity, abstraction, and hierarchy. Building on these concepts, the book then introduces and provides recommendations for each basic element of VHDL code, including statements, design units, types, data objects, and subprograms. The book covers naming data objects and functions, commenting the source code, and visually presenting the code on the screen. All recommendations are supported by detailed rationales. Finally, the book explores two uses of VHDL: synthesis and testbenches. It examines the key characteristics of code intended for synthesis (distinguishing it from code meant for simulation) and then demonstrates the design and implementation of testbenches with a series of examples that verify different kinds of models, including combinational, sequential, and FSM code. Examples from the book are also available on a companion website, enabling the reader to experiment with the complete source code.

You need a graphical user interface, and it needs to run on multiple platforms. You don't have much time, and you're not a wizard with X/Motif, the Win32 GUI, or the Mac GUI. The project seems impossible, but with Tcl/Tk it's simple and fun. The Tcl scripting language and the Tk toolkit create a powerful programming environment for building graphical user interfaces. With two lines of code you can create a simple button; with two hundred lines of code, a desktop calculator; and with a thousand lines of code, an industrial-strength groupware calendar and

appointment minder. Your applications run on all of the major platforms: UNIX, Windows 95/NT, and Macintosh. You can even embed your programs in a Web page to make them available online. Mark Harrison and Michael McLennan, two noted Tcl/Tk experts, combine their extensive experience in this practical programming guide. It is ideal for developers who are acquainted with the basics of Tcl/Tk and are now moving on to build real applications. Effective Tcl/Tk Programming shows you how to build Tcl/Tk applications effectively and efficiently through plenty of real-world advice. It clarifies some of the more powerful aspects of Tcl/Tk, such as the packer, the canvas widget, and binding tags. The authors describe valuable design strategies and coding techniques that will make your Tcl/Tk projects successful. You will learn how to: Create interactive displays with the canvas widget Create customized editors with the text widget Create new geometry managers, like tabbed notebooks or paned windows Implement client/server architectures Handle data structures Interface with existing applications Package Tcl/Tk code into reusable libraries Deliver Tcl/Tk applications that are easy to configure and install Embed applications in a Web page Build applications that will run on multiple platforms Throughout the book, the authors develop numerous applications and a library of reusable components. Learn from their approach, follow their strategies, and steal their code for your own applications! But don't bother retyping all of the examples. 0201634740B04062001

A guide to writing computer code covers such topics as variable naming, presentation style, error handling, and security.

Lesk provides an accessible and thorough introduction to a subject which is becoming a fundamental part of biological science today. The text generates an understanding of the biological background of bioinformatics.

Web designers and programmers, add JavaScript to your HTML5 development toolkit without fear Modern websites are complex, and some of the most exciting features - things like geolocation, canvas, portability to mobile and more - require JavaScript to leverage what HTML5 can create. Don't know JavaScript? That's where HTML5 Programming with JavaScript For Dummies comes in. Rather than walking you through JavaScript as a programming language, it approaches JavaScript as a tool to help you enhance web pages. Helps web designers and programmers tap the full power of HTML5 by leveraging JavaScript, even if they have no prior knowledge of JavaScript Enables readers to produce modern websites with today's technology essentials: portability to mobile devices, animation, the use of Libraries, the Canvas component, and more Emphasizes practical uses and offers many short examples for illustration Discover basic to complex ways to use JavaScript programming syntax to harness the full power of HTML5 and CSS3.

Effective Programming More Than Writing Code Createspace Independent Pub

Looks at the principles and clean code, includes case studies showcasing the practices of writing clean code, and contains a list of heuristics and "smells" accumulated from the process of writing clean code.

Kotlin is a powerful and pragmatic language, but it's not enough to know about its features. We also need to know when they should be used and in what way. This book is a guide for Kotlin developers on how to become excellent Kotlin developers. It presents and explains in-depth the best practices for Kotlin development. Each item is presented as a clear rule of thumb, supported by detailed explanations and practical examples.

Used in numerous universities throughout the United States, Canada, Australia, and New Zealand, this book provides programming insights for educators, practitioners, and students. The book will present readers with the vital tools necessary in providing successful programs for their patrons.

A hands-on, problem-solving guide to Perl programming explains how to enhance one's proficiency in Perl and includes useful examples and code, as well as self-documenting, object-oriented modules available with Perl 5. Original. (Intermediate).

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. Summary 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. Feliene 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

Classical FORTRAN: Programming for Engineering and Scientific Applications, Second Edition teaches how to write programs in the Classical dialect of FORTRAN, the original and still most widely recognized language for numerical computing. This edition retains the conversational style of the original, along with its simple, carefully chosen subset language and its focus on floating-point calculations. New to the Second Edition Additional case study on file I/O More about CPU timing on Pentium processors

More about the g77 compiler and Linux With numerous updates and revisions throughout, this second edition continues to use case studies and examples to introduce the language elements and design skills needed to write graceful, correct, and efficient programs for real engineering and scientific applications. After reading this book, students will know what statements to use and where as well as why to avoid the others, helping them become expert FORTRAN programmers.

3- Informative Books in one Bundle! The Most Comprehensive JavaScript Beginners Guide on the Market! Have you ever wondered what allows people to be able to see different things on different websites? The answer is simple: JavaScript. Many websites are written in JavaScript so that you can be able to see what they are all about and what is going on in each of the sites. It is a language that can be written in many different formats so that different websites can use it for different purposes. JavaScript is able to do everything from creating a website to adding buttons and even disabling the ability to click on a button unless an option is chosen. While JavaScript is a multilayered language that will take some time to learn all of the levels of, the basics are quite simple. You can learn how to begin writing JavaScript by knowing only the basics, and you can build on your knowledge of the basics and what you initially learned. To get started with writing JavaScript, all you need to do is learn the beginning process. It is easy for you to do this if you have the right tools. This book will act as a way for you, as a beginner, to learn the process of JavaScript. While it will teach you some of the simplest JavaScript codes, it will not be overwhelming with codes. Instead, it will teach you what you need to know before you become a JavaScript expert and before you make the decision to truly dive into it. If you are ready to learn about JavaScript, what it can do and how you can get started, start this book right away. When you are finished, check out some of the other books in this series to learn more JavaScript codes and how to become a true professional who is great at writing JavaScript and can do more than you ever thought possible. Follow the series on an easy way to become a JavaScript expert! Sail Past the Beginners Level with these valuable tips! JavaScript is a language that you will always be able to learn more about and always be able to expand your knowledge of. Once you have learned the very basics of it, you should work to make sure that you are trying to find out as much as possible. JavaScript can be very rewarding, and you will need to be able to do as much as possible with it if you want to get the most benefit out of it. The tips and tricks that are contained in this book will give you some insight into what JavaScript is really capable of and what you can actually do with it if you learn as much as possible about it. There is a lot to learn, and you will be able to reap all of the benefits from JavaScript if you follow this book. The tips and tricks are designed not only to show you how to use the codes to build a beautiful interactive website but to also wow all of your visitors with everything that you have to offer on the website. Reading the book will not make you a JavaScript expert, but it will have you well on your way to being one. Read on for some of the best tips that are available and how you can make them work when you are trying to learn JavaScript in the easiest and most efficient way. Javascript- Simple and Effective Strategies: JavaScript isn't necessarily a simple language or a simple code to learn, but there are some very simple strategies that will get you to where you want to be with your JavaScript career. Following these strategies will allow you the chance to make sure that you are getting the most out of the JavaScript experience and the learning process that comes from it. Grab this 3-book bundle Today!

Jeff Atwood began the Coding Horror blog in 2004, and is convinced that it changed his life. He needed a way to keep track of software development over time - whatever he was thinking about or working on. He researched subjects he found interesting, then documented his research with a public blog post, which he could easily find and refer to later. Over time, increasing numbers of blog visitors found the posts helpful, relevant and interesting. Now, approximately 100,000 readers visit the blog per day and nearly as many comment and interact on the site. Effective Programming: More Than Writing Code is your one-stop shop for all things programming. Jeff writes with humor and understanding, allowing for both seasoned programmers and newbies to appreciate the depth of his research. From such posts as "The Programmer's Bill of Rights" and "Why Cant Programmers... Program?" to "Working With the Chaos Monkey," this book introduces the importance of writing responsible code, the logistics involved, and how people should view it more as a lifestyle than a career.

Learn to track, branch, merge, and manage code revisions for real-world development scenarios Key Features: Master Git and understand the significance of version control Get to grips with Git's typical workflows, advanced functions, and their implementations Understand important Git commands to manage your repository Book Description: Git is the most popular version control system in the world. It allows developers to keep up with frequent code changes in a project, ensures there are no code conflicts between the developers, and reverts to an older version of code when required. Git for Programmers comprehensively equips you with actionable insights on advanced Git concepts in an engaging and straightforward way. This book will help you gain expertise on Git with many practical use cases as you progress through the chapters. The book begins with a quick history of Git and instructions on how to get it and install it, after which you'll dive into the creation and cloning of your repository. As you progress through the book, you'll explore Git places, branching, and GUIs. Once you understand the fundamentals, you'll learn how to handle merge conflicts, rebase, amend, interactive rebase, and use the log. You'll also explore important Git commands for managing your repository. Finally, the book concludes with coverage of bisect, blame, and several other problem handling techniques. By the end of this book, you'll be able to use Git with confidence; saving, sharing, and managing files as well as undoing mistakes and rewriting history. What You Will Learn: Create and clone repositories Understand the difference between local and remote repositories Use, manage, and merge branches back into the main branch Utilize tools to manage merge conflicts Manage commits on your local machine through interactive rebasing Use the log to gain control over all the data in your repository Use bisect, blame, and other tools to undo Git mistakes Who this book is for: If you have basic understanding of Git and want to strengthen your command over advanced techniques and navigate different functions, this book is for you.

Modern Fortran teaches you to develop fast, efficient parallel applications using twenty-first-century Fortran. In this guide, you'll dive into Fortran by creating fun apps, including a tsunami simulator and a stock price analyzer. Filled with real-world use cases, insightful illustrations, and hands-on exercises, Modern Fortran helps you see this classic language in a whole new light. Summary Using Fortran, early and accurate forecasts for hurricanes and other major storms have saved thousands of lives. Better designs for ships, planes, and automobiles have made travel safer, more efficient, and less expensive than ever before. Using Fortran, low-

level machine learning and deep learning libraries provide incredibly easy, fast, and insightful analysis of massive data. Fortran is an amazingly powerful and flexible programming language that forms the foundation of high performance computing for research, science, and industry. And it's come a long, long way since starting life on IBM mainframes in 1956. Modern Fortran is natively parallel, so it's uniquely suited for efficiently handling problems like complex simulations, long-range predictions, and ultra-precise designs. If you're working on tasks where speed, accuracy, and efficiency matter, it's time to discover—or re-discover—Fortran..

About the technology For over 60 years Fortran has been powering mission-critical scientific applications, and it isn't slowing down yet! Rock-solid reliability and new support for parallel programming make Fortran an essential language for next-generation high-performance computing. Simply put, the future is in parallel, and Fortran is already there. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About the book Modern Fortran teaches you to develop fast, efficient parallel applications using twenty-first-century Fortran. In this guide, you'll dive into Fortran by creating fun apps, including a tsunami simulator and a stock price analyzer. Filled with real-world use cases, insightful illustrations, and hands-on exercises, Modern Fortran helps you see this classic language in a whole new light. What's inside Fortran's place in the modern world Working with variables, arrays, and functions Module development Parallelism with coarrays, teams, and events Interoperating Fortran with C About the reader For developers and computational scientists. No experience with Fortran required. About the author Milan Curcic is a meteorologist, oceanographer, and author of several general-purpose Fortran libraries and applications. Table of Contents PART 1 - GETTING STARTED WITH MODERN FORTRAN 1 Introducing Fortran 2 Getting started: Minimal working app PART 2 - CORE ELEMENTS OF FORTRAN 3 Writing reusable code with functions and subroutines 4 Organizing your Fortran code using modules 5 Analyzing time series data with arrays 6 Reading, writing, and formatting your data PART 3 - ADVANCED FORTRAN USE 7 Going parallel with Fortan coarrays 8 Working with abstract data using derived types 9 Generic procedures and operators for any data type 10 User-defined operators for derived types PART 4 - THE FINAL STRETCH 11 Interoperability with C: Exposing your app to the web 12 Advanced parallelism with teams, events, and collectives

Want to tap the power behind search rankings, product recommendations, social bookmarking, and online matchmaking? This fascinating book demonstrates how you can build Web 2.0 applications to mine the enormous amount of data created by people on the Internet. With the sophisticated algorithms in this book, you can write smart programs to access interesting datasets from other web sites, collect data from users of your own applications, and analyze and understand the data once you've found it. Programming Collective Intelligence takes you into the world of machine learning and statistics, and explains how to draw conclusions about user experience, marketing, personal tastes, and human behavior in general -- all from information that you and others collect every day. Each algorithm is described clearly and concisely with code that can immediately be used on your web site, blog, Wiki, or specialized application. This book explains: Collaborative filtering techniques that enable online retailers to recommend products or media Methods of clustering to detect groups of similar items in a large dataset Search engine features -- crawlers, indexers, query engines, and the PageRank algorithm Optimization algorithms that search millions of possible solutions

to a problem and choose the best one Bayesian filtering, used in spam filters for classifying documents based on word types and other features Using decision trees not only to make predictions, but to model the way decisions are made Predicting numerical values rather than classifications to build price models Support vector machines to match people in online dating sites Non-negative matrix factorization to find the independent features in a dataset Evolving intelligence for problem solving -- how a computer develops its skill by improving its own code the more it plays a game Each chapter includes exercises for extending the algorithms to make them more powerful. Go beyond simple database-backed applications and put the wealth of Internet data to work for you. "Bravo! I cannot think of a better way for a developer to first learn these algorithms and methods, nor can I think of a better way for me (an old AI dog) to reinvigorate my knowledge of the details." -- Dan Russell, Google "Toby's book does a great job of breaking down the complex subject matter of machine-learning algorithms into practical, easy-to-understand examples that can be directly applied to analysis of social interaction across the Web today. If I had this book two years ago, it would have saved precious time going down some fruitless paths." -- Tim Wolters, CTO, Collective Intellect

Python's simplicity lets you become productive quickly, but this often means you aren't using everything it has to offer. With this hands-on guide, you'll learn how to write effective, idiomatic Python code by leveraging its best—and possibly most neglected—features. Author Luciano Ramalho takes you through Python's core language features and libraries, and shows you how to make your code shorter, faster, and more readable at the same time. Many experienced programmers try to bend Python to fit patterns they learned from other languages, and never discover Python features outside of their experience. With this book, those Python programmers will thoroughly learn how to become proficient in Python 3. This book covers: Python data model: understand how special methods are the key to the consistent behavior of objects Data structures: take full advantage of built-in types, and understand the text vs bytes duality in the Unicode age Functions as objects: view Python functions as first-class objects, and understand how this affects popular design patterns Object-oriented idioms: build classes by learning about references, mutability, interfaces, operator overloading, and multiple inheritance Control flow: leverage context managers, generators, coroutines, and concurrency with the concurrent.futures and asyncio packages Metaprogramming: understand how properties, attribute descriptors, class decorators, and metaclasses work

Widely considered one of the best practical guides to programming, Steve McConnell's original CODE COMPLETE has been helping developers write better software for more than a decade. Now this classic book has been fully updated and revised with leading-edge practices—and hundreds of new code samples—illustrating the art and science of software construction. Capturing the body of knowledge available from research, academia, and everyday commercial practice, McConnell synthesizes the most effective techniques and must-know principles into clear, pragmatic guidance. No matter what your experience level, development environment, or project size, this book will inform and stimulate your thinking—and help you build the highest quality code. Discover the timeless techniques and strategies that help you: Design for minimum complexity and maximum creativity Reap the benefits of collaborative development Apply defensive programming techniques to reduce and flush out errors Exploit opportunities to

refactor—or evolve—code, and do it safely Use construction practices that are right-weight for your project Debug problems quickly and effectively Resolve critical construction issues early and correctly Build quality into the beginning, middle, and end of your project

Are you looking for a deeper understanding of the Java™ programming language so that you can write code that is clearer, more correct, more robust, and more reusable? Look no further! Effective Java™, Second Edition, brings together seventy-eight indispensable programmer's rules of thumb: working, best-practice solutions for the programming challenges you encounter every day. This highly anticipated new edition of the classic, Jolt Award-winning work has been thoroughly updated to cover Java SE 5 and Java SE 6 features introduced since the first edition. Bloch explores new design patterns and language idioms, showing you how to make the most of features ranging from generics to enums, annotations to autoboxing. Each chapter in the book consists of several “items” presented in the form of a short, standalone essay that provides specific advice, insight into Java platform subtleties, and outstanding code examples. The comprehensive descriptions and explanations for each item illuminate what to do, what not to do, and why. Highlights include: New coverage of generics, enums, annotations, autoboxing, the for-each loop, varargs, concurrency utilities, and much more Updated techniques and best practices on classic topics, including objects, classes, libraries, methods, and serialization How to avoid the traps and pitfalls of commonly misunderstood subtleties of the language Focus on the language and its most fundamental libraries: java.lang, java.util, and, to a lesser extent, java.util.concurrent and java.io Simply put, Effective Java™, Second Edition, presents the most practical, authoritative guidelines available for writing efficient, well-designed programs.

[Copyright: 46f027697ee5a0d4539727115da22e59](#)