5 Books That Can Help You Become a Better Programmer

What to read when you're a novice coder who wants to grow

Matthew MacDonald
Aug 9, 2019 · 7 min read ★

Image by IvanPais from Pixabay

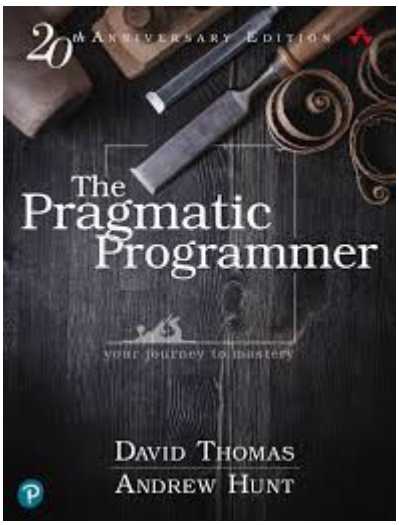
I love books. There's something about the unhurried pace of ink on paper, the way it lays out its concepts in neat, orderly pages.

But tech books have some serious problems. They're out of date almost immediately. They have no interactivity, and therefore no way to demonstrate concepts in action (or to let you *play* with those concepts). They have relatively low bandwidth — for example, long code examples need to be broken down into bite-sized chunks, or reading them is a chore. And don't even get me started about black-and-white screenshots.

But some tech books defy the odds and stay useful for years. These gems don't teach the technical details of the latest programming framework. Instead, they illuminate the philosophy of software development. They teach the craft of good code and good design. They tell you what it's like to work in the software industry, and they show you what it *means* to be a programmer. In this article, you'll meet five of my favorite examples.

1. The Pragmatic Programmer

Can a programming book still teach 20 years after its first edition? If can if it's packed with timeless wisdom about the art of writing good code.



The Pragmatic Programmer is the sort of book that inspires programmers to think more deeply about how they approach challenges, work with other people, and become more effective coders. It explains how to fight *software rot* — the powerful force that drags the code of every big project into an increasingly disordered mess. It gives a basic introduction to unit testing and refactoring, and sound advice about debugging (don't panic!) that just might help you solve problems without stepping on any oversized programmer egos. And *The Pragmatic Programmer* will help you pick up some cool programmer lingo, including one of my favorite terms: "orthogonality."

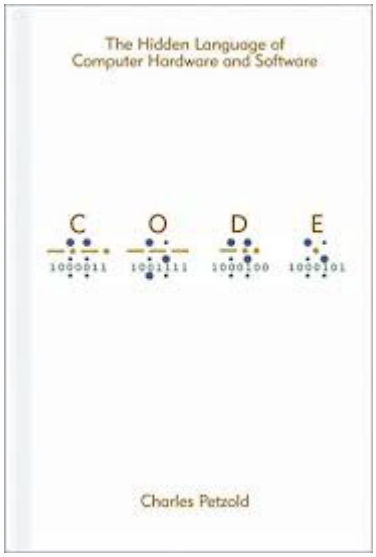
At roughly 300 pages, *The Pragmatic Programmer* doesn't seem particularly long, but it is dense. Fortunately, you don't need to read it cover to cover. Any time you crack open the cover and dip a toe in, you're likely to come away with some practical advice.

Runner ups: There's a small group of time-tested books about code philosophy that everyone likes to recommend (or at least display on their desks) alongside *The Pragmatic Programmer.* One of them is Bill Gates's favorite bedtime story, Code

Complete. Like *The Pragmatic Programmer*, it's packed with valuable advice, but it's also much longer and a bit drier, which makes it read more like a textbook. Another quality book, Clean Code, covers some of the same ground and is definitely worth a read, although it's starting to show the signs of its age and its tight focus on Java.

2. Code: The Hidden Language of Computer Hardware and Software

What if you wrote a book that was equally interesting to a programmer and the average curious layperson? Code is that book — an ambitious exploration of how computers perform their miracles.



It's difficult to explain *Code* without seeing it for yourself. We've all read books about how computers work, often with cute diagrams and long lines of 1s and 0s. But they aren't like this book. *Code* is a deep dive into computing that begins with Morse code and eventually reaches CPU schematics. It reads like a novel — albeit one with very detailed asides.

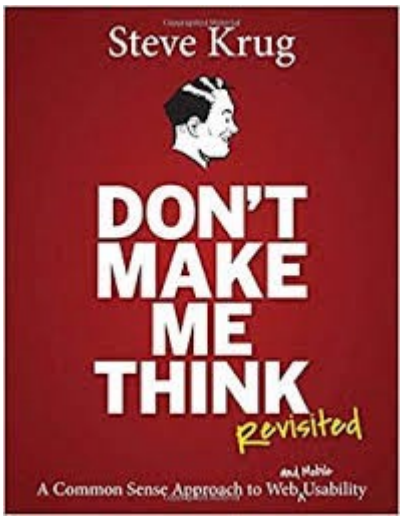
The central theme of *Code* is the way information is encoded in electronics. Often, *Code* seems on the brink of verging into one-too-many interesting history lessons, only to suddenly turn relevant by pulling back the curtain and showing you that you've been learning about modern computers all along. It's written by coding legend Charles Petzold, one of the first trailblazers to explain to Windows API to frustrated coders. ("Look it up in Petzold" was the mantra of the time.) Several decades later, he gave a similarly detailed exploration of 3D programming in WPF — a fascinating but somewhat cumbersome technology that was ignored by virtually everyone.

The bottom line? If you're curious about what's going on inside your electronics and why code is the way it is, *Code* is the book that can illuminate the mystery.

3. Don't Make Me Think

Here's a truth about computer programming. Sooner or later, every programmer will end up designing or implementing a user interface, whether they want to or not. And even though every programmer is also a user, and even though it seems like a straightforward task to make a logical, usable interface, odds are your first attempts won't be very good.

Design issues are like icebergs — they seem small on the top, but are unexpectedly vast and treacherous down below. Many programmers resent design because it takes them outside of their toolkit of unique skills, and it has an uncomfortable way of exposing bad decisions. That's where the wildly popular guide Don't Make Me Think comes in.

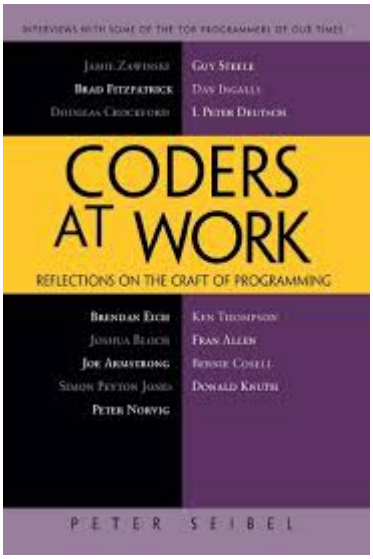


Don't Make Me Think is an exploration of good design. It's packed with website examples, but its advice applies to any kind of software — or any kind of product that one human makes for another human. The real trick is the way that *Don't Make Me Think* unlocks that bit of knowledge you already had, deep down, and shows you how to put it into practice. And it just so happens that learning to think like a usability expert is fun. Lots of fun, in fact. *Don't Make Me Think* is easily the breeziest, most entertaining book in this list.

4. Coders At Work

Talking about code is important, but talking *with* coders is essential. That's the idea behind Coders At Work, a collection of interviews with 15 seriously brilliant

programmers.



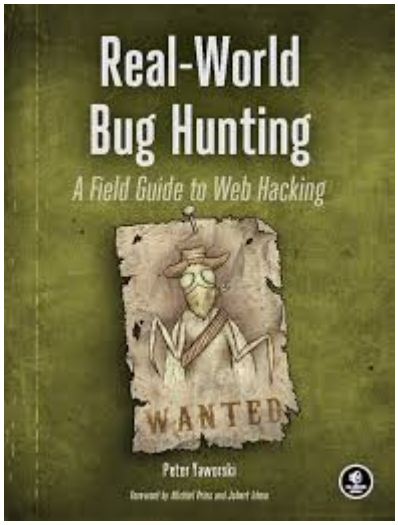
Most working coders can share some useful advice. But *Coders At Work* brings you to the table with luminaries like Brendan Eich, the creator of JavaScript, and Ken Thompson, the designer of Unix. It's like you're having an informal conversation with some of the most successful programmers on the planet. You'll learn how they approach challenges, how they solve problems, how they write code, and what they predict for the future. It's also interesting to see where their approaches diverge — sometimes dramatically. For example, Dan Ingalls talks about how he begins coding immediately when facing a new challenge, while Joshua Block carefully maps out an API before writing a single line of code. Donald Knuth goes even farther, and describes how he wrote a complete implementation of the typesetting system TeX in pencil before approaching a computer.

Because each chapter in *Coders At Work* is written in interview form, it feels more like a series of magazine articles than a weighty book about programming theory. But the more time you spend in the company of these programmers, the more you'll learn about what it's like to be a programmer working at the forefront of a number of different fields.

5. Real-World Bug Hunting

No list of programming books would be complete without a deep dive into some aspect of security. There are plenty of good topics to cover. Every programmer could benefit from learning the underpinnings of encryption, the way hackers penetrate networks, and the best practices that can help code defend itself against attacks. But if you want to

cover a lot of ground but not get bogged down with too much theory, Real-World Bug Hunting is a nice place to start.



In *Real-World Bug Hunting*, you get a realistic description of the most common ways that bad people attack good programs. You'll see how poorly validated input can wipe out a database or compromise a website. You'll watch malicious websites spam good ones, and see hackers use leaky memory to take over computers. Some readers may find these scenarios just a shade too technical, but the book is filled with actual hacker exploits — and there's no better teacher than the real world.

Runner-ups: Every programmer should learn at least a bit about cryptography, the science of secure communication and identity verification. There are plenty of books on the subject, although some are ancient and others are intimidating textbooks stuffed with math. For a very light start, try the illustrated Manga Guide to Cryptography. For a more classic approach, read the seminal Cryptography Engineering, written by no less than three of the world's leading experts in cryptography. And if you want to take a break, I can heartily recommend The Cuckoo's Egg, a page-turner about astronomy nerd (and accidental computer expert) Clifford Stoll and his detection of a Russian hacker at Berkeley Lab. It's the closest you'll get to summer beach reading.

. . .

As a programmer, you'll always feel that you're just a half step in front of an oncoming tsunami of change and new technology. But it's important to pause every once in a while. Take a break from constantly catching up on new frameworks and languages.

Read one of these books. You'll get a broader perspective of the craft and career of software development. And you just might elevate your code.

. . .

If you enjoyed this article, check out 5 Famous Programming Quotes, Explained. And for a once-a-month email with our best tech stories, subscribe to the Young Coder newsletter.

Programming Software Development Software Engineering Education Code