Software Engineer roadmap via books

medium.com/@iamjwr/software-engineer-roadmap-via-books-a6aabdc2589c

Jake Russell 2021年12月20日

Top highlight

Jake Russell

Throughout my career, I've found two methods of learning far more useful than all the others: and This may very well apply to me in particular as methods of learning proficiently, but I can say with certainty that the topic I'm writing about here — reading — is immensely useful. Some engineers find video-based learning (or something else entirely) to provide the most value, but books are jammed pack with gold — chances are those videos are inspired in some sense by timeless books. If reading isn't your thing, you may want to skip this post.

A friend of mine recently decided to take up the journey of becoming a programmer. He's been passionate about tech longer than I have been, but never really got around to the software side. He purchased a mac mini, I pointed him to The Odin Project, and off he went.

Earlier this week he mentioned how he wants to supplement his learning, and a reddit thread he stumbled upon had a list of books that could help. I have over 50 technical books sitting on my shelf, so I figured I could give him a roadmap that I would follow (with books) if I was in his shoes.

The image above is a hand-picked selection in a specific order from top to bottom. The bottom section is where I would start, eventually ending at the top. The bottom is a good overview of what to expect in a career as a software engineer, and how to approach it. I read these far too late in my career, and a lot of mistakes could have been avoided. Things like, leaving a job; leading a team; being a good junior; where to focus my talents, can be found in these 2 books. A few honorable mentions would be *,* and . Both are great in their own right, but much more dense than my recommendations.

> Note: If you want to skip to the roadmap scroll down, all books in the image are linked and discussed in more detail.

The next section (working from the bottom up) is what I would classify as fundamentals. The middle book is the staple here, while the other two are more ancillary.is a mythical beast in our industry. You've probably heard about it somewhere. It's a tough read, but very rewarding. A good alternative is *The Elements of Computing Systems,* it basically has you build a working computer from the ground up. The other books in this section are more language references/resources. I chose JavaScript and Go as the two languages for the image, but this could be replaced with some staples for other languages (python, java, etc).

Now we're onto the third section, which in my opinion is the hardest to truly get right. I wholeheartedly believe that if you grokked all the books in this section you'd be a senior-level engineer. Design patterns, algorithms and data structures, TDD (or really, testing in general), and code cleanliness. All mostly overlooked and underappreciated, but so beneficial. There are a thousand books that could land in this section; the ones above just so happened to be on my bookshelf.

Finally, the big dogs. These are books I'm still struggling with. The one exception here is *Explain the Cloud Like I'm 10,* it's a joyful, simplistic read that I think every programmer should eventually read. The others though… whew. This is where you'll start reading about scalability, reliability, architecture, and how to write long-lasting code. Again, I could add plenty of books to this section. Frankly, these are enough.

I want to provide some links and a quick summary of each book in the roadmap. This should make it easy to find a substitute or even skip.

1. by Chad Fowler

This is first on the list because it's a super easy read, not too technical, and provides loads of value around how to build and structure your career. If you're entrepreneurial, you'll take to the book quickly. Essentially, it teaches you how to manage your career from a business perspective.

2. Building a Career in Software by Daniel Heller

Leading teams, switching jobs, promotions, staying sharp, you'll find it all in this book. I wish I had this as a resource to navigate my career optimally. The predicaments I found myself in over the years would have been far less harsh with this to reference.

Moving onto section 2, the fundamentals.

3. Elements of Computing Systems by Noam Nisan, Shimon Schocken

To this day, I haven't found a better book of exploring the art of programming and the rewarding outcomes. If you can make it through the book front-to-back, you'll learn so much about computers, how they work, and how to get them to do what you want.

4. You Don't Know JS by Kyle Simpson

These are free on Github, but I'd advocate for buying the hard copies. I think they're hands down the best JavaScript resources on the market (they are a series). He has great talks and even better books.

Note: As I said above, these should be replaced if you're learning another language (like python).

Section 3, the advanced stuff.

5. Grokking Algorithms by Aditya Bhargava

Cracking the Coding Interview is a fantastic resource for algorithms, but I feel it's more targeted for engineers mastering technical interviews. *Grokking Algorithms* is a smaller book that provides simpler explanations. This will eventually be necessary learning material (algo's and data structures) whether or not it's for an on-the-job task, or for interviewing.

6. Design Patterns by Gamma Erich, Helm Richard, Johnson Ralph, Vlissides John, Grady Booch

Now, this could very well come up in an interview, more likely though is you actually utilize patterns at work — sometimes you may not even know you're using one. Memorizing all of these is probably not the best approach, but knowing a few and when to use them will change your codebases for the better.

7. Test-Driven Development by Example by Kent Beck

You'll probably encounter this earlier in your journey (or at least the concept of testing). This book will solidify your knowledge of test-driven development and how to test your code.

8. Clean Code by Robert Martin

Code cleanliness is a hot topic in the programming world, and a lot of engineers have differing opinions on what's considered "clean." Uncle Bob does a good job of detailing some of the more proven practices when it comes to writing code.

Last section, the books I need to read 3 times over.

9. Explain the Cloud Like I'm 10 by Todd Hoff

A fun, short read. This book explains a variety of cloud concepts (and computing in general) with short, easy to read, summaries. You could give this to your dad and he'd come back with some new knowledge. I find the juicy bits near the end, but the whole book is great.

10. System Design Interview by Alex Xu

Alex does a great job of prepping engineers for the system design questions in interviews with this read. I'd consider this the *Cracking the Coding Interview* for system design questions. As an aside, this is a good book for system design overall.

11. Designing Data-Intensive Applications by Martin Kleppmann

A massive book, with a plethora of stories and practices on how to build reliable, scalable, data-intensive applications. I'm currently in the middle of this, and it's amazingly complex. If I was at the head of Netflix early on, this would be my go-to resource.

12. Domain-Driven Development Distilled by Vaughn Vernon

I heard about DDD early in my career from a senior I worked with. I never got around to working on an application that had DDD conventions, but I did architect a few hobby projects using DDD.

It's a useful technique for aligning the business and technical side of a product, and for the greenfield projects I've started, I wish I could go back and implement them with DDD in mind.

That's the list. I hope you find some of these books as useful in your journey as I have. I'd love suggestions to add to the list (I need an excuse to buy **even** more books), or alternatives.