Including tips by top developers from Google, Microsoft, Spotify, Amazon, and more

14 Habits of Highly Productive Developers

Zeno Rocha

Part One: Principles

Hello World

"There is no elevator to success, you have to take the stairs" — Zig Ziglar

I started developing software more than ten years ago. Since then, I've built many applications, created dozens of open source projects, and pushed thousands of commits. Besides that, I spoke in more than a hundred conferences, and had the opportunity to chat with a ton of developers along the way.

I was fortunate enough to be in contact with some of the best software engineers in the industry, but I also met a lot of programmers who are still doing the same thing for many years.

What separates one group from the other? What's unique about people who work in the biggest companies in our industry? What's special about individuals who create the most used applications in the world? How can some developers be so prolific at work and also outside their jobs?

These questions stayed on my head for a long time. I realized that I could buy the best mechanical keyboards, go to the most famous tech conferences in the world, and learn all the newest frameworks. Still, if I cultivated bad habits, it would be impossible to become a top developer. Because of that, I decided to reach out to the best developers I know and ask them tips on how to be more productive.

This book doesn't offer a straight path or pre-defined formula for success. This book is the result of a quest. A quest to uncover what habits can be cultivated to help become a better software engineer.

Methodology

This is not a traditional book. You won't find the same format or structure that a regular book has. In fact, this book was designed to be as simple and objective as possible. You can follow the order of chapters, or you can read them individually. Everything is standalone and doesn't depend on previous knowledge.

At the end of each habit, you'll find a section marked as **"Questions & Answers"**, where I interview senior developers and tech leads from various companies to understand how they got there. I went after tech giants such as Google, Amazon, Microsoft, and Adobe. Powerful startups such as GitHub, Spotify, Elastic, Segment, GoDaddy, and Shopify. All the way to established organizations such as Citibank, BlackBerry, and The New York Times.

These people come from all over the world and have a pretty diverse background. From San Francisco to New York. From São Paulo to Montreal. From London to Stockholm. The idea is to present you not a one man's point of view, but a collection of insights on how to navigate your career.

You'll also find sections marked as **"TODO"**, where I encourage you to reflect on certain topics and take action with specific directions. I highly recommend taking a few minutes to dive into them since this will generate even more knowledge.

And finally, you may find some sections marked as **"Bonus"**. These are extra content that I prepared for you. They can be found outside the book and will complement what you're reading.

Why Habits?

If you've ever tried to lose weight, you know how frustrating that entire process is. You can exercise as hard as you can for three hours, but if you do that only once in a week, it will have zero effect on you. What truly generates results is when you go multiple times per week. Then suddenly, a few months later, you'll start noticing changes in your body.

Consistency matters, and that same concept applies to your professional career as well. Things take time, and intensity is not always the answer. The habits you decide to cultivate (or not cultivate) will determine your future life opportunities. As described in the book *Atomic Habits*:

> *"Habits are the compound interest of self-improvement. The same way that money multiplies through compound interest, the effects of your habits multiply as you repeat them. They seem to make little difference on any given day and yet the impact they deliver over the months and years can be enormous. It is only when looking back two, five, or perhaps ten years later that the value of good habits and the cost of bad ones becomes strikingly apparent." — James Clear*

I often get emails from other software engineers asking what programming language they should learn. And while that's a very important question to ask, I feel like a more valuable question would be: "What habits do I need to cultivate in order to be effective in *any* programming language?"

That's why I decided to focus this book on habits instead of tactics.

Now let's dive in! Are you ready?

Habit 1: Look For The Signals

"The oldest, shortest words – yes and no –
are those which require the most thought." — Pythagoras

Every single day we're bombarded with tweets, newsletters, and videos telling us what we should do, what we should learn, what we should focus on. We are constantly faced with FOMO (the fear of missing out). What if we're wasting our time with the wrong programming language? What if the most productive framework is not the one that you're currently using?

For me personally, it all started with choosing the right Operating System (OS). At the time, I had a Windows machine, but everybody was telling me that Apple computers were better and I should switch. However, their prices were way out of my league, so getting one was not even a possibility. Fast forward a couple of years and one of my employers gave me the option to choose between Windows and MacOS. I went straight for the MacOS to see what the fuss was about and what was so incredible about it. After some time using MacOS, Ubuntu started to become very popular and everybody was telling me I should switch to Ubuntu. So I thought I'd give it a try and started using it. My point here is not to tell you which one you should choose, the point here is that **there's no such thing as the best tool**.

Instead, we should practice JOMO (the joy of missing out), which is mostly about being happy and content with what you already know. Give yourself some credit and look back on everything you learned so far. Of course, you shouldn't be complacent and stop studying new technologies. It would be best if you find some balance between practicing your existing skills and learning new ones.

People will try to convince you which is the best OS, the best programming language, the best framework. They will tell you about all the amazing things that *their* tool does and that *your* tool does not. The reality though is that every single tool is different and we are also different as users. What is best for you, may not be the best for me or for others.

Think of it as a radio station that you're trying to tune into. I know you probably don't use radios anymore, but stay with me. Imagine you turn the dial and it's just picking up static noise, but after a few frustrating seconds, it finally manages to pick up a signal and tune into a station. The signal is the meaningful information that you're actually interested in. The static noise is just the random, unwanted variation that interferes with the signal. That's why self-awareness is so important, you need to be able to identify what is the signal and what is just noise.

It's crucial to understand that the noise will always be there. You don't need to necessarily abandon social media, unsubscribe from all newsletters, and stop watching YouTube videos. A digital detox can definitely help for a while; however, it's not a long term solution. What you need to do is to cherry-pick what is relevant to you at this point in your career.

Accept the fact that you simply can't learn everything. **Remember, desires are endless; needs are limited**. Accept the fact that newer is not always better. There are people working with ancient programming languages and still making a lot of money. Practice daily the subtle art of saying 'no'. No to that newest library. No to that fancier platform. Say more 'noes' so you can say 'yes' to what really matters to you.

1 YES !== 1 NO
1 YES === MANY NOES

Hear the noise, but only pay attention to the signals.

// TODO

Create a list of all technologies and tools that you would like to learn. Now label each of them with a different priority: "This Week", "Next Month", "Next Year". Whenever you feel like you're missing out on some new shiny trend, revisit this list and reorganize the priority.

# Questions & Answers: How do you decide which technology to learn and invest time in?

Daniel Buchner (Microsoft):

> *"Earlier in my career, I remember paying close attention to every new framework that would land on the front page of Hacker News. I would read up on whatever unique approach they would take, and spend extra cycles tinkering with the framework or library to get a sense of it. This is perfectly fine if you have ample free time to explore and learn, but can often lead to fatigue, because the list of hot new things is never-ending.*
>
> *These days, I try to focus on a few key things that drive my evaluations:*
>
> *1. What are the absolute must-have technical requirements for whatever I am working on? This may include things like performance, desired UX, or interoperability with target systems.*
> *2. How easy is it going to be for others to work on what I am building, and what will it look like for them to integrate it in whatever projects they are working on?*
> *3. Is what I am doing aligned with the trajectory of the open web, standards, and specifications I believe will stick around over the long term?*
>
> *The three points above tend to weed out many of the libraries, frameworks, and other utilities I come across. This has saved me time and allowed me to focus more on delivering what I need to, instead of getting caught up in dev-tourism."*

Addy Osmani (Google):

> *"Accept that you can't learn everything, but you can learn enough to be effective. When it looks like an idea, framework or technology is gaining some traction, I'll invest an afternoon in trying it out myself to get a sense of two things: 'Does it improve my productivity?' and 'Does it improve the user-experience of the types of projects I usually build?' If the answer to either of these is yes, I'll consider spending time learning about the framework or technology in more depth.*
>
> *Because our time is finite, there are plenty of technologies that I'll try, will find compelling enough to keep an eye on, but will make a trade-off about learning and investing in favor of something else. I think that this is healthy. I originally tried (and punted) on React the first year it came out, but now use it regularly. I like a lot of the ideas in Svelte, but because I had already learned React, Preact and Vue, decided that it was a better investment of time to level up in completely different areas with that time, like the Web Animations API.*
>
> *As I said, with time being finite, it's good to find a balance in what you choose to learn to keep yourself effective. When you pick just a few, high-impact choices you set yourself up to have enough time to get more depth in your technology of choice. This can be immensely useful when you're trying to build anything non-trivial in the real world."*

Thanks for taking the time to read this free chapter.
I can't wait for you to read the full book :)
Wanna help? Share it with your friends!

- Zeno