How You Can Become a Great Software Engineer

A Quick Guide

Fatos Morina

How You Can Become a Great Software Engineer

A Quick Guide

Fatos Morina

Contents

When some software engineers see an article mentioning the word *senior*, they may immediately run away and ignore it. They can be intimidated just by the title.

Yes, being a senior engineer requires a lot of time and experience, but you can still be a great engineer. A great junior engineer. A great intermediate engineer.

You write clean code. You follow best practices. You do a really good job.

Being good at what you do can help accelerate your path to becoming a senior engineer, or even a CTO.

Now, without further ado, let's get started.

The first thing that I believe you should keep in mind is that you must…

Take Responsibility for Everything in Your Life

Do not blame others for anything. Simply accept that you are the common denominator of all the problems and difficulties in your life.

Are you not understanding a particular design pattern?

Do not blame the course, the book, or the instructor. Learning is your responsibility.

Every teacher tries their best, and we live in this time when you have plenty of options to choose and learn from.

There is no excuse.

Did you make a commit last week that broke an important API call?

Even though this may be a rare occasion or something that you may not predict that could happen, you should still take responsibility.

First, admit that you made a mistake. Then, go and fix the problem.

Use this as an opportunity to learn and document what went wrong and what you could do better in the future.

Do you feel that you are not able to keep up the pace with new things?

I believe you already know that you are responsible for that as well.

It can be hard and very time consuming, especially if you have kids to look after and also want to have a social life. Still, your growth is your responsibility. You have to own that.

You should try to make time a few times a week to improve your skills, whether that's during your day job when there are no tasks to work on and you are free to learn, or before leaving for work.

You could try to wake up one hour earlier before work and invest that period of time in learning new things, or you could also set aside a few hours on a Sunday morning to do that.

Are you living too far away from your office and have to take a long commute on a daily basis?

Most likely it was your decision to live there, not your manager's decision.

Find another location that's closer. It may be more expensive, but you should consider that you are saving a lot of time and energy, which can outweigh money. You can invest that saved time into learning new things, which could as a result help you earn more.

Even if you cannot afford it, you could still make use of that time and not simply scroll on Hacker News or Reddit. You can listen to audiobooks or podcasts, read books, or even work on your side projects.

Learn the Fundamentals

Most people tend to avoid learning the fundamentals, as they may not see the fruits of their work in the short-term.

It is more satisfying to tell others that you have your application ready to be downloaded in Play Store rather than informing them that you are still focusing on the basics.

Be different than those people.

Learn the fundamentals of computer science — more specifically, data structures and algorithms — and learn them really well.

You have to become capable of implementing some of the most commonly used algorithms and data structures yourself without using a textbook.

When you finally understand them at a deep level, you will have gained the ability to solve real-world problems in software development in an intuitive way, which can be a much faster and more efficient approach.

According to the biography of Einstein[1] written by Walter Isaacson, Einstein believed that "intuition is nothing but the outcome of earlier intellectual experience." In software development, that prior intellectual experience requires a solid grounding in the basics.

As Linux kernel developer Linus Torvalds says[2]:

> "I will, in fact, claim that the difference between a bad programmer and a good one is whether he considers

[1] http://www.amazon.com/exec/obidos/ASIN/0743264746/makithecompsi-20

[2] https://www.happybearsoftware.com/how-learning-data-structures-and-algorithms-makes-you-a-better-developer

> his code or his data structures more important. Bad programmers worry about the code. Good programmers worry about data structures and their relationships."

Large tech companies such as Microsoft, Google, or Amazon recruit new software engineers by challenging them with hard technical interviews that mostly revolve around fundamental concepts.

In other words, no matter what type of programming language you use, your code is just the way to express the algorithms and the data structures beneath the surface.

Value Your Relationships With Others

No matter how hard you work, your relationships are extremely important.

No amount of money can buy the time you spend with your family or loved ones.

Sacrificing that time and instead simply trying to focus your whole life on software engineering will make you less happy, which can also make you feel less connected to others and more miserable. As a result, you are more likely to get fewer things done.

As the founder of YC Combinator Sam Altman puts[3] this:

> "Don't neglect your family and friends for the sake of productivity — that's a very stupid tradeoff (and very likely a net productivity loss, because you'll be less happy)."

In the end, we are human beings, not human doings.

We need time to recharge and connect with our loved ones.

Professional relationships

Whether or not you are the only engineer working on a project, you must not let your ego cause you to think only of yourself.

For engineers, ego is the worst enemy.

[3] https://blog.samaltman.com/productivity

Some engineers tend to think win/lose, in which others have to lose for someone to win. You think you are going to get promoted or get a raise if your colleagues look bad in front of your manager.

As a result, you will orient your actions toward showing off as much as you can, taking credit without merit, or belittling your colleagues, all to represent yourself as superior.

This will not only give you a superficial type of contentment (as you are continuously trying to belittle your colleagues), it'll also probably damage your team productivity as a whole, as you and your colleagues will not be willing to work together at all.

If you want to be an effective engineer, you need to set yourself apart from what the majority of people are doing.

In every situation, you should try to reach the best alternative for the whole team.

You should think about mutually beneficial solutions that will ultimately lead to a better long-term resolution, rather than if only one person in the situation gets their way.

Cultivate the habit of asking yourself questions like these: "What's in it for them that I can also benefit from?" "How can we both get some portions of what we want without damaging our relationship?"

You may sometimes have misunderstandings and short tempers with your colleagues, but this does not mean that the harmony and the relationships you have will be destroyed.

Your ability to deal with people is one of the key areas in your life.

Avoid arguments as much as you can and focus on long-term benefits.

When you begin with the end in mind, you do not want to be alone, even if you reach your biggest goals. Living on a big island with no one nearby is probably an adventure you do not want to experience.

There is no doubt that large projects are developed by a lot of people that work as a team.

Bill Gates started[4] Microsoft with Allen, who even came up with the name Microsoft.

Elon Musk created his first IT company, Zip2, with his brother Kimbal, and he then sold the company to AltaVista for $307 million in cash and $34 million in securities. Instagram was initially started[5] by Kevin Systrom and Mike Krieger.

Amazon has more than 341,000 employees[6]. Microsoft has more than 120,000 employees[7].

Google, after tons of research on building the perfect team, has found[8] that some of the most productive teams were the ones with an environment that cultivated synergy and embraced psychological safety.

Harvard Business School professor Amy Edmondson describes[9] psychological safety as "a sense of confidence that the team will not embarrass, reject, or punish someone for speaking up."

Everyone has the freedom to contribute, collaborate, and get involved in everything they want. Whenever you find yourself part of an argument with a colleague, try to understand their core interest. Address them so that they become reciprocally beneficial and productive.

In this way, your team can be a lot more productive by taking advantage of the strengths of every member. Teamwork is greater than the sum of its parts.

[4] https://www.theguardian.com/technology/2011/may/02/paul-allen-microsoft-bill-gates-ideas

[5] https://www.quora.com/Instagram-company/What-is-the-genesis-of-Instagram

[6] https://www.geekwire.com/2017/amazon-soars-340k-employees-adding-110k-people-single-year/

[7] https://news.microsoft.com/facts-about-microsoft/#3uT4yq8pCixa1Bui.97

[8] https://www.nytimes.com/2016/02/28/magazine/what-google-learned-from-its-quest-to-build-the-perfect-team.html

[9] https://www.nytimes.com/2016/02/28/magazine/what-google-learned-from-its-quest-to-build-the-perfect-team.html

If you want to be an effective programmer, try to help your team become more synergetic. That can be done by valuing every teammate's freedom and helping them feel very comfortable expressing different points of view without the fear of embarrassment. Allow everyone to contribute to and collaborate on everything they find interesting and worth contributing and paying attention to.

Be very kind.

Say thank you and please a lot.

Praise other people's efforts.

Apologize as soon as possible.

Focus on the Most Important Things

We live in a time of distractions, where it is very hard to resist the countless emails or Facebook and Slack messages that we receive on a regular basis. They may represent an important part of our work, as we need to understand the requirements before we start with the implementation.

Still, there may be occasions where our attention and energy is depleted by unnecessary interruptions, which will not only take us a lot of time but also increase our levels of guilt and stress.

We may then start to wonder why we were not able to finish what we were supposed to do.

Obviously, Einstein didn't have a cell phone, nor Slack, but he had his own types of distractions and had ways of dealing with them that we can learn from. His son reported that "even the loudest baby crying didn't seem to disturb Father," adding, "He could go on with his work completely impervious to noise."

His ability to concentrate deeply and work for long periods of time without interruption helped him achieve historical breakthroughs. "I was able to do a full day's work in only two or three hours. The remaining part of the day, I would work out my own ideas," says Einstein of managing his time at his intellectually unstimulating job at the Bern patent office, which gave him time to focus on more challenging engagements.

Software engineers can learn from Einstein and practice blocking off distractions. Don't pretend you are able to focus on the same level of intensity after countless sessions "just checking" Facebook messages!

Tim Ferriss, the author of the best-selling book, Tool of Titans[10], writes how "Single-tasking is a superpower." He adds, "In a digital economy, where distraction is a currency, if you can focus on one thing for three to five hours at a time and block out all the distractions, you have a huge leg up, you are able to do deep work and connect dots that are unconnected."

Try making a compromise with your manager about setting aside certain portions of the day for communication and invest the other parts in valuable and cognitively demanding endeavors, such as learning new concepts or technologies and writing code.

When we are able to distance ourselves from those interruptions and apply productive habits, we increase the chances of diving deeper into our learning, and as a result, we become a lot more productive.

The author of the popular book, "The 7 Habits of Highly Effective People[11]", Stephen Covey said it clearly, "Most of us spend too much time on what is urgent and not enough time on what is important." He further explained,

> "The key is not to prioritize what's on your schedule but to schedule your priorities."

Obviously, not every task has the same type of importance and urgency. Covey mentioned the order that we should follow to be the most effective:

1. Important and Urgent
2. Important and Not Urgent
3. Not Important and Urgent
4. Not Important and Not Urgent

[10] https://www.amazon.ca/Tools-Titans-Billionaires-World-Class-Performers/dp/1328683788/ref=tmm_hrd_swatch_0?_encoding=UTF8&qid=1603209632&sr=1-1

[11] https://www.franklincovey.com/the-7-habits.html

If you want to be an effective engineer, you need to make the most of your time by putting first things first on your agenda. You need to become a leader and manage your agenda according to the above ordering schema.

As Covey said, "The main thing is to keep the main thing the main thing."

There are many books and articles that mention "time management." In reality, you cannot manage time. You can manage only your priorities.

Prioritizing can seem difficult, as you may be afraid of missing out on other things. However, once you are used to doing it consistently and see the benefits and the emotional rewards that it brings, it gets easier and very pleasurable.

As Sam Altman puts[12] it:

> "It doesn't matter how fast you move if it's in a worthless direction. Picking the right thing to work on is the most important element of productivity and usually almost ignored."

[12] https://blog.samaltman.com/productivity

Love Learning

Software development is one of the most in-demand professions of our time. There are constantly new job openings, which attract both youngsters and already employed people from a wide range of professions.

They know that you can get a really good salary working as a software engineer, and consequently, they start their careers with great ambitions.

Even if you get a job in software development, however, you can only become great if you love learning and programming. It is one of the few professions in which you constantly have to learn something new. It's not just the existing concepts you need to know. You have to adapt to the constant invention of new technologies. You have to adapt to the changes in the market and learn whatever new things are currently considered as valuable.

You will have to consistently learn a lot. In order to keep your brain constantly engaged in learning, you will have to be curious and stay humble.

Often, programmers get employed on the basis of their existing knowledge and start to think they don't need to learn new things anymore. However, you will only be able to excel as a software engineer if you are curious and love learning, as the need to learn never ceases.

In November 1915, after writing only two pages of what he referred[13] to as "one of the most beautiful works of my life," Einstein sent his 11-year-old son Hans Albert a letter. He praised his son for his learning efforts, saying, "That is the way to learn the most…

[13] https://www.brainpickings.org/2013/06/14/einstein-letter-to-son/

When you are doing something with such enjoyment that you don't notice the time passes."

"Curiosity has its own reason for existing," Einstein explains. "One cannot help but be in awe when one contemplates the mysteries of eternity, of life, of the marvelous structure of reality."

Become curious about how a framework works and is structured. When you learn something thoroughly, you gain a clear picture of its mechanisms and functionalities.

Do not simply accept that scikit-learn is a great framework for machine learning — learn how it works, behind the curtains. Moreover, consider the possibility of contributing to and improving it, as it is open source.

Watch out too that you don't become obsessed with money. Of course, we all need money to pay our bills and buy food for ourselves and our family, but if you want to excel in software development, you need to cultivate a passion for learning and developing for the sake of it.

You need to love programming and feel grateful that you have the opportunity to positively impact the lives of millions of people with the lines of code that you write.

Einstein believed that "love is a better teacher than a sense of duty."

What we can learn from this as software engineers is the importance of not just working for a good salary but because we love learning and our work in general and have an intrinsic curiosity and drive that keeps us engaged after work or during the weekends when our boss is not watching.

Establish Effective Habits

I agree that some of these things are easier said than done, but I believe it is very important to apply them in our lives.

If you want to become a great software engineer, first you have to decide to do so.

As Stephen Covey said, “I am not a product of my circumstances. I am a product of my decisions.”

Excellence is not accidental but comes as a result of careful planning, hard work, and devoted dedication.

As Aristotle said, “We are what we repeatedly do. Excellence, then, is not an act but a habit.”