Don't Kid Yourself About Software Bootcamps

W. Adam Koszek

Koszek ORG

wojciech@koszek.com

2022-08-26T22:29:40Z

Abstract

Don't kid yourself about bootcamps. These intensive trainings last couple of weeks and promise marvels. It's not a good way to become a software developer. It's worthwhile experience and a good investment if you want to start, but assuming you can become a "software engineer" or even "software developer" during software bootcamp may end up in a surprise. I write this because I met enough people discouraged by software bootcamps, since they often end up disappointed and drop software work whatsoever. This is bad, so below you have some suggestions about bootcamps.

High-tech has this explosion of jobs which need people. There are electronics jobs for electrical engineer, robotics jobs for mechanical engineers and a lot of programming jobs for software engineers.

Everyone know shortcuts don't exist Now I'm sure you haven't seen any bootcamps for electrical engineers. The reason why is that it's hard to become one: you need a blend of math and physics with a great deal of effort put on practical implementation of circuits, chips and design. If you attend the university to become an electrical engineer, you're not having a vacation for 4 years of your bachelors studies. You're being trained hard to become an EE engineer. Assuming there'd be a magic shortcut to become an EE engineer in 12 weeks, people would jump on it right away. I can guaratee you that.

Mechanical engineers are no exception. Frankly my university was famous for mechanical engineering program. My father graduated from it in times when Google and photocopiers didn't exist, and people his age taught there. This transpired to the quality of ME at my school. While during my CS studies I had to take "Principles of Physics", both part 1 and 2, mechanical engineering people had parts 3, 4 and others. And while I had to take maybe 3 other similar classes (Solid State Physics for example), they studied this all day long, for 4 years. Same here, if shortcuts for getting 90% of mechanical engineering degree existed, I can easily name individual people who'd pay $50k to just do it.

Everyone says shortcuts are possible When it comes to software, many people somehow assume that you can get into software through a shortcut. That you can short-circuit normal engineering study program to something really short. Something that would extract an absolute essence, a golden rule of software, and just teach you that. Examples I see around are bootcamps that last for 12–20 weeks.

Think for a moment about your computer passion, interest and hobby as a relationship. You want to get to know the person who you got attracted by. Instead of committing 4 or 5 years of getting to know each other well enough to survive normal day-to-day life, you decide to get into an accelerated mode. A mode, where you spend 8 hours a day for 12 weeks with this person, and keep asking all possible questions out there. Do you think that would work?

Medical industry in the US has the same problem as high-tech industry: there's just not enough people graduating to fill primary care physician roles. The need for primary care physicians is so big that people look for ways to shorten the usual five year medical school program and three years of mandatory residency. But it's at most shortened to four years. Not several weeks. You wouldn't want to be taken care of by a doctor after 20 weeks of schooling, wouldn't you? Don't expect that people with industry software and production systems running and making money will let you operate in their hospital.

In other words: while shortcuts can be taken, and study programs compressed, there's a limit. There's a border which you can't cross while attempting to produce a human being capable of solving techical problems.

Do you need a bootcamp?

As I said at the beginning, bootcamp might be good for you if you know zero. When you hear people getting into the software industry from a bootcamp, I think they are exceptions. Probably they're smart enough that they didn't need a bootcamp anyway. Maybe they were just too lazy to find necessary books, materials and articles to self-study. You must remember that survivorship bias is a serious problem during pros/cons analysis, and it's pretty common in many domains. We won't know what happened with legions of people who spared $5–$20k and didn't get anywhere.

If you're persistent and don't want to invest so much into teaching help, do know you can get into software industry by yourself. It'll be hard and take longer time, but will be more successful.

Do this instead This is my "no shortcut" list of getting more competent at software.

BOOKS.

Learn enough to be able to program and start building software. The best case is when you can show it.

OPEN SOURCE.

Publish it on GitHub if possible. Don't just drop your program there–make it a product. Imagine that you'd like to charge money for your creation. It has to have a name, a copyright, well-written documentation. Everything must be well explained. If I go to see your software, I must know what it is, how it works and why it's there. What problem it's solving.

Bootcamps help here just a little bit, because they show you bite-sized pieces that you can understand. The problem is that you're solving someone's problem that you may not care about. It's like any other assignment. The studies on learning show you grasp concept fully if you work with them. Play with some of your ideas in your mind, ask yourself if software could be a solution to any of them. And if the answer is yes: act. Go and implement some of your ideas in code.

I'll be honest: even if your idea is simple, but is not a template from a bootcamp's assignment, I value it more. When I see a candidate showing his "project", and after searching in Google I see 97 other "real estate catalog" project from the same bootcamp, I question whether it's really a "creation" or just a copy.

And by all means: make it work. If what you're building is a website or a web application, it must be up and running at all times. There's nothing worse than showing something that doesn't work, and being questioned about this later. This will cost you money (books, VM costs, bandwidth costs), but treat it as a mandatory investment in yourself.

ALGORITHMS, PUZZLES, INTERVIEW QUESTIONS.

Want it or not, people will interview you in one way or another. Interview hacks range from studying from books like Cracking the Coding Interview to just studying actual algorithms Algorithms. Next if I were to give you a life hack, I suggest you interview with small teams, maybe startups. If possible, startups done with first time entrepreneurs. In the ideal world, you want a person just like you to interview you. This is one side of the scale. Another is an interview at Facebook or Google, where people will ask you tricky questions over the phone and figure out how much you know without even seeing you.

Summary

You won't be surprised to hear that there aren't many shortcuts in getting into one way or another. Make sure you study and do "high leverage" things. Doing your own projects, publishing them and writing articles about your creations I'd

consider high leverage, along with the study for the actual interviews. Just like a study with SAT or GRE exams, this will help you just a little, but is more likely to score you an interview with a company that would be your "foot in a door" of a software industry. Let me know if you need any other advices. If you do, I may end up squeezing the topic of your interest into my writing schedule.

Subscribe for updates

Once a month I send updates on the new content and hints for software engineers.