Learning By Doing

This blog post was translated by ChatGPT 4o.

Today, let's talk about "Learn By Doing". This concept is incredibly important. Learning itself is vital, but learning how to learn is even more crucial. I have found that to create something, learning by doing is the best approach. For example, to learn Android development, I must actually practice Android development by undertaking one challenging project after another.

Paul Graham discusses how computer science students should spend their time in university: by solving numerous difficult problems. Firstly, from the beginning to the end, tackling difficult problems allows you to learn a lot. For instance, working on a highly challenging project like developing a WeChat client would mean that by doing everything yourself, you would eventually know how to use Photoshop to process images and understand all aspects of Android. If you don't use third-party push services and instead write a real-time communication server yourself, you would learn even more.

After finishing a book, we might feel like we understand most of it. But this is far from enough. When I finished reading "Crazy Android Lectures," I felt like I knew quite a bit about Android programming, having covered the main points. It took me about 10 days to read the several hundred pages of the book. However, it took me four months to become proficient with the knowledge through hands-on practice. It was only after working on a few small projects and a larger one that I became somewhat familiar with Android development.

Reading books alone is insufficient. It is a superficial way of learning. Of course, since I only spent 10 days reading the book, it wasn't very time-consuming.

I've realized that most of the time spent on elementary, middle, and high school education might be wasted. These are all based on textbook education.

I once discussed with my uncle how lime plaster sticks to walls. Honestly, I still don't know. Does CaO react with O2 to form CaCO3, and then CaCO3 is a solid and quite hard? In daily life, I can't differentiate between CaO and CaCO3, even though I could balance their equations and know their properties through practice and exercises. But soon enough, I forget. It's been almost a year and a half since I left the high school exam, and I don't recall much. I scored 590 in the college entrance exam, but if I were to take it again now, I might score 300?

Luckily, in my three years of high school, I spent a considerable amount of time learning algorithms and programming, instead of just playing with boring textbooks and exams. However, it's a pity that I didn't learn more programming back then.

Although learning subjects like physics, chemistry, biology, math, and English also honed my patience and

perseverance for perfection, it also had its downsides. It made learning feel painful, knowledge seem boring, and everything about grades and scores. It affected my innate interests and curiosity, making me focus on rankings and who is better in programming, rather than enjoying the knowledge itself.

TJ (vision media) says, "I never go to school and I never read books. I learn programming by looking at other people's code, then out of curiosity, I always want to understand why."

Paul Graham says, "Ambitious adults focus their curiosity on a narrow field."

The more complex the knowledge, the less it can be driven by fame, reputation, or self-discipline, and the more it relies on spontaneous curiosity.

Since childhood, I have acquired knowledge more for fame, ranking, and being better than others. I crave external recognition. Of course, I also genuinely like programming.

The core driving force for me is that compared to some friends, I still feel inadequate. They are very impressive, the same age as me, second-year university students, aged 20, one writing compilers in his spare time and doing well in an outsourcing company, another has two companies and a full-time job handling backend development, and there's also a friend from Tsinghua with various world-class awards.

Clearly, I can feel the gap. It's not that they started learning earlier, but they are more efficient and spend more time every day than I do.

The pressure from peers of the same age, the desire to be better than peers, brings enormous motivation. One must admit. Colleagues are also very impressive. But subconsciously, I think they are older, and maybe I'll be like them at their age. Thinking this way makes it seem like nothing.

So, we should try to know people our age who are better than us. Although it's harsh on ourselves, continually admitting our inadequacy, the results are good. As long as you face everything bravely, don't give up on yourself, see the gap, and work silently, the results will be very good.

There might be more benefits. Because they are better, more excellent, and more skilled at something, they are more likely to genuinely enjoy it. They are more likely to love programming. Thus, they bring positive influences, allowing you to experience the joy of programming.

The good thing about programming is that once you get into it, you will involuntarily like it, forgetting about rankings, whether you are good, and fame and fortune.

I admit, I spend time programming because it's my job, and I have to report something to my boss. Secondly, I feel inadequate, and if I don't work hard, it will be too late. Thirdly, I have a dream of becoming a great programmer and then starting a full-stack business. But because I put in the effort, I can quickly feel the joy of creating things by doing it. The sense of achievement from solving one problem after another.

By getting hands-on, you can learn better and it is more fun.