How To Become a Better Programmer by Not Programming

Last year in Programmers as Human Beings, I mentioned that I was reading Programmers At Work. It's a great collection of interviews with famous programmers circa 1986. All the interviews are worth reading, but the interview with Bill Gates has one particular answer that cuts to the bone:

> *Does accumulating experience through the years necessarily make programming easier?*
>
> Bill Gates: No. **I think after the first three or four years, it's pretty cast in concrete whether you're a good programmer or not.** After a few more years, you may know more about managing large projects and personalities, but after three or four years, it's clear what you're going to be. There's no one at Microsoft who was just kind of mediocre for a couple of years, and then just out of the blue started optimizing everything in sight. I can talk to somebody about a program that he's written and know right away whether he's really a good programmer.

We already know there's a vast divide between those who can program and those who cannot.

But the dirty little secret of the software development industry is that this is also true *even for people who can program*: there's a vast divide between good developers and mediocre developers. A mediocre developer can program his or her heart out for four years, but that won't magically transform them into a good developer. And the good developers always seem to have a natural knack for the stuff from the very beginning.

I agree with Bill. From what I've seen, there's just no crossing the skill chasm as a software developer. You've either got it, or you don't. No amount of putting your nose to the grindstone will change that. But if you accept that premise, it also presents us with a paradox: if experience doesn't make you a better programmer, what does? Are our skill levels written in stone? Is it *impossible* to become a better programmer?

To answer that question, you have to consider the obsessive nature of programming itself. Good developers are good at programming. Really good at programming. You might even say fanatically good. If they're anything like me, they've spent nearly every waking moment in front of a computer for most of their lives. And naturally, they get better at it over time. Competent software developers have already mastered the skill of programming, which puts them in a very select club. But if you're already in the 97th percentile for programming aptitude, what difference does a few more percentile points really make in the big scheme of things?

The older I get, the more I believe that **the only way to become a better programmer is by *not programming***. You have to come up for air, put down the compiler for a moment, and take stock of what you're really doing. Code is important, but it's a small part of the overall process.

This piece in Design Observer offers a nice bit of related advice:

> Over the years, I came to realize that my best work has always involved subjects that interested me, or -- even better -- subjects about which I've become interested, and even passionate about, through the very process of doing design work. I believe I'm still passionate about graphic design. But the great thing about graphic design is that it is almost always about something else. Corporate law. Professional football. Art. Politics. Robert Wilson. And if I can't get excited about whatever that something else is, I really have trouble doing a good work as a designer. To me, the conclusion is inexcapable: the more things you're interested in, the better your work will be.

Passion for coding is a wonderful thing. But it's all too easy to mindlessly, reflexively entrench yourself deeper and deeper into a skill that you've already proven yourself more than capable at many times over. To truly become a better programmer, you have to to **cultivate passion for everything else that goes on *around* the programming.**

Bill Gates, in a 2005 interview, follows up in spirit to his 1986 remarks:

> The nature of these jobs is not just closing your door and doing coding, and it's easy to get that fact out. The greatest missing skill is somebody who's both good at understanding the engineering and who has good relationships with the hard-core engineers, and bridges that to working with the customers and the marketing and things like that. And so that sort of engineering management career track, even amongst all the people we have, we still fall short of finding people who want to do that, and so we often have to push people into it.
>
> I'd love to have people who come to these jobs wanting to think of it as an exercise in people management and people dynamics, as well as the basic engineering skills. That would be absolutely amazing.
>
> And we can promise those people within two years of starting that career most of what they're doing won't be coding, because there are many career paths, say, within that Microsoft Office group where you're part of creating this amazing product, you get to see how people use it, you get to then spend two years, build another version, and really change the productivity in this very deep way, take some big bets on what you're doing and do some things that are just responsive to what that customer wants.

You won't-- you *cannot*-- become a better programmer through sheer force of programming alone. You can only complement and enhance your existing programming skills by branching out. Learn about your users. Learn about the industry. Learn about your business.

The more things you are interested in, the better your work will be.