The Secret Technique for Learning How to Code

Step 1: Don’t be intimidated.

BY VICTORIA FINE
AUG 31, 20154:59 AM

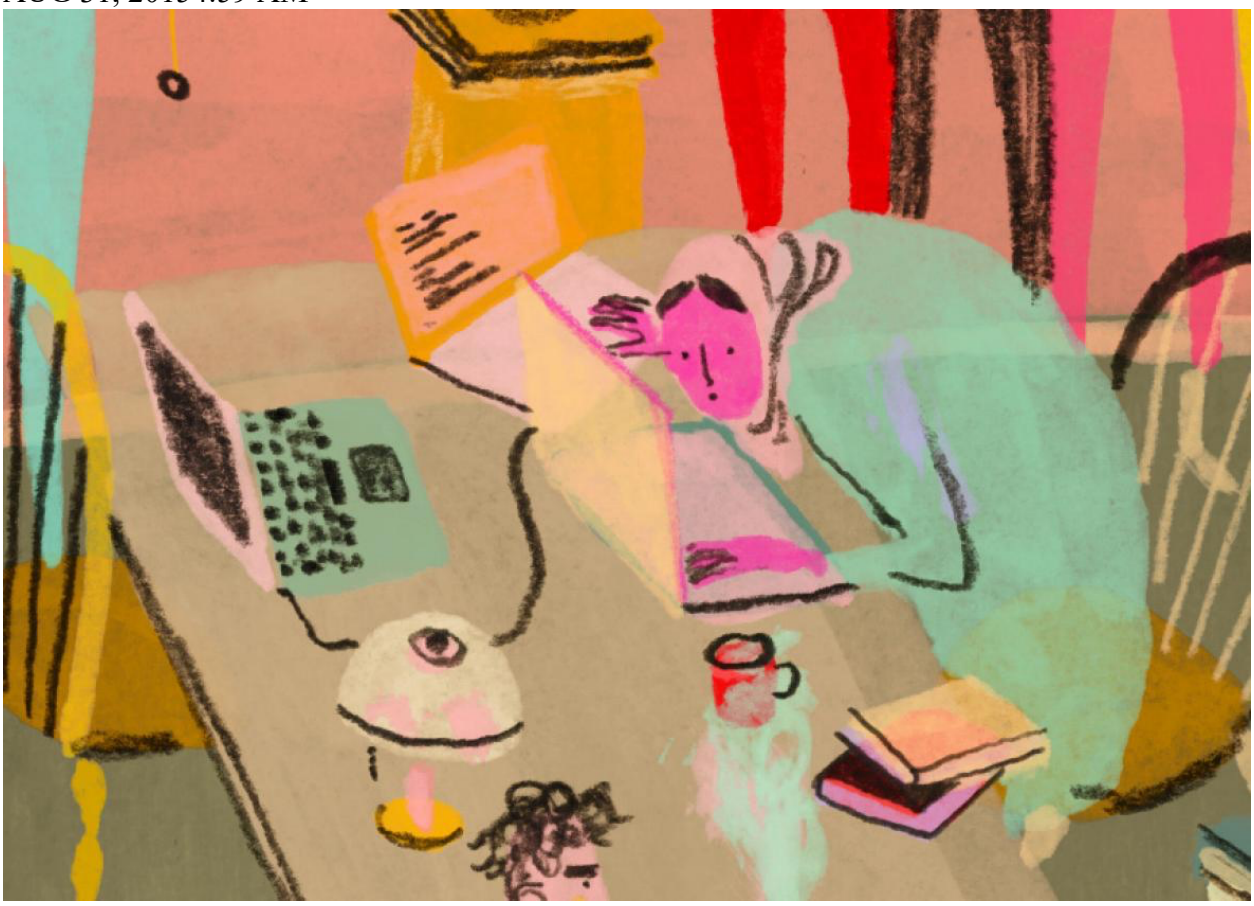

Illustration by Mouni Feddag

TWEETSHARECOMMENT

“You don’t need to go to grad school. Save your money. I’ll teach you how to code.” Seven years ago, in a bar near downtown Los Angeles, I was sharing a drink with a new friend. We had met on a travel website and he had just finished giving me some excellent advice about Southeast Asia, where I was heading for a month. He’d asked me what I was planning on doing when I got back. I told him I was moving to Chicago for my master’s degree. He told me I was crazy.

“That’s nice of you. But I’m going,” I told the almost stranger flatly. That was enough advice for one night.

“Seriously, you can teach yourself anything you want to know. Sign up for Lynda.com. That’s how I taught myself.”

I was annoyed. It wasn’t the first time I’d heard this dismissive advice, and it was not what I wanted to hear. I wanted to learn how to build websites, shoot video, record podcasts, do all those things that made up a perfect “backpack journalist,” which was shorthand back then for “one-woman-band-that-failing-newspapers-could-afford-to-hire.” It was 2008, the recession was looming, and I was afraid my journalism career would be obsolete before it would even qualify financially as a career and not a hobby.

There was another reason I’d signed up to go back to school—I was intimidated by trying to learn about any kind of math-like technology, coding in particular, that didn’t fall into a strict curriculum.

So far, I had been able to qualify as a “digital expert” out of sheer youth. At my first job at a 100,000-person, multinational company, I was tasked with writing the first social media policy ever, because I had a Facebook account. I also used Skype regularly. That was the extent of my technical ability, and I knew it wouldn’t fool people for long.

I had a reputation among my friends of having all the wrong kinds of paranormal abilities. Once, a friend I often called for help troubleshooting said, “You’re like the guy who bends spoons with his mind, but instead of bending spoons you can look at a computer and it gives you a blue screen of death.”

I was also horrifically bad at math and good at writing and reading. I knew what I had been told: that math people were coding people and language people were not coding people. If I were bad at math, this was going to be an excruciating uphill battle and I didn’t want to go it alone. To be self sufficient, I’d need an entire faculty’s worth of structure and encouragement to help me out of this hole of my own intellectual making.

That night at the bar, I politely changed the subject with my new friend and we continued the night pleasantly. Three months later, I moved to Chicago for my higher education (and by then, the new friend had graduated to boyfriend).

The class that changed everything was called “405: Techniques in Interactive Storytelling”—which is what, I believe, every journalism school called any class that had to do with building websites at that time. In terms of my recommendation for you, dear reader, any intro to coding class will do. *It actually doesn’t even matter what you learn in the class.* It’s the process of mental self-discovery that changed me, and hopefully will change you, too.

In this case, the goal of the class was to create a multimedia news story and then build a simple HTML website to host it on. I was so nervous.

I got the same feeling I’d gotten in every math class I’d ever taken, where the symbols written on the board that seemed so simple to everyone else looked like hieroglyphics to me. The instructor also echoed something else I’d heard in a million math classes.

“It’s just simple logic. There’s nothing hard about coding, you just have to understand the sequences and rules.”

The problem was, I had never understood math as logic. Math was inherently illogical—I had learned that in fifth grade when a teacher gave me a word problem about a yard sale. The yard was a certain number of square feet, and there were only so many tables to set up to put objects on, and the objects were all different sizes. So how do you most efficiently set up the tables and objects in the yard for display?

I wrote in the answer box: Lay out three blue tarps and put everything out nicely. It’ll save you a lot of trouble, you won’t have to wake up as early, and you’ll have more time to negotiate!

I failed that assignment, but I literally couldn’t understand why. That was a much easier solution than the one they were proposing. It was the beginning and the end of my understanding of math as logic.

So here I was in class, feeling that same feeling, with the mounting fear that on top of my math inadequacy, my computer could randomly implode at any time because I was touching it. And my instructor, though kind, was as indifferent to my fear as every math teacher had been before her.

At the beginning of class, my now-boyfriend instant messaged me. “If they teach you tables, ask for your money back.” Tables had long since been ditched in favor of building with divs, apparently.

From my desk, 15 minutes into class, I answered him. “We’re learning tables!?” I wrote it with a sinking feeling. I knew nothing about coding but I already knew that the safe structure I’d hoped would help me break past my fear was letting me down. If I wanted to be relevant, I would need to hack what I was learning.

Every class I attended that quarter made almost no sense to me—just like my old math classes. So I stayed after class, or I asked friends or my boyfriend to help and watched as they did something curious—they all Googled. They Googled to look for code, they Googled to double-check their answers for me, I’d ask them something and they’d say, just Google it.

That’s when the first light bulb went off. I started to learn that, unlike my math classes of yore, almost any answer for the quandaries of writing code was online, and instead of being considered an act of cheating or plagiarism, you were encouraged to build off of the work of others. So I began to Google.

Obviously, Google was not unfamiliar to me. I could Google-stalk with the best of them. I had long prided myself in being able to background check anyone, and to be able to find any source information for anything or anyone. (For journalistic purposes only, clearly.)

In the world of Googling to code, the principles were the same. Like any good Google query, a successful answer depended on asking the right question. “How do I make a website red” was

not nearly as successful a question as “CSS color values HEX red” combined with “CSS background color.” I spent a lot of time learning to Google like a pro. I carefully learned the vocabulary of HTML so I knew what I was talking about when I asked the Internet for answers.

Once I got the answers, it was all matter of sequencing the code I found in the right order to make it work. I started spending a lot of time looking at other people’s code to see where mine was right or wrong. There wasn’t a single way to things correctly—every coder was a little bit different, but there were basic principles to follow that made your site function correctly.

And this was my second breakthrough. Coding wasn’t like math at all! It was just like learning a language for the first time. You had to learn a basic vocabulary and then put it in the right grammatical order so the computer could understand it.

Suddenly, I came to class unafraid. Just as being terrible at math made me believe I’d never be good at coding, my talents at reading and writing had come with the social reinforcement that I had a predisposition to learning new languages. I grew up believing that I was allowed to be good at them. I had learned Spanish, French, even a little bit of Swiss German. I was never great at any language apart from English, but learning new languages had never come with the same stomach-dropping fear that math did.

I built the website for our class final and it was not easy. But it also wasn’t scary. It just took a lot of patience, reloading, and Googling. Unlike math, there was no mythic and absolute answer that I was missing; I could do it my own way as long as I followed some basic rules. And unlike learning a new language, I didn’t have to stare into someone’s face and watch her cringe when I mangled a new phrase. My ugly textedit coding document was a judgment-free zone.

I passed the class respectably and went on to take several more. All of which, according to my boyfriend, were a little out of date, and I probably should have just signed up for Lynda. But I welcomed the structured challenges, the deadlines, and the ability to compare my work to others’ and validate that I was doing OK. I needed to take those classes to realize I didn’t need them, and progressive curriculum forced me to advance my own self-learning.

I also began applying my newfound skills to the other courses I was taking—video editing, podcasting, marketing. I found that my heightened Googling ability served me well there too.

This isn’t the part of the story where I tell you I became a professional coder. I never did. I’m still not particularly good at coding anything, and I never learned a coding language besides HTML or Flash (RIP). But my fear had disappeared, and with it, I suddenly became “good at” technology.

After graduating, my boyfriend and I went on to run a multimedia journalism nonprofit, where we taught the same kinds of technology I learned in grad school to people living in conflict and post-conflict zones. I could troubleshoot 90 percent of my and others’ problems, and the more problems we had with virus-ridden student computers, the more I learned.

There's another thing I learned. When I was younger and sure that I was good at reading and language and bad at math and technology, I checked every box in the stereotype of what girls were thought to be good and bad at. I was perennially embarrassed by this, like I was letting down an entire generation of feminists.

When I taught women around the Middle East and Eastern Europe, they were often comforted that they had a female instructor. But they had none of the same qualms about learning that I had had. I didn't know if it was an age thing or a geography thing, but the socially imposed expectations that had made me feel like I would suck at technology didn't exist for them. And, be still my feminist heart, the girls were usually the best ones in our classes.

Five years later, my boyfriend (who has since graduated to husband) and I landed back in L.A. I started teaching some interactive courses at the University of Southern California, and I realized that things hadn't changed as much as I'd hoped.

On the first day of class, when I surprised students with the in-class assignment to build a "Hello World" HTML page along with me, a female student broke down in tears.

"I won't be able to do this! I need more time," she told me.

When I pressed, she said she didn't know what she needed more time for. She just wasn't expecting having to code on her first day of class, or at all. It was a required class and she didn't realize that coding was part of the curriculum. She told me she was bad at math and she was worried now that she'd fail the class.

I tried to reassure her, but knew that in that moment, she'd just have to live through it to get it. I probably came off as insensitively as all of her math teachers had.

By the end of class, every student had a colorful "Hello World!" website on his or her screen. The same girl was elated and high-fived me. "I built a website! I am awesome!" she declared.

Everyone should have that moment: The moment when you realize that an entire lifetime of society's expectations for you was wrong.

Learning to code didn't turn me into a coder, but it made me approach learning new things completely differently. I question my fear of new things instead of listening to it. I ask smarter questions, I embrace new technology more quickly and work harder to find more efficient ways of doing things, because I know they almost always exist. I'm also quite excellent at Googling things.

At ***Slate***, as our Director of Strategy, I use these skills on a daily basis. I can read code well enough to identify problems. I can talk to developers and vendors about the solutions they propose with the right vocabulary and know when they're shining me on. I also stare at a lot of numbers and spreadsheets, to look for patterns and come up with alternatives for doing our work better.

By not letting the things I'm "bad at" determine the things I should try next, I've invited a host of new skills and passions into my life, and for that I'll be forever grateful that I took that first coding class.