It took me 10+ years to realize what I'll tell you in 8 minutes.

Student: Elias D. Franco.
Date: 17/07/2025.
Link Video: https://www.youtube.com/watch?v=RGaW82k4dK4.

So, I've been coding since 2012, and I really wish someone told me these 10 things before I wasted years figuring them out the hard way. If you're stuck, overwhelmed, or doubting yourself, let me save you 10 plus years. My name is Pete, and I've been a professional programmer for more than 13 years, and I've helped hundreds of beginner devs learn how to code properly and land a job in tech.

You don't need to know everything. All right, here's the first thing I wish someone told me: you don't need to know everything. Not even close. But when I started out, I thought real developers had all of JavaScript memorized. Like they were just built different. I used to think real developers could just wake up, grab their keyboard from under their pillow, and instantly write perfect JavaScript, like they had the entire language stored in their brain. Spoiler: they don't. Nobody does.

Imagine you're learning to cook. You don't need to memorize every recipe. You just need to know the basics—how to chop, how to season, how not to burn the house down. Coding is the same. It's more about understanding patterns than remembering every detail. Back in the day, I used to feel bad googling how to get the last item in an array. Like, wasn't I supposed to already know this? But here's the thing—I still Google that sometimes. And so do developers who've been doing this way longer than me.

Being a good developer isn't about having everything memorized. It's about knowing how to find answers, how to think through problems, and how to stay calm when nothing works. So, if you're googling basic stuff, congrats. You're doing it right.

Learn how to learn. All right, here's a big one that would have saved me years of feeling stuck: most beginners try to learn code before they learn how to learn code. See, I thought if I just watched enough tutorials, eventually I'd get it. Spoiler alert: 10 tutorials later, I could follow along, but I couldn't build anything on my own. It was like learning to ride a bike by watching YouTube videos. You feel productive right up until you actually try pedaling and crash into a bush.

Learning to code is like learning a language. You don't become fluent by listening. You become fluent by speaking. Same with coding. If your fingers aren't on the keyboard, your brain isn't really learning. What really changed everything for me was switching from consume mode to create mode. Instead of just watching someone else build an app, I started building my own stuff, even if it was terrible. Here's the rule I wish I had from day one: for every hour you spend watching a tutorial, spend at least 4 hours building without it. Get stuck, Google stuff, break things. That's the real course.

Perfection is a trap. Okay, confession time: early in my dev journey, I once spent 3 hours trying to name a variable. Not even kidding. Just staring at the screen like, should I call it dataInfo or maybe superImportantThingy? Why? Because I thought everything I wrote had to be perfect. But here's the problem—perfection is a lie. You're never going to write flawless code. Nobody does. Even the senior dev you look up to is pushing code that breaks sometimes. They just know how to fix it faster.

It's like learning to paint but never putting a brush on canvas because you're scared that the first stroke won't be a masterpiece. Well, it won't be. It's not supposed to be. Once I stopped obsessing over making everything clean or elegant and just started shipping stuff, things changed. Projects got finished. I started learning faster. And guess what? The code got better as a result. Not because I chased perfection, but because I gave myself room to mess up. Done is better than perfect. Ugly code that works will teach you more than beautiful code that never leaves your laptop. So stop polishing. Start building.

You will never feel ready. Start anyway. You're never going to feel ready. Not ready to build your first project. Not ready to apply for that dev job. Not ready to charge money for your work. I kept waiting for this magical moment where I would feel like a real developer. But it never came. Even after years of experience, I'd still think, "Who let me touch production?" It's kind of like going to the gym. You don't wait until you're in shape to start working out. You just show up. You start small. You get stronger over time.

And same thing with coding. Confidence is built by doing, not waiting. My first freelance gig, I was terrified. Impostor syndrome on full blast. But I said yes anyway. And for sure, I Googled half of it on the fly. But I delivered. And that one yes opened doors I didn't even know existed. You're more ready than you think. You don't need to feel confident. You just need to be willing. Start messy. Start scared. Just start.

The real skill is problem solving. Here's something no tutorial thumbnail ever says: the real skill in coding isn't writing code—it's solving problems. Anyone can memorize a for loop. But can you break down a feature request into tiny, buildable chunks? Can you figure out why something's broken when nothing looks broken? Think of coding like being a detective. The syntax, that's just your notebook. But the real magic is in asking the right questions, tracing clues, and piecing things together.

Early on, I'd freeze every time something didn't work. Like, "It's broken. I must suck." But eventually, I realized debugging *is* the job. It's not a failure. It's the process. The devs you admire aren't just fluent in a language. They are relentless problem solvers. They stay curious. They ask questions. They keep digging. If you want to level up fast, start focusing less on *what* to write and more on *why* it's written that way. Build your thinking muscle, not just your typing speed.

By the way, check the description below for my beginner dev video series because I'm sure you'll find value in it.

Nobody cares about your code. They care about what it does. This one might hurt a little, but you need to hear it. Nobody cares about your code. Not your client, not your boss, not the user. They care about what it does. You could write the cleanest, most elegant code known to humankind. But if the button doesn't work or the site loads like it's on dial-up, it's useless.

I remember building this beautifully abstracted component once. I was so proud of it. It was reusable, efficient, DRY. But the client? They just said, "Cool. Can we make the button blue instead of green?" They didn't care. They just didn't care about my clever hooks. They just wanted results. Think of your code like plumbing. Nobody looks under the sink and claps because your pipes are tidy. They just want the water to run. Write code that works. Write code that solves problems. If it's clean and elegant too, great. But don't lose sleep over the perfect solution no one sees. Value is better than vanity.

Burnout is real. Protect your energy. Let's get real for a second—burnout is real and it can hit hard. You start off excited, motivated, watching tutorials at double speed, drinking way too much coffee. Fast forward a few months later, you're exhausted, confused, and wondering if you're even meant for coding. Been there. I once spent an entire weekend trying to fix one tiny bug. Did I eat properly? Barely. Slept? Just stared at the screen hoping the code would magically heal itself. When I finally solved it, sure, the bug was gone—but so was my energy. And honestly, that wasn't a win. That was a warning.

This idea that real developers grind 24/7? Total nonsense. The best developers I know—the ones who last—they take breaks. They have boundaries. They rest. They go on holidays. They play games. Think of your brain like a battery. You wouldn't run your phone at 1% all day. Why do that to yourself? You don't need to hustle every second to prove something. Productivity isn't about burning out. It's about sustainability. Sleep. Touch grass. Go for a walk. Your code will thank you.

All right, let's recap fast and real:

> You don't need to know everything.
>
> Learn how to learn.
>
> Perfection is a lie. Ship it anyway.
>
> You will nev er feel ready. Start anyway.
>
> Coding is problem solving, not just typing.
>
> Nobody cares how clever your code is—just that it works.
>
> And yeah, burnout is real. Protect your brain battery.