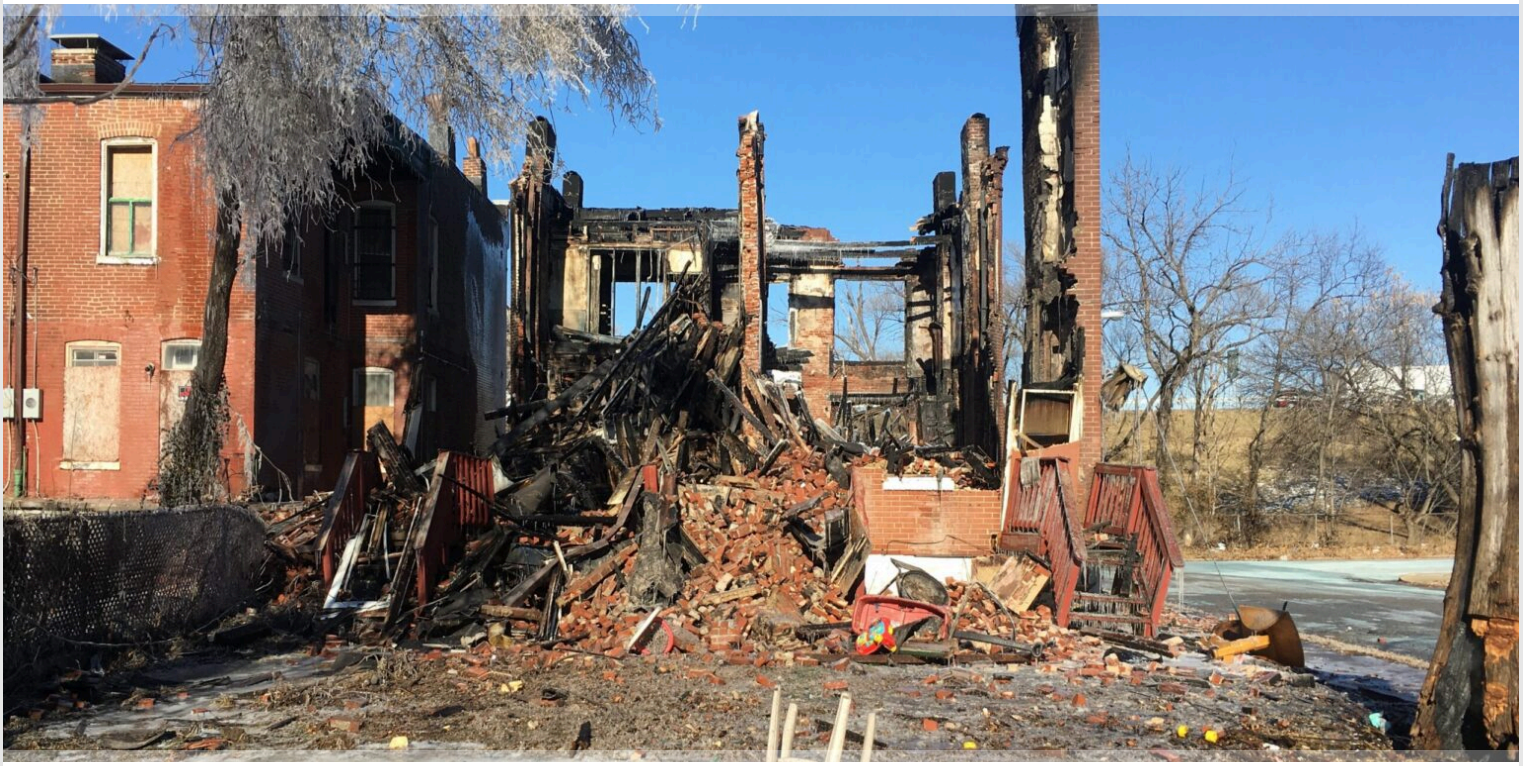
Confessions of a Software Developer: No More Self-Censorship

Kerrick Long November 28, 2025 4 Comments



I haven't published since April because I've been afraid. I also avoided social media, news aggregators, and discussion forums for months. I'm done letting fear stop me. **What was I afraid of?** In this post I detail *every single thing* I've avoided admitting on this blog.

Knowledge Gap Confessions

First, why am I admitting these things now? I realized I am not the only working software developer missing crucial skills. My learning path through my career looked a lot like a slime mold seeking morsels of food: strengthening what has utility, but letting the rest wither. But lately, I've been building a better base of knowledge. Writing or speaking about what I learn–which helps me learn better–requires me to admit I didn't know. Plus, I'd like to show others in my situation that it's never too late to learn what you don't know. I *can* fill in those fundamentals, and so can you.

> It's from that very ignorance that sprouts the drive for knowledge.

> — *Leticia Portella, A Friendly Guide to Software Development*

I Didn't Understand Polymorphism For a Decade

Learning about polymorphism over the past twelve months was the first time I was embarrassed to admit I didn't already know something. I've been writing ostensibly object-oriented software since 2012. And yet, my lack of awareness of polymorphism showed me I've been writing little more than structured programs. That I could replace conditionals and case staments with specialized classes had never crossed my mind.

Why I Was Afraid to Admit It

As a hiring manager I interviewed software engineers and tried to filter for object-oriented knowledge. Retroactively, it's clear I was hypocritical. This gap reveals that I spent the early part of my career learning tools, not principles. Plus, it highlights my lack of formal education. Polymorphism is covered in every college OO course.

I Forgot SQL

I took a college database course as a student. As a working professional, I read and worked through the exercises in *Learning SQL, 3rd Edition*. For a while, I *could* write SQL. But I specialized in front-end web development, and had no professional use for SQL. Like any unused skill, it atrophied. I remember how to write basic queries, but not much more. For example, I cannot tell you the difference between a left inner join and an outer join without looking it up.

Why I Was Afraid to Admit It

I'm not accustomed to forgetting. Growing up I had a remarkable ability to remember almost everything. It didn't matter whether I did it, read it, or heard it. It could be a fact, a skill, or an event. Four years later I could access that knowledge, with the slightest reminder unlocking a flood of memories. Now that I'm in my mid-thirties, that isn't always true. SQL is the first time I've lost *an entire skill* to atrophy. It's tough to come to terms with the start of aging. It's tougher to admit it publicly.

I Don't Write Automated Tests

An estimated 95% of the code I've shipped to production had no automated tests. Early in my career, I had no exposure to the concept. Later, I was writing front-ends in Ember, whose testing story was looked good but felt pretty bad at the time. More recently, I've been working legacy code, and I haven't put in the work to make it testable. The only time I tend to write new tests is when I'm writing a new subsystem, which can be designed testable from the start. I'm convinced that writing automated tests needs to be part of my daily practice, but I haven't gotten there yet.

Why I Was Afraid to Admit It

This may be my most professionally-damaging confession. If you believe Uncle Bob, shipping production code without tests is not more than risky, it's unethical. I stopped myself from posting about my learning journey for fear that a future hiring manager would decide that I was unfit to work with them on this basis.

> How much of the code should be tested with these automated unit tests? Do I really need to answer that question? All of it! All. Of. It.
>
> Am I suggesting 100% test coverage? No, I'm not *suggesting* it. I'm *demanding* it. Every single line of code that you write should be tested. Period.
>
> Isn't that unrealistic? Of course not. You only write code because you expect it to get executed. If you expect it to get executed, you ought to know that it works. The only way to know this is to test it.
>
> — *Robert C. Martin, The Clean Coder, Chapter 1: Professionalism*

Personal Confessions

I Didn't End Up Learning Blazor

People have been waiting for a follow-up from me about my journey learning C#, .NET, and Blazor. This isn't that post. I don't know if that post will ever come.

C# was never the language I wanted to learn for side projects. .NET was never the platform I wanted to work with professionally. I was learning them for one reason: my job. My engineering department decided to switch our tech stack from Angular to Blazor. I was the only person on the team with no C# skills. I started fixing that immediately.

A couple months later, almost as suddenly, the decision was undone. Our tech stack would *not* change after all. With no intrinsic motivation to push me along, I abandoned the C# / .NET book I was reading without finishing. I've got more important things to learn.

Why I Was Afraid to Admit It

No matter what software thoughts crossed my mind, I intended to post about them. Writing helps me solidify ephemeral thoughts. Publishing offers an opportunity for feedback. But I made two errors that locked me into a pattern of fear. First, I promised a follow up article at the end of my last post about the Blazor stack. I then felt worse every time I published an article other than the promised follow-up. Second, I began to see value in the *amount of traffic* a blog post got. The posts about my first steps in that learning journey were the winners. Admitting I changed tack when the company did felt like admitting defeat.

I Want to Write More Ruby

I love Ruby. I use it in my code examples. It's my default language for open source projects. I write Ruby for code katas, etudes, and hackathons. But I haven't been paid to write Ruby since 2013.

The best possible option–and yet the most improbable–would be for my current employer to start a Ruby project. I've worked with a few of my teammates for 12 years across two companies. I've always chosen to keep working with fantastic people at the cost of working with a less-than-fantastic language.

Sadly that means I'm limited to being a Rubyist after work and on weekends. I spend fewer of *those* hours than I'd like writing Ruby, instead favoring other obligations, hobbies, and professional development goals. The only way I foresee getting to work with Ruby as much as I'd like would be to paid for it.

Why I Was Afraid to Admit It

My manager and *his* manager, the CTO, read this blog. I found it difficult to write freely about my distaste for the tools I use every day. I found it ever harder to admit that I *actively want* my daily job duties to be different. I feared they might take it as a hint that I'm quitting (I'm not), or that I'd push to use a tool at work nobody else is familiar with at work (I won't).
A bonus, even more personal confession…

▶ A bonus, even more personal confession…

Workplace Confessions

Your SaaS Team Doesn't Need a Special Process

Hundreds of companies, thousands of researchers, tens of thousands of workers, and millions of dollars have gone into shaping our industry's best practices. The agile manifesto is old enough to drink. Software as a Service has dominated the market for over a decade. Your company has a limited innovation budget. Do you want to spend it on coming up with a custom software development lifecycle, or making a product that wins in the marketplace? Follow Scrum, Lean / Kanban, or eXtreme Programming to the letter, and let your team focus on the product.

Why I Was Afraid to Admit It

Like any author, I write what I know. Here, I was motivated to write because a co-worker pushed to create a custom software development process. I don't know that I have the tact to avoid it seeming to be a takedown of that person or their ideas. I admire the ability of authors like Kent Beck and Martin Fowler to write about how to work better without calling out coworkers who made mistakes.

Remote Work Sucks

Remote work eliminates a lot of problems with office work: commutes, inefficient use of real estate, and land value distortion. But software development is better when you breathe the same air as the folks you work with. Even with a camera-on policy, video calls are a low-bandwidth medium. You lose ambient awareness of coworkers' problems, and asking for help is a bigger burden. Pair programming is less fruitful. Attempts to represent ideas spatially get mutilated by online whiteboard and sticky note software. Even conflict gets worse: it's easy to form an enemy image of somebody at the end of video call, but difficult to keep that image when you share a room with them and sense their pain.

Why I Was Afraid to Admit It

When COVID-19 hit, the company I worked for went remote "for a couple of weeks." After a few months of productive work without an office and with no vaccine in sight, it became permanent. I took the opportunity to move to a rural area. Geographic arbitrage meant I could afford 27 acres, and I even bought a family milking cow. My family has since put down roots: close friendships, community involvement, and a lifestyle built around the lack of a commute.

I feared that writing negatively about remote work might jeopardize my current remote job–and every future remote job I might look for. I thought, "who would hire a remote worker who prefers in-office work?" Because even though I prefer working side-by-side with others, I won't likely move for a job. I have a 30-year mortgage with a low interest rate. My house was purchased before the post-pandemic price spike. I have an acre of lawn & garden, not to mention the farm acreage. I'd need to *double* my current income to maintain my current lifestyle in a city, which is unlikely.

What Now?

Now that the dam has burst, nothing is holding me back from publishing. I'm going to continue to work on skill building, but now I feel free to write about it. If this article resonated with you–whether you also have knowledge gaps you'd like to fill, you'd like to help me fill mine, or you just want to see what happens–please let me know in the comments. Subscribe via Mastodon to see everything I post, use RSS to customize your subscription, or subscribe to my mailing list to get notified when I post a larger article.

Mastodon & Fediverse

To help you follow along, I've enabled ActivityPub on this blog, meaning it's a fully-functioning Mastodon account: `@kerrick@kerrick.blog` .

Kerrick Long (blog)
@kerrick@kerrick.blog

Follow

RSS & Feed Readers

If you prefer to kick it old school, grab an XML feed reader like NetNewsWire for MacOS, Feedmill for Windows, or Newsflash for Linux and choose one or more feeds:

- The "Everything" feed
  - Just the "Articles" feed
    - Specifically the "Advice" feed
    - Specifically the "Blog Posts" feed
    - Specifically the "Book Reviews" feed
    - Specifically the "News" feed
    - Specifically the "Tutorials" feed
  - Just the "Micro-Posts" feed
    - Specifically the "Responses" feed
    - Specifically the "Quick Tips" feed
    - Specifically the "Shared Links" feed
    - Specifically the "Tutorials" feed

Email Newsletter

Of course, the classic (and my favorite) way is to subscribe to email notifications. I'll only send emails when I publish a new article, not for every micro-post.

Get notified when I publish new articles

Get notified when I post new articles. Privacy policy applies.

you@example.com Subscribe

Categories: *Blog Posts*

Automated Tests | Blazor | C# | Career | Learning | Object-Oriented Programming | Polymorphism | Professionalism | Remote Work | Ruby | SQL

Previous Post
Ship Software That Does Nothing

No Newer Posts
Return to Blog

4 Comments

thomas

Bravo! Never fear! And continue writing!

November 28, 2025 | *Reply*

> *Kerrick Long*
>
> Thanks! I plan to keep writing 🙂
>
> November 28, 2025 | *Reply*

G. Davies

I wrote a small system from scratch (20,000 lines of mostly Java). I've been maintaining it for about 8 years now.

It's earned over $2.8 million USD over its lifetime from customers paying directly to use it.

Each month I receive on average of one email from a customer thanking me for this system.

Unit tests: zero.

Test coverage: 0.000%

November 28, 2025 | *Reply*

> *Kerrick Long*
>
> That perspective is helpful, thank you. The production code I've shipped, even without tests, has also been economically successful. A lack of automated tests tends to make change harder, but it doesn't doom the software.
>
> November 28, 2025 | *Reply*

Likes

Reposts

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name*

Jane Doe

Email*

name@email.com

Website

http://google.com

Save my name, email, and website in this browser for the next time I comment.

Post Comment

To respond on your own website, enter the URL of your response which should contain a link to this post's permalink URL. Your response will then appear (possibly after moderation) on this page. Want to update or remove your response? Update or delete your post and re-enter your post's URL again. (Find out more about Webmentions.)

URL/Permalink of your article

Ping me!

Email Newsletter

Get notified when I post new articles. Privacy policy applies.

you@example.com Subscribe

Search

Search

Recent Posts

Confessions of a Software Developer: No More Self-Censorship

Ship Software That Does Nothing

Kerrick's Wager: on the Future of Manual Programming

What Books Should I Read to Learn ASP.NET Core and Blazor?

I Started Reading 26 Books About C# and .NET. Here Are the 2 I'll Actually Finish ASAP.

Recent Comments

Kerrick Long on Confessions of a Software Developer: No More Self-Censorship

Kerrick Long on Confessions of a Software Developer: No More Self-Censorship

G. Davies on Confessions of a Software Developer: No More Self-Censorship

Kerrick Long on I Started Reading 26 Books About C# and .NET. Here Are the 2 I'll Actually Finish ASAP.

💡Smartman Apps📱 on I Started Reading 26 Books About C# and .NET. Here Are the 2 I'll Actually Finish ASAP.

Shift WordPress Theme by Compete Themes.