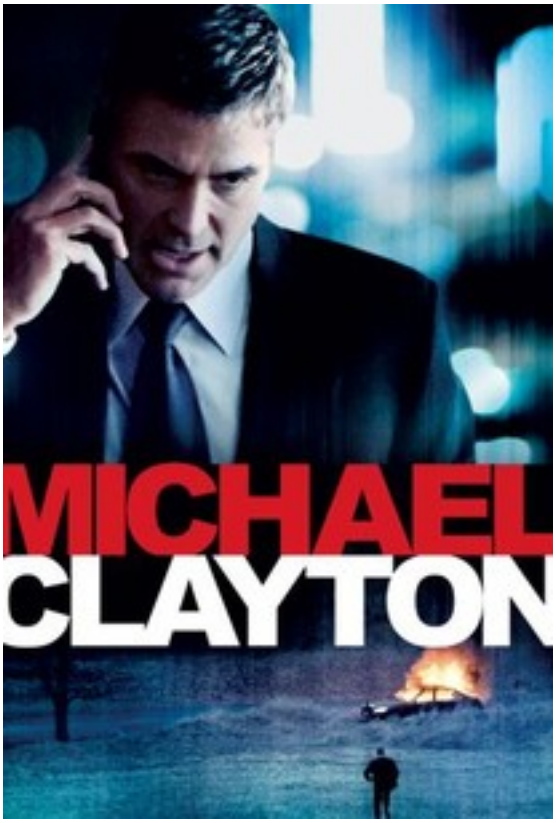

THE MICHAEL CLAYTON OF CODERS

James Gosling invented Java. Ed Catmull founded Pixar. Jim Blinn developed significant computer graphic software. I operate on a smaller scale. I am the Michael Clayton of software engineers.

If you've seen the movie, you know it's about an attorney whose career is in a nose dive. He refers to himself as a fixer; the guy you call to clean up legal messes, a janitor. When it comes to coding, that's me. I'm a code janitor. I'm the guy you call when nobody else wants to work on code that no longer works but has to.

I wasn't always a code janitor. When I graduated with my spanking new BSEE from the University of Wisconsin - Milwaukee, ten years after my BFA, I began a quest in the sprawling tech industry. I was a "Knight of Technology" sworn to conquer electronic and software problems.

Thirty years later, after two dozen jobs in failed startups, halted projects, and large companies that get reorganized every four months, I cannot honestly say it was worth the effort. Yes, I did get paid, but not for overtime. I was out of work for about twenty percent of my life. When I did have a job, the work was often misguided and mismanaged. There were maybe eight projects I worked on that saw the light of day and only three of those made any money. Sure, I learned a lot. I was able to write code to draw a rectangle on a

screen in a dozen computer languages. Woah.

I say "was" because recently I took a battery of those coder assessment tests and scored in the mid 60s for Java and C++. Not surprising since I haven't coded in Java in five years and C++ in twelve years. I also didn't do any studying ahead of time, did not look anything up while taking the tests, and took the tests at my normal working pace. I'm pretty sure for a specific job, I could still be productive and get back up to speed in a month.

But enough with the whining and excuse making, what I really want to talk about is an error I found in one of the C++ questions. It wasn't a coding error. It was a spelling error. In some code that is supposed to scan emails from customers and differentiate between "complaints" and "compliments," the variable name for "compliment" was spelled "complement."

I wouldn't have paid much attention to it except for two reasons. First, I'm a screenwriter and spelling matters to me. Second, this exact kind of error happened on a project I worked on and caused endless trouble.

I forgot what variable name was, but in a team of over a dozen programmers, some of them spelled it correctly and others didn't. Because we were using C++, the variable name could be spelled either way and still compile as long as it was consistent in a class. The problem was, it got very confusing for us humans because nobody was sure if the variable was intended to hold the same kind of values or not. Also, it made it really hard to do file searches and be sure you got every instance of the variable.

So, we had an hour and a half meeting arguing over whether we should bother correcting it and if so, what spelling to use. The team lead, being a pure engineer, decided it was better to let the misspelling stay in there and change the correctly spelled variable names. Thereafter, we continued to run into the same problem over and over because as programmers came into and left the project, there were always people spelling it correctly, which was incorrect for our project.

To me, it's a good example of how using existing conventions, like spelling words correctly, can prevent having to do a whole bunch of technical gyrations to get code working and keep it working.

It's also a good example what I learned in 30 years of designing and coding software.

Luckily, one of the three projects I worked on that made money was a comedy computer game called "Leisure Suit Larry." When I wasn't working overtime coding scenes for it, I dreamed up ideas for other games. After writing the backstory for one of the characters in one of those games, I turned it into a screenplay.

Since then I've been writing and pitching my screenplays and, when I can afford it, getting them made into small movies that few people have heard of, but some pay to watch. For

me, it's way more satisfying than drawing rectangles. As I get better at marketing my movies, they make more money. I'm still a code janitor; but, it gives me enough money to keep making my movies until they make money for me.

I need the money because I pay production teams to make my movies. I do that because during 18 years of pitching feature length screenplays to producers and creative executives I've learned that the best way to pitch a movie is to say you'll pay to produce it and that you have a market for it.

So, for instance, if you were to pay me to fix your legacy code, some of the money goes into financing my movie productions. Thus, you can be a patron of the arts and get your code working, probably. Another thing I learned is that unless the problem is a real life and death situation, failure is always an option because time and money are finite resources.