Extracted from:

The Developer's Code

What Real Programmers Do

The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina

The Pragmatic Programmers

The Developer's Code

What Real Programmers Do

Ka Wai Cheung

Edited by Brian P. Hogan

Our Pragmatic courses, workshops, and other products can help you and your team create better software and have more fun. For more information, as well as the latest Pragmatic titles, please visit us at *http://pragprog.com*.

Cartoons courtesy of Mark Anderson, reproduced with permission of the artist. *http://www.andertoons.com*

The team that produced this book includes:

Brian P. Hogan (editor)
Kim Wimpsett (copyeditor)
David J Kelly (typesetter)
Janet Furlow (producer)
Juliet Benda (rights)
Ellie Callahan (support)

Printed in the United States of America.
ISBN-13: 978-1-934356-79-1
Printed on acid-free paper.
Book version: P1.0—February 2012

CHAPTER 1

Introduction

I've been had by code. Twice.

The very first time was when I took a programming class during my freshman year of college. It was a mandatory course for the curriculum I had decided to enroll in. It wasn't like what I had seen in so many movies during my childhood. I didn't type in a few simple commands, press ENTER, and watch a trash-can robot say "hello."

There wasn't even a trash-can robot in this class. Instead, it was about pointers, memory allocation, and object instantiation. I was too in the weeds to see what all of it meant. However, the evidence was overwhelmingly clear: programming was not for me.

I wanted to be an artist or perhaps a mathematician. I wanted to be both creative and exact—both right- and left-brained, as they say. Programming seemed to lean too far to the left, and no other career options I could think of let me play in both worlds simultaneously. I was lost.

Just a couple years later, the Internet boom changed the landscape of programming. Suddenly, it was real-world, it was approachable, and it had *a lot* to do with design. It valued both artistry and logic almost equally. For the first time, I really could foresee myself enjoying this work. I could now channel my passion for creativity and logic into web applications. So, I returned to programming, albeit with great apprehension.

Truth be told, I also came back to it for an entirely different reason. For that two-year hiatus, I studied many other

subjects that seemed to have too many unanswered questions. Devising the Grand Unified Theory in particle physics or finding the largest prime number? Impossibly ambitious and daunting undertakings, for sure. They just weren't for me. In addition, that course on existentialism didn't clear things up either. As a young adult, I simply wanted *answers*, not more *questions*.

Programming. The very subject I had once eschewed was now my refuge. After all, computer science was man-made. All the answers had to be there. I yearned for a career where answer seekers like me could thrive, where you turned elixirs of code into always-happy customers, comrades, and clients. The rules were already written. We just had to build. All the obstacles existed solely in code, I thought.

In my second return to programming, I was duped again, because this was certainly far from the truth.

Who Is the 21st-Century Programmer?

As I would find out over the next fifteen years, programming isn't a job for the reclusive. It *certainly* is not about the übernerd sitting in a dimly lit basement, sweating away for months on end, and then emerging with the final product in all its glory.

Today's applications are mainstream. We build for every user. Our clients may or may not have any idea about how we work. Our turnaround times are sometimes on the order of whirlwind weeks, not months or years. Burnout can come on suddenly; procrastination can be the path of least resistance. For us, the developers of today, building software involves *obstacles* that go far beyond what we encounter in our development environment.

One of my good friends jokes with me on a regular basis. "What exactly is it that *you do* for work?" She knows I'm a programmer but doesn't really know what that means. She questions me in that same sarcastic, probing way that Bob Slydell does as the office consultant in *Office Space*.

I tell her this: I am a nonaccredited, overly logical psychologist, therapist, mechanic, diplomat, businessman, and

teacher working in an industry that is *still* defining itself each and every day.

That is as concise a definition I can give for the modern-day programmer.

Discovering the Lessons Firsthand

My name is Ka Wai Cheung. I'm a developer, designer, and founding partner at We Are Mammoth in Chicago.[1] We build applications for a variety of clients and create some of our own web-based software. You'll hear a bit more about those later.

This book is a collection of lessons, observations, and missteps I've gathered, firsthand, in our industry. For seasoned programmers, you might find some of my anecdotes similar to your own experiences. We can laugh, cheer, and cry through them together. For newbies beginning the journey, let this be a helping hand through your first few years in the industry.

In the last fifteen years, I've encountered myriad lessons. Here are just some of the topics we'll address in this book:

- Why many traditional development processes and role definitions in this industry are antiquated—and how to sniff them out
- Why saying "no" to the software pet project and open-ended timelines is essential to productivity
- How collaborative work environments can make us so much more productive—and how they can also make us so much less
- How to make code generation a natural part of the development process and how it benefits us in ways besides faster code output
- How to best work with clients who don't see eye-to-eye to us and how to handle angry customers who immediately dismiss new changes to our software

1. Our website is at http://www.wearemammoth.com, and our blog is at http://blog.wearemammoth.com.

- Why big raises and the old mantra "Employees are our greatest asset" don't equate to a better tech job
- How to recognize when software is becoming too complex for its own good
- How to become a better teacher so we can pay our knowledge forward to future generations of developers

This Book Is About Us

This is a book for developers of all kinds. However, it has little to do with code. It doesn't matter if you program in C# or Ruby or Python or PHP or Java or JavaScript or ActionScript. It doesn't matter whether you're working on databases, writing server-side code, or scripting the interface. This book is about everything that surrounds the professional developer beyond the bounds of markup and objects.

That doesn't mean we'll leave programming in the dust, though. There will be *some* talk about code. However, when we talk about code, we'll approach it in a less technical, far more holistic way. You won't see a laundry list of best practices or design patterns. Plenty of books do a great job of that, and we'll mention a few along the way.

This book is about what real, modern-day programmers do to flourish in our industry. Let's begin.