Extracted from:

The Passionate Programmer

Creating a Remarkable Career in Software Development

The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina

The Pragmatic Bookshelf

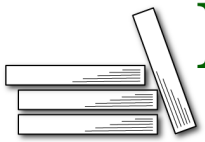
PRAGMATIC LIFE

THE PASSIONATE PROGRAMMER

CREATING A REMARKABLE CAREER IN SOFTWARE DEVELOPMENT

CHAD FOWLER

FOREWORD BY **DAVID HEINEMEIER HANSSON**



Pragmatic Bookshelf

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*.

Printed in the United States of America.
ISBN-13: 978-1-934356-34-0
Printed on acid-free paper.
Book version: P6.0—June 2012

Tip 52

Better Than Yesterday

Fixing a bug is (usually) easy. Something is broken. You know it's broken, because someone reported it. If you can reproduce the bug, then fixing the bug means correcting whatever malfunction caused it and verifying that it is no longer reproducible. If only all problems were this simple!

Not every problem or challenge is quite so discrete, though. Most important challenges in life manifest themselves as large, insurmountable amorphous blobs of potential failure. This is true of software development, career management, and even lifestyle and health.

A complex and bug-riddled system needs to be overhauled. Your career is stagnating by the minute. You are steadily letting your sedentary computer-programming desk-bound lifestyle turn your body into mush. All of these problems are much bigger and harder to *just fix* than a bug. They're all complex, hard to measure, and comprised of many different small solutions—some of which will fail to work!

Because of this complexity, we easily become demotivated by the bigger issues and turn our attention instead to things that are easier to measure and easier to quickly fix. This is why we procrastinate. And the procrastination generates guilt, which makes us feel bad and therefore procrastinate some more.

As I mentioned in Tip 49, *That Fat Man in the Mirror*, on page ?, I've struggled with getting and staying in shape for as long as I can remember. Indeed, when you're miserably out of shape, "Just get in shape" isn't a concept you can even grasp much less do something concrete about. And to make it harder, if you do something toward improving it, you can't tell immediately or even after a week that anything has changed. In fact, you could spend *all day* working on getting in shape, and a week later you might have nothing at all to show for it.

This is the kind of demotivator that can jump right up and beat you into submission before you even get started.

I've recently been working on this very problem in earnest. Going to the gym almost daily, eating better—the works. But even when I'm getting with the program in a serious way, it's hard to see the results. As I was wallowing in my demotivation one recent evening, my friend Erik Kastner posted a message to the social messaging site, Twitter, with the following text:

```
Help me get my $%!^ in shape...ask me once a day: "Was today
better than yesterday?" (nutrition / exercise) - today: YES!
```

When I read this, I realized that it was the ticket to getting in shape. I recognized it from the big problems I have *successfully* solved in my life. The secret is to focus on making whatever it is you're trying to improve *better today than it was yesterday*. That's it. It's easy. And, as Erik was, it's possible to be enthusiastic about taking real, tangible steps toward a distant goal.

I've also recently been working on one of the most complex, ugliest Ruby on Rails applications I've ever seen. My company inherited it from another developer as a consulting project. There were a few key features that needed to be implemented and a slew of bugs and performance issues to correct. When we opened the hood to make these changes, we discovered an enormous mess. The company employing us was time- and cash-constrained, so we didn't have the luxury to start from scratch, even though this is the kind of code you throw away.

So, we trudged along making small fix after small fix, taking much longer to get each one finished than expected. When we started, it seemed like the monstrosity of the code base would never dissipate. Working on the application was tiring and joyless. But over time, the fixes have come faster, and the once-unacceptable performance of the application has improved. This is because we made the decision to make the code base better each day than it was the day before. That sometimes meant refactoring a long method into several smaller, well-named methods. Sometimes it meant removing inheritance hierarchies that never belonged in the object model. Sometimes it just meant fixing a long-broken unit test.

But since we've made these changes incrementally, they've come for "free." Refactoring one method is something you can do in the time you would normally spend getting another cup of coffee or chatting with a co-worker about the latest news. And making one small

improvement is motivating. You can clearly see the difference in that *one thing* you've fixed as soon as the change is made.

You might not be able to see a noticeable difference in the *whole* with each incremental change, though. When you're trying to become more respected in your workplace or be healthier, the individual improvements you make each day often won't lead directly to tangible results. This is, as we saw before, the reason big goals like these become so demotivating. So, for most of the big, difficult goals you're striving for, it's important to think not about getting closer each day to the goal as it is to think about *doing better* in your efforts toward that goal than yesterday. I can't, for example, guarantee that I'll be less fat today than yesterday, but I *can* control whether I do more today to lose weight. And if I do, I have a right to feel good about what I've done. This consistent, measurable improvement in my *actions* frees me from the cycle of guilt and procrastination that most of us are ultimately defeated by when we try to do Big Important Things.

You also need to be happy with *small* amounts of "better." Writing one more test than you did yesterday is enough to get you closer to the goal of "being better about unit testing." If you're starting at zero, one additional test per day is a sustainable rate, and by the time you can no longer do better than yesterday, you'll find that you're now "better about unit testing" and you don't need to keep making the same improvements. If, on the other hand, you decided to go from zero to fifty tests on the first day of your improvement plan, the first day would be hard, and the second day probably wouldn't happen. So, make your improvements small and incremental but *daily*. Small improvements also decrease the cost of failure. If you miss a day, you have a new baseline for tomorrow.

One of the great things about this simple maxim is that it can apply to very tactical goals, such as finishing a project or cleaning up a piece of software, or it can apply to the very highest level goals you might have. How have you taken better action today for improving your career than you did yesterday? Make one more contact, submit a patch to an open source project, write a thoughtful post and publish it on your weblog. Help one more person on a technical forum in your area of expertise than you did yesterday. If every day you do a little better than yesterday toward improving yourself, you'll find that the otherwise ocean-sized proposition of building a remarkable career becomes more tractable.

Act on It!

1. Make a list of the difficult or complex improvements you'd like to make; they can be personal or professional. It's OK if you have a fairly long list. Now, for each item in the list, think about what you could do today to make yourself or that item better than yesterday. Tomorrow, look at the list again. Was yesterday better than the day before? How can you make today better? Do it again the next day. Put it on your calendar. Spend two minutes thinking about this each morning.