DEAL WITH IT

ATTITUDE FOR CODERS

BY GAVIN DAVIES

Deal With It

Attitude for Coders

Gavin Davies

This book is for sale at http://leanpub.com/dealwithit

This version was published on 2013-09-02

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Tweet This Book!

Please help Gavin Davies by spreading the word about this book on Twitter!

The suggested tweet for this book is:

I just bought Deal With It: Attitude for Coders

The suggested hashtag for this book is #attitudeforcoders.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

https://twitter.com/search/#attitudeforcoders

Contents

Dedication

For Mum and Gail,
teachers to the core

Introduction

"Deal With It." That's a strong phrase.

One interpretation is "here's how it is, you have to put up with it". It can be a bratty, unilateral, condescending, dismissive statement.

Another interpretation is "let's cope with things how they are, but work hard to change them for the better". An encompassing, generous statement.

This book is about choosing how you, as a software developer, deal with our industry and your day-to-day work.

"Most of the time, it's your thinking, not your talent, that holds you back." - Rick Warren[1]

[1] https://twitter.com/RickWarren

What's all this about?

I've been working in software for a long time, and I wanted to write the book I wished I'd had when I started out. I wanted to encourage myself, and encourage others, and pass on the development experience of a decade and a half. That might sound like a lot to you, or it might not!

There are a lot of technical programming books, this isn't one of those. I wanted to write a book on the attitude side of things, which is just as important to being successful as a coder - in terms of being good to work with, a proper attitude is absolutely vital!

I had long wanted to do this, but felt a book was a lot to take on. I mentioned this, and one of my colleagues, Rod, responded "nah, you can just sling together 60 pages these days and call it 'the good parts'!"

So here it is - the good parts of what I know! To borrow a phrase from Zane Lowe - this is our book. Mine and yours. It should get you thinking, give you ideas, perhaps help you out of a rut.

You might not agree with everything that I say. I'd be worried if you did!

This book is in short chapters. One thing at a time.

Isn't that the best way to do things?

Who is this guy?

I'm just some guy, you know? If we're going to spend some time together, though, I'll fill you in on what I do.

My job title is "Principal Software Developer". I work at Box UK, a software consultancy I've been at for over 5 years, during which time we've grown from about 22 staff to over 75. We take on all sorts of projects; streaming media, CMS builds, framework development, throwing huge hunks of data around, responsive websites, mobile, bespoke apps… Before Box UK, I worked mostly for small outfits, although I had a brief stint at a large company that felt rather too Dilbert for comfort.

Day-to-day, I operate as developer-in-test and quality evangelist. I work with devs to improve their skills. I perform code reviews. I give training. I will automate anything.

Vitally, I write code.

"Can this guy teach me anything?" you may ask. Perhaps I can't - I doubt I'm any smarter than you - but this is our book, and as you hold it, I hope you take space to think, to let your mind wander, to reach your own ideas. That's what books do for me. That is why I love them so.

Part 1: Overall attitude

Attitude over Aptitude?

"We Are All Fallen Creatures and All Very Hard To Live With" - C.S. Lewis

So I've told you a bit about me, but it's just as important who you are. This may sound strong, but I feel attitude is as important as aptitude. I've had a lot of very clever colleagues - some of whom I just couldn't work well with. I'm hard to work with too, but there are some things that the greatest people I've worked with had in common.

1. Open to unfamiliar approaches
2. Challenges hirself to develop new skills
3. Willing to share knowledge

An archetype that you will likely encounter in your career is the supersmart computer scientist who works in his own isolated silo and spends a lot of time sneering at those who know less. No matter how smart Mr Sneer is, he can be a detriment to morale. Don't let this kind of person put you off in your quest to improve, or drag you down to his level.

This book is about dealing with the challenges of the software industry. My belief is that if you get your attitude right, and be willing to learn, grow and share, then the knowledge will come.

Be courageous in ignorance

"Endeavour to be the dumbest guy in the room" - unknown

You will be ignorant at some points in your career. That's OK. I've stopped trying to hide my ignorance, I will ask questions. You just need a little courage.

At Uni in '98, I had a fellow joint honours student in my yeargroup. Sam was extremely sharp, but being joint honours, didn't always have the cross-knowledge the straight CS guys had. Every lecture, Sam would ask a question. Sometimes, one of the hardcore techie guys would groan, eager to leave, but Sam would persist unabashed. He always got his answer, and you know what? There was invariably someone else who needed that answer and it was often a nervous young Gavin Davies. I had the pleasure of working with him on a group project and his willingness to ask questions taught me a lot about professionalism.

It's not about hiding what you don't know. It's about having the courage and determination to get it right.

You may also be helping others. Choose to be brave.

Be professional

Some people don't understand the difference between being professional and being corporate. Perhaps I don't understand it either, but here is my take.

Being professional is doing a good job to the best of your ability. It's communicating well. It's being honest and friendly and efficient.

Being professional does not imply wearing a suit and displaying no personality or sense of humour. In fact, lack of humour makes communication ineffective, because people will disengage.

Most of the most professional people I know don't often rock up to work in sharp suits (except you, Mr Knight!). Rather, they show up with sharp skills, a great attitude, a willingness to learn and share and an enthusiasm for their work that drives them on.

Putting jokes in your docs or wearing a band t-shirt is not unprofessional. If you are sloppy, slapdash, condescending, know-it-all, non-communicative, or rude (particularly to clients) - THAT is unprofessional.

Being the best version of yourself in your workspace that you can possibly be is professionalism. That's my take.

Be resourceful

Our jobs can be daunting, I know this. I have to do it scared most of the time. One way to truly annoy me, though, is a dev who doesn't try at all.

Imagine a dev, let's call him Montague Marwood. Monty starts work at a new company and is on your team. You show him the ropes and give him training on the application you're working on. You direct him to training materials and documentation and give him a simple starter problem to solve. Two days later, Monty hasn't made a single commit. You ask how he's doing and Monty says "oh, I didn't really understand *feature x* so I haven't really done anything."

You could argue that you could have perhaps pair programmed with Monty, or the problem it should have come up in daily standups. What I'm trying to illustrate, though, is that some people, if they can't solve a problem, will just kind of sit there and not even mention it. Please don't be like this.

Never just sit there with a problem doing nothing. You must be resourceful. Talk to your teammates. Study the docs. Run the unit tests. Ask questions on your IRC channels. Browse the code. Communicate! It's OK to hit roadblocks, but you MUST actively attempt to solve the problem!

Don’t try to prove how smart you are

This chapter appears in the full version of this book! Please visit the book’s LeanPub site[2] to get the whole book!

[2] http://leanpub.com/dealwithit

Don’t worry about being the brightest

This chapter appears in the full version of this book! Please visit the book’s LeanPub site[3] to get the whole book!

[3] http://leanpub.com/dealwithit

Open Source your knowledge

This chapter appears in the full version of this book! Please visit the book's LeanPub site[4] to get the whole book!

[4] http://leanpub.com/dealwithit

Seek to build community

This chapter appears in the full version of this book! Please visit the book's LeanPub site[5] to get the whole book!

[5] http://leanpub.com/dealwithit

Take the initiative

This chapter appears in the full version of this book! Please visit the book's LeanPub site[6] to get the whole book!

[6] http://leanpub.com/dealwithit

Don’t be precious!

This chapter appears in the full version of this book! Please visit the book’s LeanPub site[7] to get the whole book!

[7] http://leanpub.com/dealwithit

Do it scared if you have to

This chapter appears in the full version of this book! Please visit the book's LeanPub site[8] to get the whole book!

[8] http://leanpub.com/dealwithit

Part 2: Tools, learning and technique

Have respect for books

"No matter how busy you may think you are, you must find time for reading, or surrender yourself to self-chosen ignorance." - Confucius

A number of years ago, I wrote a blog post entitled "Coders! Y U no read books?!"[9] and I stand by it today. Books are a wonderful thing, and just because we work online doesn't mean we should limit ourselves to solely using online tutorials, StackOverflow and videos for our development.

I include e-books in this, of course, I'm not that backward!

A book allows you to sit and soak in your own thoughts. The gradual, expansive experience of reading gives me most of my best ideas. I have solved innumerable technical problems whilst lying in a good hot bath reading something by Kent Beck or Martin Fowler.

Don't try to read too fast or too much. The important thing is the space it gives you. If your mind wanders, let it wander, inspired by the text.

I found that the book "Pragmatic Thinking and Learning" by Andy Hunt was absolutely marvellous for giving me ideas. I'd read a paragraph and it would unlock whole trains of thought that I lazily sauntered down. A joyous experience!

[9] http://www.boxuk.com/blog/coders-y-u-no-read-books

Always pressure test

"You must move, have a sense of timing, and progressive resistance that resembles what you would receive on the street.. That's Aliveness" - Matt Thornton, Straight Blast Gym[10]

In November 1993, several people got exposed. At UFC #1, fighters from all around the world were defeated with apparent ease by a Brazilian Jiu-Jitsu (BJJ) guy named Royce Gracie.

Where many martial artists dubbed their styles "too deadly" to spar and pressure test in training, the Gracies would travel around testing their style against whoever would take them on. Unlike many styles, BJJ practitioners trained against resistant, non-compliant partners. Therefore, Gracie's techniques work when put to the test.

The same is true of software development. You can talk all you want, but you must pressure test your software. Unit tests should be part of a project from day one. Measure, don't guess, at load testing. Run automated vulnerability scanners. Use coding standards checkers and mess detectors. YOU want to be the one to know the weaknesses of your "style", not some script kiddy. Make it part of your build loop if you can.

Don't tolerate vague, static, esoteric "grab my wrist" nonsense. Pressure test all that you do.

[10] http://www.straightblastgym.com/interview01.htm

Think testing

Here are 7 reasons to unit test that I gave in a blog post Pragmatic Code Coverage[11]:

1. To answer the question "does the code do what I think it does?"
2. To break a problem up
3. To encourage good design and loose coupling (untestable code is bad code!)
4. To write exploratory code
5. To show your working and provide documentation (e.g. testdox format)
6. To prevent regressions
7. … and because otherwise you're being a cowboy!

People talk about BDD vs TDD like they were in opposition. Here's my take.

TDD says; "this program does what I, the developer, wrote it to do."

BDD says; "this program does what the customer wants it to do."

Both are useful, and you must write tests. Be pragmatic, though. If a region of code really is untestable and isn't just badly factored, then annotate it with @codeCoverageIgnore or whatever your framework supports.

[11] http://www.boxuk.com/blog/pragmatic-code-coverage/

Tell other developers what your code should do. Describe your code by writing descriptive tests.

Automate like your name was Dan

This chapter appears in the full version of this book! Please visit the book's LeanPub site[12] to get the whole book!

[12] http://leanpub.com/dealwithit

Performance matters

This chapter appears in the full version of this book! Please visit the book's LeanPub site[13] to get the whole book!

[13] http://leanpub.com/dealwithit

Measure, don't guess

This chapter appears in the full version of this book! Please visit the book's LeanPub site[14] to get the whole book!

[14] http://leanpub.com/dealwithit

Get your tools right

This chapter appears in the full version of this book! Please visit the book's LeanPub site[15] to get the whole book!

[15] http://leanpub.com/dealwithit

Build a solid technical foundation

This chapter appears in the full version of this book! Please visit the book's LeanPub site[16] to get the whole book!

[16] http://leanpub.com/dealwithit

Be deployment minded

This chapter appears in the full version of this book! Please visit the book's LeanPub site[17] to get the whole book!

[17] http://leanpub.com/dealwithit

Choose your libraries carefully

This chapter appears in the full version of this book! Please visit the book's LeanPub site[18] to get the whole book!

[18] http://leanpub.com/dealwithit

No source is set in stone

This chapter appears in the full version of this book! Please visit the book's LeanPub site[19] to get the whole book!

[19] http://leanpub.com/dealwithit

Learn to spot antipatterns

This chapter appears in the full version of this book! Please visit the book's LeanPub site[20] to get the whole book!

[20] http://leanpub.com/dealwithit

Don’t be tied to a single technology

This chapter appears in the full version of this book! Please visit the book’s LeanPub site[21] to get the whole book!

[21] http://leanpub.com/dealwithit

Part 3: Wisdom for the long haul

Have achievable goals

This is adapted from a longer entry on my personal blog[22]

A long-term project can be hard work. It's easy to lose sight of goals, and drift when it feels like actually DELIVERING something is simply out of sight, over the horizon. Then, suddenly, that intangible deadline begins to rocket towards you and a frenzied "crunch time" begins.

This is one of the reasons devs people tend to work on their own projects outside of their day jobs - because these projects tend to be feel achievable, and the sense of progress is tangible.

It's greatly underestimated how much developer morale affects productivity. Unhappy devs will twiddle their thumbs and fiddle around with toy projects, sighing at the thought of another arduous day working on the "goal over the horizon".

That's why agile - or any kind of iterative development - is helpful from a psychological standpoint. Short term goals are incredibly motivating because you have an achievable target. You aren't trying to fit the universe into your headspace at once - you are instead working over a short period to reach a goal that remains in sight.

[22] http://gavd.co.uk/2012/09/how-agile-can-keep-up-morale-on-long-term-projects/

Accept that failure happens

"For though a righteous man falls seven times, he rises again" - Solomon

"Experience: that most brutal of teachers. But you learn." - C.S. Lewis

You're going to get things wrong. You're going to make mistakes. No matter how hard you try, no matter what best practises you follow, some day you'll DROP DATABASE on a live server, or do what I did and accidentally email the text of Edgar Allen Poe's *"The Pit And The Pendulum"* to several thousand users (oops!).

Give yourself a break. For me, this is hard; I'm instinctively tough on myself, so if you're wired like that then I know how it is. Think, though; "would I be so annoyed if one of my co-workers had made this mistake instead of me"?

Try to put your mistakes into perspective. Definitely learn from them. Do all you can to put things in place to prevent mistakes happening again.

It can be hard to come into the office the morning after a catastrophic screw-up. That's a mark of a true professional, though.

Get up, don't give up! Know that we've all made mistakes. Forgive yourself, learn, grow and teach.

Treat people as people

This chapter appears in the full version of this book! Please visit the book's LeanPub site[23] to get the whole book!

[23] http://leanpub.com/dealwithit

Take care of yourself

This chapter appears in the full version of this book! Please visit the book's LeanPub site[24] to get the whole book!

[24] http://leanpub.com/dealwithit

Beware burnout

This chapter appears in the full version of this book! Please visit the book's LeanPub site[25] to get the whole book!

[25] http://leanpub.com/dealwithit

Dealing with burnout

This chapter appears in the full version of this book! Please visit the book's LeanPub site[26] to get the whole book!

[26] http://leanpub.com/dealwithit

One thing at a time

This chapter appears in the full version of this book! Please visit the book's LeanPub site[27] to get the whole book!

[27] http://leanpub.com/dealwithit

Part 4:
Communication is key

Documentation is communication!

When you pick up a project, it can be hard to even get it running. I've often been in the situation where I've checked out or cloned a repository and I have no idea how to use the application!

There's an old computer science term: the runbook. It's kind of a retro term, but it works for me. I use "runbook" to mean a file that tells you how to use an application.

Usage examples should always be in your documentation. Never be vague, be specific.

Above all, in this age of ludicrously named libraries, your runbook should tell you precisely what the application actually does!

Many Github projects are a good example of this; they should always have a file in the root called README.md, telling you how to test, use and extend the project.

Try to bear in mind that others don't have your knowledge. Get your colleagues to check your runbooks in the same way as they would code.

It doesn't have to be highly detailed, but it does have to get somebody going quickly. It's the information you'd want if it was you.

No unnecessary docs

We've looked at runbooks, so am I contradicting myself here? I don't think so.

If you'll forgive me something of a "businessy" term, it's all about ROI (Return On Investment) here. As are most things. Documentation reaches a point of diminishing return. Don't try to specify things precisely that will change a lot - instead, focus on communicating the things that will remain fairly steady - usually, the public interface. People can read your code if they want the details; written documentation should take the runbook approach from the previous chapter.

Always ask yourself questions like: "is this going to change? Will anybody need to know this to use the system? Am I adding complexity or clarification?"

If the documentation you were going to add can be served by simplifying the system, you may be better off spending the time in that way.

Write documentation. Your project is NOT complete without it. Just be sensible about how far you go.

Keep emails short

This chapter appears in the full version of this book! Please visit the book's LeanPub site[28] to get the whole book!

[28] http://leanpub.com/dealwithit

“Just” is the worst word

This chapter appears in the full version of this book! Please visit the book’s LeanPub site[29] to get the whole book!

[29] http://leanpub.com/dealwithit

Social problems can't be solved with tech

This chapter appears in the full version of this book! Please visit the book's LeanPub site[30] to get the whole book!

[30] http://leanpub.com/dealwithit

Dealing with suits

This chapter appears in the full version of this book! Please visit the book's LeanPub site[31] to get the whole book!

[31] http://leanpub.com/dealwithit

Beware meetingitis

This chapter appears in the full version of this book! Please visit the book's LeanPub site[32] to get the whole book!

[32] http://leanpub.com/dealwithit

Part 5: Some closing advice

Never stop learning

"Unless you are continually improving your skills, you're quickly becoming irrelevant. And when you're irrelevant, you're no longer credible." - Stephen Covey

In our industry, perhaps more than any other, we have to keep improving. As coders, we have never "arrived". Our skills are never good enough.

That's pretty difficult to accept sometimes. It's tempting to believe that some day, I will be a "complete" coder. Not so! It's not achievable. What IS achievable, however, is to be better than yesterday. Therefore, an attitude of continual, humble improvement will stand us in good stead.

This is where community is particularly important. Being around people with different skills to us is really helpful. Other languages, paradigms, ways of doing testing, patterns, deployment methods, automation and virtualisation techniques…

Don't sweat becoming "perfect". Simply keep learning, and have the humility to never make out that you've achieved programming greatness!

Keep it in perspective

This chapter appears in the full version of this book! Please visit the book's LeanPub site[33] to get the whole book!

[33] http://leanpub.com/dealwithit

Who do you think you are telling me all this?!

This chapter appears in the full version of this book! Please visit the book's LeanPub site[34] to get the whole book!

[34] http://leanpub.com/dealwithit

Signing off

I decided I would write this book to be small, concise, and direct - I wanted this book to be as readable as possible. I hope that you enjoyed it!

I wrote on 8.5" pages in Google docs with 2" margins using 14pt Arial to restrict me to columns of about 45 characters to encourage me to be concise. I then exported to HTML and imported into Leanpub and continued to work on it in the Markdown format.

I wanted each page to stand alone, to get a single point across, and to have a quote or a story. I wanted each page to be something you could hand to a co-worker that they could read in a minute or so and hand back to you.

Hopefully you've found something encouraging in here. Get out there and do it! If you have at least some aptitude and your attitude is right, then you can improve, and find your work more satisfying. You can impact your culture positively. You can be a good example to yourself and others.

Be honest.

Be brave.

Be yourself!

"Courage is not simply one of the virtues but the form of every virtue at the testing point " - C.S. Lewis

Acknowledgements

A huge thanks to Dayle Rees[35] for the encouragement in writing this book, and not least of all, for designing the front cover! Thanks also to his lady Emma for allowing me to use her awesome cover photo - that little capuchin's face is priceless!

Thanks to Box UK[36] for over 5 years of employing me! I've never worked anywhere where there's so much opportunity for innovation and development. Thanks to Stu, Carey and Benno for having a read through and sanity checking the book!

Big up Warren, Carey, Craig and Rod - team Unified Diff[37]!

Hold tight Woodville Baptist Church[38]!

Easy the City Arms, Cardiff's best pub!

[35] http://daylerees.com/
[36] http://www.boxuk.com/
[37] http://unifieddiff.co.uk/
[38] http://www.woodybap.org.uk/