Essential Career Advice for Developers

10 Keys to Happy, Prosperous Work

Cory J. Miller

PUBLISHED BY

iThemes Media
1720 South Kelly Avenue
Edmond, OK 73013

CoryMiller.com
iThemes.com

Like software, books get updated.
Or maybe my thoughts and opinions do.

Download the latest version of this ebook here:
http://corymiller.com/careeradvicefordevsbook

Introduction

Over the past 5 years I've had the privilege of working directly with web developers and software programmers. In that time, I've also interviewed and worked with a number of developers, along with managed and lead the development team at iThemes.

I can safely say, it's been both one of the delights of my work life as well as an ongoing challenge as I'm not a developer by any stretch of the imagination.

So when I think about all of my experiences working with and getting to know developers, two big things stand out to me about you:

No two developers are exactly the same. You're all unique in your own quirky way. And I love and cherish that about you.

I thoroughly love being a part of building and launching things with you. You do things I can't. You can actually build all the things we dream about together.

And if software is eating the world, and I believe it is, we — the businesses and startups focused on creating that software — need you more than ever.

We need good, solid, dependable, knowledge, collaborative developers to build the software that will eat the world … or simply make it a better place.

We need to know you can fit our unique personality and cultures. We need you to connect your own unique skills, passions and personality into our collaborative mission and purpose.

We need to know who you are. We need to know what you can do.

We need you to be prepared for the work. We need you to keep learning and expanding your skills.

We need you to help us find roles that fit you, your skills and passions.

We need you to be able to work with other human beings and communicate with the aliens, those who don't speak code.

With this book, I want to help you connect and adapt so that you can thrive in our startups and organizations so that together, we can build software that changes the world.

So … this is for you. For us. But really, for the world.

1. Build Your Portfolio of Cool Projects That Work

Employers want to see stuff that shows you can actually do things with the skills you list on your resume. Projects, like nothing else, are amazing for this.

See …. I'm not a developer. And I don't pretend to be one. So you can bullshit me. Most of the people interviewing you will probably not be programmers. (Which is why we bring in developers to talk geek with you too, assign sample projects and review your code.)

Good working projects that solve a problem for users, though, is one of the BEST ways to get a job. It shows demonstrated use of the skills you list.

It demonstrates completed work and initiative. It also shows you shipped something in the past. (But we'll talk more about shipping in a bit.)

Here are the two keys for good projects to put in your portfolio:

1. It uses skills you're trying to get a job doing. (If you're a PHP programmer, it should naturally be in PHP, etc., etc.)

2. It actually works so we can play around with it or at a minimum, see a demo.

“How big should my portfolio be?”

Enough to showcase your skills and your work thoroughly enough to convince someone to reserve a spot on their team, go through all the crappy paperwork and give you a paycheck.

“What projects should I include?”

Live, working projects, not worthless academic projects for school. Keyword: Working. Second keyword: Usable.

“What if I’m not super proud of them?”

Put it out there anyway. Having nothing to show is worse. Plus, everybody starts somewhere. Then as you get better projects to include, replace the crappy ones with the better ones. Keep doing this as you, your skill and your experience grows.

“What if I don’t have any projects right now?”

Get some. Start with small, laser-focused projects. Do pro bono work if needed. Build onto them, or take on a bigger project as you go.

connect with your customers could create new ideas, new products, new industries.

2. Get involved and contribute to an open source project.

Contributing to open source projects blends learning and networking, while having a project to show off. So it's a triple win.

I've personally lived this advice through WordPress. The community is huge and my contributions to it in some of the early days propelled my career (and helped me launch iThemes).

Find an open source project you're interested in and get involved in the community around it. If you can't find one to do so, simply release something cool you've built and build interest around your own project.

Your work on open source projects can become part of your portfolio plus it shows that you can work with others in a collaborative environment.

Here are some more quetsions for evaluating an open source project to contribute to:

- Is it something you're passionate about?
- How big and active is the community around it?
- Do other people care about it?
- Does it leverage your skills?
- Does it stretch you and your skills?
- Are you able to expand your contacts and relationships?
- Will your work be visible to show others?

3. Remember, always & forever: It's all about people.

I love geeks because you're brilliant … but you're not always the most social creatures. Without people, your code is just an academic exercise. It's about your customers-users and your team. Ignore this at your own peril and lasting joy and happiness.

Your code may survive the apocalypse, but if you bulldoze people doing it … no one will give a crap. You'll be lonely. And I don't want that for you. Neither do your parents.

Build cool stuff that makes people's lives awesome. But do it WITH people, and FOR people.

Your code won't show up when you are sick, or hurting, or at your funeral … people will.

So don't become the ogre no one wants to work with. Don't get crusty, short tempered & the angry beast nobody wants to work with. It will happen to you if you don't guard your attitude and take time off to rest and recharge (more on that soon).

But beyond that … learn how to relate to other human beings. Be able to work within a team. If you don't know how to do that, get involved in user groups and teach and mentor.

Empathy is an invaluable asset for your career for developers but sadly one that is often neglected or overlooked. Empathy is being able to walk a mile in another's moccasins. To think like them and help them. To understand and identify with their another's unique needs and wants.

I promise that if you pair empathy with your awesome technical chops, you're going to be a bright shining star for all of us. We will celebrate you because you understand us and can help us achieve what we so desperately need from you.

Empathy is simply about caring. Caring for your team and the users of your software, first and foremost.

To learn empathy is rather easy … it is simply consistently being other-centered. Letting others talk first. Listening before talking. Asking before telling.

Make it a point to empathize with others … figure out what makes them tick, what motivates them, what they care about and ask them to tell the stories of their lives.

You'll be a better team member because you care about the people you work with, even if they don't "rank" as high as you do. And I think you'll find over and over that to do big things, you need people. You need people to help with support so you can keep coding. You'll need people to help with documentation. You'll need people to help showcase the benefits (ehem, "features") of your software so people will actually buy them and use them.

And you'll be a better programmer because you care about users, and how they use your software and understanding them helps you make that software better and easier to use and to do what they really need it to do. That's powerful, world changing stuff … all because of empathy.

4. Good communication skills are almost as valuable and important as programming.

If it's always and forever about people … then good communication is IMPERATIVE … learn how to get better at it.

You can build virtually anything, but if you can't communicate with others — in their terms and language you're going to struggle mightily.

One of our business mentors is fond of saying: "Communication is not the message sent, but the message received."

So you must work continuously, tirelessly on learning how to communicate with the non-geek, dummies of the world like me. (And by the way, we both have to work on it! So it's not a one-way street.)

Communication with non-geeks is truly hard for both of us. It's not that we're stupid, we just don't speak the same language. And it takes work.

So here are some tips on communicating with the non-geeks of the work world that will benefit your career greatly:

- Be forever patient. Like the longsuffering kind of patient. Stay calm and keep calm, even when you've hit a deadend with us. Always keep your composure. Never talk down to us. And remind yourself that we don't talk the same language and this is a communication barrier. I understand how enraging it could get when we're not making any progress. (For managers of programmers reading this, you've also got to do this unceasingly. It's another two-way street.)

- Translate for us. After you mention your technical jargon, then say, "This means ..." But the best translations are typically metaphors or analogies. It offers a comparison in our terms that we can understand and more so, relate to.

- See the issue from the other person's point of view, restate what you've heard and ask if your understanding is clear.

- Tell us the WHY until our eyes glaze over

- Then tell us HOW what you're doing or what you ran into affects the project in terms of time, complexity

- If development is an expedition, share the hurdles, challenges, dangers of the journey you may run into. Then give us options. Don't just give one path of the journey, present us with different options even if they are all marginal options. And once you've presented the different options, give some input to what you'd suggest or prefer and most importantly, WHY.

- Seek help and assistance from a translator. Find one of those weird blends of people who can walk and talk in both worlds. They are your ally (and most likely your manager's trust ally too because they can translate). Stick close to them. Talk to them often. See how they might translate what you're saying to others.

- Know yourself, know your strengths, know your personality. We ask our team to take StrengthsFinder and talk a lot in these terms within our teams. We also have encouraged our team to take the Kiersey Personality Test. Explore what these discovery tools say about yourself, confirm it with others, and then start asking others about their own unique strengths and personalities. (By the way, this shows empathy.)

5. Ship

All good programmers ship. So always, ship your code.

Be someone of action and execution, not just a daydreamer or academic who just talks about theory and what could be but never produces. Finish the projects you start and ship them.

Ship because you want to change the world with your code and make people's lives awesome. Code that never ships never has the chance to change the world. And if you never ship, your code is simply a narcissistic art project, nothing something intended to change the world. In this sense, code that hasn't ship doesn't exist because it's not available for the world, just you.

Shipping is bold, brave and for the select few who want to have impact in the real world.

Getting your code into the wild is extremely satisfying. You get to see if people care or not … if you were right or wrong about the decisions and assumptions you've made. To me, people using our software is the most fulfilling thing we can do because it is our work being used by real people.

Be known as someone who ships good code and then iterates fast to make it even better.

Understand what a version 1 truly is. We know you can build v. 500. But version 1's actually ship and have the chance to be used in the wild. Learn about an MVP (minimum viable product) is via The Lean Startup. Also the concepts rewritten about by Jason Fried in REWORK are invaluable in this context, particularly under the "Progress" heading.

Beware of perfectionism, it's evil and is a farce that can kill your career. Yes, other coders will judge you, but while they are doing that, they aren't shipping. And you are. You're also making your code better while they continue to puff hot air out.

And by the way, they also don't ultimately matter anyway. They aren't your users and customers and your team. (They are just noise actually. And those who have enough time to critique are typically those who never do anything of value in the world but, yeah, critique.)

Do good solid work, test the hell out of it … then ship the damn thing.

A Quick Word on Deadlines

Deadlines are good and valuable. Don't fear or loathe them. Seek them and the accountability they afford and use them for the success and growth of your career and the projects you're working on.

They give you a goal and mark to hit. Light at the end of the tunnel. And we all need something to work toward.

They force you make important decisions that ultimately should help you ship faster and better. They force you to spend time, energy & focus on what actually matters.

They give you a sense of accomplishment. When you and your team members hit them, celebrate. Ring a bell, or a gong … throw a party for MAJOR ones but always recognize them (even to yourself).

Use them! Use them to ship your work. And if you aren't given timelines or milestones, ASK for them. Or set them for yourself and do the insane thing of sharing them with others. The accountability is good for you and the project.

6. Park your arrogance at the door.

Arrogance is usually a hindrance to shipping, by the way.

The best coders don't have to tell me they were great. They let their code speak for itself.

And let me be even more specific: Their SHIPPED code.

The best coders I've known have quiet confidence and check their ego at the door for the greater good. These are the developers people yearn to work with. Show up. Do the work. Ship. And the glory will come.

So when I mean park your arrogance at the door … I really mean, deal with it before you ever get to my door.

Don't get me wrong, we want you to know your stuff and I love and value brilliant programmers … inside and out, backward and forward … we, and I'll speak for everyone you'll ever work with, just don't want it shoved down our throats every day.

If you have an insatiable desire to tell the world all the awesomeness about yourself, all the time, then something's a little off, honestly.

In fact, I'd go so far as to say you probably have a confidence problem.

Most people in life don't want to work with arrogant people. Heck, I avoid arrogant people in general. Don't you?

Words are vapor. In fact, when you add a thick cologne of arrogance to it, it becomes hot air.

And you can achieve so much more in your work and your career with action than mere words. Action breeds a quiet confidence because we all know you show up, do the work and deliver. So the less words required the better.

So learn quiet confidence and humility through action.

You'll be more popular and people will love you and your work even more.

And then when you hit that milestone and deliver on your project, we're going to make a really big deal out of you and your work … because you didn't do it already.

7. Get out from behind a computer screen and meet people.

Learn how to network with people, not just equipment.

Go to conferences, workshops, Startup Weekends ... or anywhere the people and places you want to work for and with are also hanging out.

Introduce yourself, talk about them and their company, then the door will open to talk about your work. Ask their advice for what it means to be successful in their company and industry. You'll often get good advice plus it'll flatter them to be asking their opinion.

But be sure to exchange business cards, yes, which means you might need them if only for this. Because when you get home, you want to connect with them on LinkedIn (more in a minute on why LinkedIn), and you'll need their email address to do so.

(By the way, you can connect with me on LinkedIn here.)

All of the above -- in-person networking -- allows you to also practice your social and communication skills by the way.

At some point, try to do a talk or presentation to a user group or at a workshop or conference. (And as my friend and web design teacher Christy Sooter says schools also need guest speakers too!)

But start small and build up to bigger audiences as you can.

If you're doing cool projects, like something in an open source community, talk about it and how it's helping people and why you built it. Be practical and passionate. Show examples of what you've done.

When people like me connect you with the projects in Keys 1 (having a cool portfolio of work) and 2 (contributing to open source projects) while realizing you can do No. 5 (ship), we're going to BEG you to come work with us.

Meeting us and just getting in front of us helps connect all those dots.

Why use LinkedIn for Networking:

I highly suggest getting a LinkedIn profile (the new resume) and using it to connect with all of your professional contacts. With over 225 million members worldwide (74 million in the U.S. alone) and growing, it is THE dominant social network for professionals. It's being used more and more for recruiting and hiring. One of the biggest reasons is being able to see your "six degrees of Kevin Bacon" in the world of work. It shows you how small and connected the world is. And it'll come in handy if (and when) you ever need to go warm up your contacts and start looking for a new job. By the way, here is 10 ways to use LinkedIn's Company Profiles to find a cool company to work for.

8. Make sure you power down regularly.

This might not be a “how to get hired” tip but it’s more a “how to stay hired and also thrive” tip.

We all need to rest and recharge.

There will always be deadlines, milestones, and a feature list to chase. I promise.

Figure out what is the key to how you best rest and recharge. Then schedule it out. Give people plenty of notice and do something that almost forces you to actually take the time off (buying plane tickets, etc). If you don’t truly power down you’re a great candidate for burnout. And that’s no good for anyone.

And it’s ultimately your responsibility to watch the gauges of your energy, focus and patience. When you start to go low on all of those, you’re going to start to become hard to work with and short-tempered.

It’s so important to know yourself and then work with your team leads to ensure you’re taking an appropriate amount of downtime.

It’s amazing how much better you’ll be after some time off. And when I say downtime, I mean powering off as much as possible. Maybe you leave your phone on but you’re not checking email or on chat.

When you take time to rest and recharge, when you come back to us … you'll be more creative. You'll be more passionate. You'll be more focused. You'll be more open and eager to explore options. And it's likely during your downtime, the solution to a nagging problem has bubbled up. Those are all the great byproducts of rest.

- Read a book. About something completely unrelated to programming.

- Take a cooking class. It's working with your hands and a lot of fun when you can eat your end result.

- Start learning a new language. I don't mean PHP or Ruby, but rather Spanish.

- Experience nature. Go on a hike, or a rafting trip, or go snow skiing. Something in the outdoors.

9. Always, always, always be learning and exploring.

This is also great life advice, by the way. If you're not learning, you're dying.

So always find something new to learn — even if that's another programming language, new framework, whatever. But never be satisfied with the knowledge you have today.

The best companies hire learners. The best managers and team leads want learners on their teams.

Being a learner means you're not content to stay where you are for the rest of your life. You know you're an unfinished product and you have much more to learn in life. (If you think you've arrived and have nothing more to learn, reread Key 6: Park your arrogance at the door.)

Never forget that there is a 16-year-old genius plugging away in his or her parent's basement in rural Kansas who is going to rock your world.

The best learners read. And they read voraciously.

(And I highly suggest building your reading library with an Amazon Kindle account. The free reading apps can be install on PC, Mac, iPhone, Android, etc. Here are some ways to maximize the Kindle platform for learning.)

You should always have a new book on your desk or in your library. You should be reading things that:

- Stretch you
- Expand your knowledge
- Give you a broader perspective on the world
- Help you improve as a person

Also, learn and grow on your own time and dime.

You should NEVER solely rely on your boss or organization to invest in your skills. Invest first in yourself. Make the time. Write the check. Because ultimately at the end of the day, whether you stay at that organization or not … you take those skills with you.

So invest in books (like I mentioned above), register for conferences, training, or classes to get more skills and take personal time off to do so, if needed, get into online training membership and attend webinars and watch videos in your niche …. whatever helps you learn and grow.

And as a boss, this type of team member rises to the top too for me because they are willing to spend their own precious time and hard-earned money on their own learning.

10. Pass it on.

As a learner, as a superstar in the world of world, I believe your responsibility is to pass it on to someone else a couple of steps behind you. To the you a couple years ago.

Seek to be a mentor, teacher, trainer, helper to others on your team and in your sphere of influence.

Great companies and great leaders also love those who not only learn … but seek to build up others.

Some of the richest rewards of my career have been doing this for others. When you do this for others, people take notice. It's about multiplying yourself and your impact and that's a hot commodity in the world of software.

I talk at great length about how to "Pass It On By Multiplying Ourselves" in my book Purposeful Paychecks: How to Find Lasting Career Happiness.

I won't rewrite the entire section, but here's how to be a magnet that others are drawn to seek you out in their own journey:

- Be an unselfish giver.
- Be friendly.
- Be open.
- Be a good listener.
- Be dang good at what you do.
- Be teachable.
- Be passionate.
- Be invested.
- Be helpful.
- Be available.

Do these things, or rather BE these things … and people will seek you out as a mentor.

Parting Words and Connecting with Me

So those are the big keys I'd give developers seeking to have a great career, but there are plenty more topics and tips to share and expand on like how to get a raise, how to ship a version 1, why you should look for a startup to join, and more.

If you'd like to get updates when I post more career advice for developers, sign up for this special email list here.

And as I said, I cherish the relationships I've had with developers like yod. I'd love to connect with you. So here are some ways we can connect further:

Write to me: cory@corymiller.com

Follow me on Twitter: @corymiller303

Oh, and of course be sure to connect with me on LinkedIn**.**

I hope this book has been helpful to you and you can put the concepts and values and steps into good use to make a great career building the software the world needs!

—Cory Miller