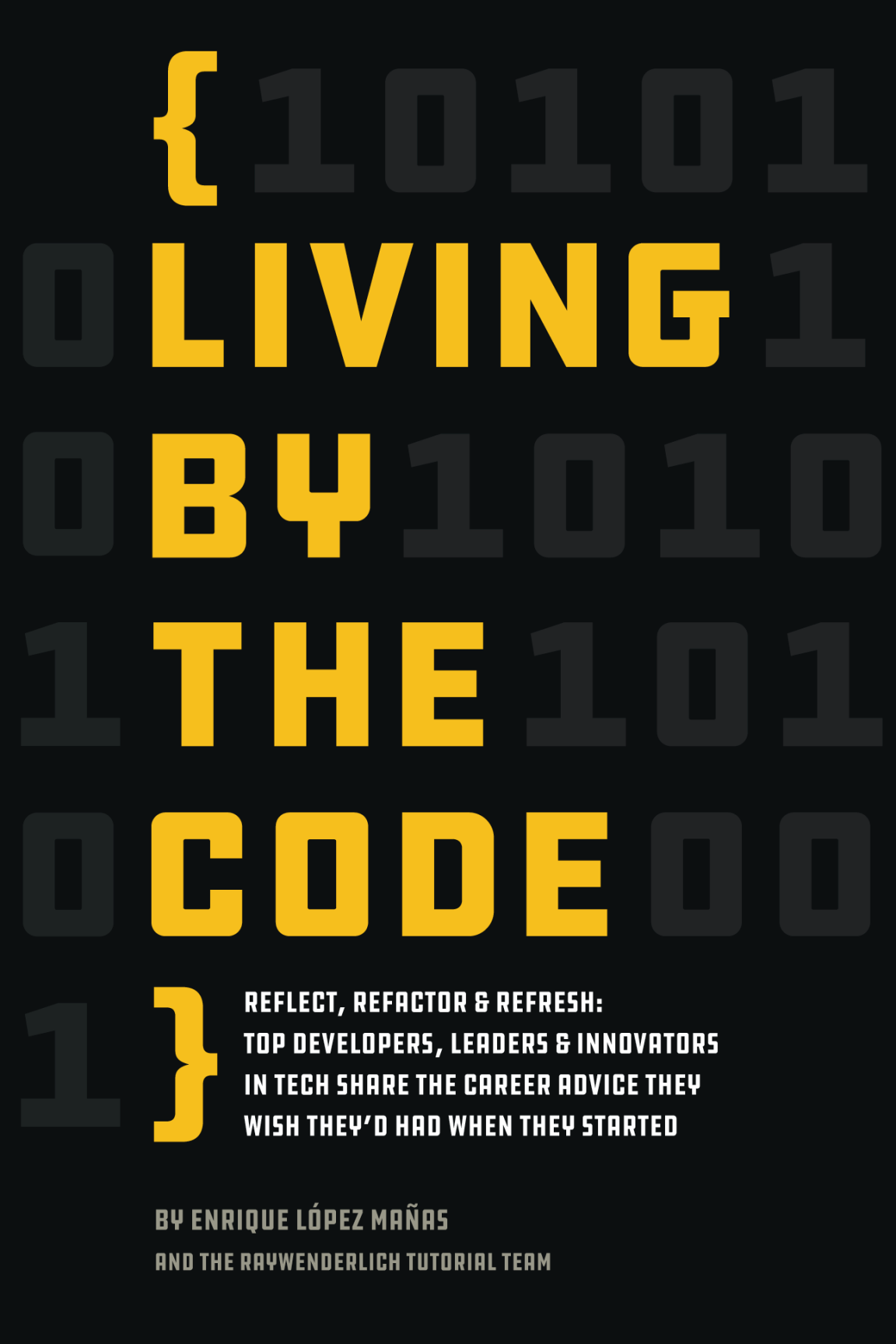

{ LIVING BY THE CODE }
REFLECT, REFACTOR & REFRESH:
TOP DEVELOPERS, LEADERS & INNOVATORS
IN TECH SHARE THE CAREER ADVICE THEY
WISH THEY'D HAD WHEN THEY STARTED
BY ENRIQUE LÓPEZ MAÑAS
AND THE RAYWENDERLICH TUTORIAL TEAM

LIVING BY THE CODE

{ LIVING BY THE CODE }

REFLECT, REFACTOR & REFRESH:
TOP DEVELOPERS, LEADERS & INNOVATORS
IN TECH SHARE THE CAREER ADVICE THEY
WISH THEY'D HAD WHEN THEY STARTED

BY ENRIQUE LÓPEZ MAÑAS

Living by the Code
by Enrique López Mañas

Cover font: DDC Hardware © Aaron James Draplin via Font Seed | fontseed.com
Cover design: Luke Freeman and Victoria Wenderlich
Interior layout: Olivia M. Croom

ISBN: 978-1-942878-82-7

CONTENTS

FOREWORD

Dan Kim
Basecamp

A career in tech is filled with tremendous promise and potential. There are opportunities abound all over the world, in every industry, for every interest. The possibilities of where your choices can lead you are endless, exciting, and insanely intimidating!

If you've ever found yourself wondering if you're on the "right" path, I can assure you that you're not alone. And it's not just a handful of us who've thought the same thing—every single one of us has pondered these important questions, none of which have easy answers.

"What does my career path look like? Where am I going to be in one, five, or even ten years?"

"Is management the right direction for me? If so, how do I develop those new skills? If not, how do I continue to improve as a developer and leader?"

"Why is being part of my developer community important? How can I get involved?"

I can relate because I've lived with that uncertainty, too. I've been in the industry for 20 years, and I had no idea what I was doing when I was starting out. Even today, while I'm more confident in my direction and choices than I have ever been, I'm under no illusion that I have it completely figured out.

That journey from constant doubt to relative confidence takes time and can be a difficult path—one that's made easier by a chorus of wonderful colleagues, friends, and mentors. Through the course of a career, many people will lift you up, point you in the right

direction, and course correct you when you need it.

Cherish these folks.

And while there's no doubt these personal connections will be your deepest, most valuable resources, it takes a bunch of time and energy to develop them. It can take many years, many jobs, and many life experiences to build relationships with the people who will serve as your guiding lights.

Surely there must be a way to learn from a broad base of successful folks more quickly?

There is, and you've found it! Enrique has pulled together a wonderful collection of experiences from a diverse group of people within our industry. He asks unique, personalized questions to each individual, helping us to understand how they got to where they are today—their philosophies on life and work, the resources that help them move forward, and the tactics they apply to be successful. If you've ever wondered how people have handled their fear of public speaking, starting a business, losing a job, work life balance, tough career transitions, toxic work environments, or remote work, you've come to the right place.

What makes these stories truly valuable is that these people are your peers. They are business owners, developer advocates, conference speakers, freelancers, managers, and developers of all levels, just like you. They aren't living in ivory towers, expounding generic self-help advice. I know many of the people in this book, and they're as real as it gets. No glamorized stories. No overnight successes. Just real stories of regular people making their way, building careers, and landing on their feet. I know these stories can help you because many of them have already helped me.

There's really only one guarantee about your career in tech, and it's this: It's going to be full of twists and turns. I never could have guessed that I would end up where I am today. You can't predict or fully control where you'll end up, but you can start laying the foundation for where you want to go. The diverse experiences of everyone in this book will help you do that.

My advice: Read through all the stories once and make note of the passages that mean the most to you. Consider them and begin integrating these ideas into your life and work. Then come back after a bit and evaluate. What's changed? What worked? What didn't?

Wherever you land, remember that the landscape is constantly shifting. So don't fret. Rather, embrace where you're at. Make adjustments that feel right to you. Keep writing your story. Share what you've learned. Mentor and lift up others. And most of all, keep your head up and take time to appreciate the good stuff.

Good luck on your journey. It's sure going to be challenging, but more than anything, it's going to be a ton of fun!

Dan Kim
@dankim

PROLOGUE

Enrique López Mañas

A few years ago, a copy of *Tools of Titans* by Tim Ferriss fell into my hands. I devoured it. It was different from the books that I usually read; it was neither narrative, nor fiction, nor an essay. Tim had interviewed several reputed folks from different fields, such as the literature, sports, or the film industry. The common denominator among all of them is that they were role models in their fields. They all had a story to contribute about their routines, beliefs or success stories.

When I finished the book (a book that you can get value from reading every few months) I thought: "I would love to have a similar manuscript interviewing people in tech." After making a living developing software professionally for around 12 years, there were a few things I was still struggling to understand—things that were not clearly available to me.

How can we make our daily routine as productive as possible? What do the daily routines of the titans of the tech industry look like?

How does a permanent job compare with contracting? How do we switch from one paradigm to the other? What are the advantages and disadvantages?

How can the transition between technical and management positions be done? Do we strictly need to become managers to make our career progress?

What is good leadership? How can one achieve those traits?

How can we effecitvely work remotely effectively? Are companies willing to accept this? What do they expect?

It is not easy to get a unique answer to those topics. There might be multiple answers. There is no formal education about it; you need to try by yourself, sometimes fail, and take your lesson. The problem is that we only have a limited number of decades in our professional career. So, in order to be successful, we need to observe what others are doing. We need mentorship. We need to hear the opinions of others that have walked that path.

This project aims to connect all those dots. I have personally failed many times trying to get those answers until I reached a professional status I felt comfortable with. Although the process of learning never ends; there is always the next step you want to reach.

In this book, I wanted to explore a set of core principles that were immutable for developers. Aspects we could research today, and that would still be valid in a few years. That was the motivation behind *Living by the Code*.

I ran this idea among some peers. I probed a few colleagues, asking them their thoughts and feelings about this. The enthusiasm grew slowly but steadily when most of the comments about it were positive. And this project finally kicked off.

I feel extremely proud to have convinced a few of the role models I have to be a part of *Living by the Code*. You will find some of them discussing technical interviews. You will find others discussing soft skills, academia, career or different work paradigms. You might find some of your answers to remote work. Each contribution to the book is brilliant on its own. I wished I could have had this book when I started my career in tech, more than a decade ago.

Our main wish and objective with this book is to deliver value to you, dear reader. We hope this will be a book that you might review from time to time, trying to disentangle some of the questions you have had in your career. If, as a result, your life improves and you're able to be more effective and successful in your job and life, we will have achieved our goal.

ABOUT THE AUTHOR

Enrique López Mañas is a software engineer, mostly focused on Android and backend development with Java/Kotlin. As a contractor, he has filled his hours with an eclectic variety of technologies: TensorFlow, iOS, Swift, NodeJS, RoR, and Python. Before *Living by the Code*, he published the books *Android High Performance*, *100 Android Questions and Answers*, and *100 iOS Questions and Answers*. He has been a member of the Google Developer Expert crew since 2014.

Besides his programming duties, he also runs a few side projects. He is the editor and maintainer of *Kotlin Weekly*, a mailing list that delivers news about the Kotlin universe every Sunday. He is the organizer of Kotlin Users Meetup Group Munich and Droidcon Vietnam. He speaks monthly on the *I/O Investing* podcast about finances and investing.

In his free time, he reads, writes, and runs long distances for extended periods of time. Over time, he has developed an interest in finances, communities, science fiction, and high-performance sports. He tries to practice transparency, and shares as much as he can on his Twitter account (@eenriquelopez).

COMMUNITY

“I wish someone had told me: You belong here. You’ll never have all the answers. Feeling like an imposter is normal and there are many others out there like you who feel the same way. Find them.”

—Sarah Olson

COREY LEIGH LATISLAW

“Bring people along with you.”

@corey_latislaw CoreyLatislaw.com

Corey is an international keynoter, an avid sketchnoter, and a technical leader at TAB/The App Business in London. She has developed high profile Android applications over the years, including Capital One, XfinityTV, and Pinterest and ran teams large and small. She’s a former Google Developer Expert (GDE) in Android and Google Developer Group (GDG) organizer.

You spend a lot of time volunteering. What drives you to do this? Can you talk a bit about the specific causes and groups that are important to you?

I want to have an impact on our world and I want my work to mean something and to benefit millions, if not billions, of people. I've worked in many organizations that are operating at different scales and I get excited by the ones that are solving real, human-scale problems.

One of the places I found to have a meaningful mission and a solid plan for implementation was installing solar power in rural Tanzania. They are connecting people to the world. I also worked with a nonprofit, installing computer labs in emerging markets. That was about connecting children with computers and technology and changing the course of their lives. That's the sort of thing I'm interested in.

In my daily work, I make space for individual mentoring, especially with women. The tech world can be a hostile environment, and I want to foster the opposite. I'm frustrated by the slow pace of change in the tech culture—there's been a lot of money invested in diversity and inclusion, but we haven't gotten anywhere—but this makes me feel that I'm making an actual difference.

Do you think we could do better than we're doing now?

Yes, absolutely—though I don't know exactly what that answer is. I've been doing a lot of thinking about how to embed diversity and inclusion into competency frameworks so that it becomes part of the fabric of the business culture and shows up on performance reviews. If we say that it's important we should be measuring it and we should be accountable for moving our companies and the industry forward.

In London last year, you were giving the keynote at a conference. How do you prepare for something like that?

I'm terrified of public speaking, believe it or not. I failed my first public speaking course back in college and slowly got to the point where I am now. Luckily, public speaking is a learnable skill.

In the beginning, I tried to look smart on stage. I overcomplicated the talks with technical details and spent loads of time preparing—for example, for my first keynote I wrote a 30-page research paper with citations for the script! I read more or less from the script on stage, but it felt too wooden. That was a lot of work and a whole lot of stress.

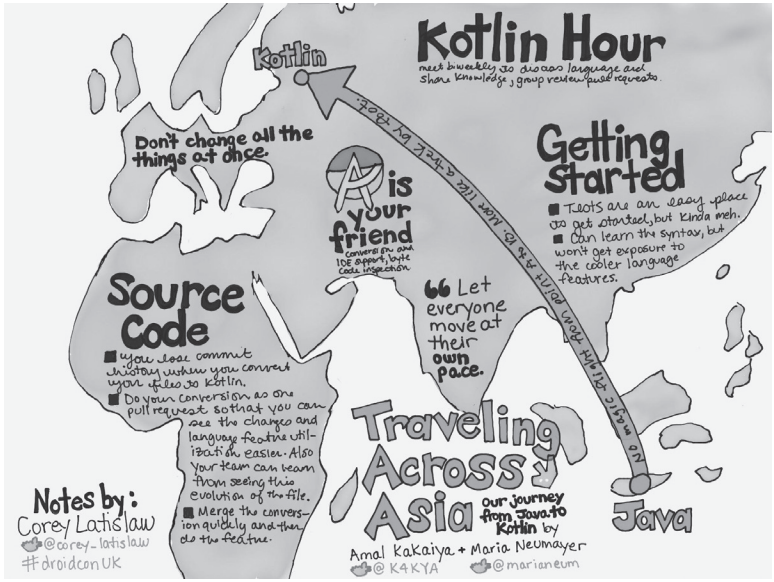
Recently, I've shifted to talking about my own experience, which is much easier to talk about. I've also shifted to a more fluid preparation style. For my "The Creative Technologist" keynote, I came up with the idea and wrote that talk in two weeks using improvisational techniques. I would record myself talking about my topic off the cuff, play it back, and keep the parts I liked.

To come up with a topic, I do a mind map to generate talk ideas. Then I find themes and write up an outline or abstract—either in Google docs or in a rough sketchnote, which is a visual note with a mix of drawings and words. I ask other speakers for feedback on which topics might be the most compelling.

Once I have settled on a topic, I do a sketchnote. This helps me outline the ideas and major takeaways. The next step is to make thumbnail sketches for the slides. I make myself practice aloud instead of just editing the script. I always do dry runs in front of people to refine the talk and mentally prepare.

The keynote I gave in London was called "The Art of Intentionality," and it was about how to be more intentional with your time and energy. At some point, I decided that I'd draw live on stage, and I ended up creating a ten-page worksheet for everyone.

I prepare intensively, which is probably why I don't do several talks in a short period. I usually like to give a talk at least seven to ten times. That gives me time to refine my presentation style and material. I try to choose topics that are going to be relevant for a long period.



Sample sketchnote, an image with a mix of drawings and words.

Your keynote speech, "The Art of Intentionality," will be relevant for years to come.

It's about figuring out your guiding light and setting goals and tracking success, as well as cutting things that are a drain on your time. It should be relevant forever.

Should all software engineers be involved in public speaking?

It's certainly accelerated my career and helped me develop better communication and research skills. I don't think that everyone needs to necessarily speak at a conference, but I do think that public speaking and storytelling are skills that everyone is going to need.

We all engage in public speaking at some point in our careers. If you're in a meeting, it's a smaller audience, but you still need to prepare for that so that you can contribute your best work. Meetups are a great place to share your unique view with the community.

The only people who shouldn't speak are the ones who can't keep sexual jokes out of their talks or people who are actively working against diversity and inclusion. Those people create a hostile environment.

> *Once you know you're building the right thing, it's important to make something that both technical and non-technical audiences will understand.*

What skills are the most underrated or under-appreciated for software engineers?

I don't like the term "soft skills," but our industry is seriously lacking them. If you look at senior engineering job specs, about half of those skills are related to communication, leadership, influencing, and growing those around you. These skills are really important, but I think that a lot of people focus on deep technical knowledge instead of growing the full toolbox of skills. It's going to limit your growth if you don't focus on them, so you might as well start now!

The ability to zoom out and see the big picture is very important to understand what we're doing and why. As engineers, we should be empowered to push back and ask questions to ensure we are building what's needed. It also helps to take a step back when solving a problem. Take a walk, take a nap, talk to someone else. You might find that the answer jumps out at you when you are "doing nothing."

Once you know you're building the right thing, it's important to make something that both technical and non-technical audiences will understand. That brings it back to thinking about the audience and to speaking and storytelling skills. Being able to bring people along with you and making your products understandable by everyone is a really important skill.

How can we improve our soft skills?

Books are my favorite resource. There's a business book I like called *Crucial Conversations: Tools for Talking When Stakes Are High* by Kerry Patterson, Joseph Grenny, Ron McMillan and Al Switzler. It's about non-violent communication. The whole concept is that we make up stories for why people act the way they do, but, in reality, it's rarely about us. For example, if I walk by you in the morning, and you say hello, and I don't respond, you might think I'm snubbing you or are angry with you. In reality, I might have a lot on my mind or not even have realized that you had said hello.

In these stories, we often place ourselves into the category of victim, and we label others as villains. This book helps illuminate that mindset and offers curiosity as a solution. The phrase that best illustrates a curious mindset is "tell me more." When we default to a "tell me more" reaction, we avoid creating these stories. I tell everyone to read this book.

What other books can you recommend?

One of my favorite books is *Drawing on the Right Side of the Brain* by Betty Edwards because it changed the way I think. It talks about left brain and right brain research, tells you how to tap into your creative side, and describes the five components to drawing. It focuses on drawing as a teachable skill, and I have seen a difference. I didn't know how to draw when I started to read this book. I first picked it up in 2005 (and twice more since), and now I can draw fairly well.

This book also really helps you to find flow. With coding, you build a picture of the architecture or code structure in your head. It's fragile. Being able to maintain it is important, but it's also important to get back into it quickly if you lose it. This book teaches you how to quickly tap into your intuitive side and see things holistically.

One of the more interesting things it discusses is how to bore your inner critic and make it drop out. For example, you would turn a photo

upside down and then draw it, and your brain is thinking, "What are you doing? That doesn't look like anything real! Why are you doing this? What is happening?" Eventually, it gets bored and disappears and then you are just drawing. Different tricks like that are really helpful.

Another creativity book I love is called *The Artist's Way*, by Julia Cameron. It's a 12-week program to recover your lost creativity. The basic idea is that creativity is a teachable skill and we all can tap into it. Its central practice is writing three pages every morning and taking yourself on an "artist date" once per week. In the morning, you get all the heavy stuff out of your head and onto paper, and then during the artist date, you're refilling the well of your creativity. Both of these books were influential to me.

Finally, I read *The First Ninety Days* by Michael Watkins just before starting this new job. Its overarching theme is how to be valuable as quickly as possible in a new organization. An organization invests in a new employee, and then it takes a while for that employee to start giving back—which can take about six months to break even. This book is all about front-loading and trying to get to value more quickly, which helps both the business and your career. Something I took away from it is the importance not only of building vertical relationships with your boss and her boss, but also horizontal relationships across the entire organization.

When I started with this company, I set a goal for myself to meet twenty people in the first week. I met thirty. I set up coffee and lunch dates with people and asked them who they thought I should meet. I kept branching out based on their recommendations. That helped me when, in the first three months, I launched an internal program for the company to get people into conference speaking and participating in meetups. I don't think I would have been able to get that off the ground without the support of the people inside the organization.

I've integrated key takeaways from each of these books into my life. You don't have to adopt everything, but there might be one or two key concepts in there that you'll find utterly invaluable.

What's something you wish you'd known at the beginning of your career?

How to understand people and their motivations. It's important to figure out how to work with and how to influence your colleagues, especially when you don't have formal authority. I think that those are all hard-earned lessons. I guess at the core of this is politics, which is seen as a dirty word in tech, but it's about building effective relationships. Learning how to work with different people, even if they are jerks or make you sad. It's important to figure out how to forge a productive working relationship with everyone.

What do you see trending in the industry?

Machine learning will be transformative, but it's hard for people to get up and running with it right now as it requires a base of theoretical and mathematical knowledge. Commoditizing it and making it easier to use will allow anyone to leverage machine learning to inform their decisions.

I'm also excited about the potential for 3D printing to solve medical and other real-world problems.

Finally, the usage of technology in art is really interesting and I have been seeing more installations and works recently. There will be a marriage of tech and art, which will redefine what art and technology itself means. These are the things I'm excited about right now.

Do you have any morning routine that sets you up for success, or in the evening if you work in the evenings?

I have a morning and evening routine. In the morning, I wake up at 6:00 AM and do my morning pages for 30 minutes and then meditate. Then I go to the gym or run in a nearby park for at least 45 minutes three to five times a week. After that, it is breakfast and coffee, and that's when I start creative time.

Usually, I will try and do at least two hours of creative work before I go into the office. That way, I get to work on my passion projects because I found that if I tried to do that in the evening, I was too tired, which made me feel like I was just a work machine. Now, I honor my values and make space for creativity.

In the evening, when I leave the office, I'm done. I don't look at emails or Slack or whatever. At home, I meditate or create, and then I make dinner and stop using electronic devices at 8:30 PM. I go to my bedroom around 9:00 PM and read—often falling asleep to the light of my backlit Kindle.

Does meditation play an important role in your creativity or concentration?

Yes. I started doing meditation in 2016. In general, it's made me more grateful, less stressed and anxious, and more tuned into what I'm thinking and feeling. I'm generally calmer. It's been a great addition to my life, so I've been doing it nonstop since then. I enjoy it.

What makes a good leader?

Empathy and the ability to create psychological safety are super important. If you're afraid of making mistakes, or if your employees are afraid of making mistakes, they're not going to tell you the truth when things are going badly, and they're not going to make the best decisions. It's better to have happy, motivated teams who believe in honesty and transparency. As much as they can, leaders should eliminate barriers that create a culture of blame, and they should create and reinforce transparency.

I also really value creativity in a leader. You can be creative in how you resource a project and build people's careers. The whole idea around creativity is you are taking what you know and somehow creating something new from that, and that can change the course of an entire business or a career.

The skill I want to develop is storytelling. I want to get people excited and bring them on a journey. I want to be able to explain what my mission is and have them be excited about it. I think I can do that in a keynote now, but I'd like to get better at doing it off the cuff and in person.

Have you had any useful experiences with failure?

I have only ever been fired once. I knew it was going to happen because I was protecting my team. I chose to do what I thought was right, and it led where I thought it would lead. Still, it was a shock because I am an overachiever and I like to excel. There was just no winning in this situation at all. I was afraid for a while about finances and whether I would be let back into tech. All these anxious thoughts were running through my head.

I was able to calm down and found out that failure wasn't so bad and scary. I took the opportunity that this unexpected time off offered me and started my own business. That shocking failure pushed me into trying something new, and it's been successful, so I'm really happy with it. Right now I have a salaried job, but I have the ability at any point to consult instead, which is a nice comfort.

COREY'S RECOMMENDATIONS

Crucial Conversations: Tools for Talking When Stakes Are High | Kerry Patterson, Joseph Grenny, Ron McMillan and Al Switzler

The Artist's Way: A Spiritual Path to Higher Creativity | Julia Cameron

The Bullet Journal Method: Track the Past, Order the Present, Design the Future | Ryder Carroll

GETTING TO WORK

“There’s always going to be people saying, ‘This is the one true way to do things,’ and then other people saying, ‘No it’s not. This is the one true way to do things.’ Being able to look at those claims dispassionately and figure out what actually works for you is the bigger thing to focus on in terms of learning.”

—Ellen Shapiro

MARIN TODOROV

“Focus on what you love doing.”

@icanzilb /in/marintodorov github.com/icanzilb

Marin is one of the founding members of the raywenderlich.com team and has worked on seven of the team’s books. He’s an independent contractor and has worked for clients like Roche, Realm, and Apple. Besides crafting code, Marin also enjoys blogging, teaching and speaking at conferences. He happily open sources code. You can find out more about Marin at www.underplot.com.

Many of our readers are interested in other works that have had an influence on you. What are the three that have had a lasting impact on how you do your work?

If I were to highlight only three books that have affected my work culture, I would go first and foremost with *The 4-Hour Workweek* by Tim Ferriss. This book—regardless of its clickbait title—has really been essential to how I have approached work for years. It has taught me how to declutter my schedule and focus on the things I really love doing. Reading—and embracing—this book got me started on my path to finding my work/life balance; never waking up to an alarm clock, and more advice in the book, are pure gold.

The second book that I'd say was a game changer for me I, unfortunately, read some 20+ years ago and I can't remember the title; it was about rapid application development with the earliest version of Delphi. I was impressed not so much with the content but the way the book was written. Part of the book was a first-person story about a developer who was tasked with developing a database-powered prototype overnight. You could really put yourself in the person's shoes as they discovered Delphi and made the most out of its features. This book had a lasting effect on me—even now, I still always try to cover real-life problems and guide the reader through a real-life developing process in my writing—including creating problems and bugs along the way, and fixing them later on.

As third, I'd add *Writing With Style: Conversations on the Art of Writing* by John R. Trimble. It's an essential collection of writing tips for anyone, both native and non-native speakers, who'd like to put down clear and readable prose. I bought it when we started planning the first ever raywenderlich.com book, and I made a habit of re-reading it before starting to work on a new book.

In addition to drawing inspiration from these books, what about personal insight? What is something you wish someone had told you back when you started software development that you had to learn the hard way instead?

When I was young, I naively believed working on software means working with reasonable, nice, and caring people. I had this romantic idea that being with your own kind automatically means you share the same beliefs and a universal understanding of being and working together. Unfortunately, I got to discover that software companies are made up of people just like life is—mostly the same rules apply, too. I had the chance to work at some amazing companies but had my share of bad experiences in the workplace as well.

> *"I think starting a career with the expectation that one is entitled to success if they check off some kind of a list is probably setting them up for a failure as few people are really entitled to anything."*

There are many people starting into software development today, who don't have a clear idea of the direction they should take to set themselves up for future success. What would recommend to people starting out as a software developer today, that would give them the best start to a successful career?

I never planned my career; my family didn't have the means for me to keep studying—I had to figure out on my own how to get by. I'm not in a position to give any advice on what's the best way to "set up oneself for future success in software development." But I will say I always did what I felt passionate about—I truly and deeply love programming. At first, I did it in my free time and it naturally turned into a career for me. I wish everyone had the same luck as I did and could turn their passion into a career.

In any case, I think starting a career with the expectation that one is entitled to success if they check off some kind of a list is probably setting them up for a failure as few people are really entitled to anything.

As part of your career, you've written a number of books for developers. Writing a single book can be a life mission for many, but you have an impressive record of authoring eight books and counting. And anyone can write a book, but it takes something special to write a "good" book. What is your secret to writing so many "good" books?

I disagree that anyone can write a book; in the same way that not everyone can develop software or design buildings, it requires certain skills. To name a few of these skills: It takes perseverance, thorough planning and humility. What I believe makes for a really good technical book is for the author to be able to explain things not from the position of a teacher lecturing an audience but rather as someone explaining to their peers in a friendly manner.

Additionally, it's essential to have an amazing publisher. "Traditional" publishers often times outsource editing, slap a random cover, release without a beep on social media, and then pay pennies on a sold copy. I'm really glad to have had the incredible opportunity to be published by Razeware and to build a lasting relationship, which has helped me improve my writing skills and my books year over year.

What has writing books taught you that you think can't be learned anywhere else?

One of the takeaways from the *The 4-Hour Workweek* is to focus on the things one feels they are best at and leave everything else to others. Collaborating with Ray Wenderlich and the team on all these books through the last eight years was an invaluable example of this great rule at work.

Elsewhere, I've seen excellent programmers decide to write a book on a topic they feel strongly about and fail not because they aren't

masters of their domain but because writing a book requires much more than being able to write code. There are many layers of putting a book together—illustrations, diagrams, technical editing, editing, planning, updating, marketing, print layout, and much more.

Going a few times through the process of putting a book out with Ray and everyone on this team has really taught me to focus on the parts I'm good at and support my collaborators and trust them with doing their part. This formula Ray envisioned some time ago proves to be a successful one year over year—it's amazing what the broader team can pull off in terms of putting out a crazy amount of new and updated books each year.

Your comments on traditional publishers lead me to wonder what other current industry trend you think is just plain wrong.

This question reminds me of that old Simpsons meme, "Old man yells at cloud"! Let's see. I wouldn't say, for example, that I'm a big fan of "the blockchain"—it seems to me it's been around for a long while and the only purpose it serves so far is financial speculation and scams. It almost feels criminal that it's a virtual thing produced by burning coal in developing countries to enable few folks to get even richer. I really wish we were better than that as a species.

Another is the fail fast startup culture. I don't believe any company's future is burning through human resources even faster than through its remaining cash runway. I think working this way robs people from the satisfaction of nourishing successful, long-term products, building long-lasting relationships, and maintaining happy healthy lives. I hope that sooner than later founders will begin to reject pumping cash into prototypes in favor of building lasting businesses instead.

What would you suggest is a better alternative to this trend?

I'd suggest that we need to move away from the fail-fast culture as a community. The only party benefiting of fail fast are the investors

as this culture is designed to run their numbers and not to help products gets released, companies grow sustainably, or people build knowledge and teams.

In thinking of building your own sense of productivity and nourishment, how do you start your day off with a bang? Do you have any secret morning routines that set you up for success?

I mentioned earlier that never waking up to an alarm clock has done wonders for me personally. I'm absolutely not a morning person and having a solid, uninterrupted sleep each night until I naturally feel that it's time to wake up has played an essential part in pulling off great amounts of work while keeping a healthy work/life balance. Another tip that helps non-morning folks like me is to use an app to actively block social media before noon—it really helps me get rolling in the early hours while I'm still somewhat able to be distracted.

Once you're up and working, how do you stay highly productive for long stretches of time?

Less is more! When I can plan my time myself I try to work as little as possible and focus on things I really like working on. This way I'm super productive in short amounts of time while spending the rest of my day walking, going to the museum, or generally nerding out.

MARIN'S RECOMMENDATIONS

The Spring: Charity Water project | charitywater.org

The Power of Habit: Why We Do What We Do in Life and Business | Charles Duhigg

Kid A | Radiohead

LEADERSHIP

“I definitely think good leaders are made... The biggest things I look for in a leader are self-awareness and humility. These two characteristics open people up to admitting and learning from their own mistakes, which makes them very coachable—the effort you expend on coachable people is always multiplied.”

—Cate Huston

RAY WENDERLICH

“Always keep learning.”

@rwenderlich /in/raywenderlich /raywenderlich

Ray is part of a great team—the raywenderlich.com team, a group of over 200 developers and editors from across the world. He and the rest of the team are passionate both about making apps and teaching others the techniques to make them. When Ray’s not programming, he’s probably playing video games, role playing games, or board games.

Ray, we would love to hear more about your story. How do you move from writing articles as an indie developer into founding a company with over 200 elite developers and editors? Was there any common pattern along your way in this process?

Before I started as an indie iOS developer, I was working at Electronic Arts on the back-end tools for a game called Warhammer Online. I absolutely loved the job and the people there. But the thing was: I had always wanted to start my own business. And when I got married to my wife Vicki in 2010, I realized if I ever wanted to start my own business, now was the time—before any major responsibilities (like kids) might make it a lot more difficult!

So I decided to quit my job and become an indie iOS developer. I spent a couple of weeks reading books and learning, then started making my own apps. I did the programming, and Vicki did the art. At first, we weren't earning much money through our apps, but after about a year we were earning enough to pay our bills. While I was creating apps, I was writing tutorials on what I was learning on my website, raywenderlich.com. I didn't think much of it at the time—it was something I was doing just for fun. The tutorials became popular, and eventually became too much work to do on my own. So I recruited a team of awesome authors, and we wrote tutorials together and became known as the Tutorial Team.

After a while, someone on the Tutorial Team suggested we write a book together. We did, and the book was very popular—so the next year we wrote another. We continued doing this, and eventually the books did so well that Vicki and I decided to focus on the web site full time. Since then, the site and the team have continued to grow, and we are now a team of 200 authors and editors from around the world who team up and create high-quality tutorials, books, and video courses, currently focused on iOS, Android, Unity, and Server Side Swift.

How does your team create an article or tutorial that provides value to the potential readers?

It definitely takes a lot of care to craft a great tutorial. The goal is to make the tutorial fun and easy to understand, and for the instructions to 100% work. The way I see it is developers are busy people with a very limited amount of learning time. If a busy developer trusts you with their limited time, you don't want to let them down!

In order to reach this high-quality standard every time, we have a guide of tips and tricks we've learned over the years creating tutorials, that all of our tutorial authors follow. For example, we have a rule called, "Build and run, build and run." This means each instruction section should end with a command for the reader to build and run so they can check their work as they go, just like you would normally while programming. That prevents you from a terrible situation where you follow along pages and pages of a tutorial, only to realize you typed a line wrong somewhere, and have no clue where!

In addition, every tutorial on our site goes through three rounds of editing: a tech edit, an English language edit, and a final check by a senior member of our team. That way, we can be sure that every tutorial that goes out is amazing!

We see a rampant term in our industry: ageism. Elders seem to move into management positions or disappear from technical careers. Is there any life after 40? How can we keep ourselves updated and staying technical as we age?

I think part of this is natural. As you become more senior in your field, you also become more experienced and knowledgeable, and those are desirable traits for a leader to have. That's why there is often pressure for developers to progress from an individual contributor to a team lead to a manager and onwards.

For some people, that's great and a valid career path. I think

the challenge, though, is sometimes developers feel like this is the only way they can progress in their career. However, that's not true: There is definitely tremendous value in super-experienced individual contributors, and having developers on your team who have been through the "school of hard knocks" for many years, in many technologies, is incredibly valuable. If your company doesn't recognize that, find another that does.

I think the first step is you need to decide what is best for you personally: Do you want to advance to management positions, or are you happier staying as an individual contributor? And then stay firm with your choice, even if you may get pressure in the opposite direction. If you live your life according to your values, you'll be happier for it.

For those entering the industry for the first time, there are very strong opinions of how to conduct interviews: whiteboarding, homework, pair programming. Which system do you find ineffective?

At Razeware, we use the following interview process for all positions (whether technical or not). This is optimized for speed of hiring, and it works well with a small team (we're only 15 full-time people) in a remote environment (we're 100% remote).

First, we use resume screening. I look through the resumes and weed out anyone I think isn't a good fit for the job. I try to give folks the benefit of the doubt at this point and keep a wide net at this stage.

Next, we do a video interview. I send them a short seven-minute video about our company, the job, and the interview process. This allows me to give more people a chance than I would if I were to schedule interviews with each person at this stage.

We then provide a challenge. At the end of the video, I give them a short challenge that should be able to be completed in a half hour or less, and it is related to something they will do on the job. Once they are past this stage, we do a half-hour interview. This interview is mostly focused on culture and big-picture fit at this point. To better

test their skills, we do a short paid contracting job. This is usually an eight-hour task that can be completed on evenings or weekends, and we pay hourly for the time—that way people are compensated even if we don't offer the job. The job mimics real-world tasks they would do if they take the job, so is an excellent indicator to us if they'll be successful here once they take the job—and shows them if they like the work!

Then, a final 1.5-hour interview for the top 1–3 candidates. We have a final interview where we go into much more detail on the person and their background. We send the offer to the best candidate at the end.

I prefer this method of interviewing because it closely mimics how our company works: we all work independently, remotely, and (mostly) asynchronously. We've had good success with it so far!

What would you recommend to an already experienced developer to keep progressing in the career?

Always keep learning. As an experienced developer, it's sometimes tempting to settle into a rhythm and do things you're already very familiar with. After all, who doesn't like that feeling of expertise! But I think you'll grow even further if you push yourself to always be learning something new—even if you have to go back to being a beginner again! Learn a new language, a new platform, a new architecture, how to be an effective manager—whatever you're passionate about! Not only will this help your career, but I think you'll find it a lot of fun, too.

You are mostly managing a team of international people. Some people might be interested in transitioning from a technical position into a managerial one. How can they reach this milestone; what would you recommend them?

The best way to become a leader is to demonstrate leadership—even if you aren't in a leadership position already. For example, a

few years back, someone applied to our tutorial team in an entry-level position: as a tech editor for videos. He always gave excellent feedback, he was prompt and responsive, an excellent communicator, and went above and beyond in helping the video courses he worked on stay on track and be as excellent as possible. Essentially, he was leading from within. He did so well that we promoted him to a final pass editor—that senior editor we have who does the final review on all of our tutorials. Again, he demonstrated his strong ability to meet deadlines, communicate well with his team, and went above and beyond to keep his projects running smoothly—and even helped out with areas outside of his responsibility! So we promoted him again, and I'm happy to say he is now our iOS team lead—Richard Critz—and is one of the best leaders I know.

So if you want to become a leader, be a leader now—in whatever role you already have. Take on extra responsibility, and show people you know how to get things done. If you have good leaders in your company, your efforts will be noticed and rewarded.

What is the biggest mistake you see junior developers making over and over again in their work? Why do you think they keep making this mistake?

As someone running a tutorial site, I see too many junior developers getting stuck in "tutorial purgatory." They just read tutorial after tutorial, but are a bit afraid to go off and build something on their own. Tutorials are an incredibly helpful way to learn quickly, but every once in a while, it pays to take a break from tutorials, and try implementing what you've learned on your own. It may be hard at first, but that is how you learn.

By doing this, you'll solidify the knowledge that you've learned in the tutorials through practical experience. Then you can come back to some more tutorials to learn even more, but now that you have some solid experience, the concepts will make even more sense. The tutorials and practice combination is a virtuous cycle.

What is the biggest obstacle, technical or personal, that you have had to overcome to get where you are today? How did you get past that obstacle?

By far, my biggest challenge so far has been identity. Throughout my life, my identity had always been as a software developer. That's what I did and enjoyed doing. And I had this idea that managers were always "the people who did nothing," while the developers did the "real work." But as my company grew, I found myself doing less and less software development, and more and more management. So when I looked in the mirror expecting to see a software developer, and instead saw a manager, there was this mental dissonance that made me feel quite badly about myself. I faced a choice: I could either change my actions (stop being a manager and get back to programming), or I could change my identity (and think of myself as a manager/leader/entrepreneur). I obviously decided on the latter. And although it seems easy to just make that mental switch, it was still a struggle for me—and took many years to change the way I thought about myself. Luckily, it was mostly a matter of time for me, and I'm very happy with my current role!

What is your leadership philosophy? That is, what core principles and beliefs guide you when you are in a position of leadership or mentoring others?

Follow the Golden Rule. Care about your people and treat them the way you would like to be treated. If you do that, everything else works itself out. The most important part of any company is your team, so nurture them with care.

What are the best apps or tools that you just can't live without? This could be in your personal or professional life.

For project management, we use Trello, which is a lightweight tool that can be easily used to manage multi-stage group projects. We use it to manage the process by which we create tutorials, books, and videos.

For documentation, we make heavy use of Confluence, which is a wiki-like tool by the creators of Jira. As our company grows, it is important to have a central place to store links to all the information we create. This way, we can have a "one-stop shop" for all the information we have as a company.

For project plans, we designate a person who takes the lead and creates a proposal in Google Docs, then sends it to the rest of the team for asynchronous comments and review.

For my personal notes, I use Evernote, which is a tool to write notes and sync them across multiple devices, quite heavily. For example, I have kept notes on every video chat I've had for the past 10 years in Evernote, which is incredibly helpful when I forget a conversation I had a few months ago. I also use it for TODOs and other notes I need to keep for future reference.

Although we prefer asynchronous communication as a remote team, sometimes we need to talk to each other instantly, and we use Slack for that.

> *"It's critical to give yourself permission to take breaks when necessary to recharge your batteries—again this isn't weakness, but is normal and essential."*

What is something you have learned "the hard way" in your career, that you wish you'd known 10 years ago?

Know your limits. Jobs can be demanding—especially for developers and leaders. There are times you'll need to burn the midnight oil, but it's essential to be aware of your own limits, and to understand what you can get done each day/week, in a sustainable way.

When you are an organized and responsible person at work, people will notice that you can get things done, and next thing you know they'll throw more and more work at you. As a person who gets things done, you might have a temptation to keep saying yes, and keep working harder, doing more and more.

But part of being dependable is to understand your limits, and sometimes you may need to push back—even to your boss—and say, "I'd love to do this, but right now my plate is full. Can you help me prioritize these tasks so I can focus on what's most important, or is there anyone else who can help with this?" This isn't a sign of weakness; it's actually a sign of strength, and you'll gain even more respect for it.

In addition, it's critical to give yourself permission to take breaks when necessary to recharge your batteries—again this isn't weakness, but is normal and essential. Remember a career, or a business, is a marathon and not a sprint. It's more important to have the energy to keep going day after day, than overdoing it and burning out. Burning out is real—I've been there many times. It took me a while to learn and accept my limits, and I hope you can find that perfect work/life balance in your life faster than I did.

RAY'S RECOMMENDATIONS

Toastmaster's International organization | toastmasters.org

"Identity" RWDevCon Inspiration Talk by Vicki Wenderlich | raywenderlich.com/1855-rwdevcon-inspiration-talk-identity-by-vicki-wenderlich

ACKNOWLEDGEMENTS

A book is never a single-person journey. There are so many folks and friends that have supported me through this project.

First of all, my eternal gratitude goes to our clan of mentors. I would have never dreamed to gather such a gang of brilliant folks for this project. Each of them has an amazing story to tell, and after having this virtual conversation with them you will surely acquire knowledge that will stick with you during your career. They are the main ingredient of this book.

Thanks to Chris Belanger, Manda Frederick and Nicole Hardina who served as editors for this book; Tiffani Randolph for her marketing and promotion efforts; and Vicki Wenderlich and Luke Freeman for their cover design work. It has been several months of collaborating and working together. I have learned so much from you, and it was delightful to do this together. Thanks, Ray Wenderlich for believing in this initially crazy idea and agreeing to be a part of it.

A special thank you to our beta readers who generously provided feedback and editorial support: Annette Marlette, Márton Braun, Ray Wenderlich, Mark Powell, Katie Collins, Vincent Ngo, Scott McAlister, Joe Howard, Richard Critz, Victoria Gonda, Chris Belanger, Tiffani Randolph, Arthur Garza, and Ben MacKinnon.

Thanks to all my colleagues that have supported this through informal conversations, barricaded behind a cafe. Nick Skelton, Marius Budin, Xavier Jurado, César Valiente, Iñaki Villar. You listened and proposed ideas, and many of them were incorporated into this book. My gratitude for sharing your feedback.

Thanks to the GDE program, who actively supported *Living by the Code*. Thanks to Benjamin and Alina for your input and faith on the project. Thanks, David for having supported during all these years my efforts as a GDE. I am so glad to have had you as mentors

as well. Thanks to all the folks that have created and worked with me on content over the years. You are the best school of knowledge I could have hoped for.

I could not have thought of a better foreword for the book than the one written by Dan Kim.

Last but not least, thank you Le Vu Nhu Quynh for your eternal patience with me. You supported me through months of ups and downs in this project. You make this world shine.

It is not possible to write each individual name on this section. That doesn't detract from all the support I had. Big thanks to all of you who made this project possible. An author is only a small gear of this fantastic machinery.

GROW YOUR CAREER

It's hard to make it as a developer in today's tech world, and even harder to find mentors who can give you the straight advice on what it takes to go from good, to great, to amazing.

But — what if you could pick the brains of today's top developers, leaders and innovators in tech...

...discovering the paths each person took to get where they are today...

...learning from the mistakes and pivots they've made in their careers...

...and start using the exact tips and techniques that keep them at the top of their game?

LEARN FROM EXPERTS

Living by the Code brings the experiences and insights of over 40 of these influencers together in one single book, to help you grow your career in today's ever-changing technical landscape.

If you're struggling to make your mark in the competitive tech industry, then this book is what you need to make your best career move — no matter whether you're a developer for a big corporation, a scrappy solo entrepreneur, or someone in between.

It's like having dozens of tech's best mentors — right at your fingertips.

ABOUT THE AUTHOR

Enrique López Mañas is a Google Developer Expert, indie developer, and avid contributor to the open source community. He runs the Kotlin Weekly newsletter, and is a part of the global Google LaunchPad accelerator initiative. He's interviewed dozens of developers, leaders and innovators for this book, to discover the advice they would give to anyone else trying to make it in the tech world.