LAND *the* JOB

Six Months to Start Your Software Career

RYAN LATTA

Land the Job

Six Months to Start Your Software Career

Ryan Latta

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

This version was published on 2020-02-19

ISBN 978-1-7344861-0-0

Tweet This Book!

Please help Ryan Latta by spreading the word about this book on Twitter!

The suggested hashtag for this book is #LandtheJob.

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

#LandtheJob

For my family, writing community, and the people who put their faith in me to help them with their career.

Contents

Introduction

Welcome to *Land the Job*! Before we get into the meat of the book, which will equip you to land that first and future job in the software industry, I thought I'd take the time to share why this book was written in the first place.

A few years ago I was working in a small consultancy, and we brought in a friend to join us. He didn't have any previous software experience, although we thought we could get him up to speed quickly. This began a two-year-long mentoring program between our new hire and me.

We covered all of the essentials needed to be a software developer. We taught what variables and functions are. We showed how to organize code. We demonstrated source control. What we taught grew in complexity as he learned on the job and from his mentors.

Eventually, he was ready to spread his wings and get out into the industry. He asked me for help in finding that first job, and our mentoring relationship changed. Over the coming months, I helped him develop his resume and sharpen his ability to find employment through scouring job postings and leveraging his network. We talked through how to get through the interviewing process and even salary negotiation.

After a few months, he landed his job. He got the salary of his dreams. My payment was a bottle of bourbon, and it was one of the best bourbons I've had.

I wondered if I could help someone else in the same way and I signed up to be a mentor again, this time with the focus on getting that first job in software. I suggested it would likely take six months to get that first job, and at the end of their six months they had their first signed offer.

While I wouldn't claim I have all the answers, I have developed a repeatable way to get a job in software. I honed this system through my career of ten years, and then taught others to do the same.

My goal is to equip people with the skills needed to get that first job and several afterward.

The Industry

If you look up the fastest growing industries, software development[1] has remained one of the most in-demand for years, and that trend isn't slowing down. If you keep looking, you'll find companies everywhere expressing the sentiment that they cannot find good people quickly enough.

You'll also find many people struggling to get jobs.

Companies are working hard to attract talent while simultaneously failing to hire it.

How can this be? Is the job that difficult?

Not really. My current hypothesis is that the software industry is so bad at finding people that it excludes perfectly qualified people because the candidates don't know how to navigate the hiring process.

I aim to equip everyone with what they need to be successful and maybe even prompt better hiring practices that render this book obsolete.

Beyond that, if we look at what it's like in software development compared to a lot of industries, you could do much worse. A given day for a developer will involve showing up to work in whatever clothes you're comfortable in at a time that is convenient. Your manager may show up at 8:30 AM, but developers might not show

[1] "Fastest Growing Occupations" bls.gov. https://www.bls.gov/ooh/fastest-growing.htm (accessed December 26, 2019)

up until 10 AM without any worry. You'll work at a computer that you've set up just for you. You can get some coffee and maybe a bite from the snacks that they provide. A few morning meetings with the team and you're into code. You can get up and take breaks with your colleagues. There will usually be an area to relax in, maybe play a game, have some drinks or snacks, and chat. The day ends, and everyone heads home.

The demand is high which means compensation is quite good considering the amount of training needed. The mobility of the workforce means that you will be able to search for increasingly better opportunities with little to no cost to you. This job can and will pay six figures—if you can get in.

There will be the other trappings that come with any job. Deadlines, poor communication, frustrating decisions, and so on is the stuff that makes any position feel like work. But you'll be paid well to get through those things, and there will always be another opportunity if it is too much to bear.

Software development isn't a job where you'll be at significant risk of physical injury. It isn't one where you'll be at high risk of being fired. It is a job that asks that you know the tools, techniques, and that you work well on a team. You can learn to do this job, and I can show you how to get it.

Thanks

Getting a job is hard work. I don't think enough people talk about how hard it is. My preferred phrase is "Soul-crushing." Applying for jobs and hearing nothing or hearing rejection over and over is tough to accept. I've often wondered if I wasn't good enough or if I wasn't cut-out for the industry.

I don't think that's true, but rather that the game is rigged. I believe at this point that getting a job is a unique set of skills that can be

taught and learned. If you have the right skills for this maybe the search for that next job won't be soul-crushing.

There are so many things I'll cover in the book about the nature of getting a job in software that I don't want to spoil what is coming up in the chapters ahead, but I do want to make it a point to write this:

Thank you so very much. May you find a way to your job in the pages ahead.

3 - Resumes

Resumes are the currency primarily used in gaining interviews. It may seem old-fashioned, but as of 2020, this is still the case. Every company you apply to will ask you for a resume. Even if you have someone recommending you on the inside, you'll have to give your resume. If it's that ubiquitous, why take any chances with it? Make your resume a tool that gets you the interview.

When I started my career, I began by reading a book called, "What Color is Your Parachute?"[2] I used this book to develop my first resume and have since refined my process and taught others how to build resumes. This chapter is a dive into how to make a resume that leads to interviews—a resume that gets a yes in sixty seconds or less.

You may be wondering why I say sixty seconds? Well, if you're in the industry long enough, you'll start reading them too. You'll begin interviewing people after a few years. You'll find yourself on the other side of the table from a hopeful developer! After reading a few resumes, you'll notice that you can very quickly conclude whether you would be interested in speaking with them or not. Other people over the years have confirmed this experience. In some cases, it takes longer to print the resume for review than it does to review it.

The challenge behind a resume is to pique their interest in sixty seconds or less. To do this, know your audience and write compelling, outcome-oriented statements.

[2]Bolles, Richard N. (2019). What Color is Your Parachute? New York, NY: Ten Speed Press

No Lying

I'd be remiss if I didn't pause to say this is where ethics come into play. There are people who will happily lie to get a job. You may find that after reading this book, you'll know exactly how you could lie your way into a career. I suspect some people will do just that. Don't take this path.

The software development community is smaller than you'd think. After a few years in a market you'll wind up with a network of peers that connects you to almost every other developer and company in the area. If people find out you lied, it will follow you, and you will not be hired again.

What constitutes a lie? If you claim you worked at a place you never did, that is a lie. If you say you have a skill that you do not have, that is a lie. If you say you have an education you don't have, that is a lie. If you're using someone else's resume, that is a lie. I wish I didn't have to write this, and for most people, this paragraph will seem completely unnecessary.

I know from experience it is necessary. I've bumped into enough frauds over the years.

Myths

Now may be a good time to talk about some odd things you've heard about resumes. First, let's talk about resume length. You may have received advice that your resume has to be one page. You may have heard two pages. I've yet to meet anyone that discards resumes that are more than two pages, so length isn't your problem. The amount of time they'll read is. I am guilty of this. I sometimes receive resumes that are six-to-nine pages long. I'll have made a decision by the end of the second page.

The bottom line when it comes to length is not to worry about it. Do front-load your resume to have the highest impact as early as you can, but I have no reason to believe that length will work against you.

There are also some interesting bits of advice people give around making your resume stand out more—things like colored paper or artistic fonts, or other crazy stuff. Yes, there will be a chance that someone will be blown away by your creativity. Busy people, though, are more likely to find it jarring and irritating instead.

If you want to show your artistic side, *Cover Letters* and *Portfolios* are an excellent way to demonstrate creativity that is unlikely to turn people away from you prematurely.

There is a lot of advice out there about formatting your resume. As long as your resume falls into one of the three major kinds, you'll be just fine. I recommend a skills-based resume in this book, but I'll cover the others briefly. Here are some example formatting topics that seem not to matter when you focus on impact:

- Objective statements
- Career synopsis
- Correct number of bullets

Disqualified

Now that the myths are out of the way it is time to cover some ground rules. I don't recommend breaking these rules, as it comes with a higher chance of failure. The major things to watch for are:

- Spelling Mistakes
- Poor Readability
- Wrong Names
- Grammar Mistakes

Spelling Mistakes

If your resume has spelling mistakes, you're finished. That doesn't mean everyone will be scanning with an eye for typos, but if someone finds one, it will be brought up within the company. It may not disqualify you outright, but it will be a turn-off. Remember, resumes are still the primary currency in use for deciding to interview. Spelling errors are so egregious when spellcheck is electronically built into almost every tool on the planet. It shows an extreme lack of detail and general laziness.

Eliminating Spelling Mistakes

When it comes to catching typos there are a few things you can do:

- Have someone else read it
- Uncapitalize your entire resume
- Search for easy-to-mistake words like their and its
- Double-check spellings of specific technologies

Another set of eyes will assuredly find more typos than you reading it yourself. Get it in front of someone and ask, "Do you see any mistakes?"

Next, you can make sure no words in your resume are capitalized. Capitalized words are treated as proper nouns to most spell-checkers, and that means they'll skip them. So lower-case everything and your resume will light up. A lot of the new found things you will realize are fine. Consider adding them to the dictionary and fix it as you go. Uncapitalizing gives you a chance to catch mistakes with a wider net.

There are a few words that most of us mess up all the time. Search for them and see if you have the right one in the right place.

- Their/they're/there

- Its/it's
- Affect/effect

Last, for any key technologies and unique words you're using in your resume, take the time to double-check their spelling by going to the source, copying it, and pasting it back. Nothing stands out more than a typo in a core element of the tech-stack that you're interviewing for.

Poor Readability

If your resume is hard to read, you're finished. This advice mostly comes down to remembering that I as an interviewer will only give your resume sixty seconds. If you make that sixty seconds irritating due to formatting or a very difficult to understand style, I'd rather pick up another resume. Keep your resume format and design clean and consistent. Build it for the reader and not for your own sake of design. You'd be surprised how easy it is to see when things are inconsistent. It is off-putting and often begs the question, "How did they not notice and fix that?" As far as resume blunders go, it's right up there with spelling mistakes.

You can distinguish yourself easily by avoiding any spelling errors and having a clean, simple format.

As a small story, I recently worked with a manager who showed me how they read resumes. It was alarming. After they had done their sixty-second read, they would turn on the invisible formatting. They would make all the spaces, tabs, and returns appear. If they saw that sometimes you used space instead of a tab, you were finished. They wanted the resumes they saw to be crafted by someone who took the time to do it right even in the areas you cannot see.

Wrong Names

Here is another common and yet foolish mistake I see all the time: A resume with different names on it. This most commonly happens with agencies and consultancies, but it can happen as a first-time job-seeker too. It is very easy to start with someone else's template and forget to make that one last name change. Suddenly the resume has two names on it. If someone sees that, you're finished.

Grammar Mistakes

Lastly, keep an eye on your grammar and punctuation. They're not likely going to notice if you missed or added a comma in the wrong spot. However, they'll notice that some bullets have periods and some don't. They will notice sentences that run on or are incomplete. They'll notice the obvious grammar mistakes. If they find these mistakes, you're finished.

Audience

Knowing who reads your resume is critical in the process. Broadly, three distinct types of people will read your resume. Writing a resume that can appease each of them is challenging but very doable. The three major audiences are:

- Human Resources Professional
- Hiring Manager
- Peer Developer

Human Resources Professional

The Human Resources professional is the easiest to write for. They'll be involved throughout the interviewing process, but it'd

be a mistake to assume they understand the job you're applying for. They have specific training and skills unrelated to software development. Yet they will read your resume first. How do you write for someone that doesn't know the job?

Well, the answer is in the job posting. The job posting was written by hiring managers and peer developers. It contains elements that the Human Resources professional can quickly scan to see if the candidate is a good fit. For example, if the HR professional knows that Java is a core skill, they can search your resume for the keyword Java. If it isn't there, they'll pass.

Writing for Human Resources is all about putting keywords in places that are highly prominent and searchable. You'll hone your ability to place keywords as you become proficient at reading job postings and writing resumes.

Hiring Manager

Next up is the hiring manager. This is your primary reader. They will decide whether to interview you in sixty seconds. The HR professional scanned your resume to ensure you have all the pieces there in terms of skills and claimed experience through keyword matching. The hiring manager will read your resume and weigh it against what they know they need.

For this person, writing your resume for impact and outcomes is key. This makes a very clear connection to how you work to achieve a goal. Almost every resume in the wild simply lists the duties and tasks performed. Yours will go further to say how you made a difference in those duties. You'll stand out. They'll want to ask, "How did you do that?" Writing to pique this person's curiosity is the real game.

We'll cover later how to adjust your resume for each job you apply to, but the hiring manager is your primary target. You can list skills you have that you know they need. You can adjust some

of your experience statements to emphasize certain things you've learned by researching the company. For example, if the company is a consultancy or agency, maybe you want to emphasize how well you worked with clients or external people. This highlights a highly desirable competency that agencies and consultancies would almost rather have instead of technical prowess.

Peer Developer

Last on the list is the peer developer. These are your co-workers, though likely in a more senior position. They are unlikely to inform the decision to interview you or not. They will see your resume as they prepare interview questions. For this audience what you put in your resume in terms of skills and experience will lead them to ask questions more predictably. As an example, even though it had nothing to do with software development, I kept the fact that I lived abroad on my resume for years. It always garnered questions that allowed me to tell a series of stories that highlighted specific qualities, traits, and circumstances.

Pay attention to the things in your resume that consistently generate intrigue. I was surprised at how living abroad did this for me. You can, in time, find ways to put things into your resume that your peers will find too intriguing not to ask about. This is great for you, as you can prepare your story ahead and change the interview dynamic.

You can have the same effect with the skills you list, though this is a bit more challenging. Expect questions about every skill you list that is in line with the job posting. You may find that certain other skills also generate questions that you can similarly use to your advantage. Two examples that come to mind are mobile development and test-driven development. These two items have generated numerous conversations for me in interviews over the years. I've been more than happy to oblige with a well-practiced response.

Skills-Based Resume and Friends

I recommend building what I call a skills-based resume. There are others I'll cover at the end of this section, but I want to spend time covering the type of resume I think works very well in software development.

If you transition away from software development, you'll likely need one of the others. I'll include them for completeness' sake, but for now, let's pretend that skills-based is the only one.

A skills-based resume emphasizes the technical knowledge or skills you have in combination with practical experience. You might be thinking that this is your first job and that you don't have those things. Bear with me. You have a lot more than you think.

The basic layout I recommend in a skills-based resume is this:

1. Skills and Knowledge
2. Awards, Speaking, Relevant certifications
3. Experience
4. Education & Miscellaneous

Skills and Knowledge

Skills and knowledge come first for a reason. These are the things the HR professional and hiring manager are going to look for first. Don't make them search. Give them the goods right out of the gate. Are you applying for a job that requires React? Put it in your skills section if you can without lying. Do they want to see experience with agile development? Put that up there too.

I break my skills section down into three sub-sections. First are the technical skills. These are things like programming languages or source-control. Then come the frameworks and libraries. Here I put things like React, Spring, JBoss, etc. Last I have things like

techniques or knowledge. Here I put things like agile development, automated testing, design patterns, etc. Those three sections have worked well for me, so find your way of splitting those things up.

If you're not sure what makes sense, you'll likely see common patterns after reading job descriptions for a while. They'll almost always include some fundamental technology backed with specific frameworks or libraries, and then ask for specific knowledge or techniques. See what makes sense.

As a small aside, every word you choose matters. If you're applying for a Java job and you can put it down without lying, put it first. Don't make them read to the end of a large list of things that don't matter.

Last, this skills section isn't every single skill. They're the ones you are choosing to emphasize. I phrase mine *Selected Skills*. I want to imply that I'm only showing the things that are relevant while having more to offer. Also, if you try to list everything you'll annoy the reader, and remember you only have sixty seconds of their attention.

Outstanding Awards and Other Flash

For the first time job-seekers, this section may be omitted depending on your background. I put this second because if you have something you can use to highlight excellence then put it here. Maybe you got an award at a different job. Maybe you've done some public speaking. Maybe you're published. You decide what and how you want to show those things off, but if you can keep it short and sweet put it here.

I list an award I received in college, though I have since pushed it further down in my resume to make room for other things. Now I have certifications and public speaking that I put in its place.

The point is that you can use this spot to show off a little. I recommend doing it now because you've already shown you have

the skills, now show them your excellence. After that, they can learn more about the impact you've made.

Experience

This is where you'll spend the most time and energy on your resume. Learning how to present your experience, even if it isn't software-related, is going to take time, practice, and a lot of refinement.

Before I go further, remember what I said about length? It doesn't matter. I do want you to try filling out at least one full page. Beyond that, go crazy.

The name of the game with writing about your experience is to write with an outcome or impact in mind. This is a sharp contrast to listing the activities, tasks, or duties you did on the job. It separates you from everyone else by showing that while everyone else writes code, you create more profit, customers, and so-on. It takes practice and time to learn how to write this way, but it will distinguish you quickly.

Let's have an example contrasting the outcome-based experience statements from task-based ones.

Task-based

Developed software across the entire tech stack

Outcome-based

Developed software product that led to a 20% increase in customers

Put yourself in the shoes of someone who can only hire a few people to add to their team. If you had to pick between a resume that listed task-based experience or an outcome-based resume, you'd be drawn to the one who wrote outcome-based experience.

It is natural to feel like you cannot come up with any kind of outcome for what you've been doing. This takes time, practice, and reflection to identify. So we start the practice here. Here is a way to get started. Imagine telling a stranger about the project or the job you had. Try to complete the phrase, "We accomplished," or "We set out to accomplish." This will get you closer to the outcome you attempted to deliver. You may feel like you can't claim that outcome, but you did play an active role in achieving it. You may also realize that this outcome wasn't achieved while you were there. That is ok as well because you can word your experience in a way to reflect that. You may, through your reflection, also note that there were other impacts or outcomes. That is wonderful as they give you more material.

Here is a list of potential outcomes you might find in your experience and history so far:

- Financial gain or protection from loss
- Cost-cutting or efficiency gains
- Improvements to process
- Handoff reductions
- On-time, on-budget delivery
- Customer delight
- Customer acquisition or retention
- Increased community engagement
- Removal of defects

The list is endless for what you can find, but this list is here to get you thinking about what outcomes you can find and speak to in your experience.

You'll spend most of your resume-building effort trying to pinpoint the outcomes and tie them back to the job you had. Take the time to find your outcomes.

Now, in terms of how to write your experience sections, there are a few key elements to put down.

- Company
- Dates worked (Year is fine)
- Title
- Bulleted, outcome-oriented list

For the company, dates worked, and title you can list them all together in any way that is easy to read, and below that the experience follows.

When it comes to how many bulleted outcome-oriented items to put down, I advocate for two things. First, that you keep the number fairly consistent across your jobs, this may have no impact on anything whatsoever, but it gives an appearance of tidiness to your resume, and I think setting a few limits helps refine the impact on the few that you have. Second, I recommend that you stay somewhere between 3-5 bulleted items per job.

Putting all that together you may have something, though not formatted well, that looks something like this:

> Acme Widgets
>
> 2018-Present
>
> Software Wizard
>
> - Co-created architecture of software product that gained 30k users in its first quarter
> - Developed data transformation pipeline that processed 250 million records every day
> - Implemented automated testing suites that removed serious production defects prior to release

And just for comparison, a similar task-oriented one that your competition will use:

Acme Widgets

2018-Present

Software Wizard

- Architected software platform using Spring, Java, React
- Developed ETL process using Hadoop
- Wrote automated unit tests

Hopefully, this contrast seems extreme, if not comically so. While I made up these examples to point out the differences, the task-oriented example is very common to see in the wild. As you begin to see others' resumes, you'll see the same thing, unless they read this book. Even a few outcome-based bullets will put you in the pile of people a manager wants to talk to over the person whose experience amounts to "Showed up to work."

You may have noticed that in the outcome-based example, I didn't mention any specific technologies that were in play to achieve those outcomes. The reason is that, in the broader picture of this resume format, you can list the skills and technologies you want to be interviewed on there. Listing them here in the experience is redundant. Also, by leaving them off, you have a bit more freedom with the story you tell when someone asks you about that experience.

You can use your skills section to highlight the ones relevant to the job posting mixed with others you can speak to and impress them with. You use your experience to show that you're more than someone who writes code and you have a chance to tell a story beyond the technologies you used. If you mentioned technology in those experience bullets, you might be asked very specific questions regarding the technology's role and limitations in achieving the outcome.

We'll cover this idea about controlling the interviews in the interviewing chapters. For now though, writing your experience in a way that invites a question like, "Tell me how you did that," is preferable to, "What scaling limitations did you bump into using Hadoop?" One is a story, and the other is a test.

Other Resume Formats

There are two other types of resumes[3] that exist, but they have their uses outside of getting a software job.

First, is a purely experiential resume, sometimes called a chronological or traditional resume. This resume relies almost entirely on your experience starting from now and working backward. After a few years, you may find this resume format makes sense. In the beginning, you won't have enough relevant experience to rely solely on that. To get through the HR professional, you'll have to mix your keywords into those sections carefully to not be too overt about it, but also allow them to find the information they need to move on.

The other major type of resume you'll see is a functional resume. This resume is all about saying what you can do. Again, you'll note the format I've provided contains this element in a skills section. A functional resume takes this a bit further though and will sometimes have a summary section where you try to very convincingly list the traits, qualities, and skills you bring to a situation. There may be no experience on resumes like this. This resume may sound perfect for the first-time job-hunter, but in truth, it is very challenging to write in a way that looks reasonable.

If you transition to management, you'll likely find this style of resume is more common and want to build one yourself. In this format, you can outline a summary of your accomplishments as a leader, then highlight the skills and traits you have, and finally, list

[3] https://blog.simplyhired.com/jobsearch/resumes/3-main-types-resumes/

experience if you choose. Again, as you see more and more resumes on the job and in the wild, you'll get a feel for the styles that exist.

Technically the one I offered is a combination but has served well over the years for myself and others. Feel free to experiment over time with yours. The name of the game is to build a resume that gets you an interview.

But I Don't Have Experience

This statement gets its own section. You do have experience, but you aren't aware of it yet. I want to be very clear about what experience is and is not.

Experience is not just writing code.

Experience is what you bring from your education and work history that is valuable to a company.

Every job you've had so far, or projects that you've been on at school, or internships, or open-source projects you've worked on are fair game. You'll want to emphasize the experience that resulted in a paycheck, but it is all fair game.

So then what do you do with all of this non-development experience? Well, this is where things get interesting. Here are several non-development skills and traits that are extremely important and seldom emphasized.

- Organizing groups of people
- Clear and concise communication
- Teamwork
- Problem-solving
- Helping get decisions made
- Supporting organizational goals
- Growth and learning

- Working with clients and customers
- Discovering new products
- Trying new things
- Teaching
- Mentoring
- Being great to work with
- Becoming competent quickly
- Working through stressful situations
- Handling a crisis
- Making a difficult decision
- Finding new tools or techniques
- Providing measurements
- Meeting commitments
- Accepting responsibility

Many managers and companies are hungry for these traits, and you can easily highlight them from any previous job now that you know they are desirable. So yes, that summer job where you flipped burgers can be part of your resume when you show that you were able to handle the stress of the job and mentor new employees.

In my resume, I highlighted a few specific experiences I thought would be intriguing, or invite questions, and I was right. One was teaching English abroad, and another was ocean lifeguarding. The first would prompt questions about what it was like to live in Japan. I'd give a five-minute talk about what it was like there and weave in stories about how I developed new training material and rose to become the supervisor in the company. For the second they'd want to know if the job was like Baywatch, and I'd tell some stories to emphasize my willingness to train and how I could handle literal life-and-death situations.

The point is you can take your previous non-development experience and use it to show that you bring a lot more to the table than just technical ability. In fact, in several types of companies like agencies and consultancies, being able to work with customers and clients is incredibly valuable.

Adjustments and Experiments

Ok, at this point you have the basics of writing a resume. Start with your skills and knowledge. Follow that with any major accomplishments or awards. Move on to your experience and wrap up with education and other miscellaneous items.

The task at hand, if we remember back to the basic plan, is that now you have to write your first draft. Build a resume with these elements and pay attention to writing your experience so that it is outcome-oriented. This may take several passes.

When you read your resume, read each line in your experience and see if you can find ways to make it more impactful or powerful. Use numbers where you can as it hooks the mind of the reader.

When this is done, you have a prototype resume.

Going forward in the process you'll take this resume and then for each and every job you apply to you'll evaluate your resume again. Re-read it and see if there are some adjustments you can make to turn that dial up even higher. Sometimes, seeing it again after a little time gives new ideas.

Then make a few adjustments you think will help you stand out for the specific job at hand. Some simple ideas are to adjust the words in your skills section. Move the ones they emphasize to the front. If, as you research the company, you see how you could tweak your experience to line up with something that you've learned, make that adjustment as well.

Don't spend too long doing this, but you'll likely find that you'll make a lot of small adjustments early on which will slowly taper off as you get more and more interviews. That's because you've honed your resume to the point that it is almost a given your phone will ring after you send it in.

This process will start over if you apply to different industries, positions, or tech-stack. This takes less time than you'd think,

so don't hesitate to try. You'll notice, for instance, that financial institutions are interested in different skills than a start-up would be. Your resume will change based on that. If you go from back-end to front-end development things will change again. As you improve your resume writing it'll be only a little extra work to start getting interviews with a new resume.

In terms of organization, I keep my prototype resume separate from the ones I use when I apply. That's because I write my prototype resume with more bullets and more skills than I apply with. This serves as a menu I can use to tailor my resume specifically to the job I'm applying for.

Tricks and Gotchas

I realized as I was concluding this chapter that there are a few items I want to mention specifically. These are miscellaneous practices I've learned that won't change much in the long run, but they're part of my process, so I'll put them in here.

First, only submit your resume as PDF. You'll sometimes get asked to send it in a Word format. Recruiters are notorious for asking this. PDF will always display the same no matter where someone looks at it. That means all the formatting you've done will be exactly the way it was when you wrote it. If you submit using something like Word, then it may change drastically based on the software versions they have. Don't take that chance.

You may also remember the story I told of someone who looks at all the invisible characters in a resume to see how consistently and correctly the applicant formatted their resume. They can do it in Word. They can't do that in a PDF. While I doubt I'll encounter anyone else that does this, I was comforted knowing that my PDF habit would have prevented an early disqualification.

Next—and this is a key reason to submit your resume in a non-editable format—recruiters are notorious for changing your resume. At their most benign, they'll simply put it into their format and stationery. More commonly they'll reformat and then truncate it to fit within two pages. Less commonly they'll alter the contents to make sure you have all the keywords sprinkled throughout the resume. I cannot stand this behavior. Also, they'll rarely tell you they do this. You'll find out after you see your resume on the job. Believe me, it can be a shock to see your name on a resume you don't recognize.

I was at one company and saw my resume in a file of resumes as I began to interview people. I noticed it had my name on it, but it had been reformatted entirely. I asked to look at it, and my jaw dropped. Years of experience were missing, pieces of the experiences that I did have were missing. This was not my resume. This was some hack job with my name on top. I gave my manager a copy of my resume and asked if they had any idea these recruiters were doing that. The thought had never crossed their mind.

Let's talk about references. The rule here is that references are available upon request. They are used less often than you'd think and they are a pain in the butt. Also, you know when they are likely going to pick up the phone and call someone. You can give your references a heads-up that they may get a call soon. This is a courtesy. Nobody likes getting a call out of the blue from some random number only to be put on the spot to give their opinion of you to a total stranger.

Last up is some formatting tips. I mentioned above that I aim for readability. I like a very even and clean looking resume. Your three primary tools for doing that are an established template, tables, or tab-stops.

Starting with a template is the easiest way to begin if you find one you like that isn't obnoxious. See if there is one that speaks to you and looks clean. Then fill it out and make it your own.

Tables are a way for you to create columns on a page. So, for example, you may want to list your experience where your company is on the left and the years worked on the right. Inserting a table into the page and adjusting the columns can give you that effect nicely. The downside is that working with tables can be a major pain.

Last up are tab-stops. These are like a poor version of tables. Essentially you can use tab-stops to control how much indentation happens on a line. This combined with right, left, and center justification can give you a similar effect to tables. The main thing here is that you can get some odd effects when you adjust a tab stop and use a bulleted list.

Maybe start with a template.

So for now, write that prototype resume or, if you're re-reading this chapter, dust yours off and clean it up!