Standing Out In Tech

Learn how to create profiles that stand out, get more attention and help you to land more interviews as a software developer

Table of Contents

Table of Contents

Introduction
- About this book
- Why write this book?
- Who am I?
- What are we going to cover?

The Problem
- Why you need to understand Recruitment

Recruitment
- What do Recruiters do all day?
  - Putting out a job advertisement
  - What if there are no applicants?
- What do Recruiters see on my LinkedIn profile?
- How Recruiters filter the applicants

LinkedIn
- User Experience (UX)
- Banner Image
- Headline
- About
- Featured
- Experience
- Activity
  - You don't have time?
  - Hashtags
  - Network like a pro!
- Certifications and Endorsements
- Profile Picture
- Contact Information

GitHub
- Overview
- Bio
- Commit Graph
- Popular and Pinned Repositories

        - ReadMe
    - Profile Picture
    - Repositories

- Side Projects
    - How often should I commit code?

- Portfolio Website

- Theory
    - Tech Focus
    - Team Ready
    - Related Technologies
    - Testing

- Miscellaneous
    - Career Change
    - Changing Stack
    - Self Taught

- Summary
    - The best developers rarely get the job
    - Evidence wins
    - Why are you different?

Introduction

About this book

Let me start by saying a huge thank you to you for supporting me and for choosing this eBook! It means a lot and I know that by spending your hard earned dollars, you are putting a lot of faith in me. I'm confident you won't regret it and my intention is to leave you seeing a bigger picture.

Accompanying this book, there is my website (https://crushing.digital), where you will find more resources and also links to my social media accounts where I put out content on a regular basis.

I have interviewed thousands of software developers and witnessed almost every error a developer can make when presenting themselves on a résumé, LinkedIn profile and during an interview. In this book I hope to teach you those mistakes and how to avoid them, greatly increasing your chances of success.

Breaking into the industry is particularly hard. It is a common gripe that job adverts appear to request more experience than is sometimes humanly possible. Increasing numbers of developers are graduating bootcamps, only to find themselves being rejected quickly from almost every job to which they apply.

The techniques in this book are tried and tested with real developers and I have had great success with developers of all levels, often going from a position of zero attention from recruiters and employers to a steady flow of interest.

It won't be easy and for the techniques to have real impact, there are no shortcuts. If you are up for the challenge, the rewards can be great!

It’s hard for junior developers to prove they are ready!

The problems are not, however, limited to developers with less experience. Junior developers are struggling to prove the value of their bootcamp certificate or that they are ready for a real world project. Self taught developers are trying to prove that their skills are at the right level and they have the discipline whilst more senior developers are having to prove that their skills are great and current, they have great communication skills and can handle the pressure.

I have experienced all of these issues and realised there is a solution. It is not, as most developers would hope, for the recruiter to go and “learn more about tech”. The problem rests with us, the developers. More senior developers often state that they already get a good level of interest in their profile, but to this I always respond: “Imagine if you fixed your profile and got the attention you deserve?”. On the flipside, would the industry benefit from recruiters who have a technical background? Sure, though I’m not going to hold my breath for that to happen any time soon. It would be a costly exercise to put developers into recruitment roles!

This book, therefore, is my solution to these problems. I believe I can show you how to stand out and, whether you are trying to break into the tech industry or hoping to move up the ladder, show you how to stand out from the crowd and start landing more interviews.

Why write this book?

The first reason is that I have found a system that works and I want to share it! Initially I helped friends and colleagues to improve their career prospects and helped them gain more attention to their profiles and résumes as well as helping them to land more interviews. That kick started a journey and the result is the book you are reading.

Secondly, most of the questions I am asked by developers fall under the same subjects. Regardless of their level of seniority or field of expertise on software development, the problems are the same. In addition to this, more and more people are flocking to the industry and whilst they have a basic grounding in software development in order to start their career, many of these developers have not been taught how to position themselves and stand out in the market. Those that have received some rudimentary training, seem to have been taught the old fashioned way. Sometimes the old ways are the best and, other times, you have to break the mould. This is one of those times where a modern approach is required.

Who am I?

My name is David Roberts and I have over 20 years of software development experience. I began as a junior software developer in 2000 after graduating from university. I have worked for small companies, large companies, good companies and bad ones. I’ve written tools that I’m proud of and that have been used by thousands, even millions. I’ve written things that I hope are now lost in the sands of time or I’ll certainly keep in a private repository!

I have been relatively successful in landing interviews and job offers. This is not boasting, quite the opposite, in fact. Perhaps it was the golden age of tech and companies scrambled for any developer they could get their hands on? Maybe I was better at the interview? I don’t know 🤷 but I know that this caused me to develop bad habits that I would not recognise until much later in my career. It is these lessons that I hope to teach you here. Learn from my mistakes!

I built my way up from being a junior developer, through to senior roles and on to leading technical teams. I built various teams for my employers and ran development teams of varying sizes. Hiring was an essential part of my leadership role. It is for this reason that I felt I already knew how recruitment functioned and operated. I was wrong!

I began helping developer friends of mine, as I previously mentioned, who were struggling to get noticed in the industry. They were either applying and not getting interviews, or failing at the next step. To their delight and mine, after making a few tweaks here and there to their strategies, they went on to land great roles and secure the salaries they desired and more. I then began to open up this service to other developers. I created a small website (https://crushing.digital) and began to market my services. It was great fun and only a few months later, I was approached to work with an international agency providing support to developers during their application process. This role morphed into finding, interviewing and hiring the best developers across the globe. I built teams for their partners and grew the business. Little did I know, I had become a recruiter and this was never the plan!

In the coming months and years, I built various recruitment teams for different agencies. I developed the systems, processes and strategies to handle technical recruitment at scale across the globe. I interviewed thousands of software developers, engineers, projects managers, DevOps engineers, designers, quality assurance specialists, data analysts, team leads and more. I had hired hundreds, but I had also rejected thousands!

My eyes were opened. I got to see the other side of the fence. As I rejected candidates, I realised where I had been going wrong all this time and I saw developers lining up to make the same mistakes over and over again. I always felt that I could help those developers who I could see were technically great, but failed to show it during the selection process. This is my mission now!

I wasn’t supposed to be a recruiter, that was never the plan. Judging people makes me feel somewhat uneasy. However, the pace is so fast that you rarely look up to see what is really happening. I always maintained, in contradiction to most recruiters, that I never wanted to work for the clients. I wanted to work for the developers. I am a developer! How developers were treated, communicated to and paid were subjects close to my heart. The upside is I feel I have learned all about recruitment so you don’t have to! I’ll show you why you are receiving so many rejection emails and how to make sure you receive less of them in the future.

> I’ll show you why you are receiving so many rejection emails

Today, I have returned to my original path with Crushing Digital. I want to help developers. I feel the best way I can do that is to show you how the game really works, so you can navigate with greater success in the future. I want to fix all those problems I saw on a daily basis and help those developers who might otherwise continue to receive rejection emails.

What are we going to cover?

To understand any business, you need to understand your customer. For developers who are in the job market, that means understanding recruiters. They are your customers. Sometimes it is external recruiters and agencies. Sometimes, especially in larger organisations, it is internal recruiters, but they are almost always there. We need to understand recruitment, how it works, what their motivations and pain points are. Solving the pain points of your customer is a solid strategy, no matter what your industry!

Once we understand recruiters, we will discuss how you get their attention and cover the various ways that a recruiter might find you, whether that be a job application, LinkedIn, GitHub, your personal website or they simply rediscovered your profile from within their ATS.

We will cover how to write a great profile, how to showcase your value and how to encourage the recruiter to interview you or move you on to the next step in the process. We will look at LinkedIn and, by extension, your résumé, as well as how to leverage your supporting information such as a portfolio website, your GitHub profile, side projects, blogs and other social media. You probably don’t want to put content on social media? Neither did I and by the end of this, I hope to change your mind!

After reviewing your profiles, we will delve into the theory. We will explore the mentality behind the hires at each level. Knowledge of these theories and truths will allow you to adapt your strategy and how you present yourself both on paper and in an interview to the type and level of role you are seeking.

Finally, we will discuss a few of the more common edge cases and questions that arise. We will discuss common problems that developers face at each stage of their career and how you can mitigate those problems via your personal branding and presentation.

At the end of this, I’m sure you will have questions. I want you to know that this is not a one way street. Please reach out to me on any of the social links I have detailed above and I will always respond to you and help where I can. Where possible, I would urge you to ask the question publicly so that others may benefit from it. Likewise, I will try to answer questions publicly whenever possible.

The Problem

Most developers fall into the same traps when they begin their job search. Senior developers would like to believe that their problems are vastly different from those new to the industry or graduating from a bootcamp. In truth, the problems are the same. Of course, how and where they present themselves will differ.

Anyone graduating from a bootcamp looks the same as all their peers. You have all completed the same modules on largely the same course to achieve the same certification for a course that is commonly mistrusted by the industry. This mistrust is a little bit of jealousy. You are graduating after weeks where those waiting to interview committed to years in a university and yet, some might argue, arrive at the same point.

It is true that all these developers look the same. University graduates look the same but it took 12 months for the next batch to arrive. Employers snapped up what was left in fear of being left without a team. Today, developers are graduating from bootcamps every month. They are coming thick and fast!

If I were to ask you how you stand out from the crowd, you might struggle to answer. Well, let’s narrow that down. How do you stand out from your classmates? If there is no real difference between graduates, you realise that your chances of securing the position you applied for are directly related to the number of applicants and I’m here to tell you, most job postings get hundreds of applications, if not thousands, depending on where you apply. For senior developers, standing out is equally important. There is an extra layer of differentiation too. If the employer is going to pay the increasing wages of a senior developer, they are going to want a developer who is almost tailor made for them. Experience in tech is no longer the only requirement. Experience with the right stack and the surrounding toolset, solving the right problems in the right industry are common requirements.

More and more of the job postings are from the exciting startups which have recently announced massive funding rounds in order to scale. This, in turn, attracts the developers who like technical challenges and so standing out becomes even harder. If the company has lots of money, they are more inclined to pay for the right developer. Simply stating you are a ‘Programmer’ is insufficient. They are waving the dollars in the air, but you have to be the right fit and a generic profile simply won’t cut it any more.

Why you need to understand Recruitment

Marketing is not something that developers commonly think about, however, an understanding or an appreciation, can really be beneficial.

When you consider the marketing ads you usually see, the format depends on the context we are seeing them in. Some ads are designed to get you to stop scrolling or even stop strolling past the shop window. Some are timed to perfection when you’re at your most hungry or susceptible to the lure. Paying attention to the context is key!

Now, recruiters are the customer of the developer. I know, I know, the real customer is the client, as they are the ones who pay, but don't be distracted by the money at this point. You are selling a product and that product is YOU! The result is a transaction, of sorts. The recruiter puts you forward for the role. We need to maximise the chances of this happening.

The recruiter is scrolling. They are not scrolling through social media in a haphazard manner. They don't have time for that! They are scrolling through profiles and we need them to stop on yours. We need them to realise they have found 'The One'. Morpheus needs look no further!

When a recruiter does read your profile, we need them to put you forward, without hesitation and, as I will cover, this requires them to feel confident in your application. To achieve this, there are certain things we can do. This won't be a lengthy appreciation of the recruiting industry. We are going to try to understand them, so we can target them!

Recruitment

What do Recruiters do all day?

You probably don't like recruiters? Don't worry, neither did I until I became one and even then I wasn't sure how I felt about myself?! Where did this antagonistic relationship come from? Whose fault is it?

"Recruiters need to learn more about tech!" is a phrase that I hear frequently from developers. I may have uttered those words myself on occasion. The truth is that recruiters are purposefully against you. Honestly, they don't have time. Time is a luxury that they don't have and I believe this is where most of the problems arise.

A job comes in and the specification is vague, to say the least. It can vary from "find me a developer" to a wishlist of some of the most obscure tech you've seen cobbled together. In case you are wondering, the employer is usually reluctant to flex on those requirements, hard as the recruiter may try. Cue frantic Googling to find out what these technologies do and hope for some guidance on common or alternative things to look for (note: FYI, no guidance is coming!). Here comes the kicker:

"We need someone in 3 days"

You read that right. Three days to put a job posting post out to the public, market it, contact the existing network of developers, manage responses and gauge interest, filter for suitability, interview and put a handful of candidates forward. In their downtime, recruiters respond to the thousands that applied for the last job but were unsuccessful. The company might not care whether this happens, the client certainly doesn't, but this is the recruiters network and, effectively, their livelihood! For clarification, there is no downtime and, I hope, you are starting to see the scale of the problem!

"the best developers rarely get the job"

The time constraint is a real problem. It leads me to catchphrases that I use often such as "the best developers rarely get the job", but I'll clarify this later. Time is our most precious commodity, or so we're told. It is the one thing we cannot get back. It's the one thing that developers demand more of, not just in their daily lives of coding, but when they are begging the recruiter that even though their profile might not deliver the value they seek, should they offer the interview, the recruiter will realise the true value of that developer.

“Please give me a chance!?”

On such a short timeline, the recruiter has no time to read around the subject. No time to waste on the wrong candidates. An interview commits at least 60 mins of the time, excluding preparation and organisation. Time is precious.

No time to waste on the wrong candidates

Where does this time constraint come from? Well, sometimes the client will have their own agenda, but most commonly it is the recruiters themselves that apply this limit. It is there because they know this is a speed game and their competitors (other agencies trying to fill the same role) are vying for the same payday. The early bird catches the worm in this instance! Agencies know that if they can get their developer, ideally a few developers, in front of the client before anyone else, there’s a good chance they will begin to interview and the chances of succeeding in filling that role and not losing out to another agency improve greatly.

Back to what the recruiter is doing all day! Well, they are reading through developer profiles. Actually, allow me to correct that? They are ‘scanning’ through developer LinkedIn profiles. Of the hundreds, if not, thousands that apply to each job, they scan through to find the best matches in the limited timeframe they have.

It is common to use LinkedIn over résumés as the formatting is always the same. They spend about 15 seconds of time evaluating whether you are worth more of their precious time and energy. Recruiters don’t want to commit to an interview without the developer showing great promise that they might be a strong match for the role. If they do not see a strong match, they bounce to the next applicant. They have a LOT to get through, remember! Offering an interview to a candidate that later indicates they are not available, unsuitable or not interested in the role is a huge drain on an already tight schedule.

Putting out a job advertisement

Let’s take a step back and consider an imaginary role vacancy that might fall to a recruiter to help fill. As is common, there are many applicants and the recruiters go about assessing the LinkedIn profiles of the applicants.

Let’s give the imaginary role a title and perhaps some requirements:

Front End Developer

Hard Requirements: React, Redux, Context API
Nice to have: TypeScript, React Testing Library, GraphQL, Next.js, UX

So, what does the recruiter look for? A front end developer? That’s a bit vague, right? Let’s narrow it down. We are looking for a React developer. We know that if a React developer has worked at any scale they are going to have used tools for state management, testing and more.. Next it comes down to who can tick the most boxes in the 'nice-to-have' section. We want to impress the client and, most importantly, ensure that if a competing agency puts forward a candidate, our candidate wins. Evidencing usage of some of those skills in the 'nice to have' section could be the icing on the cake to seal the deal.

HINT: Look at jobs advertised and begin to tally the skills in the required and nice to have sections. FInd out what are the most common skill combinations. This tells you what to learn next.

What if there are no applicants?

If there are limited or even no applicants for a role, the strategy remains the same. The difference is that the recruiters are then scouring LinkedIn and using the products provided by the platform to find suitable candidates. This means even greater numbers of competitors for you, the developer.

Whether the recruiter finds your LinkedIn profile via a search or you apply for the role, once your profile is in front of them, success is going to come down to you and the value of the evidence that you are presenting. So, let's open your profile and ask ourselves: "What do recruiters see?"

What do Recruiters see on my LinkedIn profile?

Sadly, for the most part, there's not a lot to see on most profiles. It's one of the driving forces behind writing this book! Most LinkedIn profiles are quite sparse. Some developers seem unsure of how to complete a profile effectively, others write things that they value personally and forget about their intended audience. Either way, uncovering the real value of a developer can be an arduous task.

How a LinkedIn profile (and a résumé, by extension) are filled out should differ depending on your level of experience and the role for which you are applying. Most developers, including those wanting to break into the market, focus on their experiences. Senior developers are fortunate enough to be able to throw out a few dates, job titles and company names and leave it there. Nothing more is required, right? Junior developers follow suit, only they don't have the titles and company names to offer. It becomes either a lengthy plea for someone to take pity on them and offer them a role or a mission statement that sadly falls flat.

> "I want to change the world, via code!"
>
> "If you offer me a job I will promise to deliver fabulous work!"

Most developer profiles are either vague or overly wordy. To find out more or clarify what they have read, the recruiter would need to speak to you directly or engage in an email exchange. That may take days to organise and complete. Any time a request for more information is made, there's a danger of a delay. I call them 'gaps' and they are the enemy of the recruitment team. Contacting you and having a chat would be wonderful, if the clock was not against the recruiter. Your job is to make your profile as concise and direct as possible. Be the answer to someone's question!

I've thought about the subject of what constitutes a good developer profile. So much so that I once conducted a piece of research. I wanted to find more developers like the ones I had already found. Take the best developer and mirror that success. Was there something on their profile that indicated that this candidate was going to be great? It seemed like a no-brainer, so I collected the names of the best developers I had met over recent years and set about reading their profiles. What qualities connected them all? If we can isolate that, at least on paper, then we can instruct the recruitment to find similar candidates and maintain a very high bar of quality. Ten names, all great developers, bucket loads of experience and successful careers. What more could I ask for? Well, the results were interesting. The only thing that I could find to link them all was that they all had awful profiles. Every single one of them, without exception. I asked around and dug deeper into how we found each candidate. In the end, it became clear that these developers had been found in spite of their profiles, rather than because of them. These developers were very successful and my guess is they didn't even need a good profile?

You might think this to be the case, especially if you have a few years of experience as a software developer, but I would suggest we ponder where these developers might be if they did have a better profile? They had been successful in securing good opportunities but what if they had landed the opportunities that, one could argue, they deserved?

The good news is that creating a profile that stands out is not overly difficult or taxing. You have the luxury of knowing that most developers won't do it, so a few simple changes already puts you ahead of the competition. As with most things, you can take this as far as you want to and the requirement to do so will depend on the type of job and employer you are looking to attract.

How Recruiters filter the applicants

With so many applicants to get through, you probably already know they don't process them all, at least not in any depth. A quick glance is all they need to assess if your profile is a good or great fit for what they are looking for. Can they see enough evidence that you match the requirements for the role? This is where my slogan "The best developers rarely get the job" comes from. Recruiters will interview the candidates whose profiles show the most promise. It is their best bet against wasting their time. Many great developers are overlooked at this stage.

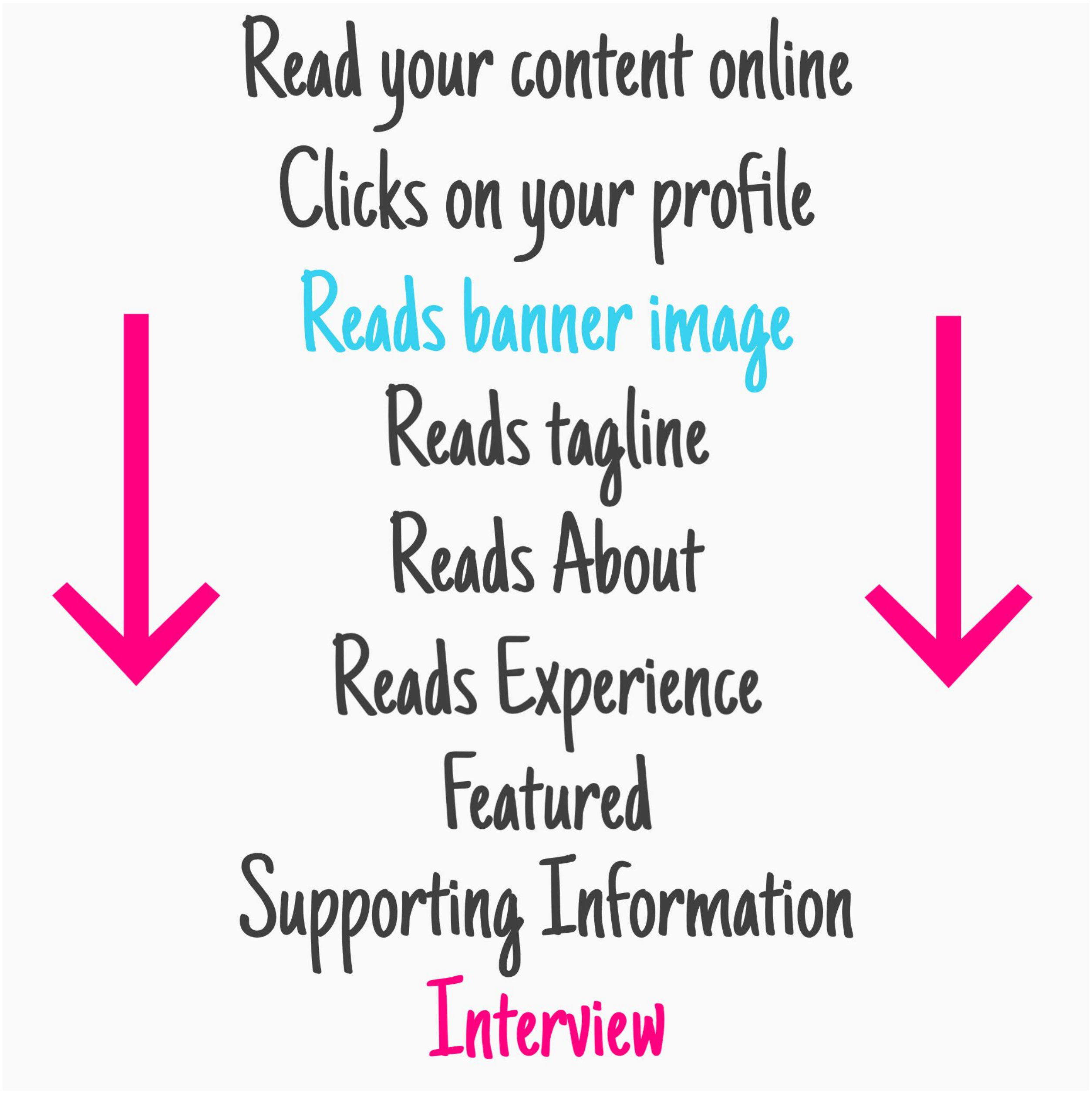
Each profile receives 10-15 seconds of attention and the recruiter decides if the candidate is a potential match. If so, you go into one of two lists. The strong list or the backup list. If they see no potential at this point, they leave you where you are. No need to be hasty at this point and send a rejection email. This is where that long period of radio silence begins when you are an applicant. They are processing and interviewing the candidates with a stronger presentation than yours. They will come back to you if all else fails. It's not a good place to be, you'll agree!

The ideal situation for a recruiter is to get to 3-5 strong applicants with, ideally a handful of backups in the alternate list. Once they have 3-5 strong candidates, the recruiters stop processing applicants. They push these developers through the process and await feedback. If one of the developers succeeds, then the job is finished. However, negative feedback about the applicants might mean they need to adjust their filter, reassess and present the next batch of applicants. Therefore, they are only coming back for you if something goes wrong. Being in that first batch is your priority.

LinkedIn

User Experience (UX)

The truth is we don’t know how each recruiter or employer is going to find us. It might be that we have applied for a position and sent them a link. Commonly recruiters find developers by searching on the platform. If you’re an active blogger, they might find you via your content. Perhaps they came from another of your profiles, such as GitHub. The point is, they are here and we need them to stay, at least long enough to realise you are the developer they have been looking for and send you an invite for an interview?! For this to happen, we must consider the funnel:



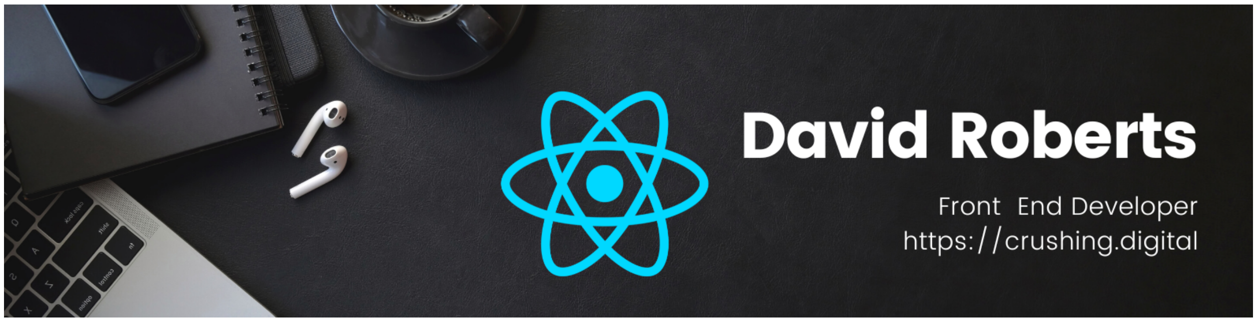
The first part is optional because, as discussed, someone could find our profile in a number of ways. I would encourage you to leave as many opportunities for people to find you as possible and that means creating content. More on that later! Regardless, they click and arrive at your profile but do they see your true value? When I ask developers about their value I'm mostly met with a wall of silence. It's something that we don't often think about. We just assume everyone can see our value. If I press further and state that it might be a repository or a project they are proud of, they will direct me to a link. Sometimes it's to their GitHub (or similar) or portfolio website. You can see the relief on their faces immediately. They had an answer to the question. When I then ask how I might get there from their LinkedIn profile there is a two second delay before the pin drops. Suddenly, everything becomes clear. Even if recruiters or employers were directed to their LinkedIn profile and liked what they saw, will they ever successfully make it through the funnel to see the real value you possess?

What if the real value is presented clearly on my LinkedIn profile? Great, but what if I find you via a tweet you wrote, a blog post that received some attention or via a commit on an open source repository? Wherever that value resides and no matter which of your profiles I land on, you must ensure you take me to the evidence that tells me you are great. Then, make it easy for me to contact you. We talk about this in web design all the time. How many clicks does it take to get the user from the landing page through to completion of the purchase. We now know that customers rarely arrive at the home page. They arrive via a search or an advertisement placed offsite. Regardless, once we draw them in, the execution needs to be swift. We need to make the process of offering you an interview the easiest thing they have ever done!

Banner Image

Something I had overlooked, at first, is the impact a good banner image can have. I learned this from Justin Welsh who was talking about entrepreneurship and how to gain more followers and, therefore, customers.

The banner image might not seal the deal, but it is a very prominent visual cue and opportunity to tell the reader that they are in the right place. You have found a suitable candidate. My advice here is to confirm the tech you are pursuing and, perhaps, where they can find out more about you and your skills? If the recruiter is looking for a React developer and they see a React logo and a link to your personal site with some text affirming you are a Front End developer specialising in React, we are off to a good start!



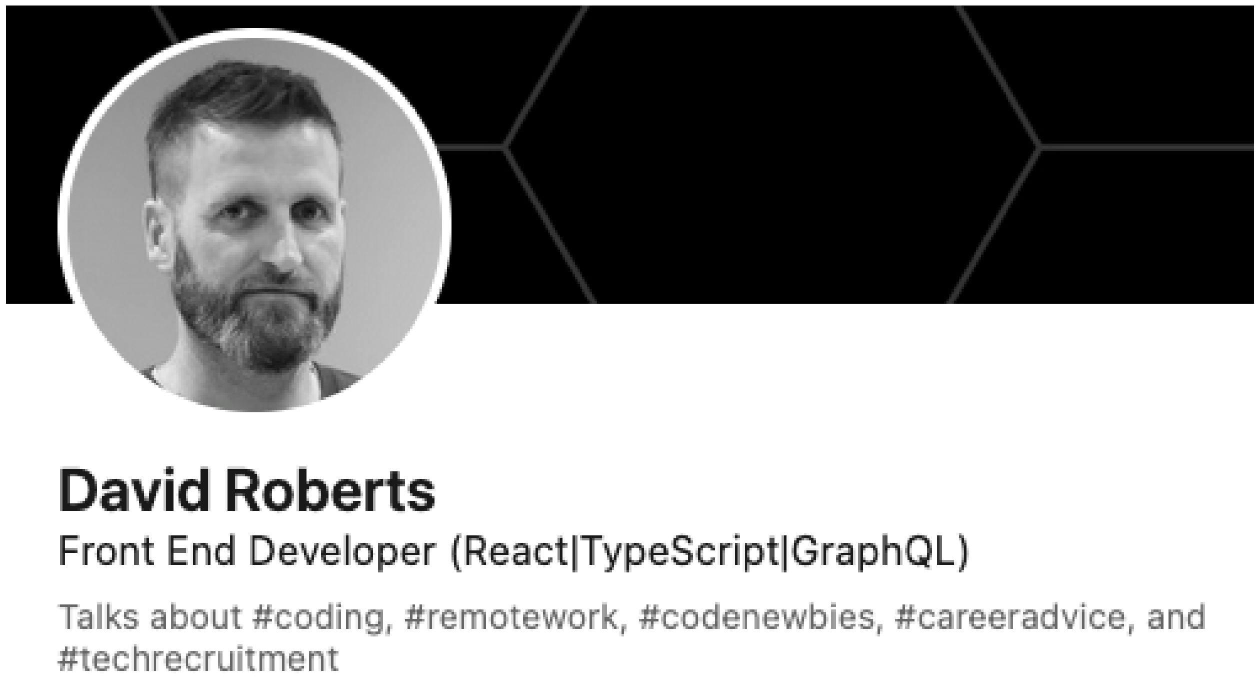
Headline

The headline is one of the most misused fields on LinkedIn and for various reasons. In 90% of reviews my feedback is that their headline is vague. Entries like “Programmer”, “Software Developer” are common. Some go a little further and might use something like “Front End Developer”, but if we consider the scenario of the recruiter or employer arriving at your profile in the search for a React developer and also the mantra that we want to ensure that they keep reading, does this headline confirm to them that they are in the right place? I grant you it might not say to them that they are in the wrong place but we have missed an opportunity to confirm.

“Recruiters are lazy!!”

Another common catchphrase of mine is “Recruiters are lazy”. Let me assure you, they are not. It is a demanding and often stressful role where time is always against you. I say this phrase to get you to think differently about how you present yourself. If you had 30 seconds to pitch your business idea to investors, what would you say? It would differ greatly from what you would say if you had more time, right? You don’t have time when someone views your profile. Recruiters are in a hurry, so I use the phrase to push you. Think of them as lazy and that they will not read or dig to find more value, especially if they have hundreds of other applicants. So, learn to make it easy for them. Spell it out in no uncertain terms. Leave no doubt!

Consider the headline “Front End Developer (React/Redux/GraphQL)”. It is a subtle change, but the confirmation is there. Instead of wondering if you are a React developer, they’re already moving on to the next stage of confirming your skills and experience. Job done!



David Roberts

Front End Developer (React|TypeScript|GraphQL)

Talks about #coding, #remotework, #codenewbies, #careeradvice, and #techrecruitment

About

If we are already off to a good start, we are building confidence in the reader that you might be worth more of their precious time. Possibly even warranting an interview?! They are at least intrigued to find out a little bit more and I emphasise the word ‘little’. The About section is probably the most misunderstood and the things people write there are either never going to get read or not helping their cause anyway.

When I said that they are intrigued to find out more, I was careful not to say ‘read’ more. Recruiters rarely read large volumes of text unless absolutely necessary as it has great time implications. Recruiters will scan but, again, they are only looking for confirmation of what they already suspect: You match their search criteria. You are the developer they have been searching for. This is not the time to write your memoirs!

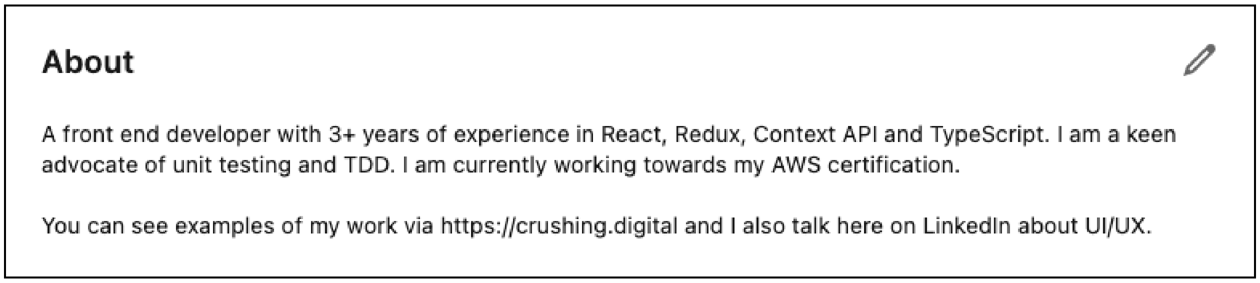
Some of the more common entries I see in the About section on LinkedIn are long paragraphs of text with mentions of tech peppered throughout. It’s a start but you are making it more difficult for the reader to confirm that you are a suitable candidate. You are making it more difficult for the reader to scan the document for the relevant information.

The next problem is killing them with detail. Did you have an idyllic childhood with regular trips to the beach? Great, good for you! How does that help progress this story? Perhaps you like cats or coffee and cannot even consider starting a day's work without a cappuccino and scrolling through cat pics on Instagram. How is this helping? If it does not add value, it detracts from the value. Be concise, tell the reader what they want to know and funnel them onto the next step in the decision making process.

Something I have begun to think is that the About section is the successor to the Cover Letter. I’ve always hated cover letters and rarely, if ever, written one. It was drilled into me that you write a cover letter for each role for which you apply, explaining why you are eager to join the company. This was far too much like ‘sucking up’ for my liking. However, I’ve had a change of heart more recently. Purely because I believe the content of a cover letter was misrepresented to me. I believe it should not be about why you want to join the company and therefore has to be written for each role for which you apply. It’s supposed to be why you are a great prospect for them to hire! Why you are of great value and that does not change. Welcome to the About section.

The About section should look to the future!

Now, don’t get carried away, again, we are not writing a novel. Do not over-sell. Think, quietly confident. We need to demonstrate that we are of value and do so in as few words as possible. Here, I say, your About section should look forward. If you want to look back, which I would argue is the purpose of the Experience section, do so after you hit them with your eloquent depiction of your future. Make it clear what you are passionate about in the direction you have chosen for your career. Tell them the job you are going to get next by explaining this passion and purpose. Please note, you cannot say you are passionate about something, it has to be implied!


About

A front end developer with 3+ years of experience in React, Redux, Context API and TypeScript. I am a keen advocate of unit testing and TDD. I am currently working towards my AWS certification.

You can see examples of my work via https://crushing.digital and I also talk here on LinkedIn about UI/UX.

Featured

The Featured section is underused by so many developers. Creating content for social media is not high on the list for many and we will revisit this subject later, but for now I want you to consider it as an opportunity. Perhaps you have written a blog post elsewhere or you have a screenshot of a new feature or project you are working on? What if I told you that most people will not open these links, at least, not at this stage and they only serve as more confirmation that this is a passionate and proactive developer that talks about the tech (for which the recruiter found you!) on a regular basis?

Your mind is probably whirring and you’re uncomfortable with the subject? For now let me say that I think I can change your mind on this. This section can be super valuable and does not require hours and hours of work. If you are a junior developer or looking to break into tech, this is going to be your favourite and most powerful section very soon. I shall explain later!

Experience

Here’s the big one. The undeniable powerhouse of any LinkedIn profile. This is where the magic happens, right? Well, not really. This is one of the most important points of this book. Whilst everyone focuses on the experiences, you are going to focus elsewhere. We do, however, have to get it right. Never miss an opportunity! Of course, it is an important section, but in the end it’s a record of your past and needs to do little more than confirm your skills and experience and funnel the reader into offering you an interview.

Most developers either write too much or too little in their experiences. Senior developers, for the most part, do as little as possible and rely heavily on the magical number that is their years of experience. Juniors tend to over egg it, preferring to write volumes of text in order to persuade the reader. Persuasion is hard and often does more harm than good. It’s kind of like dating. Put on a good show and entice, but begging will get you nowhere, trust me!

When a recruiter or employer gets this far into your profile, they are trying to answer 3 questions. Remember, they have gotten this far after only 10-15 seconds of scanning and confirming they are in the right place. Those questions they now want answered are:

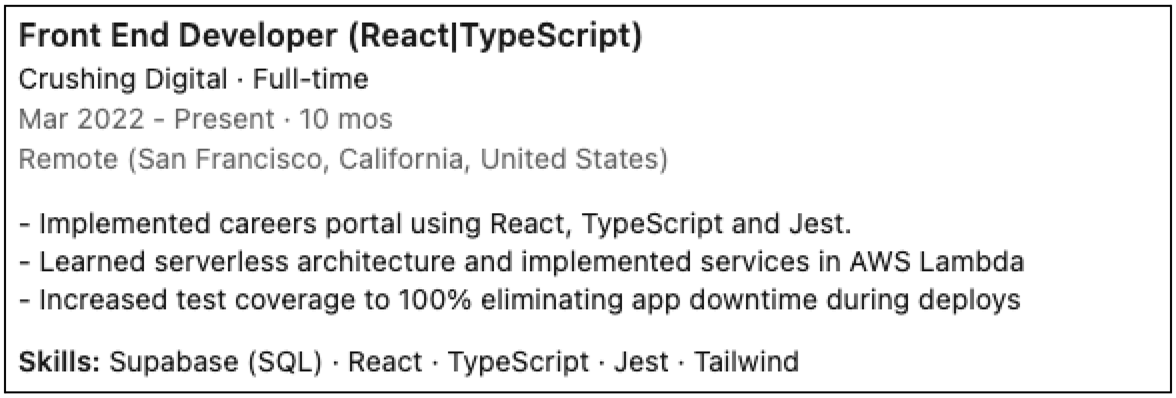
1. When did you start using the tech they are trying to hire for?
2. Are you currently using that technology?
3. How frequently have you been using the technology between points 1 and 2?

For these reasons I always encouraged developers to list the tech they have used on a role and to do so at the top. Don’t make the reader click the ‘see more’ link to simply establish the basics of what they need to know. Today, LinkedIn has addressed this and now allows you to include the skills for each experience. This is paramount!

Believe it or not, the wording on each experience may not even get a read at this point. That might be for a later filter or even during the interview. Recruiters mostly don’t understand the technical intricacies that developers write in these sections. Writing verbose paragraphs is mostly misguided. Bullet points are the preferred format and they need to be high level enough to be readable but also enticing enough on a technical level. I like to stick to 3 subjects by answering the following questions:

- What did you build?
- What did you learn?
- What was the business benefit?

This is not the time to show off. Keep it transactional. This is telling the reader in the shortest way possible that you are experienced and professional. Something that is often overlooked is the job title. You might consider this a fixed entity. The title is what was given to you when you signed the contract? Titles mean nothing and the sooner we all admit that, the better. Elon Musk once said that all titles except President, Secretary and Treasurer are meaningless. He's right! They might have different labels in other countries but the theme remains. Today, developers are no longer just developers. They are junior, mid-level or senior, tech or team lead, staff or principle engineers. All wonderful ways of establishing pay grades, allowing someone's pay to increase without having to increase the salary of the person sat next to you! You may well have been given the title of Software Engineer or similar, but as per your headline, sometimes the inclusion of the tech (certainly for juniors) can help establish a career technical focus. It's so much easier to scan and nobody really cares what you call yourself on LinkedIn in your previous roles. Please note, I'm not advocating that you lie. I'm saying that sometimes a more verbose title might help smooth things in the mind of the recruiter or employer.


Front End Developer (React|TypeScript)
Crushing Digital · Full-time
Mar 2022 - Present · 10 mos
Remote (San Francisco, California, United States)

- Implemented careers portal using React, TypeScript and Jest.
- Learned serverless architecture and implemented services in AWS Lambda
- Increased test coverage to 100% eliminating app downtime during deploys

Skills: Supabase (SQL) · React · TypeScript · Jest · Tailwind

Activity

The Activity section is the parent of the Featured section from earlier. It is said that only 1% of LinkedIn users create content, so standing out is almost inevitable if you take this path, though I know it can be a daunting experience for many. The Activity section is either baron or filled with banal comments. I'm not being negative, but most of our conversations on social media have little to no real value, at least not to a recruiter or employer.

Are you missing an opportunity? This is about standing out, is it not? If only 1% of users are creating content and you are in the 1%, you're already on your way to standing out. You already look like a more exciting and enticing prospect. When the hiring manager asks the recruiter if you are a good developer, they can only really respond with “I don’t know”. They are not developers and rarely have the technical knowledge to be able to comment comprehensively. Instead of a shrug of the shoulders in response to this question, what if the recruiter was to say “...well, they do seem to talk about [insert tech here] a lot!”. That has to be of more value, right? Sharing knowledge, documenting projects and your studies makes the recruiter feel confident in your dedication, if nothing else!

When I was leading recruitment teams I would be sent developer profiles in a constant stream. I would, effectively, give a thumbs up or down as to whether we should continue with this candidate in the process. This step was necessary as, quite often, the recruiters were not sure if the candidate was relevant. Examples include putting DevOps candidates forward for Web Development roles because they had Python as a skill. They did not realise that Python could be used in different contexts. Old cliches spring to mind such as Java developers being put forward for JavaScript roles. Recruiters are rarely technical and, as we have discovered, have little time to address that issue. Each time you write about your chosen tech stack, mention it in a job title or on your experience, it has a bolstering effect for the recruiter. The luxury of being able to say “...well, they do seem to talk about [insert tech here] a lot!” is a warm blanket or hug to them. Don’t you want to hug a recruiter?

What to write or say on social media posts is a topic that stumps most people. Fear plays a big part and often developers try to mimic what they have seen before. Developers are usually reading tutorials or articles that teach a particular tech subject to them. So, they start writing articles to teach people what they can! If you’re a junior this can generate anxiety and the onset of imposter syndrome. Luckily, this is not what I advocate at all, though I do agree that everyone can teach. You might have taken that first step in learning to code, but there are people still waiting to take that step. You just might help them by teaching something from a beginners perspective?

Let’s remember that recruiters are lazy, or that I want you to think of them that way as they rarely have the time to dig deep for information about the candidate. Recruiters and employers are rarely going to have time to read or listen to your posts. Recruiters would rarely understand them even if they did?! Developers will often run away and embark upon trying to write the perfect post. One post that people are going to marvel at and will lock you in as the next Kent C. Dodds. It's not going to happen, trust me. At least, not unless you are incredibly lucky. Articles that get read by thousands are always preceded by hundreds of articles from the same author that were largely ignored. Even if you were trying to become known for this, it would take a LOT of time and effort, otherwise we'd all do it?

"Write like nobody is reading"

The trick is to write for you. Write like nobody is reading. Talk like nobody is listening … because they're not! Write for your future self to read in years to come. In fact, don't write, but instead journal.

If nobody is going to read, then what is the point? Each post is like a piece to a jigsaw. On its own it is mostly worthless, but put them all together and it creates quite a picture. Your posts are the picture of you. They are adding to that confidence that the non-technical recruiter needs to present you to their manager or a client. Each post is a seed, so go and plant a forest and before long they won't be able to see the wood for the trees!

You can write on a blog, sites like Medium.com or where you choose. I would encourage you to consider writing or, at least, sharing on LinkedIn. This is where the recruiters hang out. They're hungry for developers like you who make life easy for them. Feed them!

With regards to the frequency of posting, let it come naturally. Don't force it. I often say to developers that each commit should trigger you to consider posting and pondering a similar set of questions to the ones you answered in the Experience section:

- What did you build?
- What did you learn?
- What are the alternatives?
- What challenges did you face?

You don’t have time?

Ahhh, my favourite excuse! The way I feel about this is the same way you feel when your development manager says there’s no time for testing on the project. You either spend the time now or spend the time later. For testing, it’s because you’re going to have to fix the bugs later. In our case, if you don’t decide to document your journey now, you’re going to spend the time later trying to convince recruiters and hiring managers! It is so much harder to convince someone to do something, ask a sales person!

When I speak of documenting your journey, most developers immediately think about crafting long posts, explaining deep technical concepts and processes. I don’t mean that at all. My suggestion is a post per commit. A screenshot or code snippet and a couple of lines of explanation. Nothing more. If a post takes you longer than a few minutes, You’re doing it wrong. More on this later!

Hashtags

The most common strategy for hashtags on your posts is to choose the ones with the largest audience. You might opt for #coding or #programming and possibly narrow down with the tech such as #java or #react. These are fine but you’re also a small fish in a VERY big pond.

My tip is to write your post. Consider the subject matter and do a search for the same topic. What do you find? It may take you a while to find someone walking about the same or similar topic. Once you find them, steal the hashtags. Next, can you comment on their post? Perhaps you have struggled on the same issue and possibly you can help each other out? Doing this every day is powerful and I encourage you to start searching for the active discussion and joining it as well as creating your own. Both serve different purposes but are powerful strategies in their own right.

Network like a pro!

Once you begin to document your journey, you’ve overcome the biggest hurdle. I jokingly say that nobody will read. They will, but not in the numbers you are hoping or expecting at first. The upside is that when you do finally get comments and reactions, you have the beginnings of both a network and a job hunting strategy.

Reply to **every** comment you receive. Well, apart from the abusive ones. Let them fall to obscurity. The best response, unless they have a valid point, is no response. In time the good people of the world will weed out those bad comments and come to your defence. For now, let’s focus on the good comments and reactions.

For each reaction or comment, after you have responded to the comments, take a look at the author. If your content is about a technical subject, then the chances are the commenter is also in the same field. Most likely a developer, possibly a recruiter. If it’s a recruiter, connect and build a bridge. If the author is a developer, then they probably use the same technology? Where do they work? Can you look up their company via LinkedIn and find their website? They hire developers just like you! Scroll to the bottom of the home page and find their careers page. If they are hiring, apply! If not, why not contact them anyway? They value your skills and everyone likes a proactive approach that comes without agency fees. This is a ninja move and really starts both your networking strategy and your proactive application process.

You can apply the above strategy not just to those that react to your posts, but also look at those writing about similar topics. Join their conversations, steal their hashtags and then have a look where they work. In time, your strategy to document your journey will feed your networking strategy, which in turns feeds your job search.

Certifications and Endorsements

Certifications are far more valuable than endorsements, unless they come from prominent places and that is unlikely for most people. The problem with endorsements is they usually consist of reciprocal endorsements by friends or acquaintances. There’s so much doubt as to their validity they are often overlooked. They do no harm, but, in my opinion, they are not worth spending time trying to garner endorsements in greater numbers when that time could be spent on more impactful strategies.

Certifications do have value. They are a nice little confidence booster to your application. Do you have experience with AWS? Being able to respond “Yes” is one thing, but saying “Yes, I’m certified” carries a lot more weight. Now, the one thing I will add on this subject is certifications appear quite far down the page on a LinkedIn profile and the reader is only going to go this far if you have enticed them to do so. That certification, if it is to have the impact it deserves, needs a far more prominent position to do so. A featured post including your certificate or spelling it out in your Headline or About section is advisable. It’s not one or the other, in terms of whether you do the featured post or have it in your certifications, it’s both!

Profile Picture

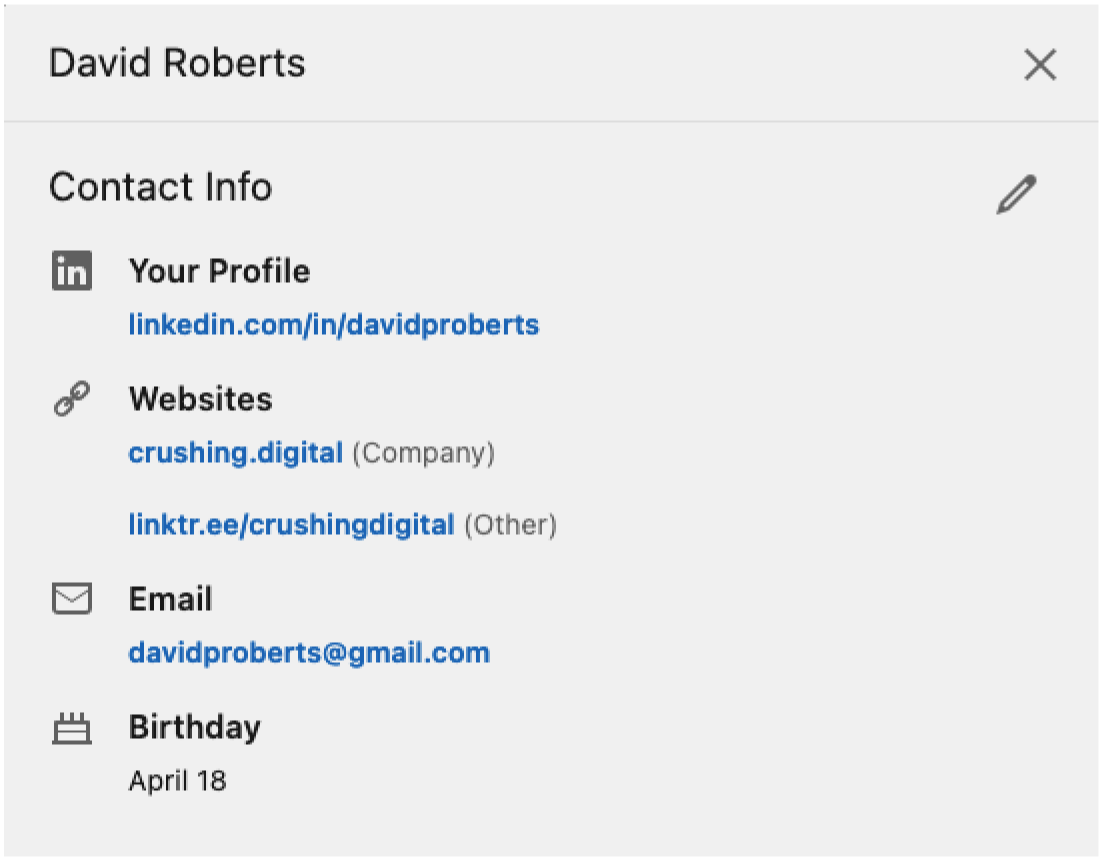
You might ask why I have left this until so late in this section about LinkedIn to mention your profile picture. It is, after all, a prominent feature on any profile. Yes, it is, but it’s also a subject that divides opinion. Should we judge people based on their profile picture? Absolutely not. Does it happen? Yes, sorry. It happens because clients demand it. They might not openly express it, but they make it clear when expectations have not been met. It’s not about being the best looking person in the world, it’s about being presentable and professional. When it comes down to considering which candidate takes the final slot for an interview, don’t let something as trivial as a profile picture get in your way. Here’s a checklist to make sure you are hitting the mark:

- Can we see your face clearly?
- Is the picture in focus?
- Is your face well lit?
- Are you smiling or at least not looking too angry or sad?
- Does this picture look like a professional person who is ready for work
- Are you fully clothed (Yes, it’s important!)

Contact Information

You would be surprised how often I ask a developer where the greatest value in the LinkedIn profile lies and the response comes back that it’s a link in their Contact Information section. Their greatest value is hidden behind a link before we even begin! Having good links in this section is a good thing, but you have to drive people there! Referring back to the section on User Experience, if you want the reader to be taken to a personal website or project, then ensuring they can get there is vital. I would argue these links should form part of your Featured posts section, but duplication is ok in this instance. I want you to look at this section in 2 ways:

1. What else can you cram in here? Make use of it and if someone happens to click on this but is currently undecided about how they will proceed, this is a golden opportunity to seal the deal. Give them reasons to feel more confident in you!
2. What can you take out of here? Yes, these two contradict, but what I’m saying is that these links will only be seen if the reader happens to click here. Don’t leave any treasure covered up. Which of these links could be turned into featured posts?


David Roberts

Contact Info

Your Profile
linkedin.com/in/davidproberts

Websites
crushing.digital (Company)

linktr.ee/crushingdigital (Other)

Email
davidproberts@gmail.com

Birthday
April 18

GitHub

Overview

Some developers are heavy users of GitHub with lots of repositories and a green wall of commits, others have very little to show and it’s common for repositories to be private or owned by an organisation and therefore hidden from public view. Let’s not forget that GitHub is only one of many hosting and version control platforms. I've tried to keep it simple here and talk about only one, but the same principle applies to GitHub’s competitors.

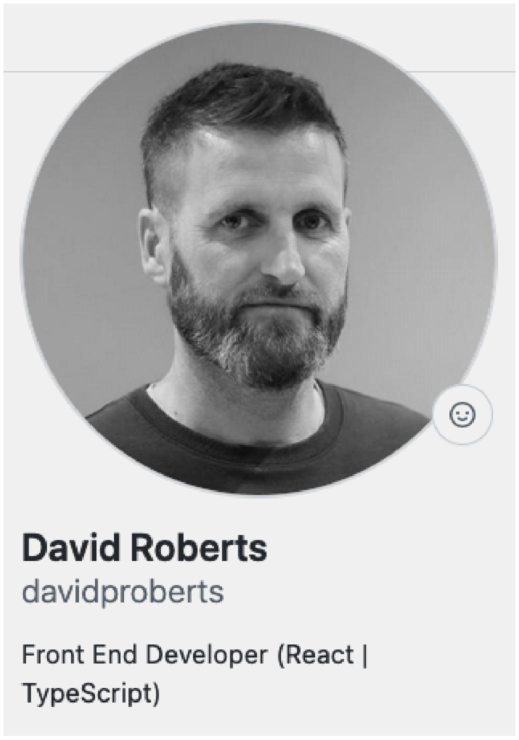
If you don’t use GitHub or don’t have anything to show, fear not. The lack of a profile is not a showstopper. As with many other things, it can help, but rarely hinders. I want to illustrate how it can help and can be leveraged to your advantage.

Where I have said that it rarely hinders your progress, I should caveat that a weak or mismanaged profile can let the developer down. These errors can mean the difference between an interview and a rejection email, all because of the introduction of doubt.

Mirror your LinkedIn Headline of GitHub

Bio

Most people don’t put much effort into their Bio. It rarely contains the same information as their LinkedIn. Perhaps it is deemed a ‘developer only’ zone and few developers care for what another developer may say about themselves. It’s their code that we are here to see, right?. It’s not LinkedIn, after all? Well, as previously discussed, you never know how or where a recruiter or potential employer may discover you and if, in this instance, it’s via GitHub, then you want that same impact that your headline had on LinkedIn. Therefore, mirror it here. If they clicked through to GitHub from LinkedIn then it’s immediate confirmation and affirmation of what they already knew. Just as per traditional marketing advice, repeating your message helps it to sink in eventually!



David Roberts
davidproberts

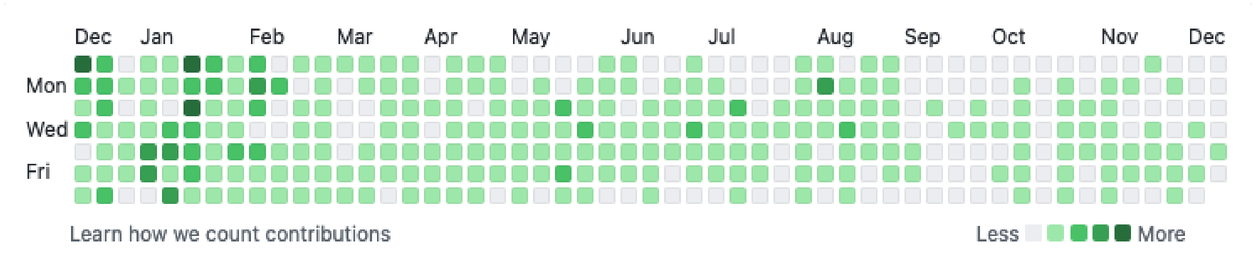
Front End Developer (React | TypeScript)

Commit Graph

We all crave and even marvel at a filled commit graph, right? Do people really commit every weekend? I'm not sure that's healthy. To be honest, I'm not sure a commit graph without a few gaps peppered through it and a few weeks completely missing here and there should be legal, but that's just me!

An empty commit graph won't hurt if you are an experienced developer. In fact, the more senior you are, the less it is expected. It is expected that you would always be working on large scale, private projects under the veil of secrecy? You're conquering the world, no?

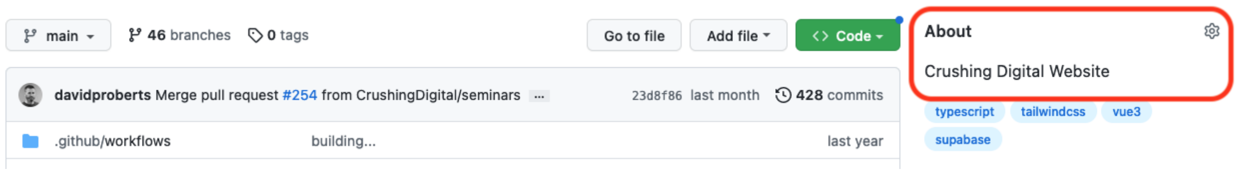
If you're a junior developer, it's still not a bad thing, necessarily to have no commits but the doubt it raises is the level of commitment you are demonstrating. The danger is recruiters give priority to a developer that appears more enthusiastic. Proactivity is powerful at this stage. You might be on a course or a bootcamp. You might be early in your career already, but you're not still learning and growing. If you have no evidence here, be ready to demonstrate where this evidence lies. At least, don't look baffled when they ask the question!



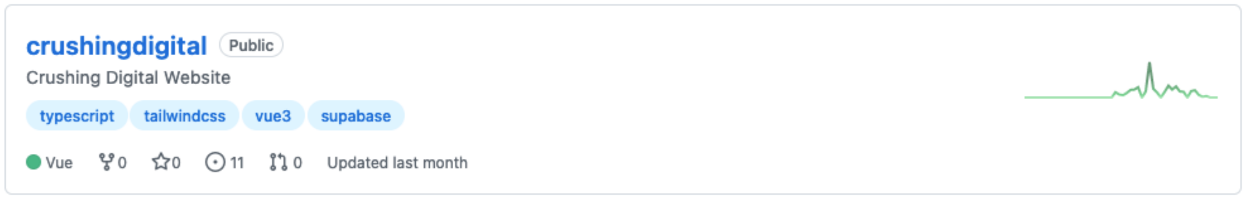
Popular and Pinned Repositories

In most cases, recruiters and employers arrive at your GitHub profile via the overview page. Before they enter the chaos that is your list of repositories, you are given the opportunity to show them your best work and guide them to your value. Sadly, most neglect this opportunity!

Firstly, pin the repositories you want them to see, in the order you want them to see them! Secondly, under each repository snippet is a description, or at least there should be. I'm not talking about adding a ReadMe to the repository, I'm talking about the 'About' section at the top right of the repository home page.



The programming language is indicated to the reader via the footer of the snippet but often this is not enough. An example would be a snippet that says 'JavaScript'. Does that confirm this is a ReactJS or a NodeJS repository? How would the recruiter know that this is yet more evidence or your usage and exploration of this technology? You're assuming they either read the code, which we all know they won't or can't, or that they are assuming that JavaScript in this case must mean React or Node? We are making massive assumptions and it rarely pays off. Also, consider the various other technologies used in that repository from testing tools to external libraries. This is another chance to build confidence. Keep the snippet brief but tell them the tech and what purpose the repository serves!



When pinning repositories, think about what you are telling the reader. If you have presented yourself on LinkedIn as a NodeJS developer and I arrive at your GitHub and the pinned or popular repositories are all in Python, what does that say? You might argue it means you are a rounded developer with a plethora of programming language experience? In reality, you have clouded the situation and introduced doubt. Does this developer really prefer to focus on NodeJS or Python? Which language and stack do they have more experience in? Which one do they love? In order to find out, the recruiter must do research or ask you those questions. All of this takes time and, unless you haven’t been paying attention, you know that this is something they do not possess. Be consistent in your message.

ReadMe

The GitHub ReadMe has long been hailed as the résumé killer. No need for those old fashioned documents. Keep it all in one place? The home of developers. Well, that didn’t quite work out, though some put a lot of effort into this (relatively) new space.

It’s not that the ReadMe is a bad idea. Far from it. In fact, I do like it but the idea is a classic startup failure though we can hardly call GitHub a startup. The failure is they forgot about the customer, the recruiter. Just as I said previously, the recruiter is your customer when you are searching for a new job or pushing marketing material out there for recruiters to find.

So, why have they forgotten about the customer? Well, they have and they haven’t? They made it easier by giving the developer a chance to put everything together as a one-page for the recruiter to find. However, GitHub still feels like the home of the developer or technically advanced. Recruiters feel uneasy in that space. For this reason I tell developers that recruiters are only here to confirm what they have understood elsewhere.

The next problem is quite similar to those I have described for LinkedIn. Developers are inefficient in their presentation. Colours, graphs and charts are nice but think about the overall message. It’s common for the ReadMe to light up with a plethora of technologies. I would urge you to tell a less bright, but more targeted story. Think about the narrative. Funnel the reader into contacting you for an interview. Tell a very narrow story. If you are here for [INSERT TECH HERE] (e.g. React) then you are in the right place!

Use the ReadMe to showcase your skills and passion. Use it to direct the reader to your best evidence. Think about achieving a goal and that goal is to contact you!

Profile Picture

There is little need to go over this again as we covered the specifics in the section on creating a profile photo for LinkedIn. My only suggestion is to keep the photos the same. There’s nothing more frustrating than being unsure that you are on the right profile. Often the link to a GitHub profile from a website or similar to GitHub is broken. The recruiter might have to search and try to find you and a profile picture helps a lot here. When 20 people are returned after a search for your profile by name, the chances are the recruiter is moving on unless there is a visual confirmation. Keep it simple!

Repositories

In addition to the overview screen, the other place that recruiters tend to venture is the ‘repositories’ tab. No, they’re not looking for code reusability or a test coverage percentage, merely a quick scan down your repositories. A nice feature for recruiters is that the repositories are displayed in chronological order. What you have been working on in recent months is there for all to see!

Just as we discussed during the section on the pinned repositories, if the recruiter cannot quickly see evidence of you working in a repository that matches the tech they are searching for, it raises doubts. Again, don’t assume that the recruiter will know that this JavaScript repository is Angular or React, for example. Spell it out! To be clear, repositories in different technologies are not a bad thing, however, recruiters do need that little confidence boost. If the list shows your most recent repository in a given technology is over 12 months ago, what does that mean? It could mean that you have moved on to something else? It could mean it’s not your preferred technology or that you are now a little rusty? For the recruiter to find out takes time and we know what that means?

Another red flag is developing your personal site in a different stack to the one you are applying to work in. An example might be that you’re moving into React development from Angular, but you’ve just started your portfolio site in Angular. As developers, you and I know that it’s just JavaScript, but it is wrong to assume that a recruiter will know or trust this.

For the junior developer or anyone learning new technologies, there is commonly a list of what I call ‘shallow’ repositories. Common signals are a title like ‘create-react-app’, two or three commits followed by abandonment.

This is natural when someone is learning, but it can give you away. It can be a tell-tale sign of a junior. For the junior it raises concerns of ‘tutorial hell’. The student is stuck in the cycle of completing tutorials but never applying the knowledge in real world projects. For the senior it shows you are not yet overly confident, or potentially competent in this technology. Beware of the signals your repositories might be sending.

Beware of shallow repositories!

For the junior developer, particularly web developers, I often encourage them to not create a new repository for each tutorial, but to build them all as pages in a single repository or site. To the outsider or recruiters this appears like a larger repository and something closer to a real world project that you work on daily and is growing. Now you appear closer to a mid-level developer than a junior as working on larger projects is the sign of greater experience.

Side Projects

I’m never going to tell you that side projects are mandatory. That is for you to decide. Most cite time as the limiting factor when it comes to having projects outside of their working day. If you’re trying to break into tech then I would argue that the requirement for a side project is more prevalent. You at least need some way of showcasing that you are learning and, most importantly, applying what you learn in a way that allows recruiters and employers gain confidence that you are ready for real world projects.

The mere mention of side projects as a subject has most developers fearful of a huge time commitment outside their usual 9-5 employment. It need not be this way. Just as in the previous section on activity within LinkedIn, we are looking for a drip feed, a window into your world to show that you are being proactive in this journey and you are immersed in technology. Can you take what you learn each day and apply it and showcase it?

Even if you write blog posts and engage in discussions about software development, most people will not read your posts. You can rest assured that the vast majority of recruiters will not. The posts merely serve as a confidence booster, once again. Even developers don’t like to look at each other's code. Just look at anyone performing a code review on a pull request! The phrase “Looks good to me?!” springs to mind!

Employers, if technical, will prefer to have you work in controlled environments for their screening tests than read code from your personal project. This way they can set the task and the starting codebase so they can better compare solutions from competing candidates. Reading other people's code is rarely something they choose to do. Therefore, for your posts about a side project, a code snippet and a few words about it are usually sufficient. The point is that now you are making regular updates on a project that appears to be growing and the evidence of it is easy for recruiters and employers to see.

For example, consider you are a junior developer pursuing a career in web development. At the start of your journey, as early as possible, start a website. Literally, create a blank site and deploy it. Write a blog post to accompany it. You have started and remember, you’re not trying to showcase talent on day 1. Talent on day one for a junior developer is a hard sell. You want to take people on a journey. Rarely does a good movie begin with the hero winning the day in the opening scene. The audience needs to see them suffer and face enormous challenges and by the time they win in the end, they have the full support from the audience as they know what the hero has gone through. You need to go through that too and taking the audience with you is key!

Months later, when quizzed about your experience with a problem or a technology, instead of having a boolean answer (e.g. Q: “Have you ever used TypeScript?”, A: “Yes”) or trying to convince someone of your experience, you can direct them to your evidence. You have a body of work that showcases every step of your journey.

Evidence is undeniable

If you are more senior, the solution is the same. Again, few will read what you write outside of developers wishing to tread the same path. You will, however, have hundreds of posts, a lengthy commit history and a growing project that shows your commitment to the industry and this technology. Convincing people of anything is hard. Evidence is undeniable!

This side project brings the other profiles together. You have a solid commit graph. One project that is growing that utilises the technology you want to work with. Accompanying posts are on LinkedIn and featured so that recruiters and employers cannot fail to stumble upon them. The force is strong with this one!

How often should I commit code?

I get asked this question regularly and it is difficult to answer. The truth is the commit frequency is irrelevant as long as there is consistency. Remember, this is not about impressing someone with a single commit or feature. This is about showing your dedication, passion and enthusiasm for the subject. If you commit daily, all credit to you. If it’s once a week, that’s still 52 commits and accompanying posts per year. It’s for you to decide how quickly you need that body of evidence. As with your code, smaller commits and more frequent comments are always welcomed!

Portfolio Website

When web developers create portfolio websites, I always feel like they misunderstand the intended purpose. At least, what I believe should be their intended purpose. I’m sure I’ll receive passionate feedback on this subject!

Most portfolio websites are carbon copies of small business websites. I’m not referring to the default bootstrap theme. I mean they follow the same format and user experience. Let’s unpack this because it does not only apply to web developers. Many developers who work in a variety of other technologies create personal websites to attract potential employers, even if they merely download and edit a template.

The strategy behind most small business websites is to funnel the customer into purchasing a product or service. Contacting the business is low on the agenda. Even contacting a company to make a support request is actively discouraged. Contact points are usually found at the bottom of the page, via links in the footer and are therefore the last thing you see. Remind yourself, what do you want your customer (the reader of the website) to do? The goal is for them to contact you, is it not? Everything else, whilst wrapped up in an explosive showcase of coding talent, is merely supporting evidence?

The mistake that we make is to think that people arrive at the homepage and are immediately blown away by our delicate alterations to the base Tailwind theme. They're not, trust me. They then click through each project in turn, gasping in amazement at your usage of FontAwesome icons and more. I'm joking and being somewhat sarcastic, but what really happens is they have a quick glance and mutter "Oh, they've done a weather app and a Netflix clone". It's hardly inspiring. How many times a day do you think recruiters see examples like these?

Housing all of your previous work and tutorials is a good way to utilise this space whilst bolstering your GitHub and LinkedIn profiles, but remember, this is secondary to the purpose of the site which is to ensure you get an invite to an interview. You are the product and the reader is the customer. Ensure your customer and product unite!

Theory

We've been looking at specific platforms and how to present yourself in the best way possible so that recruiters feel more confident in putting you forward or, at the very least, spending more time with your profile. Now it's time for a little theory. This applies to the previous platforms but then extends out to the wider reaches of your evidence. No matter where you have a presence, be it a personal website, a blog or something else, applying the theories we will discuss here will help you keep your presentation focused and build a funnel that leads to landing you an interview.

Tech Focus

You're a polyglot developer, I get it. Unfortunately, for the most part, the market doesn't share your optimism. Employers are searching for something specific and even when they are open to the possibility of hiring a polyglot, they engage a recruiter who struggles to understand or search for such a developer. What do you type into Google to search for such a requirement? So, they pin them down on specifics and we are back to square one. You need a focus and all of the rules discussed in this book are applicable once more.

If a recruiter cannot establish your focus, it will take too long to establish it via an email or interview. Unless they are desperate, they will move on to the next candidate. Where possible, all your marketing materials (résumé, website, blog, LinkedIn and GitHub profiles) should be targeting one stack. Examples of breaking out of this are languages like Rust where roles are infrequent and you have extensive experience in another language.

JavaScript is not a focus. It is to you and the rest of the development world, but not to your customer. Recruiters are searching for React, Angular, Vue and Node. Can you have success by defining yourself as a JavaScript developer? Yes, absolutely, but this is about my best advice for improving your chances of an interview.

Team Ready

When people are learning to code, they think it's about syntax, loops and lines of code. Later, you realise this is a smaller part of the role. The real life of a developer is pull requests, standups, refactoring, pulling out libraries and replacing them. It's about the team. Working with other people is a skill and it usually takes people a long time to realise this.

Coding on your own is a natural beginning, but eventually you have to work with others and that's where the pain begins. Having that experience and then being able to demonstrate and evidence it is valuable. In my opinion, this is why open source contributions are so valuable. It's not that you've written something incredible. It's that you've coded to a public standard, written tests and proved you can slot into a team.

To be team ready, you cannot just have knowledge of your chosen programming language or framework. You need demonstrable experience with the whole toolset that is being used including testing, continuous integration and deployment pipelines. For clarification, you don't have to set all these things up, but having worked in that environment means your transition and time coming up to speed will be greatly improved and that is attractive to employers as the effect of slowing a team down is often considered when hiring.

Related Technologies

A recruiter is rarely looking for ‘just’ a React.js developer. There are always accompanying technologies. To continue the React example, it could be as simple as Redux or Context API. Perhaps they use TypeScript and a developer able to hit the ground running in that respect would be sought after? Going back to being ‘team ready’, that could mean experience with testing? The point is, employers are rarely looking for simply a ‘software developer’. Even if they are, this strikes fear into the heart of the recruiter. They crave a detailed job specification as it helps them with their search criteria and filtering strategy. It also helps them to interview candidates as they have more specific technical questions to ask. You would be quite shocked to hear of the time spent in most recruitment departments campaigning for, and developing strategies to extract a better job specification. If you are not overly technical, how can you measure how ‘good’ a developer is? It’s a challenging prospect for even the most experienced developer. Recruiters are often dealt a bad hand in this respect. You have to feel for them.

With this in mind, your job is to research the roles you would like to apply for. As most job advertisements are very similar in their content, you can scan through the hard requirements for the role and then the ‘nice to have’ elements. If you were to tally these on a spreadsheet you could easily establish which skills are most commonly requested on roles that are of interest to you. This is your list of technologies to learn and showcase in your marketing if you are already knowledgeable about them.

The closer your match to all of the requirements, the more confident the recruiter will be to put you forward. Once their list of candidates is nearing completion, they apply their favourite ORDER BY clause and it is always:

```
SELECT * FROM developers ORDER BY recruiter.confidence LIMIT 5
```

You’ll note that I have added a LIMIT clause too! I want you to remember that if you are outside of this initial filter then they will interview and present every acceptable candidate until the list is exhausted. Then, and only then, will they return to the rest of the list for more candidates. This is where the recruiter can go quiet for a few days. The client did not go quiet, they were interviewing other candidates. If the client has, indeed, gone quiet, then this is because they are interviewing candidates from another recruiter. Does this sound familiar?

Testing

For all of the above reasons, testing was a subject I almost always introduced into a software developer interview. It helps me gauge whether you are team ready. It helps me understand the scale of the projects that you have worked on and it always gives me an insight into your position and stance with regards to code quality. The most common response to the questions: “Do you test your code?”, “What types of testing do you perform?” and “Do you practise Test Driven Development (TDD)” was “My employer does not value testing and therefore we don’t write any tests”. It is often caveated with “....but I want to”. We are far enough into this now that I hope you recognise that this introduces an element of doubt?

Perhaps you are experienced with testing? Perhaps you have never had the chance but realise there is some value in it? My strong advice here is to NOT have the above answer to these questions. Honestly, any different answer is preferred as it is so refreshing. I encourage you to embrace testing and become comfortable with the tooling. Answering the question about whether you test your code with: “I do, they don’t. I write unit tests in …. ” immediately tells me you are experienced, see the value and care about code quality regardless of whether anyone else around you does so. You do not have to implement tests at work, against the wishes of your employers and land yourself in trouble. Write tests on your own projects instead. Ticking the box on testing puts you into a smaller group of developers. I.e. the ones that recruiters interview!

Miscellaneous

Career Change

The field of software development is so popular right now that many people are coming to the industry as a career switch. This can be daunting as it feels like wasting all the skills and experience that you have collected in your years of experience. You are, essentially, starting again. I don’t believe it needs to be this way.

Soft skills are finally getting the attention they deserve. These go with you. Life experience is always valuable. Take a fresh look at your experience section on LinkedIn and where you might not have entries related to software development, the bullet points on your previous roles can be rewritten with a renewed focus, emphasising the soft skills that are now your most valuable asset from these experiences.

The next thing is life has probably already taught you that if you want something, you have to go out and get it. This point is well documented above in showcasing your proactivity. This is even more important for the career switcher. Can you showcase how you are making inroads to making this change. How are you attacking this problem of learning a completely new set of skills? Career switching is not a negative, you are making a positive step in your life rather than simply arriving by default after finishing up in education.

Another neat trick is to try and utilise your previous experience in your development journey. Instead of writing the standard Netflix clones and weather apps, writing a tool that may have helped you in your previous roles is powerful. You are solving real world problems and that is a far greater show of both development prowess and maturity. The question of whether you can take academic concepts and apply them to real world situations is answered beyond doubt!

Changing Stack

The bugbear of every developer is the lack of understanding and dismissive attitude towards developers wishing to change stack. It can feel like the industry is telling you that you must start over again from the bottom. Just when you thought your junior years were behind you, you are thrown back to the starting blocks once more.

As a developer, I know that switching tech is part and parcel of life in the industry. There's always something new to learn. This is the best and worst thing about the industry. You are constantly evolving.

Recruiters feel differently. Well, if truth be known, they want to believe you, but they are less confident about putting you forward. If there is anyone with any level of experience in their list, they will automatically rank above you and take your place in the queue for interviews.

My advice is to start a side project but just as it is for the career switchers, you need to drag along your evidence and validation from your previous skillset. As you document your journey exploring this new technology, can you write about how this language, framework or technology compares to that which you have been using previously. Can you detail where it is better and areas where it may be lacking? You are subtly reminding the reader that you have a wealth of experience that you are applying to this new technology.

Another thing is to write a real world project. When learning new tech we always go back to tutorials. This is a red flag and causes you to resemble a junior. Do the tutorials, if you wish, but quickly apply what you learn in a real world project. This is a powerful signal and reminds everyone of your prowess!

As new technologies come along, there is always a queue of developers hoping for the chance to jump into it. When I ask how they are pursuing this technology on their own, they stare blankly back at me. They want to be paid to learn. Don't we all?! This is the same problem as those hoping to break into the industry. Again, your chances of being selected are as good as everybody else's, unless you can demonstrate proactivity. Everyone wants to support someone who is on a journey. We are all on a journey, only most of the time people don’t adequately advertise that fact!

Self Taught

It is my opinion that the self taught developer is at an advantage. You weren’t expecting that, right? When you are self taught, you realise that you cannot rest and rely on a piece of paper or a qualification to get you what you want. When you’re self taught, you’re more accepting that marketing is a prerequisite?

The strategy for this marketing is what we need to discuss here. If you are self taught then you have studied and researched your subject. It is unlikely that you have documented that journey. Imagine if you had? Let’s not dwell on that and consider our options. Firstly, start documenting what you are doing as discussed above. If you do have existing blog posts and research evidence, consider reposting it on LinkedIn or at least sharing it where recruiters are waiting.

The next step is to address the fears. When you’re a self taught developer, doubters will assume you have some knowledge but lack the formal training. Can you provide evidence that you either do have that formal element to your understanding or, one better, document the merits of a different approach. This shows that you have the understanding and the development maturity to question it. Please note, hating on the formal side of things will likely be ill received. You may well be offending the reader. The world is full of choices and you have chosen a different path to the same goal. Your job is to show you have arrived at the same goal with the same credentials, you just don’t have the certificate, but as I champion in this book, evidence wins. Always.

Summary

I hope you have gotten some value from this eBook. I have certainly seen, first hand, the impact these subtle changes can make. I’m often asked about timelines for seeing the impact of these strategies and I had long avoided the question. It’s too subjective and also depends on so many factors:

- How frequently you are posting your evidence
- Your choice of side project
- The quality of your posts
- Market demand for your skills
- Years of experience
- Your geographical location (sadly)

On the flip side, I have received lots of feedback over recent years. Many have reported seeing a noticeable change in the volume of attention they receive in just a week. This is without posting on social media and purely down to the changes on LinkedIn alone. One developer told me they had gone from receiving zero requests for interview from recruiters to two requests per day within a couple of months. The impact can be dramatic.

To my delight, many developers get in touch to tell me they have landed a job offer or signed a contract. This is the nice part of my job, if I can call this my job? It’s more a mission and a labour of love. From breaking into the industry and increasing salaries to working remotely and earning a different currency, it is wonderful to hear and these are the reasons I keep doing this.

Testament to my statement that these strategies do work, one fun thing I hear with increasing regularity is that developers sometimes undo the work on their LinkedIn profiles once they land the job they want. They feel the need to ‘turn it off’. Too much attention is, apparently, a possibility!

I hope these strategies work for you and I hope you will let me know how you get on.

Hopefully, some of my catchphrases will now not just make sense, but resonate?

The best developers rarely get the job

It is the ones that present themselves the best that do get the interviews and the job offers. Time is the key factor. Make your profiles efficient for the reader and funnel them into giving you an interview.

Evidence wins

Every developer wanting another job is hoping. They are either hoping to break into the industry, hoping to work for a big brand name, hoping for a title or hoping for a salary. If you follow the ideas set out in this book, you will know that behind every hoping candidate are thousands of candidates just like them.

Rarely do developers show evidence of their true value. They prefer to hope that others will uncover it. Don't hope. Show what you have and make it **very** easy to find. This is how you stand out!

Why are you different?

Most developers look exactly the same. We have the same bad profiles, the same side projects and personal websites. We read the same articles, share the same posts and have the same qualifications. If you really want to stand out, you want to show them how you are different from your peers, not how you are the same.

Be proactive, very few are doing it!