As someone that's been a developer for 8 years, I've had plenty of interviews and a majority of them didn't work out. I went to college and got a bachelors in Computer Engineering after finishing the first half of a computer science degree and realizing that I wanted to know more about what was going on with a computer. After spending 10 months finding my first position I got stuck in that position for 5 years before I figured out how to take charge of my career advancement, and within a year of shifting gears, I took a position from a LinkedIn cold contact that gave be a 33% pay raise.

I'd love to help others to figure out how to short circuit those first years and get right into the career advancement. Before you can even get the interview you have to become intentional with your learning and you resume design. The short version of that is that you have to really understand your story and how to effectively tell it. Every step from resume submission to the first month of work at a company is about telling your story. You need to make sure that you present your relevant skills and experience in a way that allows potential employers to quickly determine that you are a good fit.

The interview process differs dramatically depending on the company and job you're interviewing for. Some companies will want you to interview for a full day, and others will want you to do a half dozen interviews. What most interviews will have in common are a set of questions related to your general technical knowledge, culture fit, peer interaction, prior experience, and likely some type of problem solving puzzle. Let's take a look at how to tackle these interview questions.

Pro Tip: If you want to learn as much as possible then you may want to look into Cracking the Coding Interview by Gayle Laakmann McDowell.

General Technical Knowledge

While there is a good bit of specific technical knowledge that is necessary to be a developer, it's important that you understand more generalized principles regarding an application's deployment and interaction channels particularly.

Some common general knowledge questions you might get in include:

1. How does a REST api work?
2. What is CICD and why is it important?
3. What are some common ways to prevent application/website failure?

These can be a little tricker than the learning for more specific questions due to the fact that they aren't something that you are gonna just go and study for. They tend to be things that come from working with tech and taking the time to understand what something is or how it works in base principles. One way that I learn about these different things is that I'll use wikipedia to look things up while reading job descriptions. I also tend to keep a list of terms that I've come acrss in meetings/documentation/online reading to research later. These are very high level pieces of research and shouldn't take up much time.

One of the worst answers that I ever gave was when I'd noticed the tech stack defined on a whiteboard wall, and said something about it. The interviewers then asked me how my team was using Spring. Were we using Spring Framework or Springboot? I didn't know, and I'd been advised once that I didn't need to know about that stuff, just don't mess with the xml files unless you know what you're doing. I said something to the interviewer about "we just use spring configuration" because we had lots of secondary applications that were configured to run by altering xml files parked on the application server. Failing at this question would later lead me to dive into learning Spring Framework and Springboot. After completing the **Java Brains** playlists on both I was more knowledgeable about Spring than anyone in my organization and able to help with much more advanced configuration things than others were able to understand. You have to let your failures guide your growth.

Culture Fit

When you work closely in a team, it's important that everyone gets along well enough. You don't want any major personality conflicts. It cand be quite costly to hire a person, train them, and then ultimately find out that they aren't going to work out. This can lead a company to spending a lot of time and money before they are able to get the help that they need.

Some common culture fit questions you might get include:

1. What about this role appeals to you most?
2. Tell me about a lesson you've learned from the workplace over the years.
3. If we asked your coworkers how they'd describe you, what would they say?

To me the thing about these types of questions is that you don't exactly fail them. If a company doesn't believe that you are a good fit with their culture, that means that you likely wouldn't enjoy the job anyway. Its a two way street. You want to make sure that you are working for a company that is going to fit with your values, and they want to find an employee that fits with theirs.

I personally tend to bring out my culture fit concerns as early as possible, and this is possible with many different open ended questions. When answering #1 above, I'd say something about their modern cloud based tech stack, automated deployment process, and that it didn't seem like a place that would have major issues with life-work balance. This is because I have 4 kids, and a wife that I love dearly and I don't enjoy spending nights and weekends doing work. If these things prevent a company from wanting to hire me then it's for the best.

Regarding that #3 question, I actually took the time to ask all of my closest coworkers to list how they would describe me and what makes me different than every other developer, and it allowed me to learn a lot about myself. I wrote about this previously in **Know Your Brand**.

Peer Interaction

If someone works well with others then you can overcome a lot of technical knowledge deficiencies. On the other hand, a really technical person that can't get along with their peers well will have major issues and can prevent a team from being productive.

Some common peer interaction questions you might get in include:

1. Tell me about a time that you had a problem with a coworker and how you resolved it.
2. What would you do if a manager assigned you a new task or project right before you were about to leave for the day?
3. How do you prefer to receive feedback on your performance from your managers and peers?

With these there's a lot more nuance than just right and wrong. In my opinion the answers should show professionalism, a bit of compassion, and respect to authority. Again we're talking about personality and not something that is entirely clean cut.

Prior Experience

There are many questions of this type that most people feel are dumb. In the end, your ability to work within the framework of these sometimes silly questions can demonstrate your ability to think critically under pressure. Many people don't do well with being questioned under pressure regardless of the topic. One of the best ways to do this well is just to practice. There are many lists of questions that can be found online as well as suggestions for how you might approach an answer. I'd urge you to think through answering many of these questions on your own so that you have some examples prepared already.

✓ need and I learned to do [x] in order to fill that need".

Some common prior experience questions you might get in include:

1. What's a time that you faced significant difficulty at work and how did you overcome it?
2. How do you normally learn about new things?
3. What kind of projects have you worked on before?

Problem Solving Questions

These are probably the most dreaded of all interview activities. These can be actual LeetCode puzzles, take home assignments, or whiteboard problems. The more that you know about data structures and algorithms, and the more coding puzzles that you've completed, the better you'll be at completing these types of interviews.

LeetCode (or similar) tasks can be the most frustrating, particularly the ones that you are supposed to do at home. I specifically remember doing the one for Amazon some years ago and there were 2 puzzles that should each take a half hour. If you are currently employed and have children then it is highly likely that you'll be doing as I did and working on this after you should have been in bed. That's a horrible thing. I think that the take home assignments are basically the same issue. Practicing with coding puzzles is ultimately the only thing that's gonna help you with these. Unfortunately these don't test your ability to do the actual job. If you are finding yourself coming up against these in your job search then I'd suggest that you might spend a good chunk of time doing puzzles, but I wouldn't let it take up more than half of your time spent preparing for interviews and the like.

✓ that takes [y, z, w] arguments and returns [q]" is a wonderful thing to get through at the beginning. It allows you to clarify the data that is available, and needed. I also like to include "We'll assume that there exists a function that does this other thing called [fubar]" and that can allow you to address the algorithm needed at a higher level than otherwise. If you are sorting something then it'd be really helpful to have a function that compares the priority/order of two objects. Ultimately the thing being determined in this interview is what your thought process and problem solving approach looks like.

Some common problem solving questions you might get in include:

1. Implement bubble sort on this very specific data.
2. How would you figure out if there was a loop in this graph structure?
3. What are some ways that you can break down a very large data set so that you can analyze it?

Final Tips for Nailing Your First Interview

Practice:
There's definitely something to be said about mock interviews, but talking to other developers can really help in any context. You can show up to meetups, give lightning talks, attend code reviews, find communities discussing coding challenges, or many other things. I've also just had a one on one lunch with people that I knew were transitioning into tech in order to talk about their experience and translating back to them what they told me in terms of development experience. What's important here is that you learn your story and figure out a way that you can talk about it comfortably while still telling the interviewer the things that you want them to hear.

Come Prepared:
Recruiters like it when you’ve done your research and can relate a question back to the company motto or one of the brand pillars. It's always a good idea to the look at the company's website and read through every bit of "about", "mission", or "bio" sections that are available. This helps you to make sure that the company aligns with your values, and helps you to figure out what are the types of things that you should be highlighting as you talk about yourself.

Ask Questions:
Come ready to ask a few questions about the position or the direction of the company. Asking questions shows that you are also checking to see if the position is a good fit for you and is impressive to a hiring manager. There are many online lists of questions that you can ask, but in my experience the best questions are the ones that are important to you. If you've had a really bad experience with working nights and weekends and you'd rather avoid that in the future then I'd ask how often there is work outside of normal hours. Regardless of your particular values, you can always jump off with "Describe what a typical day in this role looks like." And then be sure to listen through the answer and see if there is anything that you want to hear more about. For me "we have a pretty good work/life balance, but sometimes when there's a deadline we do what we have to to finish" would naturally lead to "How often in the last year have there been deadlines that required working outside of normal hours?" The answer here could quickly make me not want the job. Ask the things that you care about, and after you've had one or two jobs in tech you are likely to come across many things that irritate you and then you'll come up with your own list of things to ask.