10/18/2023 - These notes cover the Introduction I through V.

**The Interview Process**

Right off the back, the book dives into why it is important to understand the interview process. The interviewer is trying to gauge, among other things, Analytical skills, and by going through a coding challenge, they answer some questions such as:

* Did you need much help solving the problem?
* How optimal was your solution?
* How long did it take you to arrive at a solution?
* Were you able to think through the problem well and think through the tradeoffs of different decisions?

Even though most companies don’t handle data structures such as binary search trees daily, it doesn’t mean problem-solving skills are not valuable. For example, I am working on a recursive function in React to iterate through a JSON configuration to create components dynamically.

Sometimes, interviewers use some whiteboard, whether in person or virtual, allowing them to watch a person talk through the problem as they draw it out. The first presentation I gave touched on this in the sense that it’s a great idea to speak out loud while going over completing a problem-solving question.

**Behind the Scenes**

The beginning of this chapter goes over what to expect when getting started with a “screening” process. I recently went through a couple of interviews, which began with an HR person. After some pleasantries, they both asked several technical questions that, in my opinion, you should have some knowledge of whether you are new or not. For example, in both interviews, some questions were asked about JavaScript, such as explaining the difference between var, let, and const.

The rest of this chapter goes over some things to expect while going through an interview with a major tech company like Google or Facebook. There is not much difference between the two, except the interview at other companies may be a little easier. Again, the interviewer wants a way to see how you tackle a problem.

**Special Situations**

The first topic in this chapter is what the process will be like for “Experienced Candidates.” I have to say that after recently going through a couple of interviews, the book is right on the money. Whether you are experienced or not, make sure you can speak about projects you have on your resume or any projects mentioned during the interview. You want to be able to answer questions like: “What was the hardest challenge you faced?”

There is a slight cheat sheet here because some companies may also read this book to find ways to challenge their interviewees. This section gives us an actual behind-the-scenes preparation for creating an interview and a sense to tell if your interview is going well or not.

**Before the Interview**

Ensure you get the “Right Experience” by doing a big project, getting an internship, or starting something by participating in a hackathon or contributing to an open-source project.

How to write a great resume.

* Make sure to keep your resume short. One page if you have less than 10 years of experience.
* Employment History only includes relevant positions. Show what you did, how you did it, and what the results were.
* Listing everything you’ve ever worked with is dangerous. Many interviewers consider anything on your resume to be “fair game” as far as the interview.
* Place 2-4 projects that you are proud of on your resume. Make sure you can thoroughly talk about each one.

The book talks about learning more than one language. The idea here is that if you only know one language then the assumption is that you haven’t experienced very many problems. Everyone agrees that one should probably get good a one language first, or at least feel comfortable with the ideas and concepts, before diving into another language. I would pick Go/Java/C# as these are good general purpose languages that have GCs, Rust/C/C++ as these will force you to think about low level concerns and concepts as well as planning out an app for memory, python since it’s a good to have a scripting language to bash out quick issues, but first get good at one language. Also if you’re doing front end you should look at JavaScript before anything else and maybe in lieu of something. In the end it’s about expanding concerns and understanding not being able to just read syntax

Review the preparation map as a checklist for your readiness (page 30). A brief checklist of items you want to cover.

* Do several mock interviews.
* Build a project.
* Study interview questions.
* Do mini-projects to solidify understanding of key concepts
* Implement data structures and algorithms from scratch.

**Behavioral Questions**

I highly recommend the Interview Preparation Guide on page 32.

Be able to talk about your weaknesses.

Questions to ask the interviewer. Examples:

* “What is the ratio of testers to developers to program managers? What is the interaction like? How does project planning happen on the team?
* “I’m not familiar with technology X, but is sounds like a very interesting solution. Could you tell me a bit about how it works?

Make sure you know your technical projects. Be able to discuss challenges, mistakes, technical decision choices, and the things you would do differently.

* Responding to Behavioral Questions
  + Be Specific, not arrogant, by giving facts.
  + Limit Details by allowing for hooks.
  + Give Structured Answers by following the examples on pages 34 and 35. I was recently asked to give an example of when I had to convince the team of a decision and how I went about it. The interviewer gave me feedback right away and let me know how the story I gave demonstrated quality team leadership.
* So tell me about yourself.
  + The book has a great structure; you can make this yours and dig into what you are working on and be excited about it.