# Technical Interview Workshop -- "Preparing for the Trivial"

Monday, March 19, 2012
7:04 PM

*The Process*

- Interviews are **not** tests! Interviews are more like auditions or conversations!
  - Don't just throw answers around… pretend that you have to prove what you know!
- Questions:
  - What data structure would you use to store a phone book?
  - Design a rate limiting system for an API?
- "OMG hire this person now!"
  - Strong engineer
  - Qualified for role
  - Culture fit
- For each step of interview, it's still an audition! Just because you answered it for previous guy doesn't mean you're off the hook!

*The Interview*

- Typical interview -- generally 45 minutes long
  - You -- 10 min (resume, transcript, interests, etc.) → used to figure out where you might fit in company
  - Technical Element: Coding, Data Structures, Algorithms, System Design, CS Fundamentals -- 30 minutes
  - Product/Company -- 5 min "Why do you want to work here?"

*The Resume*

- Avoid expert! You're "experienced" with it… easier to impress as well
- Include what your projects did… but remember it's all fair game for questions

*What are your interests?*

- Even if it's something as simple as "frontend vs. backend" technology

*Coding*

- Be comfortable with writing code on all of these mediums! Remember, interviewer will be watching you, and you'll have some time constraints…
  - Whiteboard
  - On paper
  - Chalkboard
  - On a laptop

*@chanian's Checklist* -- Ian Chan's plan, but just come up with your own way of approaching a question

0) Clarify, ask questions (interviewer *may* omit some information! Will you realize that you need more information?)
1) Reiterate the problem ("Are we sorting… integers?")
2) Work out a solution (TALK OUT LOUD. Maybe quick sort would work, or maybe merge sort, which splits the list up...)
3) Plan out your implementation code
4) Code
5) Run through your code (Now that you have code, continue! Don't stop there! It's not a test! Show how you understand everything about it!)
6) Think about corner cases (At least mention them!)
7) Do a simple space/time analysis (What separates CS graduates from hobbyists!)
8) Optimizations/improvements (memoization, helper function, etc.)

Example: `add1(arr)`

```
// does addition in place       << Assumptions!
// assumes an integer array     << Corner cases: null case?
function add1(arr) {
```

```
    for (var i = 0; i < arr.length; i++) {
        arr[i]++;
    }
}
```

*Data Structures*

- Trees
- Hash Tables
- Lists, Sets
- Queues, Stacks
- Graphs

*Algorithms*

- Sorting functions
- String manipulation
- Greedy algorithms
- Dynamic programming

*Other CS Fundamentals*

- OOP Concepts
- Programming Languages
- Stacks/Heaps
- Process (AGILE, Waterfall?)
- Computer Networks (TCP, HTTP)
- Testing
- System Design

**Prove to me that this stuff is trivial to you.**

- Freebies!
    - Design and implement a hashtable.
    - What happens when I visit twitter.com?
    - Reverse a string in place
    - Implement doc.getElementsByClassname (How does it work?)

- Summary:
    - Be thorough
    - Talk out loud
    - Have a plan for answering problems
    - Practice like hell
    - Practice some more
    - Study the basics
    - Demonstrate interest
    - Have some fun!