


# Okay, folks, here's how the Google interview process really works

 [google.com/blog/2011/09/07/okay-folks-heres-how-the-google-interview-process-really-works](http://google.com/blog/2011/09/07/okay-folks-heres-how-the-google-interview-process-really-works)

6 September 2011

Somehow, many candidates have gotten the impression that the interview process is some elaborate system, and if their process is different from their friend's, it must be a reason for it.

The truth is so much more straightforward than that, and once you get, everything will make sense. Or that's my hope, anyway.

**Here's how the process works at Google for software engineers.** We'll look at this from the interviewer's side and from the recruiter's side. Although this is technically just about Google and Software Engineering, the advice / structure is largely universal across tech companies.

## ***What Your Interviewer is Doing***

This is more or less how an interviewer becomes an interviewer:

1. *Training:* Your interviewer takes an interview training course to teach them how to interview. Actually, they're really just told things like, "Don't ask the candidate if they're married," and "Don't ask where their accent is from." In other words, don't do anything that's going to get the company sued.
2. *Shadowing:* Next, they "shadow" two interviews... you know, in case they didn't get enough of Google interviews when *they* were a candidate (yep, your parents were once children themselves, and your interviewers were once candidates). This lets them see the process again, freshly, and chat with the "primary" interviewer about what they thought.
3. *Instruction:* Then... they're thrown into a room and asked to interview a candidate. Where do their interview questions come from? Well, where would *you* come up with interview questions if you were in their shoes? You'd probably bring them from you own interview experience or find them online on sites like [CareerCup](#).
4. ***Evaluation: Interviewers evaluate how well you did relative to other candidates.*** (This point is so important that I'm obnoxiously

bolding and highlighting it. If there were a <BLINK> tag still, I'd use that.) There are two interesting parts of this statement. **(1)** It's "how well," not "% correct." It's a multifaceted, qualitative evaluation that takes into account how you solved the problem, how long it took you, how many mistakes you made, how much help you needed, and how optimal your solution was (note the "hows", not the "ifs"). I've never once made a simple statement like, "the candidate got this question correct," because that statement doesn't make sense for anything other than simple factual questions. **(2)** Performance on a question is judged in comparison to other candidates on the same question. Taking 10 minutes to solve a question optimally may be great performance on one problem, but horrible performance on another. How do you know if you did well or not? You don't.

Note that no one here has told them what to ask, or given them a list of potential questions, or asked them to focus on a particular topic.

In other words, they have about as much interviewing training / instruction when they're getting started as *any candidate does*.

Think about this. *There is no system.* "" are no different from "old Google onsite questions," or, for that matter, from old *Amazon* phone interview questions. When interviewers ask more or less whatever they want, there's little consistency\* across a company, interview type (phone vs. onsite), or timeline.

[\* There are some differences, but most of these are minor. Phone interviews will generally focus slightly less on coding, though there is still coding. Non-web based companies aren't likely to ask about scalability, unless it's relevant for their team. And some companies have a slight preference towards certain topics, such as Amazon's focus on object-oriented design. The differences between Microsoft, Amazon, Google, Yahoo, Facebook, and Apple are covered more in Cracking the Coding Interview, 5th Edition.]

### ***What Your Recruiter is Doing***

You might not know this, but your recruiter is a person too.

Ideally, your recruiter wants to usher you through the process efficiently. If you are going to get you an offer, they want to tell you as quickly as possible. If you're not going to get an offer, they *still* want to tell you as quickly as possible.

But, that doesn't always happen because stuff comes up - reorgs, vacations, general life / work busyness.

Next time you ask why your recruiter took a while to respond, ask yourself why *you* sometimes take a while to respond. More often than not, it's just that stuff came up that has nothing to do with the other person.

----

So with all of that as preamble, let's see if we can answer some quick questions.

### **I made a mistake in coding. Am I going to get rejected?**

See above. Do most other candidates make that mistake (or similar mistakes)? [FYI: on a medium difficulty or higher problem, very few people solve the problem "perfectly."]

### **I'm preparing for a Microsoft phone interview. What should I focus on?**

See above. The fact that it's a Microsoft interview, or that it's a phone interview, is mostly irrelevant. Look at software engineering interview questions. If there are particularly points of knowledge you're struggling with (e.g., you forgot how to traverse a binary tree), you should study those. You shouldn't worry too much about who is giving the interview.

### **How long do I have to solve an interview question?**

This is sort of like asking how long you have to solve a math problem. Arithmetic problems are solvable in seconds, basic calculus problems in minutes, and complex theory in hours, weeks, or even years.

For a *specific* interview problem, taking "too long" might indicate poor performance, but that amount of time varies significantly across problems.

**When my buddy interviewed with Apple, he was asked to solve 3 questions in 30 minutes. I didn't even finish one problem in that amount of time. Do I have any chance?**

My imaginary 10 year old niece solved 5 math problems in only 10 minutes, while my math professor has been working on this other math problem for a year now. My imaginary niece, therefore, is smarter than my math teacher.

The above question makes about as much sense as this statement.

Unless you and your friend were asked the same interview questions, you really can't conclude anything from your experiences.

**My friend heard back from Google the day after his interview, but it's been five days and I haven't heard a word. Is this just Google's way of rejecting me?**

Nope. Doesn't mean a thing.

**I am an experienced candidate. Will I held to the same standards and asked the same kinds of questions?**

More or less, yes. Depending on who you talk to, experience either helps you on standard coding / algorithm questions (since you've been coding for longer) or hurts you (since you're further away from these academic topics).

The slightly unfortunate reality is that interviewers tend to repeat their favorite questions across candidates, so, all else being equal, someone with 30 years or experience will probably be asked the same things as a recent graduate.

However, there will probably be somewhat higher expectations when it comes to behavioral / resume questions.

## How long does Facebook take to respond after an interview?

See earlier section about recruiters. Asking how long they take to respond is like asking how long you take to respond to an email. The company may *target* responding within a week (which is a fairly standard amount of time), but delays can happen for all sorts of reasons.

---

I hope this little window into the interview process helps you next time you wonder why something happened the way it did. Mostly, it's just people running around doing whatever they want. Yep - that's it. ***There is no system.***

*Shameless plug (but, hey, lots of candidates swear by it): Just because there's no grand, overall system doesn't mean you can't prep for your interviews. You can and you should. Check out Cracking the Coding Interview, 5th Edition: 150 Programming Questions and Solutions. Lots of advice, and none of the fluffy "be the best you can be!" stuff. Straight, to the point, and lots and lots of cool coding problems.*