JOEL ON SOFTWARE

I'm Joel Spolsky, a software developer in New York City. More about me.

OCTOBER 24, 2006 *by* JOEL SPOLSKY

The Phone Screen

RECRUITER, ARTICLES

It happens all the time: we get a resume that everyone thinks is really exciting. Terrific grades. All kinds of powerful-sounding jobs. Lots of experience. Speaks seventeen languages. And saved over 10,000 kittens!

Look! kittens!

And then I call them up, and I can't stand talking to them. Within ten minutes, I realize they are not going to make it as programmers. I've had people with great resumes tell me a pointer should fit in one byte. Sometimes they just can't answer the simplest questions, or you feel like you have to wrestle the answers out of them.

Before moving on to a full-fledged in-person interview, we usually use a phone screen to make sure that we're not wasting time and money on someone who is just seriously not smart.

A phone screen has distinct advantages over a normal in-person interview. First, it's cheap. It takes 45 minutes to an hour and actually does eliminate about half of the people who looked really, really good on paper.

of the people who looked really, really good on paper.

More importantly, it's more fair.

With a phone interview, because you can't see the person, it's easier to focus on the quality of what they're saying rather than other external factors not relevant to their job, like their appearance, or their nervousness. Ever since Malcolm Gladwell wrote Blink, I've been terrified of the prospect that we might be judging candidates too quickly based on things which are not relevant to their ability to do their job—their appearance or confidence or height or general nerdy demeanor might make us way more apt to look on everything else that happens during the interview with rose-colored glasses.

The great thing about a phone interview is that it's much harder to form these kinds of snap judgments; you actually have to listen to what the person is saying and decide if that corresponds to what a smart person might say. This isn't completely true, of course: you may have prejudices about certain accents or dialects. But at least you are moderately less susceptible to appearance prejudice.

My phone interviews have three parts. In the first part, I ask the candidate to describe their career history and basically tell me about themselves. This is mainly intended to get them loosened up and feeling comfortable, to eliminate any nervousness, and to let them sort of present themselves the way they want to be presented. During this stage, you should be looking for evidence that the candidate is a problem solver: the kind of person who *gets things done.* You're also looking for passion. You want people who care about the stuff that they did.

During this part I'll drill down on two kinds of things: technology and politics.

> **Technology.** If someone tells me that they implemented such-and-such a project, I'll ask detailed questions about the technology they used and how they used it. I'll also ask specifically what role they played. Sole developer? Developer on a team? I'll tend to go into these questions in great detail, because this is where you uncover the people who either didn't know what they were doing, or have made things up, or have exaggerated their own roles. If someone's resume implies that they spent two years coding in

Python, for example, I'm going to drill down until I'm pretty confident that it sounds like they really have two years of Python experience.

Don't hesitate to use trick questions. I might say, "I hate Python, because it's really tedious to declare all your variables. That's why I like C better." Python doesn't make you declare variables (and C does). Don't be afraid that the candidate will feel pressured to agree with your false assertion. Smart programmers have a certain affinity for the truth, and they'll call you on it. As Dave Winer says, "You can't lie to a compiler." If they're really good, they'll find a way to be diplomatic about it. But they're going to be passionate enough that they'll almost forget that they're in an interview. That's a good sign.

Politics. Behind the boring list of past employers on the resume, there's always a story. What I'm looking for is the story of how the candidate handled challenges in the past. I'm looking for people that got things done, even in the face of opposition. I'm looking for people who challenged the status quo, who overcame objections, and who made things happen. So when the resume says, "drove the adoption of .NET," I want to hear what *drove* means. In detail. When the resume says that they "founded a company," I want to hear everything. Whose idea was it? Who convinced whom? Who did what? Did it work out? Why not?

The second part of the phone screen is the technical problem. I usually ask the same question for years and years before switching it, because this makes it easier to compare candidates. The question is a wide-ranging, open design question: how would you design a data structure or a block of code to do *x?* Where *x* is something kind of big and complicated. I usually have a series of questions ready to guide the candidate down a particular path in the design of this data structure or block of code, because it's such a big question, and I can often tell how smart the candidate is by how far they get down that path in a fixed amount of time.

Here are some ideas to get you started:

- How might you design a program that lets people play Monopoly with each other over the internet?

- What would be a good data structure for a photo editor?
- How would you implement code to operate the elevators in a high rise?
- How would you implement the rendering engine of a web browser?

The ideal question takes something where the interviewee is deeply familiar with how the thing works from having used it, but is unlikely to have ever implemented it themselves. You want something that can be done over the phone, without too much writing, so "how would you write the code for quicksort" is a bad question, because we don't expect programmers to be able to recite code over the phone. You want to have a conversation about algorithms and data structures, really, the meat and potatoes of programming, where the goal is *not* to find the best possible answer, necessarily, but simply to give you the opportunity to talk about code, to talk about time/space tradeoffs, to talk about performance characteristics of code, all of which will add up to giving you a pretty good idea in your mind whether the person you're talking to is actually pretty good at programming and whether they're smart or not. If you find yourself explaining everything three times, either you're terrible at explaining things, or you're talking to someone who is not that smart.

The bottom line in my interviewing technique is that smart people can generally tell if they're talking to other smart people by having a conversation with them on a difficult or highly technical subject, and the interview question is really just a pretext to have a conversation on a difficult subject so that the interviewer's judgment can form an opinion on whether this is a smart person or not.

The third and final part of the interview is letting the candidate interview *me.* Remember, the whole philosophy of recruiting is predicated on the idea that smart candidates have a choice of where to work, and if that's true, the interview process is as much a way for the candidate to decide if they want to work for us as it is a way for us to decide if we want to hire the candidate. "Do you have any questions about Fog Creek, about working at Fog Creek, or anything else you want to ask me?"

Sometimes this part of the interview reveals a frightening lack of preparation by the candidate. "So, what exactly does Fog Creek do? And where are you

located?" Failing to do even the most basic homework before the interview, by spending five minutes on our web site, does not give me a great deal of confidence in the candidate's ability to be smart or to get things done.

By this point I've already decided if this person seems smart enough to score an in-person interview. So I treat the third part of the phone interview as if I were the one being interviewed, and the candidate was the one that had to be sold on Fog Creek. Which they do.

Passing a phone screen is never enough to get hired. It's nothing more than a simple filter designed to save time and expense on in-person interviews, and to eliminate candidates who will never make it before you've flown them all the way across the country and put them up in a fancy hotel. Still, even with the phone screen, probably only about one in three candidates makes it all the way through the in-person interview.

ABOUT THE AUTHOR.

I'm Joel Spolsky, co-founder of Trello and Fog Creek Software, and CEO of Stack Overflow. More about me.

← PREVIOUS POST

The Phone Screen

NEXT POST →
The Guerrilla Guide to Interviewing: version 3.0 now shipping!