VSLive San Diego 2018

How to Interview a Developer

Billy Hollis

@billyhollis

Disclaimer: I am not here to advise on human resources policies or laws

Before the interview – Understand why you're there

You are there to judge this person's value and fitness for your team, your organization, and your problem domain

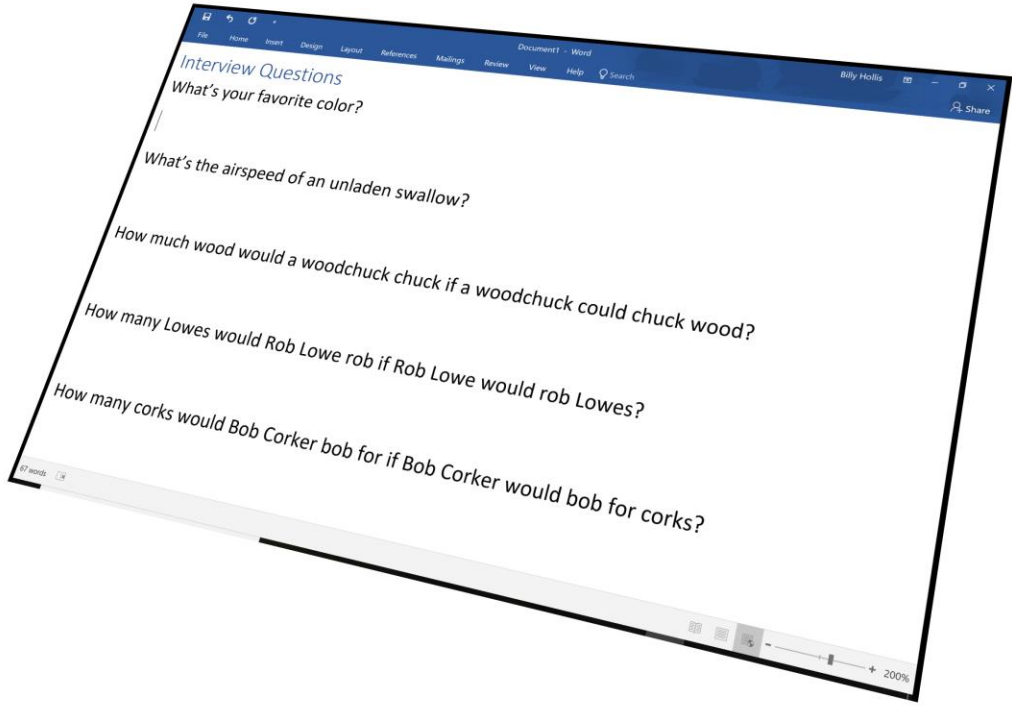
You are *not* there to socialize with another member of your tribe

"There are 10 types of people: those who understand binary and those who don't."

You are *not* there to impress the interviewee with how smart you are

Before the interview - preparation

- Don’t wing it – prepare
- Have a standard list of questions to help you find the right questions for this candidate
  - Starter kit in this session
  - No canned lists – questions should vary based on candidate
- If needed (and it probably is), work on your listening skills
  - I’ll give some tips for that later



Before the interview – request sample code

- Only a few pages
- Not a complete project
- Look at it with some of the same attitude you would use during a code review
- Some candidates can’t give you any because it’s all proprietary

Before the interview – cover letter and resume

- Review cover letter and resume
- I put a high weight on cover letter, because they probably wrote it themselves
- I don’t feel obligated to read more than two pages of a resume

- Before doing the interview, print a copy of the resume, or ensure that you have it open on your tablet

I’d like to ask you all a question:

What accomplishment in your career in the last five years makes you feel the most proud?

Analyzing a resume

- Technology focused vs. results focused
- Length of time spend on teams
  - Consultant vs. employee

Alphabet Soup - why they cook it

Languages	*Tools and technologies*	*Operating systems*
.NET	Visual Studio	Windows XP
Java	JBuilder	Windows Vista
XML	Dreamweaver	Windows 7
C	Rational Rose	Windows 8
C++	UltraEdit	Windows 10
JavaScript	Borland C++Builder	Linux
Typescript	Oracle SQL* Plus	UCD Unix
SQL	SQL Server	Mac OS X
HTML	Azure	iOS
UML	Minecraft	Android
XAML	Photoshop	MS-DOS
C#	Illustrator	CPM
F#	Corel Draw	Windows 95/98
Erlang	PowerBuilder	OS/400
Norwegian	Universal Windows Platform	CICS
Farsi		

The interview

- General guidelines
- Examples of useful questions
- Listening tips
- How to wrap it up

First impressions

- I find *positive* first impressions are on target about 60% of the time
- I find *negative* first impressions are on target about 95% of the time
- Be aware of your first impression – even write it down – but don't be certain it is correct

Use neutral tone and body language

- Clever Hans and the numerologist
- Remember – the candidate is highly motivated to please you
  - They will unconsciously bias their answers towards what they think you want to hear
- Thrashing or fishing for the answer they think you want is a serious red flag

Das lesende und rechnende Pferd mit seinem Lehrer HERRN von OSTEN (Berlin)

Watch for defensiveness

- Developers make dozens of mistakes a day
- Being a good developer means being comfortable with that
- The opposite of this attitude is being defensive about mistakes
- Defensiveness is highly correlated with being a sub-par developer

Listen a lot more than you talk

- At the beginning, let the interviewee know that you will give them a chance to ask questions at the end
  - About the company
  - About the project / team
  - About why you asked certain questions….
- During the interview proper, restrict your own talking to setting context for questions and asking questions
- Remember: you are not trying to impress the candidate with how smart you are

Taking notes

- In an average interview, I write about six to eight notes, usually only a sentence each
  - Writing lots of notes puts off some interviewees – they assume you are writing bad stuff about them
- Some notes are to remember to ask a question later rather than interrupt them to ask it now
  - I put an asterisk in the margin for these

Asking questions

- Ask a combination of technical questions and soft skill questions
- Ask mostly open-ended questions
- In this section, I'll give you over a dozen useful questions that I use in interviews
- For each, I'll explain why I ask it and what I'm looking for in an answer
- This list should just be a starting point
  - Do your own variations
  - Prepare your own questions concerning the technology, domain, and process that applies to your team

Think back to a system you worked on five years ago. What would you do differently if you could develop that system over again?

Object orientation drill down: Ever done an object hierarchy? If so, how much code was in the base class vs the subclasses? Ever done an abstract interface? How would you decide whether to use an object hierarchy or an interface?

Suppose you had a program with data validation, but the data validation rules were not known until runtime. How would you handle that?

How many dentists do you think there are in Nashville? Think out loud as you arrive at your answer.

What does Reflection do? What have you used it for?

If Microsoft asked you what you don't like about .NET and what you would like them to fix, what are the top two or three items on your list?

("Imagine you are having dinner with Sadya Nadella, and he asks you….")

What is the weirdest bug you've ever encountered and how did you track it down?

(If you really want to see them code)
Do a program on the whiteboard to reverse the characters in a string.

If you could acquire complete expertise in some leading edge technology instantly by magic, which one would it be?

What's your philosophy of user interface design? How do you ensure that your end product suits the users' needs?

What do you see as the key advantages and disadvantages of ASP.NET vs. Windows Forms vs. WPF vs. Universal Apps vs. HTML5/JS vs. blah blah? How would you decide which one to use for a given project?

What do you think of agile methodologies? What are the pros and cons of them?

What's your philosophy for becoming familiar with leading edge techs? Try to learn them all up front? Learn them as they apply to projects? Somewhere in between?

How do you get up to speed on a new technology?

(On the whiteboard) Diagram a typical n-tier architecture, and explain the reasons for each layer. Has your thinking about n-tier changed in the last few years? How? What changes do you expect in n-tier architecture because of the move to cloud-based computing?

Have you been involved in a project where the wrong architecture caused major difficulties? What did you learn from that?

Do you use generics? Can you describe a challenge where use of generics allowed a good, clean solution?

If you know a project is going to take 6 months, but you're told it has to be done in two months, what do you do?

(If they have a long list of technologies on their resume) I see you have listed fifty-eleven technologies on your resume. Are you expert in all of them?

No one has ever said "yes" to that, so the followup is "Can you break down the list into those you know really well and those for which you only have basic or superficial understanding?"

Suppose you were a team leader and you discovered that a team member was going out to lunch with each member of the team and telling them how bad a team leader you were. What would you do?

"Did you do that or did the team do that?"

Some interviewees like to take credit for anything in their projects, whether or not they had anything to do with it.

(Whenever you want more detail or you think they have missed an obvious point)

Can you tell me more about that?

Add your own context-dependent questions

- Languages you use
- Service-based technologies or database technologies you use

Listening tips - 1

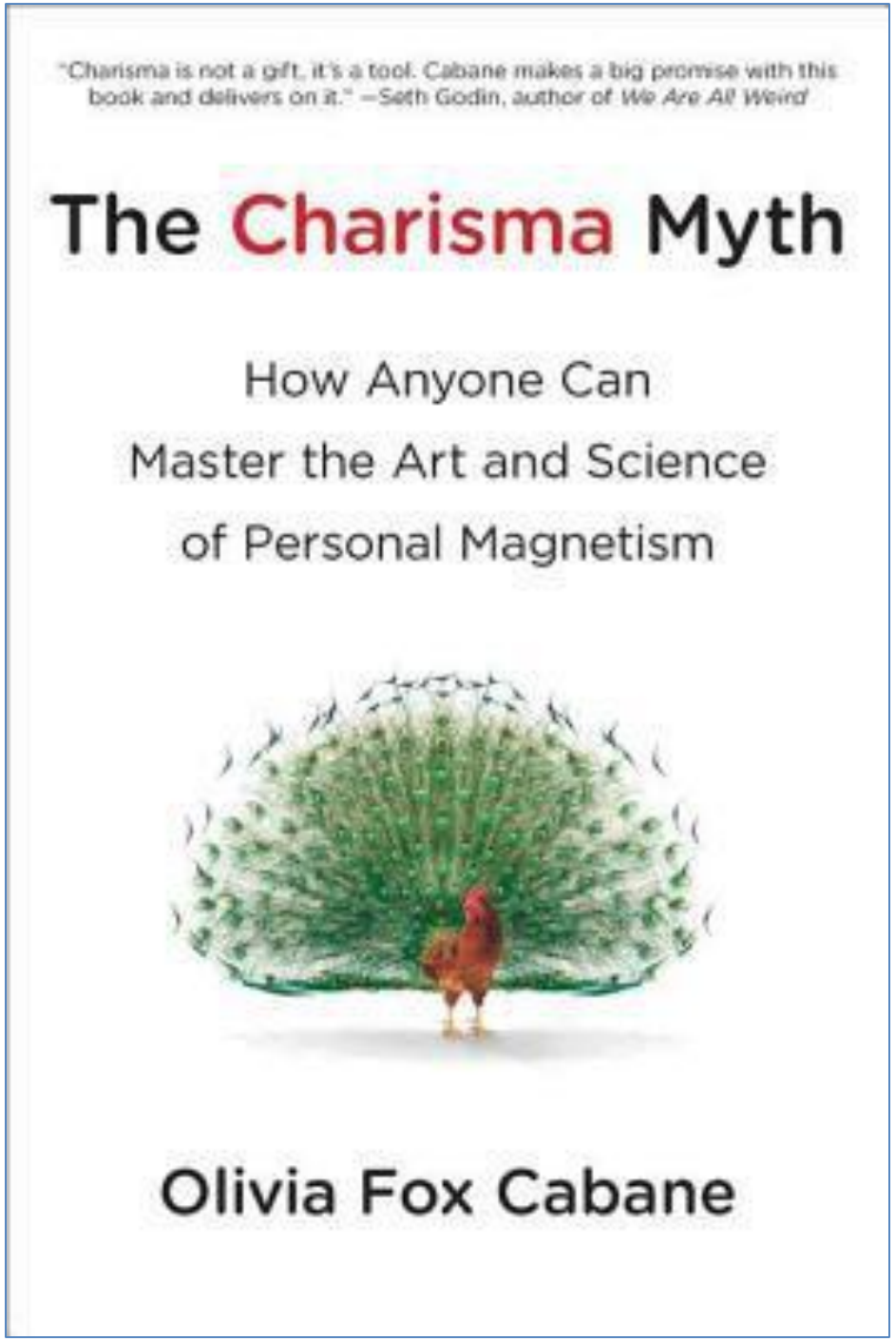
- Never interrupt
- Don’t be thrown off focus when others interrupt you
- Don’t think about how you’re going to respond while the person is still talking
- Pause before starting a response
  - Gives them more time to go into more detail
  - Makes you sound thoughtful

Listening tips 2

- Do not waste cognitive bandwidth thinking about how to refute them
- Get rid of distractions
  - Turn off your phone – even better, leave it at your desk
  - Take care of, um, physical necessities beforehand
- Be present – consciously
  - For the typical person, attention wanders every 15-30 seconds
  - Practice getting that up to three minutes
  - Relaxation exercises before an interview will help you listen better
- Avoid faking attention and pretending to listen

Practice listening – every chance you get

- With your family
- With your co-workers
- Do exercises



Wrapping it up

- Don’t forget to give them the chance to ask questions of their own
- Sometimes they are coached to ask generic questions about your organization
- I’m looking for deeper, more interesting questions about the company, or about something we discussed in the interview

Don’t wrap up too soon

- Sometimes you know in five minutes that you’re never going to hire this person
- Don’t cut off the interview at that point
  - They will feel they didn’t get a fair chance
  - They may bad-mouth you to other developers
- My minimum interview in such a case is about twenty minutes
- I end it with honest evaluation of their limitations and suggestions for improvement
  - Turns a potential bad-mouther into an ally

If you find these questions and tips useful, create a document containing them and print out a copy to remind you during the interview

After the interview

- Write down your impressions. Right now. Not after lunch. Right. Now.
- Strengths and weaknesses
- Anything that stood out about this person

Some of my personal pet-peeves

- “I’ve never had a chance to learn that.”
- “That was somebody else’s fault.”

Types to avoid

- Cowboy Coder (*Coderius Vaquerius Solus*)
  - Does everything in a maverick way
  - Usually writes code that no one else can understand
- Brute-force programmer (*Coderius Brutus*)
  - Tends towards “solution lock”
  - Will pursue brute force solution far past when it clearly isn’t optimal

Types to avoid

- World's greatest expert on programming (*Coderius Pompous Supergrediatur*)
  - Suffers ETD – Ego-Talent Discrepancy
  - Good, but not nearly as good as they think they are
- Process-obsessed (*Coderius Agilus Extremis*)
  - Emotionally attached to process
  - Will endlessly debate anyone who prefers different process

Interviewing as a team

- Two or three interviewers works well, as long as they don't whipsaw
  - More than that, split into two interviews
- Should concentrate on different things
  - Technical vs. soft skill
  - Asking and probing vs. observing body language, etc.
- If both have the same general impression at the end, that speeds up evaluation

Reference checking

- Usually done by HR, but in a small company, you might have to do it
- Be aware of legalities, and know what they are allowed to tell you vs. what they are not allowed
- My favorite question to a reference:

If this person wrote the control program for an elevator, would you ride in it?

Contacting Billy

billyhollis@live.com

billyhollis.com

- Training in UX design
- Training and mentoring in XAML and Universal Apps
- UX consulting and prototyping
- I'll come and interview candidates for you too