What It's Like To Interview For A Coding Job

First time interviewing for a tech job? Not sure what to expect? This article is for you.

Here are the usual steps:

1. First, you'll do a **non-technical phone screen**.
2. Then, you'll do one or a few **technical phone interviews**.
3. Finally, the last step is an **onsite interview**.

Some companies also throw in a **take-home code test**—sometimes before the technical phone interviews, sometimes after.

Let's walk through each of these steps.

The non-technical phone screen

This first step is a quick call with a recruiter—usually just 10–20 minutes. It's very casual.

Don't expect technical questions. The recruiter probably won't be a programmer.

The main goal is to gather info about your job search. Stuff like:

1. **Your timeline.** Do you need to sign an offer in the next week? Or are you trying to start your new job in three months?
2. **What's most important to you in your next job.** Great team? Flexible hours? Interesting technical challenges? Room to grow into a more senior role?
3. **What stuff you're most interested in working on.** Front end? Back end? Machine learning?

Be honest about all this stuff—that'll make it easier for the recruiter to get you what you want.

One exception to that rule: **If the recruiter asks you about your salary expectations on this call, best not to answer.** Just say you'd rather talk about compensation after figuring out if you and the company are a good fit. This'll put you in a better negotiating position later on.

The technical phone interview(s)

The next step is usually one or more hour-long technical phone interviews.

Your interviewer will call you on the phone or tell you to join them on Skype or Google Hangouts. Make sure you can take the interview in a quiet place with a great internet connection. **Consider grabbing a set of headphones with a good microphone or a bluetooth earpiece.** Always test your hardware beforehand!

The interviewer will want to watch you code in real time. Usually that means using a web-based code editor like Coderpad (https://coderpad.io/demo) or collabedit (http://collabedit.com/new). Run some practice problems in these tools ahead of time, to get used to them. Some companies will just ask you to share your screen through Google Hangouts or Skype.

Turn off notifications on your computer before you get started—especially if you're sharing your screen!

Technical phone interviews usually have three parts:

1. Beginning chitchat (5–10 minutes)
2. Technical challenges (30–50 minutes)
3. Your turn to ask questions (5–10 minutes)

The beginning chitchat is half just to help your relax, and half actually part of the interview. The interviewer might ask some open-ended questions like:

1. Tell me about yourself.
2. Tell me about something you've built that you're particularly proud of.
3. I see this project listed on your resume—tell me more about that.

You should be able to talk at length about the major projects listed on your resume. What went well? What didn't? How would you do things differently now?

Then come the technical challenges—the real *meat* of the interview. You'll spend *most* of the interview on this. You might get one long question, or several shorter ones.

What kind of questions can you expect? It depends.

Startups tend to ask questions aimed towards building or debugging code. ("Write a function that takes two rectangles and figures out if they overlap (/question/rectangular-love)."). They'll care more about progress than perfection.

Larger companies will want to test your general know-how of data structures and algorithms (/data-structures-and-algorithms-guide) ("Write a function that checks if a binary tree (/concept/binary-tree) is 'balanced' in $O(n)$ time."). They'll care more about how you solve and optimize a problem.

With these types of questions, the most important thing is to be *communicating with your interviewer* throughout. You'll want to "think out loud" as you work through the problem. For more info, check out our more detailed step-by-step tips for coding interviews (/coding-interview-tips).

If the role requires specific languages or frameworks, some companies will ask trivia-like questions ("In Python, what's the 'global interpreter lock'?").

After the technical questions, your interviewer will open the floor for you to ask *them* questions. Take some time before the interview to comb through the company's website. Think of a few specific questions about the company or the role. This can really make you stand out.

When you're done, they should give you a timeframe on when you'll hear about next steps. If all went well, you'll either get asked to do another phone interview, or you'll be invited to their offices for an onsite.

The onsite interview

An onsite interview happens in person, at the company's office. If you're not local, it's common for companies to pay for a flight and hotel room for you.

The onsite usually consists of 2–6 individual, one-on-one technical interviews (usually in a small conference room). Each interview will be about an hour and have the same basic form as a phone screen—technical questions, bookended by some chitchat at the beginning and a chance for you to ask questions at the end.

The major difference between onsite technical interviews and phone interviews though: **you'll be coding on a whiteboard**.

This is *awkward* at first. No autocomplete, no debugging tools, no *delete button*...ugh. The good news is, after some practice you get used to it. Before your onsite, practice writing code on a whiteboard (in a pinch, a pencil and paper are fine). Some tips:

1. **Start in the top-most left corner of the whiteboard.** This gives you the most room. You'll need more space than you think.
2. **Leave a blank line between each line as you write your code.** Makes it much easier to add things in later.
3. **Take an extra second to decide on your variable names.** Don't rush this part. It might seem like a waste of time, but using more descriptive variable names ultimately *saves* you time because it makes you less likely to get confused as you write the rest of your code.

If a technical phone interview is a sprint, an onsite is a marathon. The day can get really long. Best to keep it open—don't make other plans for the afternoon or evening.

When things go well, you' wrap-up by chatting with the CEO or some other director. This is half an interview, half the company trying to impress you. They may invite you to get drinks with the team after hours.

All told, a long day of onsite interviews could look something like this:

- 10am-12pm: two back-to-back technical interviews, each about an hour.
- 12pm-1pm: one or several engineers will take you to lunch, perhaps in the company's fancy office cafeteria.
- 1pm-4pm: three back-to-back technical interviews, each about an hour.
- 4pm-5pm: interview with the CEO or some sort of director.

- 5pm-8pm: drinks and dinner with the company

If they let you go after just a couple interviews, it's usually a sign that they're going to pass on you. That's okay—it happens!

There are are a lot of easy things you can do the day before and morning of your interview to put yourself in the best possible mindset. Check out our piece on what to do in the 24 hours before your onsite coding interview (/24-hours-before-onsite-whiteboard-coding-interview).

The take-home code test

Code tests aren't ubiquitous, but they seem to be gaining in popularity. They're far more common at startups, or places where your ability to deliver right away is more important than your ability to grow.

You'll receive a description of an app or service, a rough time constraint for writing your code, and a deadline for when to turn it in. The deadline is usually negotiable.

Here's an example problem:

> *Write a basic "To-Do" app. Unit test the core functionality. As a bonus, add a "reminders" feature. Try to spend no more than 8 hours on it, and send in what you have by Friday with a small write-up.*

Take a crack at the "bonus" features if they include any. At the very least, write up how you would implement it.

If they're hiring for people with knowledge of a particular framework, they might tell you what tech to use. Otherwise, it'll be up to you. Use what you're most comfortable with. You want this code to show you at your best.

Some places will offer to pay you for your time. It's rare, but some places will even invite you to work with them in their office for a few days, as a "trial."