Preparing your Resume

Read this article. A relevant quote:

> The number one best way to get someone to look at your resume closely: *come across as a human being, not a list of jobs and programming languages*.

Check out Gayle Laakmann McDowell's model résumé. At the bottom there's a template to make your own résumé too. Gayle Laakman McDowell is the author of Cracking the Coding Interview.

Here are some templates to get you started. **Change the font, color scheme, and layout before using them**. Employers do not like seeing similar résumés.

Here are some sample resumes other bootcampers have used:

- Ami Hays
- Dean Hu
- Edmund Li

Layout

- ONE PAGE ONLY
- Lots of whitespace
    - 1-inch margins on left and right
    -

Content

In the world of people, having a diversity of interests and experiences is a beautiful thing. In the world of tech recruiting, expertise is much more valuable. If you had various non-tech jobs before Signal, choose them carefully so you don't spread yourself thin.

You don't want to leave out large chunks of time either. In general you can include roles you held for over a year. Keep in mind that these become less relevant as you go further back in time. If your previous job was not tech-related, you need an extra good answer to anyone who asks you why you're making this transition. One of their concerns will be that you're only in it for the money.

Relevant Links

- Github
- Portfolio Site
- LinkedIn
- Personal Blog (if it’s any good)
- Twitter, Tumblr, or Quora. If you’re Internet Famous, that’s kinda cool

Skills

Don't try to organize your skills by proficiency; this makes you look less confident and desperate to throw up as many buzzwords as possible. However it's ok to do this: R, SQL, Python (learning)

Projects

This is taken from an app development guide and is presented unaltered so that you can understand the logic.

Your résumé should highlight three projects you worked on. Two projects is ok if they're both good. At four, you're leaving it up to chance which projects employers look at and which they ignore.

The three projects should be:

1. A Backbone-on-Rails app to show you know the full stack
2. A fun browser game for non-technical reviewers to appreciate (asteroids or snake, or better yet, a new game you've made that makes your resume different from other a/A grads)

3. A complex code-intensive project for technical reviewers that shows strong Object-Oriented Design skills.

The first and second **must** be deployed and available to view online. Otherwise they aren't serving their purpose. The third should include a detailed README that links to the neatest parts of the code.

Selling your Projects

Your projects are one of your biggest selling points so sell them well. First, don't make employers scan your Github profile to find each project you list on your résumé. Provide a direct link next to each project. Second, don't just list every library you used. That's boring. Only indicate the main ones like Rails and Backbone, and the unexpected ones like HTML5's Canvas. Finally, what matters the most to employers are the features you built and the problems you solved to build them. Any time you had to sit back and think about a feature before implementing it, you were solving a problem. Try to think of at least two cool solutions you came up with for each project.

To identify cool problems you solved, make a list of five neat features in your app. Step through the code that powers each feature. At every step, consider whether you could have built things differently. If looking back you find that there were actually different ways to do it, that's a sign that you solved an interesting problem.

To get you brainstorming:

- Did you find some clever way to DRY up your code?
- Did you bootstrap some data to avoid extraneous AJAX requests?
- Did you make some tough choices in your database schema?
- Did you use cookies to store anything other than a session token?
- Does your Javascript use any math to resize something in the DOM?
- Did you use a library in a way that its author probably didn't anticipate?
- Do you have any data that's nested one degree deeper than usual?
- Did you override a Rails or React.js function?
- Do your ActiveRecord models run any custom SQL queries?
- Did you make any trade-offs related to performance, eg. store information that's costly to compute?
- Are you doing any caching?
- Do you make AJAX requests to any unexpected routes?
- Does your chess game make a recursive deep_dup to validate moves without modifying the game state?

- Do your chess pieces inherit from a Slideable and Steppable class?

When answering these questions, the key is to point out features you built that went above and beyond the basic requirements. On your resume, you'll use these features as bullet points. Consider the following

Allows users to log in

vs

Uses custom-built authentication system, hashing and salting passwords with BCrypt.

In the first example, the quality of the login process is ambiguous. The second example describes a specific feature that makes the site sound more intesting, and suggests that the designer is a capable developer.

Caution:

- Students often use the past tense to describe each solution, eg. "Overrode Backbone's default model parsing method to..." It's less exciting if it's in the past. Use the present tense in the third person (ie. your project itself). "Overrides Backbone's default model parsing method to..."
- Look up the proper spelling and capitalization for the technologies you mention.

Second caution:

- Students often want to be extremely language or framework-specific with how they implemented something. Try not to do to this. Example: "Overwrites attr_writer on User#password". This makes sense to you because you know Rails, but you might not be applying to a Rails job (or the recruiter might not know anything about Rails). Your bullet is going to be incomprehensible to them.
- Instead of using language-specific details, talk at a higher-level about what engineering solutions you implemented. It shows you know how to communicate about what you really accomplished as an engineer, rather than the equivalent of reciting a stack trace.

Third caution:

- Resist the urge to just enumerate features in your bullet points! Your bullet points should be about cool things you did in the project, not features of the application. We see this mistake **all**

the time. A good resume item does *not* read like a feature list on a product's website. Your bullet points should not tell me all the features of your web app.
- Describe to me instead the *engineering problems you solved* or the solutions you found in *implementing* those features.**I want to hear about what kind of an engineer you are, not buy your product.**

More example bullet points to help with brainstorming:

- Achieves better performance by leveraging Cloudinary to cache different sizes of images.
- Original UI design, with sliding index items and toggle buttons that react to mouse hovers.
- Sensitive search functionality listens for searchbar keydown and responds automatically.
- Integrates Google Maps API with geolocation based searching to display location of other users on a map.
- Implements validation logic that identifies reservations with overlapping dates and prevents double bookings.
- Uses observer pattern to handle management of global Z-indexes for overlapping items.
- Implements custom infinite-pagination with prefetching to improve perceived load time.
- JBuilder data serialization retrieves nested data from associated database entries.

Work

The difference between a compelling work history and a boring one is whether you have identified the *results* of your work.

Lucky for you most work histories out there are in the boring category. Candidates merely go through past jobs and list the responsibilities they had at each one. That's basically useless because those responsibilities only make sense to someone who knows how the company actually worked. Your reader cares about success, not process.

In other words employers care about the outcomes of what you did at past jobs. Someone who can talk about outcomes is someone who a) made a difference at their job, however small, b) cares about results, and therefore c) is likely to produce results at their next job.

Results don't have to be achievements. You don't have to oversee a company turnaround or be named employee of the month for a year straight to earn a selling point. Simply describing something that sounds like success is a big plus. Consider the following blurbs.

Wrote scripts to test new product before shipping.

vs.

Wrote scripts to test new product before shipping. Achieved one test per feature, cutting the number of bugs to 3.

vs.

Wrote scripts to test new product before shipping using a custom implementation of Cucumber. Reduced the number of bugs which required patching from an average of 16 in previous projects of a similar scope to 3. This also saved 100 developer hours and …

The first is meh. The second is good. The third is best.

Note: This takes substantially more space to do it this way, which means you may have to cut other material. Two longer bullet points that convey success are much stronger than four bullet points that only describe process (no matter what portion of your previous job you're leaving out).

Second note: If you did data science-like technical things, talk about it. Whether that be solving problems on your own, making business plans, programming, whatever. Technical experience helps smooth out the narrative of your transition into data science.

Education

Signal

Something a lot of people have asked is “should I talk about Signal.” My initial answer was to shy away from doing so, but over time I’ve shifted towards using it as a selling point.

The case where you might not want to put Signal is where you’re trying to get a senior data scientist job. This only really makes sense for somebody who already has a strong career (most likely as a developer) and wants to transition roles. In this case saying that you took a bootcamp risks typecasting you as inexperienced.

Needless to say, if you weren't making ~100,000 before Signal you're safe including it. In fact you probably should. I would recommend describing it as a full-stack data science program designed to teach technically skilled people data science. Feel free to play around with this a bit.

College

If you took data science related courses in college, you can add a "Curriculum Highlights" section under the name of your degree and list those courses. Linear algebra, statistics, and computer science can also be included. This is not super strong; you can include it or not

Include your GPA if you graduated 5 years ago or less and if it's above 3.2. If you studied a STEM subject or Econ, you can use your Major GPA if it is higher. (Note on GPAs: most employers won't care about them, but a few will. This is just for them.)

If you have neither a college degree nor relevant work experience

First, my approach. Be open about it (and maybe just a little bit proud), and be able explain your decision process: why the things you did other than college were better than going to college (the default option for your interviewers).

If you had medical problems that's fine. Present it as "I had X that happened that caused me to fail and then I got it treated." You get a free bye for Shit Happening To You as long as it's not a continuing risk.

Now, another approach. The key in this case is to make a strong personal impression. You don't want the reader to come away thinking "OK they can math. But I have no idea what else to expect from this person." You have to be creative.

- Have you ever done volunteer work?
- Do you have some interesting hobbies that would make you stand out?

- It'll also be important to write enthusiastic cover letters that show an interest in what the company does.

You could also mention something like "Left college in good academic standing for financial reasons / to pursue data science" if this is true

Checklist

This checklist is intended for a web developer, but is presented here unaltered as a guide to strategy. Before you submit your résumé for review, make sure you can answer yes to all these questions.

- Layout
    i. Do your headers take the following order? Projects, Skills, Education, Work History
    ii. (List work history first if you were paid to write code.)
    iii. Fits on 1 page?
    iv. 1-inch margins on left and right?
- Projects
    i. Do you list three or four coding projects, including one Rails app and one browser game?
    ii. Does each project include a link to the Github repo and one to the live demo (if applicable)?
    iii. Do you describe clever solutions you came up with to build each project?
    iv. Are those solutions described in the third person, present tense?
- Work
    i. Do you describe not just *what* you did at each job but *how well* you did it?
    ii. Do you provide benchmarks for your own or your teams success?
- Misc
    i. Do you provide a simple list of the technologies you are familiar with?
    ii. Do you provide contact information and link to your Github and LinkedIn?
    iii. Does your contact information make it clear you are currently in the metro area?
    iv. Is it a Google Doc that both I and your pairboarding partners can edit and comment on? (Make sure to share it to us)
    v. No timeline gaps or minimize as much as possible