1 Before the Search

Before starting your job search, you need to prepare yourself. You shouldn't apply for jobs without knowing what kind of job you want. Just being a good coder isn't enough; you must understand what the market wants and how you can improve and package your own skills to make sure that the company with the job you want will want you.

KNOW YOURSELF

Stereotypes to the contrary, all programmers are *not* alike. Knowing what kind of programmer you are is crucial to finding the right kind of job. Although you can probably do many different kinds of programming tasks, you probably don't find them all equally engaging. Doing something you don't enjoy is fine on a short-term basis, but you need to be interested in and excited by what you're doing for it to sustain you over the long term. The best programmers are passionate about their work, and you can't truly be passionate about something that's only moderately interesting to you.

If you're not sure what you like or dislike, ask yourself some questions:

- **Are you a systems programmer or an application developer?** Systems programmers work on the code that keeps computer systems running: frameworks, tools, compilers, drivers, servers, and so on. Other programmers are their primary audience, so little interaction occurs with nonprogrammers—and usually the job involves little or no user interface work. Application developers, on the other hand, work on the pieces that those nonprogrammers use to do their own work, and often more interaction occurs with nontechnical people. Many programmers find interacting with nontechnical people about technical topics to be frustrating; on the other hand, you may enjoy creating applications that are seen and used by an audience that extends beyond other programmers.
- **Do you like coding user interfaces?** User interface design—also referred to as *user experience (UX)* or *human computer interaction* (*HCI*)—is a role that draws on a diverse set of skills, including programming, graphic design, and psychology. This work is high profile because the user interface is the most visible part of any application.

User interface design is particularly important in mobile application development, where the restrictions of the device require even greater creativity and innovation. If you have the necessary skills and enjoy this work, you're in elite company: many programmers find it finicky, hard to do well, and easy to criticize, especially when you take internationalization and accessibility issues into account.

- **Are you a good debugger?** If you think finding problems in your own code is difficult, imagine what it's like to fix problems with someone else's code. It requires strong analytical and problem-solving skills. Finding and fixing bugs can be extremely rewarding in its own right. You need to know if you'd be happy doing primarily maintenance work. (Of course, you should always expect to maintain your own code—all programmers need debugging skills.) In many cases, particularly in older companies, maintenance programming jobs involve working primarily with older technologies now considered outdated or no longer in fashion. Developing your experience and skills with older technologies may narrow the range of jobs that you're suited for, but because expertise in older technologies is hard to find, you may be highly sought after by the smaller number of companies dependent on older programs.
- **Do you like testing?** Testing—also referred to as *quality assurance* or QA for short—requires a combination of meticulous attention to detail to ensure that tests cover every conceivable use of a program and outside-the-box creativity to find bugs in the program by generating combinations of inputs that the program's developers never considered. Skilled testers are hard to find, and good programming skills are required to write tools and automated test cases.
- **Are you an architect or a coder?** Every coding job includes some kind of design aspect, but certain jobs lean more one way than the other. If you enjoy designing, particularly designing the large-scale structure of big projects, a position as a software architect might be more appealing than a coding-focused job. Although you need a good understanding of how to code to be an effective architect, architecture positions can involve a lot of meetings and interpersonal interactions and little or no coding. Unless you have formal training in software architecture, the usual route to becoming an architect is to code first and to then display an aptitude for designing and fitting together different pieces of a project.

The preceding questions deal with different kinds of programming, but you should also consider nonprogramming responsibilities that might interest you and the work environment that you prefer:

- **Does management interest you?** Some coders have a long-term goal to become a manager, but others shiver at the very thought. If management is your goal, you need to develop leadership skills and demonstrate that you can manage the human parts of the software development equation as well as the technical pieces. If management is *not* your goal, look for companies with good *technical* career paths, so you're not forced to manage people to be promoted. (You still need leadership skills to get promoted no matter which career path you choose, but leadership skills are separate from people-management skills.)
- **Do you want to work for a big company?** Working at big companies has advantages and disadvantages. For example, a large company may offer more job stability (although layoffs during downturns are common) and some kind of career path. It may also have a name brand that nontechies recognize. On the other hand, you may feel stifled by the bureaucracy, rigidness, and intracompany rivalry often found in bigger companies.

- **Do you want to work for a small company?** The pay may be less, but getting in on the ground floor at a new company can create opportunities for future advancement (and possibly substantial remuneration) as the company grows and succeeds. Also, the work environment at small companies is often more informal than at larger organizations. The downside, of course, is that most new ventures fail, and you may be out of a job within a year or two, most likely without the kind of severance package you might expect from a large company.
- **Do you want to work on open source projects?** The vast majority of programming jobs have historically involved proprietary, closed source projects, which some programmers don't like. A shift has occurred in some companies in favor of more open software development, which provides opportunities for people to work on open source projects and still be paid for that participation. If it's important to you that your work project is open source, it's best to seek out companies already involved in open source. Trying to champion open source in traditional software companies is often a frustrating and fruitless undertaking.
- **Do you want long-term or short-term projects?** Some programmers crave change, spending a few months at most on each project. If you like short-term projects and don't mind traveling, a gig with a consulting company might make more sense than a more conventional corporate job.

Realize that these questions have no universal answers, and no right or wrong way to answer them. The more truthful you are in answering them, the more likely you'll find the kind of programming job you truly enjoy.

KNOW THE MARKET

Knowing what you'd like to do is great, but don't box yourself in too narrowly. You also need to understand the current job market and how it constrains your search for the "ideal" job, especially during an economic downturn like the one that burst the Internet bubble of the late '90s or the global real estate and banking meltdown of the late 2000s.

Basic Market Information

A number of sources of information exist about what's hot and what's not in the developer job market, including the following:

- **Social networks.** The tremendous growth of social networks such as LinkedIn and Facebook has transformed social networks into virtual recruiting grounds for all types and sizes of organizations. LinkedIn is particularly important. The other social networks can provide an indirect "pulse" of the market and also valuable leads for new and even unannounced job postings.
- **Online job sites.** Visit two kinds of job sites as part of your research. Job listing sites such as Dice (which specializes in technology-related career listings), Indeed, and Monster (general job listing sites) enable you to see what kinds of jobs are currently in demand. Review sites such as Glassdoor discuss working conditions, salaries, bonuses, perks, and other information useful for finding the right kind of company for you.

- **Bookstores.** Even though more and more programmer documentation is available online, professionally published books are still important, whether printed or downloadable. The number of books published on any given topic is a good indication of the level of interest the programming community has in that topic. Look especially for niche topics that are suddenly going mainstream, but beware that in most companies, mainstream use of technologies lags the interest levels represented in books by a few years.
- **Professional development courses.** Colleges and universities try to keep abreast of what companies want and create professional development courses around those needs.

If you're not in college or university, find out what languages and technologies the local institutions and your alma mater require of their computer science students; although academic needs don't always coincide with what employers want, educational institutions try to graduate students with practical skills that employers can use.

What About Outsourcing?

Outsourcing and *offshoring*—contracting tasks to other companies or foreign divisions or companies—is an important part of the technical employment landscape. Outsourcing of ancillary business activities such as payroll administration and property maintenance has been around for decades. More recently, this has expanded to programming, driven by the advent of inexpensive computers, cheap long-distance communication provided by the Internet, and the recognition of technically educated workforces in low-wage developing countries. There was a flurry of outsourcing, particularly offshoring, in the mid-2000s. This has become less topical in the past several years because most companies that intend to outsource have already outsourced whatever they can. In addition, the costs of offshoring have risen as wages rise in the developing world, particularly in India and China. This coupled with recognition of the hidden costs of coordination with workforces from different cultures on very different schedules have led some companies to insource roles they previously outsourced. Nevertheless, outsourcing and offshoring remain a possibility for expanding companies that think they may cut costs, as well as established companies wondering if they're paying too much by keeping their work local.

If outsourcing (and offshoring in particular) is something that worries you, consider taking steps to avoid landing a job that might be outsourced at some point in the future. The following are some suggestions:

- **Work for software development firms.** A software firm's *raison d'être* is the intellectual property it develops. Although medium and large firms may open development centers in other parts of the world, the smart ones are unlikely to move their entire operations to other countries or entrust their future to outside firms. That said, some companies outsource all or substantial parts of a project to other countries for cost or other reasons, so it pays to research a company's behaviors and policies.
- **Work for an outsourcer.** Oddly enough, many outsourcing firms hire personnel in countries such as the United States.
- **Move up the programmer food chain.** Design-oriented jobs are less likely to be outsourced. Coders are relatively cheap and plentiful, but good designers are much harder to find.

(This assumes that your company recognizes that good design skills are separate from good coding skills.) Another way to make yourself more difficult to replace is to acquire *domain specific knowledge*: expertise related to the programs you write but outside of the field of programming. For example, if you develop financial software, it's much more difficult to outsource your job if it involves the application of accounting skills in addition to programming than if you're purely a coder.

- **Take a management job.** Management can be a refuge from outsourcing, so a management-oriented career path is one option to consider.

Of all these options, moving up the food chain is usually the best approach. The more nonprogramming knowledge your job requires, or the more interaction with customers, the less likely you are to be outsourced. There's no *guarantee* you'll never be outsourced, of course, or that you'll always keep your job. Your company may shutter or downsize the project you're working on at any point, after all, and put you back on the street. This is why developing reusable and marketable skills throughout your career is extremely important.

DEVELOP MARKETABLE SKILLS

In the appendix we discuss your résumé as a *marketing tool* to get you job interviews. The easiest thing to sell is something that people want, so it's important that you have *marketable skills* to offer a prospective employer.

To stand out from the crowd both on paper and in the interviews you need to develop skills and accomplishments, especially if you're entering the job market for the first time. The following are some approaches you can take:

- **Upgrade your credentials.** Companies such as Google are well known for favoring job applicants with graduate degrees. Getting a master's or doctorate degree is one way to upgrade your credentials. You can upgrade your credentials in other ways, such as taking university or professional development courses or participating in programming contests.
- **Don't bother with certifications.** The authors of this book believe that programming certificates are of limited value because very few such jobs require certification. Additionally, almost none of the programmers at the top employers have formal programming certificates, and these are the people interviewing/evaluating you. Instead of spending time getting one of the certifications, we suggest working on some of the other recommendations here (side projects, school).
- **Work on a side project.** A great way to expand your skill set is to work on a project not directly related to your primary work or study focus. Starting or joining an open source development project is one way to go. Or if you work at a company, see if it will let you spend time on an ancillary project.
- **Do well in school.** Although grades aren't everything, they are one measure that companies use to rank new graduates with little job experience. The better your grades, especially in computer science and mathematics courses, the more you can impress a potential employer.

- **Keep learning.** The end of formal education doesn't mean you should stop learning, especially when so much information about programming is available from a wide variety of sources. Whether it's books or blogs, there's always a way to keep current, no matter what type of programming you do. It's also a great way to expand your horizons and discover other areas of interest. This kind of learning doesn't show up on your résumé, but it's something you can highlight in your technical interviews.
- **Be an intern.** New graduates who manage to secure employment during their nonschool terms—especially those who participate in cooperative education programs—have a huge advantage over their peers who haven't yet ventured into the real world. Software development in the field is often different from software development in an academic setting, and potential employers are cognizant of this.
- **Use code contest sites.** TopCoder, HackerRank, CodeWars, and several similar sites have developers "face-off" to solve programming problems. If you win, meaning you solve the problem faster than the competitor or bot, you move up the leaderboards and get a higher rank, which you can make public and list on your résumé. And, even if you lose, these are fantastic practice for programming interviews. Most of these sites' revenue models revolve around sourcing candidates and making recruiter fees, so if you do well, you may find some serious job offers coming your way. Some companies try to cut out the middleman with hidden coding contests that you may be invited to participate in based on your activity on their website, such as searching for programming-related topics.

The key is to *keep learning*, no matter the stage of your career. You can't develop marketable skills overnight; they take some effort and initiative on your part but can have long-lasting effects on your career.

GET THINGS DONE

Companies look for software developers who *get things done*. You may look great on paper in terms of skills and education, but credentials and knowledge don't make products or services that a company can sell. It's your ability to *accomplish something* that truly sets you apart from the other candidates.

Getting an advanced degree such as a Ph.D., becoming a trusted contributor to a widely used open source project, or carrying a product through from start to launch are all big accomplishments. But small accomplishments can be just as important, such as adding a feature to a product, making a measurable improvement to the product's performance, starting and completing a side project, or creating a useful application for a class project. These all show that you can get things done.

Recruiters and hiring committees like to see that you have multiple accomplishments—a pattern of getting things done. This is especially true for more senior and experienced developers. You need to show those accomplishments on your résumé and your online profile. Whether your accomplishments are big or small, always be ready to talk intelligently and confidently about each one. This is incredibly important! Make sure you can clearly and succinctly describe the underlying problem and how your project solved it, even to a nontechnical person. Displaying a passion for programming is always positive; clearly communicating how your passion produces products and services that other people can use makes you really stand out from the other candidates.

MANAGE YOUR ONLINE PROFILE

Your online profile—everything public about you online—is just as important as your résumé. Recruiters use online profiles to find desirable candidates. Screeners use them to weed out undesirable applicants, and interviewers use them to prepare in-depth interview questions.

An online profile consists of any or all these things:

- **Google search results for your name.** This is the first impression you make on potential employers and colleagues.
- **LinkedIn profile.** LinkedIn is a social network for tracking professional connections. It's free to join, and you can create a detailed profile about yourself, including your jobs and your education—essentially an online résumé. Colleagues and customers can publicly endorse you or your work, which can be quite valuable.
- **GitHub profile.** Many employers will evaluate your work through your GitHub profile well before they meet you. Take a look and spend a few hours to clean up your GitHub profile so it reflects your best code. Delete or change the privacy settings for incomplete, poorly organized, or low-quality repositories. Assume your profile may be checked even before you're asked about it. If you don't have very much public, move your best code to public so it's clear that you have significant code and experience with GitHub.
- **Stack Overflow.** This will show up on your Google search result, or may be checked. If you have recent, basic questions that reflect poorly on your knowledge, you may want to delete them. If you don't have a profile, you should make one, especially if your job search is still a few months out and you have time to answer other people's questions.
- **Angel investor sites.** These sites don't just connect investors with startups, they also connect startups with potential hires. AngelList is the big player in this category. Create a profile that reflects your interests and expertise.
- **Other social network profiles.** Other social networks such as Facebook, Twitter, or Snapchat may be reviewed, depending on your privacy settings. Make sure you clean up and tighten your profile so nothing unprofessional appears public.
- **Personal website.** This is a potential source of more in-depth information about you and topics you find interesting. If you blog about political or controversial topics, you may want to remove such posts during your job search.
- **Articles and blog posts.** If you write about programming-related topics, this is a good way for recruiters to assess your experience.
- **Comments and forum posts.** These provide another way to gain some insight into your programming skills and your general attitude toward technology and technology companies.

The impression employers get from your online profile will affect your chances of being hired. If your résumé lists extensive experience with C# but they find a forum posting you made only 6 months ago asking how to open a file in C#, they'll probably conclude that you're exaggerating your experience level, putting your whole résumé into doubt. Or if they see disturbing or inflammatory material that they think you've authored, they may decide to pass you over for an interview, no matter how well your résumé reads or how long ago you wrote those things. No one's proud of everything they ever did in high school or college, but those who have grown up in the post-Internet era see things follow them that they'd rather forget about, something the older generations rarely had to contend with.

At some point before you apply for a job, take a good look at your online profile. Put yourself in a company's shoes to see how much information—good or bad—it can find about you, or link to you. If your online profile is possibly going to prevent you from being hired, take some steps to sanitize your profile. If possible, remove questionable material from the web and from the search engines.

Spend some time developing the positive aspects of your profile. This is particularly important if there's unfavorable material about you on the web that you're unable to remove. You may want to read a little about search engine optimization (SEO) and apply some of these techniques to get the positive aspects of your profile to appear before older, less favorable items in search results.

Finally, you may have access to other profile-featuring online resources that can be very helpful. Most universities have job sites where alumni can upload profiles; a few companies have similar sites for former employees.

> **WARNING** *One caveat about updating your LinkedIn profile: by default, all your contacts are notified of your updates. Many people have learned to interpret these notifications as de facto announcements that someone is looking for a new job. That might help you get the word out, but if your contacts include people at your current company and you don't want them to know you're looking for a new job, disable these notifications before you make your updates.*

Develop an online profile that doesn't have any red flags and shows you in the best possible light. Finding a good job is hard enough—why make it harder?

SUMMARY

What you do *before* a formal job search is critical to finding the right kind of job. With that in mind, you should consider the following things:

- ➤ Know your likes and dislikes as a programmer and a prospective employee.
- ➤ Understand the market to find and apply for the best jobs.
- ➤ Develop the marketable skills that employers look for and that can enhance your career.
- ➤ Manage your public profile to show you in the best possible light and make sure there are no surprises to turn off potential employers.

Once you've worked through all these points, you're ready to begin your job search.