Making Everything Easier!™

Getting a Coding Job

FOR DUMMIES®

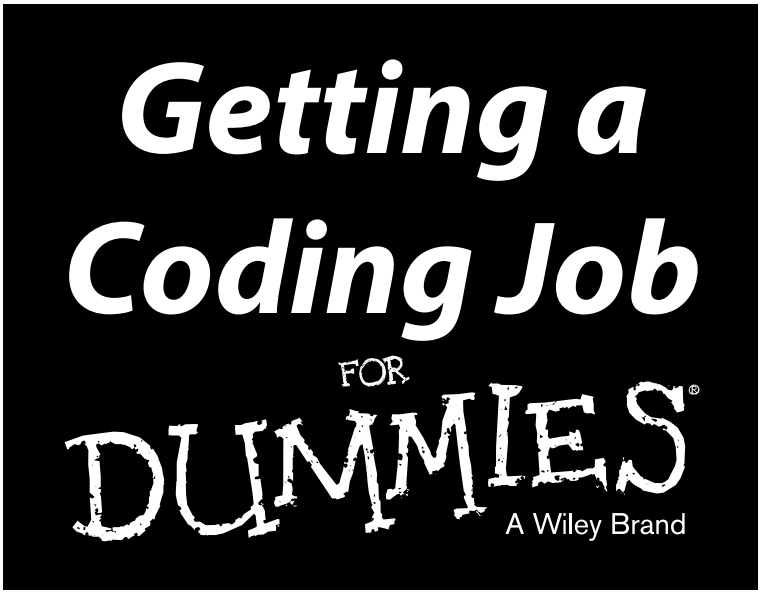
A Wiley Brand

Learn to:

- Choose the education option that best suits your needs
- Identify the skills you need to become a can't-miss candidate
- Create a portfolio site that impresses interviewers

Get access to free online resources, including videos, articles, sample resumes, and more

Nikhil Abraham
with Kathleen Taylor and Bud E. Smith



by Nikhil Abraham
with Kathleen Taylor and Bud E. Smith

FOR
DUMMIES®
A Wiley Brand

Getting a Coding Job For Dummies®

Published by: **John Wiley & Sons, Inc.,** 111 River Street, Hoboken, NJ 07030-5774, www.wiley.com

Published simultaneously in Canada

Library of Congress Control Number: 2015941960

ISBN 978-1-119-05094-0 (pbk); ISBN 978-1-119-12101-5 (ebk); ISBN 978-1-119-12102-2 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

Table of Contents

Introduction

Everywhere you turn, people are looking for coders. In offices and boardrooms, at your neighborhood bar, and around the family table, people have ideas wanting to become websites, data needing to be analyzed, and processes waiting to turn into a mobile app. Building a product requires many people — including designers, product managers, marketers, and content creators — but finding coders is always at the top of everyone's list because they are so scarce.

On the supply side of the equation, learning to code and then getting a job can feel overwhelming. However, there have never been more ways to learn how to code, including on your own, in school, at a coding boot camp, and on the job. And companies of every size and type are hiring developers.

Getting a Coding Job For Dummies will help you make sense of all the options and show you ways to get that first coding job.

About This Book

This book is designed for the person with little to no experience with coding jobs. In plain English, you discover why coding jobs are so popular, which technologies to use when coding, ways to learn coding, and how to launch your career. The topics covered include the following:

- How coding became such a hot topic and big industry
- Types of coding jobs
- Options for learning to code, including coding boot camps
- Coding technologies used to build websites, analyze data, and create mobile apps
- Building a portfolio and a network
- Interviewing your way into your first coding job

As you read the book, keep the following in mind:

- Skip around as much as you like. The book can be read from beginning to end, but if a topic interests you, start there.
- At some point, you will have questions or something will not make sense. Do not fear! Many resources are available to help, including support forums, free tutorial websites, others on the Internet, and me! Using Twitter, you can send a public message to me at @nikhilgabraham.

Foolish Assumptions

I do not make many assumptions about you, the reader, but I do make a few.

You do not need to have previous programming experience. In this regard, you need to be able to read, type, and follow directions. I explain as many concepts as possible by using examples and analogies you already know.

Before trying to get a coding job, you will spend some time learning how to code. Chapter 5 shows you some basic code examples, and Part III outlines options and resources for learning how to code in greater depth. If you don't have any coding knowledge, keep in mind that it will take at least a few months to learn enough to be able to get a coding job.

You'll need a computer running the latest version of Google Chrome if you want to complete the coding examples. Chrome is a free browser and the examples in the book and in the external resources have been tested and optimized for the Chrome browser, although they may also work in latest version of Firefox. Using Internet Explorer when learning to code is discouraged because its support for coding languages varies and it doesn't always work as expected.

I assume that you have access to an Internet connection. You can read almost all the book without an Internet connection, but you need an Internet connection to access external learn-to-code resources, such as the Codecademy website. Many listed resources are free and can be used without downloading or installing anything.

Icons Used in This Book

Here are the icons used in the book to flag text that should be given extra attention or that can be skipped.

This icon indicates useful information or explains a shortcut to help you understand a concept.

This icon explains technical details about the concept being explained. The details might be informative or interesting but are not essential to your understanding of the concept at this stage.

This icon marks a concept that likely has been explained before. It's flagged to reinforce what you've already learned.

Watch out! This icon indicates common mistakes and problems that can be avoided if you heed the warning.

Beyond the Book

Online resources are available in addition to the ones in this book:

- **Cheat sheet:** Visit `www.dummies.com/cheatsheet/gettingacodingjob` for tips while job searching and during your interviews.
- **Extras:** Additional articles with extra content are posted for roughly each section of the book. You can access this additional material by visiting `www.dummies.com/extras/gettingacodingjob`.
- **Updates:** You can find any updates or corrections by visiting `www.dummies.com/extras/gettingacodingjob`.

Where to Go from Here

With all the administrative stuff out of the way, it's time to get started. Remember, you can start at the beginning or jump to whatever section interests you the most. Congratulations on taking your first step to getting a coding job!

Part I

Getting a Job in Coding

Check out www.dummies.com/extras/gettingacodingjob for more great content online.

In this part . . .

- Understand why coding matters
- Explore coding career paths
- Follow a coder on the job
- Learn key coding concepts

Chapter 1

Seeing the Big Picture

In This Chapter

- Seeing the history of coding and where it's headed
- Understanding different types of coding jobs and salaries
- Learning about companies that hire coders

If you just focus on the smallest details, you never get the big picture right.

—Leroy Hood

Today, many moments in your daily life are affected by code. Code runs the mobile phone alarm that wakes you up in the morning, the word processing and spreadsheet software you use at work or in school to create letters or projections, the games you play on a phone or console, and the web browser you run to check your email and read the news. Many tasks in our lives have remained the same — there will always be people who need help waking up in the morning — but technology is increasingly influencing the way we complete these tasks.

Because you're reading this book, you understand coding's pervasiveness, but you may wonder about the industry's size and future. Is getting a coding job like becoming a horse and buggy driver just as Ford was starting to sell the Model T?

In this chapter, you learn where coding came from, how fast it has grown, and what the future might hold for those who can code. Additionally, you'll see the types of companies that hire coders and find out what recruiting professionals look for when hiring coders.

What Is Coding?

Computer code consists of a set of statements (like sentences in English); each statement directs the computer to perform a single step or instruction. Each step is precise and followed to the letter. For example, if you're in a restaurant and ask a waiter to direct you to the restroom, he might say, "head to the back, and try the middle door." To a computer, these directions are vague and therefore unusable. Instead, if the waiter gave instructions to you as if you were a computer program, he might say, "From this table, walk northeast for 40 paces. Then turn right 90 degrees, walk 5 paces, turn left 90 degrees, and walk 5 paces. Open the door directly in front of you, and enter the restroom."

One rough way to measure a program's complexity is to count its statements or lines of code. Basic applications such as Pong have 5,000 lines of code, while more complex applications such as Facebook currently have over 10 million lines of code. Whether few or many lines of code, the computer follows each instruction exactly and effortlessly, never tiring like the waiter might when asked for the 100th time for the location of the restroom.

Figure 1-1 shows lines of code from the popular game Pong. Don't worry about trying to understand what every single line does.

Be careful when using the number of lines of code as a measure of a program's complexity. Just like when writing in English, 100 well-written lines of code can perform the same functionality as 1,000 poorly written lines of code.

This book describes the ins and outs of careers in coding but will not teach you a programming language. In Part III, you can read about the different ways you can learn to code: by yourself, in a coding boot camp, in college, and on the job.

```
1
2  launchPong(function () {
3      function colour_random() {
4          var num = Math.floor(Math.random() * Math.pow(2, 24));
5          return '#' + ('00000' + num.toString(16)).substr(-6);
6      }
7
8
9      pongSettings.ball.size = 15;
10     pongSettings.ball.color = colour_random();
11     pongSettings.ball.velocity[0] = 15;
12     pongSettings.ball.velocity[1] = 15;
13
14 });
```

Figure 1-1: Computer code from the game Pong.

Why Coding Matters: Past, Present, Future

Today, programs written with code power so many different activities, and the work they do can almost seem like magic. With a few mouse clicks or finger taps, you can see your current location on a map, have groceries delivered to your door, or video chat with someone in another country. Although the research and development to make these advancements possible has been massive — billions of dollars invested and millions of hours worked — it has been worthwhile. In this section, I briefly describe a history of code and possibilities for the future.

Coding in the past

Unveiled in 1946 at the University of Pennsylvania, ENIAC was the first general-purpose computer. See Figure 1-2. It was the size of a large room, and programmers punched holes in paper cards to code programs that could take hours to complete. Sometimes bugs would crawl inside these large computers, causing the circuits to malfunction and resulting in errors. Removing these bugs from the computer was called *debugging,* which is the name used even today.

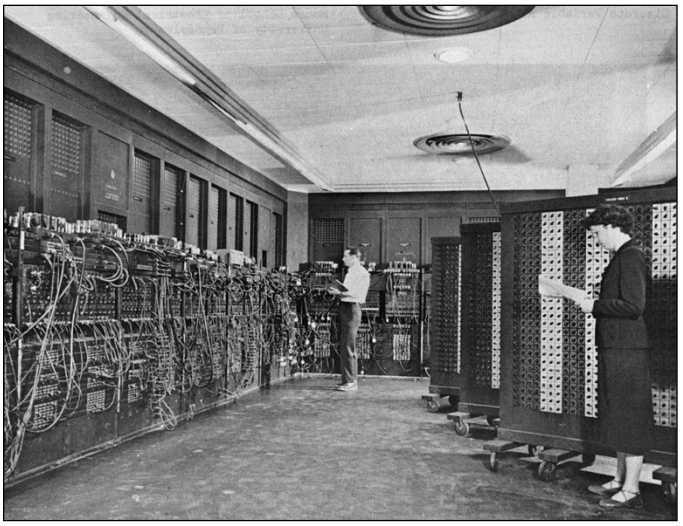


Figure 1-2: ENIAC was the size of a large room.

Gradually, with advances in hardware, computers became smaller and more powerful. Whereas the ENIAC's tens of thousands of resistors and capacitors took up almost 2,000 square feet, later microprocessors could fit all these electronics onto a chip the size of a postage stamp. Eventually, these microprocessors would be built using silicon, which is both cheap and plentiful.

Increased computing power from powerful microprocessors allowed programmers to write more complicated and resource-intensive programs. For example, computer games became faster, used more complex graphics, and displayed on-screen smoothly and realistically. Writing code, or software programming, depends on and is constrained by the underlying hardware on which the code runs. As computing power increases, code is written to provide more features at a faster speed to users.

Programming languages were also invented to take advantage of this new computing power. You may remember languages such as Basic, Fortran, Pascal, C++, and Java. Like spoken languages, programming languages were created to fill a need. If other programmers coded using the language, the programming language would survive and thrive; otherwise, it would die.

Popular programming languages can decline in popularity or die, but this can take a long time if the language is used for core processes. For example, Fortran is not nearly as popular as it was 30 years ago, but it continues to be used in the scientific community and in the financial sector, where it powers applications for some of the biggest banks in the world.

Coding today

In 2011, Marc Andreessen, creator of Netscape Navigator and now a venture capitalist, noted that "software is eating the world." He predicted that software companies would rapidly disrupt existing companies. Traditionally, software was used on desktops and laptops. The software had to be installed, and the installation process at a minimum varied by computer type and might not even work or might be incompatible with your computer hardware and software. After the software was installed, you had to supply data to the program.

Four trends have dramatically increased the use of code in everyday life:

- **Web-based software:** This software operates in the browser without requiring installation. For example, if you want to check email, you previously had to install an email client by downloading the software or from a CD-ROM. Issues arose when the software was not available for your operating system or conflicted with your operating system

version. Hotmail, a web-based email client, rose to popularity in part because it allowed users visiting `www.hotmail.com` to instantly check email without worrying about installation or software compatibility. Web applications increased consumer appetite to try more applications, and developers in turn were incentivized to write more applications.

- **Internet broadband connectivity:** Broadband connectivity has increased, providing a fast Internet connection to more people in the last few years than in the previous decade. Today, more than 2 billion people can access web-based software, up from approximately 50 million only a decade ago.
- **Coding repositories:** Anyone can publish code for others to view and use. Popular coding repositories, such as Github, are making coding a more collaborative, open, and public process than ever before. Programmers publish code to show others what they can build, to solicit feedback to increase functionality or find vulnerabilities, and to quickly spread software to other programmers.
- **Mobile phones:** Today's smartphones bring programs with you wherever you go and help supply data to programs. Many software programs became more useful when accessed on the go than when limited to a desktop computer. For instance, the use of maps apps greatly increased thanks to mobile phones because users need directions the most when lost not just when at home on the computer planning a trip. In addition, mobile phones are equipped with sensors that measure and supply data such as orientation, acceleration, and current location through GPS. Now instead of having to input all the data to programs yourself, mobile devices can help. For instance, a fitness application such as RunKeeper automatically tracks your distance, speed, and time.

The combination of these trends has resulted in software companies that have upended incumbents in almost every industry, especially ones typically immune to technology. Some notable examples include the following:

- **Airbnb:** A peer-to-peer lodging company that owns no rooms, yet books more nights than the Hilton and Intercontinental, the largest hotel chain in the world. See Figure 1-3.
- **Uber:** A car transportation company that owns no vehicles but books more trips and has more drivers in 200 cities than any other car or taxi service.
- **Groupon:** A daily deals company that generated almost $1 billion after just two years in business, growing faster than any other company in history, let alone any other traditional direct marketing company.

Figure 1-3: Airbnb booked 5 million nights after three and a half years, and its next 5 million nights six months later.

Coding in the future

The one constant in technology and coding is change. Improvements in existing computer architecture will lead to the creation of newer, faster, and smaller hardware devices, and developers will then write code to operate and control those hardware devices.

Moore's Law, a rule of thumb used in the computer hardware industry, predicts that the number of transistors per square inch on an integrated circuit will double every year. The prediction has proved to be true for the last 50 years, although some experts doubt whether it will continue to hold true for the next 50 years.

The following technology developments are increasing in popularity and should remain relevant at least for the next five years:

- **Internet of Things (IOT):** Computing power is transforming dumb hardware devices into smart, connected, self-regulated devices. For example, the Nest thermostat uses a motion detector to record when people are present, and then heats and cools homes when people are expected to be at home instead of all day. Similarly, Lockitron makes a device that allows you to lock and unlock your front door with your smartphone.

Other connected devices, such as the FitBit fitness tracker and the Apple Watch, need coders to add functionality and connect people in new ways.

- **Machine learning:** For years, databases just stored data. Now, code is finally being written to analyze the data and make intelligent predictions. For example, mapping applications use real-time and historical data to predict traffic and the time your route will take to complete. 23andme, a genetics company, compares your human genome against its database to predict which diseases you are more likely to have. General Electric has outfitted industrial machines such as hospital equipment and jet engines with sensors, and uses historical data to repair machines before they break, decreasing downtime and increasing revenue. Coders will continue to write analytics programs to crunch large datasets and generate predictions with increasing accuracy.
- **Interconnected applications:** An application programming interface (API) allows one program to talk to and request data from another external program, which provides a response. Although APIs are powerful, their functionality can be limited and they rarely talk to one another. For example, Dropbox, the storage provider, has an API to allow third-party applications to back up data, and Facebook has an API that lets third-party applications retrieve a user's photos. However, using just those two APIs, you cannot automatically back up every Facebook photo to Dropbox. Companies such as IFTTT (If This Then That) allows users to create recipes that combine APIs.
- **Virtual software containers:** Traditionally, software programs could be described as an interconnected web of your code and code written by others. To incorporate someone else's code into your own program, you had to check that both programs were compatible and that any third-party code used by the external program, called a dependency, was also compatible with your code. The process of resolving conflicts was frequently time-consuming and frustrating. One solution is to move away from the current interconnected system of software programming to an independent self-contained system. Docker is one company that hosts an open-source project to help programmers package software and its dependencies into a self-contained program called a *virtual container*. These virtual containers have standardized inputs and outputs, run on many operating systems, and can connect to each other with little need to check for compatibility. Just like standardized shipping cargo containers make it easier and faster to load and unload ships, so too do virtual containers make it easier and faster to package programs to work easily with other programs.

Tracking the Explosion of Coding Jobs

Creating applications and making computer programs work seamlessly requires many people working many hours because every instruction must be explicit. The Bureau of Labor Statistics estimates that across all industries, about 140,000 jobs in computing are being created every year that pay approximately $80,000. In some industries, computing jobs are growing by over 20 percent, which is two to four times the average growth rate across all occupations.

The demand is great, but computer programmers are in short supply. Colleges train the most computer programmers and graduate about 40,000 computer scientists per year. Using current estimates, by 2020 there will be 1,000,000 more jobs than qualified students, representing a $500 billion opportunity. See Figure 1-4.

Table 1-1 shows some of the coding occupations contributing to this boom. Each job is unique, and generally there is not a great deal of switching between jobs. For example, mobile developers don't suddenly become data scientists, or vice versa. When people do switch between these positions, there is usually a training period.

Web developers are typically self-taught; according to census data, less than 40 percent have earned a four-year college degree. Many developers also enter the profession as a quality assurance analyst and then move into a junior web developer role.

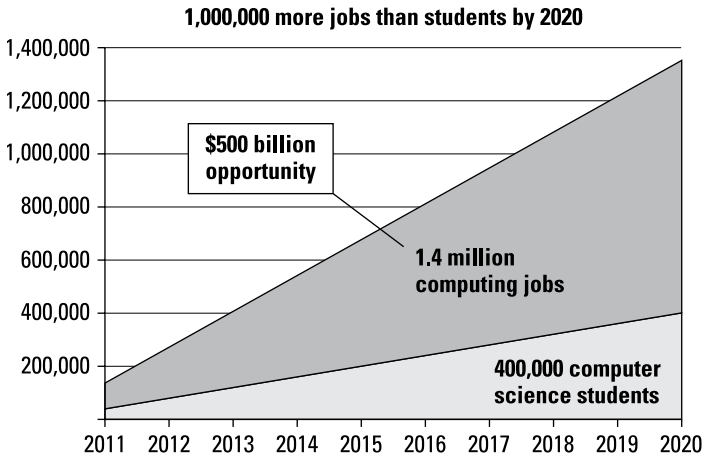


Figure 1-4: By 2020, 1,000,000 coding jobs will go unfilled due to a lack of skilled workers.

Table 1-1 Entry-Level Coding Occupations

Occupation	*Job Summary*	*Average Salary*
Mobile developer	Code applications that run on mobile devices such as phones and tablets. Also responsible for app performance and user interactions that are easy to complete on a mobile device.	$95,000
Software developer	Develop programs and write code for hardware, software, and network systems.	$93,350
Database administrator	Use specialized software to store and organize data, such as financial information and customer shipping records. Make sure that data is available to users and is secure from unauthorized access.	$77,080
Web developer	Design and create websites. Responsible for both the look and feel of the site, and for technical aspects, such as the website's speed and traffic capacity.	$62,500
Data analyst	Analyze big data using statistics and machine-learning techniques to generate insights and future predictions.	$60,000
Quality assurance analyst	Test programs to ensure that features perform according to specification, and document bugs.	$53,000

Sources: Bureau of Labor Statistics, Indeed.com, Glassdoor.com

Companies Hiring Coding Professionals

There's no way around it — all industries are experiencing a massive shortage of talent who can code. Employers are looking for talent wherever they can find it. People with traditional and nontraditional backgrounds, and those who want to work in an office or work remotely are all finding companies that need help.

The two general types of coding jobs are full-time positions in companies and contract or freelance work.

Full-time jobs

Companies of various sizes hire people who have just learned how to code for full-time positions. The size of the company can have pros and cons when it comes to hiring people who have just learned how to code:

- **Large companies:** Companies with more than 1,000 employees, such as Fortune 500 companies and large tech companies including Yahoo!, Google, and Facebook have high standards for hiring employees. Given the number of applications they receive for each open position, recruiters at these companies usually use a strict screening process and grant interviews only to people who have a computer science, math, or engineering-related major. However, for those people who do pass the hiring screen and are eventually hired, there are many resources, both formal programs and people who can help coach and train you to increase your skills.

  Almost every large company has an online application. Send in your application online, and then find an advocate, someone at the company who believes in your candidacy, to help your application pass to the interview stage.

- **Medium-sized companies:** Of the three types of companies, getting hired at a medium-sized company can be hardest. With their large recruiting departments, candidates have to interview with as many people as in large companies. In addition, medium-sized companies typically do not spend as much money on training as large companies.

  One successful strategy to getting hired permanently in a medium-sized company is to freelance first, which helps you build up your reputation and allows the company to assess your skills in a low risk way.

- **Startups:** With less than 20 employees, startups often desperately need coding talent and are small enough to make hiring decisions quickly. They don't have a formal recruiting staff, so you should develop a personal connection with the person doing the hiring. Startups don't have extensive training programs, and you are expected to contribute immediately. However, the small company size should help you form personal relationships with your engineering coworkers, who can help answer questions and informally train you.

  In the beginning, successful startups often have so much work and are so short staffed that having anyone do the work is better than having no one. For this reason, startups decide on candidates quickly, rather than wait for the best person for each role.

- **Government:** City, state, and federal governments and their agencies have thousands of internship and full-time job openings for coders. Depending on the agency, the application process can be time

consuming, and require proof of U.S. citizenship, extensive background checks, and completion of qualifying exams. Applicants can use `www.usajobs.gov` to search across all federal opportunities, and individual state government websites for opportunities in a specific state or city government.

From the source: Tips from a tech recruiter

Yoonie Kim has been a recruiter for technology companies for almost 15 years. She has held recruiting roles at Codecademy, Ning, Meetup, Google, Amazon, and Microsoft. I asked her the following questions:

- *Can you share a little about your work experiences?*

  I've worked at tech companies of all sizes and stages. I started my career at Microsoft, and worked at large established companies like Amazon and Google, all the way to small and early-stage companies like Ning and Codecademy. I also cofounded my own recruiting company to help smaller startups build out their initial engineering teams. In 2014, I joined Dropbox to help build out the New York and Seattle presence and offices.

- *How do you attract and screen candidates?*

  People use and have heard of your product, that helps, but I also reach out to candidates when employees refer them to me, and when I see candidates' work online in a blog post, open source project, or talk. When I screen candidates, I'm usually looking for what they've accomplished, and whether they have actually built something meaningful or just maintained a product. Most interview processes start with a phone screen and then on-site interviews, but I try to personalize the interview as much as possible for the candidate. If you have less coding experience, you might be asked about something you just built, while more experienced candidates will jump into a hard problem the company is currently solving.

- *What do you screen for?*

  At the resume stage, I'm always looking for something interesting that will excite the team and make people want to have a conversation with the candidate. In the actual interview, I'd say 70 percent of the evaluation is technical ability, and the rest of the evaluation is a combination of soft skills and cultural contribution. I used to look for a specific candidate profile, usually a computer science degree and previous tech experience, but I've become more open to people without college degrees, career switchers, and people who have taught themselves to code. I have recruited a few self-taught programmers, and they have gone on to have incredibly successful careers within companies.

- *What is a mistake everyone makes in the recruiting process?*

  Have a story both about what you have done previously and what you want to do at the company where you are now interviewing. Sometimes candidates don't have much to say about a topic they should know a lot about — themselves! Also, have a product or a feature you want to work on if you're given an offer. It can be hard to advocate for candidates who don't express any preferences.

Freelancing and contract jobs

Companies of all sizes hire freelancers to do discrete projects that are not overly complex and have a definite end date. For example, freelancers often build a website with a few defined pages, create mobile apps, or scrape and store data from websites into databases. Getting some of the initial work performed by a freelancer helps a company see how much time and money a project will cost and whether there is a need to hire a full-time employee.

Contract work also provides freelancers with some valuable benefits. Even for full-time coders, doing some contract work is a great way to build up skills in a new programming language or framework. Also, because coding work can be done from anywhere, freelancers have a good deal of flexibility and don't necessarily have to sit behind a desk in an office. For example, some freelancers travel often for pleasure, and can be found working in cities such as Boston one month and Bali the next month. Finally, some coders freelance full-time, and build their business by doing work for existing clients and pitching new work to client referrals.

One issue with freelancing is that you are always looking for the next job. A few websites, such as Freelancer (`www.freelancer.com`) and Upwork (`www.upwork.com`), formerly odesk.com, help provide freelancers with steady work by creating communities that connect employers and freelancers. See Figure 1-5.

These sites create online reputations for both freelancers and companies, which helps each side feel more confident that the work will be completed and the agreed upon amount will be paid.

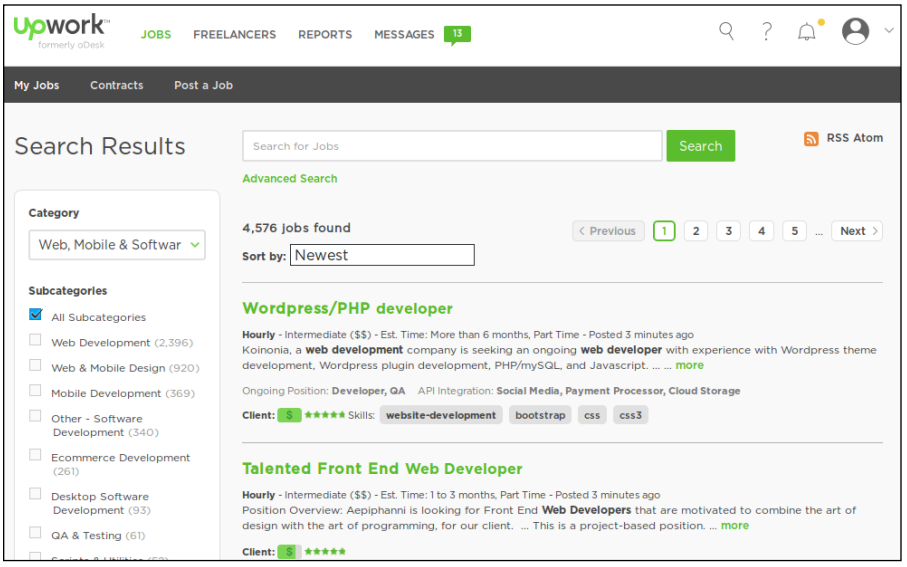


Figure 1-5: Upwork helps freelancers find and bid on contract coding jobs.

Chapter 2

Exploring Coding Career Paths

In This Chapter

- Improving your existing job
- Exploring entry-level full-time coding roles
- Understanding skills and tasks in various coding roles

We shall not cease from exploration, and the end of all our exploring will be to arrive where we started and know the place for the first time.

—T.S. Elliot

For many people, the words "coding career" evoke an image of a person sitting in a dimly lit room typing incomprehensible commands into a computer. The stereotype has persisted for decades — just watch actors such as Matthew Broderick in *War Games* (1983), Keanu Reeves in *The Matrix* (1999), or Jesse Eisenberg in *The Social Network* (2010). Fortunately, these movies are not accurate representations of reality. Just like a career in medicine can lead to psychiatry, gynecology, or surgery, a career in coding can lead to an equally broad range of options.

In this chapter, you see how coding can augment your existing job across a mix of functions, and you explore increasingly popular careers based primarily on coding.

Augmenting Your Existing Job

Many people find coding opportunities in their existing job. It usually starts innocently enough, and with something small. For example, you may need a change made to the text on the company's website, but the person who would normally do that is unavailable before your deadline. If you knew how to alter the website's code, you could perform your job faster or more easily. This section explores how coding might augment your existing job.

Choosing a career path

Coding career paths are extremely varied. For some people, the path starts with using code to more efficiently perform an existing job. For others, coding is a way to transition to a new career. As varied as the career path is, so too are the types of companies that need coders.

As more people carry Internet-capable mobile phones, businesses of every type are turning to coders to reach customers and to optimize existing operations. No business is immune. For example, FarmLogs is a company that collects data from farm equipment to help farmers increase crop yields and forecast profits. FarmLogs needs coders to build the software that collects and analyzes data, and farmers with large operations may need coders to customize the software.

To build or customize software, you'll need to learn new skills. Surprisingly, the time required to learn and start coding can range from an afternoon of lessons to a ten-week crash course to more time-intensive options, such as a four-year undergraduate degree in computer science.

Creative design

Professionals in creative design include those who

- Shape how messages are delivered to clients
- Create print media such as brochures and catalogs
- Design for digital media such as websites and mobile applications

Traditionally, digital designers, also known as visual designers, created *mockups*, static illustrations detailing layout, images, and interactions, and then sent these mockups to developers who would create the web or mobile product. This process worked reasonably well for everyday projects, but feedback loops started becoming longer as mockups became more complex. For example, a designer would create multiple mockups of a website, and then the developer would implement them to create working prototypes, after which the winning mockup would be selected. As another example, the rise of mobile devices has led to literally thousands of screen variations between mobile phones and tablets created by Apple, Samsung, and others. Project timelines increased because designers had to create five or more mockups to cover the most popular devices and screen sizes.

As a designer, one way to speed up this process is to learn just enough code to create working prototypes of the initial mockups that are responsive, which means one prototype renders on both desktop and mobile devices. Then project managers, developers, and clients can use these early prototypes to decide which versions to further develop and which to discard. Additionally,