FLATIRON SCHOOL

Beyond the Bootcamp

How to Craft a Career in Tech

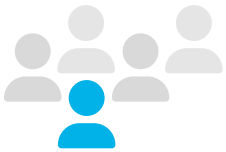
Introduction

At Flatiron School, we are dedicated to helping our students launch a career, not just land a job. But prospective coding bootcamp students are curious: What comes after the bootcamp? And after that first post-bootcamp job as a developer? Are bootcamp grads equipped to continue learning new languages? To go on to tackle new roles?

Tech newcomers often go after their first job with the misconception that their career will fall into place in a logical, stepwise progression. That's rarely the case. In a rapidly changing field, skills are acquired and careers are shaped in totally unique ways. So it's important to start your journey with a clear idea of the different paths available to you after graduating from a coding bootcamp. Right now, you can start to think about what you'd like your career to look like over not just the next one to two years, but the next five to ten.

With that in mind, we're thrilled to collect learnings and advice from our Career Services team, our inspiring alumni, and our Head Online Instructor, Peter Bell – a veteran technologist, CTO, and Columbia University adjunct professor who has lectured at tech conferences across the world on how to build a tech career. Read on to learn:

- How to identify what you want out of a tech career and how to get it.
- How to truly craft your career, not just jump from job to job.
- The paths and options available to you after you get that first programming job.
- How to build the competence and then the awareness of your capabilities to actually go out and get those positions.

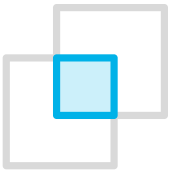


Chapter 1

Get Into Any Room: The Benefits of Learning to Code

Before we explore specific career paths you can take in tech, let's explain the real benefit of coding when it comes to charting your career path. It's really all about being able to *get into the room*. 25 or 30 years ago, if you were unsure what you wanted to pursue, people would say, "You should be a lawyer." Why? If you're interested in entertainment, you can move into entertainment law; if you're fascinated by real estate, there's real estate law; if you want to be in corporate transactions or financing acquisitions… you get the idea. Any industry needs at least one lawyer in the room. And for a long time, many people entered the field of law because it really was a surefire way to insert themselves into any domain, any business, any activity that resonated with them.

But these days, becoming a lawyer isn't necessarily the easiest or most profitable way to get into the room. What's replacing it? With software eating the world, it's becoming a technologist. Whether you want to work for a brand new three-person startup or a huge enterprise, companies are powered by software engineers. And with the increasing supply of open software engineering jobs still outpacing the number of engineers being produced by both colleges and coding bootcamps, the next decade or so is going to be a really good time to get involved with whatever you're excited about *by leveraging programming skills.*



Chapter 2

Two Programming Paths: Core and Enhancement

Before looking at more specific paths you can take in tech, there are really two broad categories of roles that allow you to leverage the programming skills you'd develop at a coding bootcamp.

Core

One general path through tech is becoming what we'll call a "hardcore" developer who spends most of his or her time writing code. This is the route that most bootcamp grads take. If you're thrilled to learn all the ins and outs of a new JavaScript framework or you light up at the idea of implementing a NoSQL data store to improve the performance of your app, this could be a great route for you. And within this category there are a number of directions to go. You might find that you prefer making things look great and focus on on front-end web development. Or, you might find that you love to focus on improving the underlying functionality, performance, or scalability of applications – then you might decide to focus a little more on the back-end of software development. Or you might combine those two things to become a full stack developer, able to examine a business problem and come up with a well-designed, powerful, and great looking app to tackle it.

> "Once you've learned those first skills, it becomes much easier to expand on them and explore other exciting domains. I see lots of

… people moving into machine learning, convolutional neural networks, conversational commerce, and more cutting edge technologies that are becoming increasingly integrated into products we use every day. For example: Isn't it kind of amazing how Facebook knows which of your friends are in your photos just by running them through an unsupervised machine learning algorithm? Or how you can just ask Amazon Alexa – sitting in everyone's living rooms right now – 'what's the weather tomorrow?'"

Peter Bell
Head of Online Instruction, Flatiron School
& Veteran Technologist

// FLATIRON SCHOOL

Enhancement

So if one broad application of your programming knowhow is to spend most of your time writing code, the other is to *enhance* the other skills and pursuits that make up your career. The graphic designer who simply delivers mockups in Illustrator for a website without considering mobile first or responsive designs – and even developing front-end prototyping skills – is quickly going away. Growth hackers and digital marketers, rather than just coming up with catchy copy and great images are now focused on A/B testing, digging into Google Analytics, and even writing scripts to test out campaigns and outreach. There are also programmers who end up using their skills to fuel product management in some organizations – to have a greater input on what they should be building.

“There are some product managers who don’t have a technical background. But if you want to really see what your developers are doing, to know how to review a pull request in GitHub or even to be able to look at their source code and make small tweaks to a color, font, or layout, having those programming skills makes it much easier. It also gives you much more credibility with your team. So even if you decide to study programming, it’s not necessarily the case that you’ll be doing it for 40 hours a week. There’s the capacity to follow your other passions and be in the vicinity of programming without coding all the time.” **—Peter Bell**

Chapter 3

What Do You Want to *Do*?
(Tasks, Not Titles)

When picturing their future career paths, so many people go after job *titles* – "I'm a software engineer so I want to become a senior software engineer and eventually a Chief Technology Officer" – without thinking about whether they'll actually enjoy the day-to-day tasks and responsibilities of those roles. They often don't take the time to consider what a title really means. What will you be doing on Monday mornings as a CTO? On Wednesday afternoons as a Product Manager? It turns out that the highest predictor of your happiness and success in a position is whether you enjoy the tasks that you spend most of your time doing.

So, what do you actually enjoy spending your time at work doing? You might not know that now, but as you spend a little more time learning how to program, and as you start to grow in your first job, notice the tasks that you truly enjoy, and the ones that you don't.

> "Sure, it sounds cool to become a CTO. But from personal experience, the primary job of a Chief Technology Officer is often hiring people and then vaguely pointing them in the right direction. Engineering management can be a great way to have more impact by working with a team of people – directing them to build more software than what you could code on your own. But understand, if

… you love coding but don't like sitting in meetings and doing one on ones, running an engineering team might not be all that much fun for you." **—Peter Bell**

It's also important to ask: What really matters to you? What are the most important elements of a job for you. Here are a few to consider.

Compensation

There's an awful lot of money to make in finance at the JP Morgans, Goldman Sachs, and Bridgewaters out there. There are some amazing nonprofits that will allow you to make a positive impact on the world, like Crisis Text Line or Charity Water – but unfortunately, no matter successful they are at preventing suicide or providing people clean water, you're not going to get rich with the stock options. Truth be told, you won't be getting rich off stock options with most startups. But it's worth asking yourself: How much is enough for you? In a 2010 study from the Proceedings of the National Academy of Sciences, they found income and happiness are correlated up to about $75,000 a year; beyond that, they start to diverge. People making $90,000 aren't necessarily that much happier than those making $75,000.

"A general rule of thumb: don't maximize for compensation in the first three to five years of your career. Take the opportunity where you're going to learn the most. Those first few years of learning and experience will put your in a much better position to get the more highly compensated jobs over time." **—Peter Bell**

Culture

Every company has its own style. If you get a job as a programmer, you're going to be spending 40, 45, maybe 50 hours a week at work depending on the team. You want to make sure you enjoy the environment and *people* you're working with. So look at your prospective coworkers. Are these people you want to hang around with? That's a great benefit to modern hiring: most companies put you in front of three or four people from the company so you can get a great sense of the people you'd be spending much of your life with.

> "A caveat: if you're coming straight out of a coding bootcamp, just get a job; get a few years experience, then decide what you want to do next. But a point will come after a few years when you'll get new opportunities all the time. When you start to see that occurring, it's worth thinking more about what kind of experiences you want to have. What kind of focus on diversity do you want? What kind of focus do you want on work-life balance? What kind of sense of *mission* do you want?" **—Peter Bell**

Commute

Interestingly, numerous studies show an inverse correlation between commute time and workplace happiness. If you're commuting more than 15 or 20 minutes a day each way, you are reportedly less likely to be happy – even if it's a great job. So, how long is it going to take to get in and out of the office? Is that acceptable to you? Can you work from home a couple days a week to break up a week? You might find a job in a shiny office, but if it takes a two hour drive on an LA freeway to get there every day, that might not be as fun as you

expect.

Creativity

How important is creativity to you? And how well supported is that in the organization you're considering joining? Also think about what *types* of creativity and what types of domains and technical problems you enjoy. Do you want to create the next big thing? Or do you enjoy the precision and the focus of cleaning up someone else's mess or operationalizing something more effectively?

> "Most engineering managers sit on a spectrum between telling their engineers 'We'll tell you exactly what to build and how to build it' across to 'Here are our business goals and we want to know how you think we should leverage technology to solve them.' In the latter case, you could get freedom over anything from which tech stack you use to the way you implement the solution to the design or UX/UI patterns you utilize." **—Peter Bell**

Chapter 4

Picking a Track

One of the decisions you'll be making when you're looking for a new job is choosing between working for a smaller startup, a classic mid-market company, or a large global enterprise. Generally, there are strengths and weaknesses associated with each of them.

Startups

The great thing about joining a startup is that it gives you a chance to have a real impact. Often in the early days of a startup, it may just be 3, 5, 10, 20 engineers. You'll be building nontrivial features along with a few people and shipping those in *weeks*, not months or years. The flip side is that many startups fail. You may have a less broad impact because, sure, you're shipping new features, but fewer people are using the product. It's not the impact of working for a large enterprise where you code may affect hundreds of millions of people.

Midmarket

You have to be careful if you're considering a job at a midmarket company because many of them aren't focused on technical innovation. Often they just plod along doing whatever made them successful 15 years ago. On the other hand, you may find more flexibility in this category. Connecting with a stable midmarket company can be a great way to get a 9-to-5 that allows you to do something fun during the day, but not to have to worry about whether it's going to be a problem if need to go pick up your kid from school when she gets sick.

"I wouldn't say don't work for a mid-market company, but look very carefully at who your boss will be. Is he or she someone who you think you could learn from, from a technical perspective? Do they care about a good engineering culture? What's their opinion on Test Driven Development? Do they have opinions on Continuous Integration or Continuous Delivery? They may or may not be 100% behind it – there are some good reasons to not do CI or CD – but if they don't at least *have* opinions about those things, they're probably not going to help move your career forward." **—Peter Bell**

Enterprise

The great thing about enterprise companies is the huge power that they can bring to bear – it's quite easy for you to do things that will affect millions or tens of millions of people. The downside is you can be a very small cog in a very large wheel, and so sometimes it's hard to have a substantial role in that impact. Also, as you grow, you can spend a lot of time in a political environment, fighting to get a product built. It's important to note that there are no good or bad enterprises, but these companies do have some team leads who are amazing and some who are not. So be thinking about who's hiring you. If you respect and can learn from that person as a mentor, then it's probably a good position to consider.

Chapter 5

Roles to Grow Into

It's strange to imagine leaving engineering just as you are getting into it in the first place as a prospective or current coding bootcamp student. But let's take a moment to think not about the first and second job you accept but about the third, fourth, or fifth. Generally, when you've been a software engineer for a while, you face a choice: where do you go now? Say you've been doing Ruby on Rails for eight years in a production environment, you're capable of doing continuous development and integration and creating dev environments, setting up hosting and solving business problems, and maybe even running or at least collaborating with a team on building new features. *What's next?*

Management

a.k.a. Team Lead, Engineering Manager, Director of Engineering, VP of Engineering, CTO

One option is the traditional path of moving from engineering to engineering management. This is a way to impact a bigger swath of product functionality than you could as an individual programmer. But keep in mind, this isn't about coding. It's running one-on-ones and doing whatever it takes to help the people who work under you to build careers they love – to ask, "How can I help my organization succeed while making the lives of team who work with me as awesome as I can?"

"It's a fun and fascinating challenge. But typically, if you're not careful, this is the track where you take the best software engineers – who are really good at writing software but may not be great communicators – lock them in a room and say, 'don't write code; go hire and talk to people.' That doesn't always work out so well! So it's worth taking the time to decide whether you really want to manage a team of engineers. Do you like tasks involved with that? If you do, technical management could be a great way to have impact by driving delivery of code that requires ten to twenty engineers – more code than you could write yourself." **—Peter Bell**

On the other hand, there are still two more directly technical approaches to consider.

Technical Specialist

a.k.a. Senior Ruby Engineer, Senior JavaScript Engineer, Senior iOS Developer

This is the person who knows everything about something. The good news is that technical specialists can really get deep into specific technologies and become true experts. But if you're not careful, as a technical specialist, your success is dependent on the technology you specialize in – and technologies don't live forever.

"It's awesome to know every single thing about the Ruby language. But bear in mind, one day people won't be using Ruby. Make sure that you have a plan to evolve as the technologies people use evolve. Don't go down with the ship. Keep an eye on new technologies and keep learning." **—Peter Bell**

Technical Generalist

a.k.a. Architect

This is somebody with a good understanding of a range of patterns and approaches for solving technical problems. They're the person you go to say, "I need to build an app that does X. What tech stack would you use? What would be an appropriate database? Do we need microservices? And how do we use all these things?" Technical generalists can really think *architecturally* about how a company can write code that will be both quick to develop and easy to maintain.

> "The interesting thing about this path: as long as you continue to invest in your skills, it's much harder to become irrelevant as technologies change because you know a little bit about wide range of technologies." **—Peter Bell**

Chapter 6

Freelance, Contract, or Full-Time?

Ss you move through your career, you'll have to choose between freelance, contract, and full-time employment opportunities. But know that it isn't a one-time decision. You may choose to move between these states, each of which has its own benefits and downsides.

Freelance

There is nothing greater than the flexibility of the freelance lifestyle – to be able to hop over to Italy as you do a React project for a friend, or to live in South America for six months of the year so you never need to hit a winter while you build the MVP for a startup. But it's important to realize that as a freelancer, you don't succeed because you're technically competent; you succeed because you're good at selling stuff. You're good at persuading people to give you $25,000, $50,000 or $100,000 to build something and drive enough revenue to pay your bills and hopefully make a profit as well. Many people do not enjoy that, so think carefully before you double down on the freelance lifestyle.

> "A rule of thumb: If you're a freelancer, you'll find that even if you're working 40-50 hours a week, and you're fully booked, you still should assume you will bill no more than a thousand hours a year. Because a big part of your job is meeting people and doing free proposals and cleanups, configuring computers, reading blogs,

… getting up to speed with the latest version of Angular or React, etc. No one pays you to do all those things if you're working on a freelance basis." **—Peter Bell**

Contract

This is full-time position that isn't necessarily long-term. There are recruiters out there who, once you have three to five years experience in a given technology, will "body shop you out," typically to larger enterprises. You can get a better salary than full-time employees, but won't get the same benefits or stock options. But if you're looking for another way to have the ability to jump between projects without your resume looking like swiss cheese, working for a company that will hire you out can be a way to go.

Full-Time

Then of course, there's the path most of us think about. It's kind of awesome. You get a paycheck every month! You don't have to hustle clients to make the payments they owe you. There are people who are not built for freelance – selling themselves or charging money for what they do. If that's the case, go full-time, not freelance. You and everyone else will be happier.

Mix and Match

Some people move between these types of roles on a three to five year cycle. They work for one or two companies full-time for a while, then they get that itch. They just want to have more freedom, and they've got a good network, so they freelance for a bit, have fun and put some extra money in the bank. Then after a few years they think, "This is OK, but I never know where my money is coming from," and

they swing back to full-time. It can be a useful cycle jumping between these paths, flexing different muscles and technologies, depending on what you value and how you work most effectively.

Chapter 7

Going Off the Rails: Other Things You Can Do as a Technologist

For technologists not interested in diving deeper into engineering or managing other engineers, there are still other types of roles to pursue.

Product Manager

One of the more common paths you can take when veering away from hardcore coding is getting into product management. It allows you to move your level of abstraction one level higher. You don't have to worry about memory leaks or how to figure out why you can't pick the right ID or why your jQuery isn't working. Product managers focus on high-level issues. *Why would someone use this? How do we simplify the signup process for people who don't want to type a lot on a mobile app?* It's a way to still leverage technical understanding and collaborate and communicate with an engineering team, but here you can focus on solving user problems and figuring out what kinds of features should be built in the first place.

Evangelist

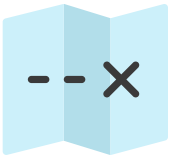
Turns out that not that many technologists love speaking to people, at least rooms full of people. If you happen to be someone who is tech savvy, you enjoy giving presentations, and are at a stage in life where travel excites you, there are some really cool opportunities to become a technology evangelist – to travel around the world presenting technologies at conferences and meetups.

"At the peak, I did 104 presentations in a year. And then I got lots of interesting offers, because there are lots of companies looking for people to present about their technologies at conferences – sharing how cool Stripe is, or why you should use Google Apps for business, or the best way to leverage the Twilio API. These are awesome opportunities if you enjoy travel and communicating with people."
—Peter Bell

Entrepreneur

A small percentage of people will start their own businesses. If you have an entrepreneurial drive, it's a worthwhile path to follow. One of the great things about having the ability to write code is that you can actually build your own MVP and run your own experiments.

"I've made a lot of startups, and I'll give this advice that I wish someone gave me: Go work for *someone else's* startup for a few years before you build your own. Why? It turns out there are lots of mistakes you make when you work at startup – and it's much much better to make mistakes on someone else's dime rather than desperately trying to make money and pay others while making mistakes on your own dime." **—Peter Bell**



Chapter 8

How Do You Get There?
Standing Out as a Coding Bootcamp Grad

Once you've decided on the path you want to follow, you have two huge tasks in front of you: building the capabilities to get the roles you want and then building an awareness of your capabilities in a competitive field – getting better and getting *known*.

So, how do you become a coding bootcamp graduate who really stands out? It comes down to building a brand for yourself as a technologist – standing for something more than just being a *"yet another full-stack web developer."* There are a number of practices you can employ to help you present a clear value proposition for yourself that makes you stand out from the other people graduating from similar programs.

Present

One possibility is that once you're getting around two thirds of the way through your course, you're at a point where you're actually starting to understand code and talk about it relatively fluently. It's not too soon to push yourself to find a local meetup and give a talk. What most people don't realize is the hardest thing about organizing a technical meetup is finding anybody willing to speak for 20 minutes. So even if it seems a little bit outside of your comfort zone, you have a shot at presenting in public – one of easiest ways to be perceived as a top technologist in your field. Not sure what to present? Look for the intersection of something you know about and something you want to know about: Unity programming for VR; mobile apps and geolocation; something that will become important

that doesn't have an expert yet. That expert could be you.

"I've presented at technical conferences around the world in a lot of technologies and a lot of domains. The interesting thing is, I'm often offered jobs that I have absolutely no competence to fill because people think I'm way smarter than they are *just because I stand up in front of people and talk*. On top of being perceived as an expert, there's another bonus: If you happen to be a little introverted, as a presenter, you'll find people will come up and talk to *you* – you don't have to introduce yourself or make small talk. And you're speaking with other techies, so you can actually be talking about, say, metaprogramming in Ruby, not trying to come up with a great conversational opening gambit." **—Peter Bell**

Blog

It's incredibly important to build the skill of blogging on a regular basis. What you're doing, really, is proving that you care about technology and can communicate about it. Note: it doesn't have to be the one true blog; you don't need to figure out something about Ruby that no one else in the world has ever known. Just give someone a great introduction to a gem you used, and three tips to get it configured more quickly. Give people your take on why you got benefits out of taking an object-oriented approach to your Ruby, and what you liked about the information hiding. You can even just take some of the concepts you're learning in a given lesson and rewrite them in your own way. By doing so, it's not only deepening your learning, but it's differentiating yourself as a person with enough passion about technology to take an extra hour to put up a blog post every week or two.

"When looking at candidates for a job, one of the first things I'll ask is: what are the things the candidates have done that they didn't *have* to do? What have they done that they weren't paid to do? Have they blogged? What have they blogged about? Is it at least somewhat thoughtful and coherent? It doesn't necessarily need to be the technology that I'm asking them to use, it just needs to be a technology that they're passionate about." **—Peter Bell**

Contribute to Open Source

There are some people who have a hard time talking in front of people at a meetup. Such people may be more interested in focusing on the code. A third way to become known in the programming community, then, is to pick an open source project or two and contribute to them. In an ideal world, you want to start with an open source project that is small enough so that the core committers will notice you, but quickly gaining speed – try to catch a moving train.

"The best part is you don't need to be an expert. You actually don't actually need to be a programmer yet to do this. Just offer to go in and improve the help docs or the README or the configuration information. Fix some typos in the text that they've put, and they will love that contribution, and know that you're contributing to a real open source project, which is another way to connect with and build a community of peers online. And if the project starts getting more attention, you'll be getting your name out there without necessarily blogging or presenting." **—Peter Bell**

Expand Your Network

There are some phenomenal technologists who are just getting by because no one *knows them*. A big part of building a career in tech is connecting with people. So – sorry, introverts – you have to *network*. Go to meetups. Find a community of developers. Those are the people who work at the companies who could hire you. Those are the people who are networking and have contacts who people go to to say "build me a Rails app for 15 grand." And if they are too busy, they may say, "I don't have time but I met this person who might be interested." It's an opportunity to find full-time and freelance opportunities. There are regional conferences and meetups around the country and around the world. Find them, spend time with them – it's one of the best ways to get a sense of what it's like to be a professional developer and build the contacts to help you to get jobs.

Be Nice

There are some incredibly smart people in tech. The problem, sometimes, with being that smart is that you may not be willing to work with other people. You simply have to be nice to people and respect them, even if you consider them to be a little less intelligent than yourself. Even if you're the hottest JavaScript developer in the world, and you might be, if you're an asshole, you're probably not going to get most of the jobs you're seeking. And any jobs you do get, you're not going to like because now you're working for a company who hires assholes! Really take the time to think, to take a breath before you communicate.

Do More

If you can find two or three hours a week to do one thing that you didn't have to do for your school, or for your job, your opportunities

will multiply over time. Read just one extra book every quarter; write just one extra blog post every month. When you start doing a little bit more, after a little time, you will notice that the quality of opportunities you're getting – and the quality of enjoyment and fun you get from the career – will be substantially better than people who are just doing the minimum they can do to get by.

"I've had the opportunity to work with a lot of capable technologists over the last few decades. And what I've noticed about them time and time again is that they're committing to doing a little more than everyone else. The interesting thing about going from good to great is that it's kind of like going from second to first in a race. If you're running a 26 mile marathon, often the distance between first and second place is 100 meters or less – it's not like you have to run twice as far or twice as fast. Similarly, in tech, if you consistently go two, three, four percent further than most of the people you spend time with, over the course of two, five, ten years, you'll find you know way more about programming and way more about solving business problems than your peers. It's not about pulling all nighters all the time. It's just a little more care, time, and precision. The engineers who do that become much more successful over five to ten years." **—Peter Bell**

Get Started

Imposter syndrome is common in tech. New developers often can't believe they got the job, that they're actually getting hired. They think, "I don't know what I'm doing. There's all these people with more experience."

> “The funny thing is, I’ve been doing this for a while so I have friends who are CTOs and VPs of engineering. But when we sit down in a room together, we say *exactly the same thing*. ‘I can’t believe they hired us to run this engineering team. We don’t know what we’re doing. Sure, we might have been doing it for 15 years, but technology changes so fast, we can’t keep up!’” **—Peter Bell**

So don’t wait until you know everything (it’ll never happen). Focus on building a skill that you’ve sufficiently honed so you can hit the ground running and add some value to a company. Leverage that into the first job, and keep learning. Don’t wait until you believe you’re the best to talk, write about, or use a technology. Just *start* and let things grow from there.

Chapter 9

Alumni Stories

Flatiron School grads work across diverse roles and industries. They also have plenty of advice to offer about learning how to code and charting a successful career path through tech. Read on for their stories and their tips for aspiring technologists.

StreetEasy

Steven Brooks: From Pro Baseball Player to Engineering Manager at StreetEasy

Steven Brooks had a unique path to his tech career: before Flatiron School, he played baseball professionally. Now, he's the Engineering Manager for StreetEasy, one of Zillow's New York City real estate brands.

Learning to Code

So *why* switch from playing pro baseball to coding? Steven spent his off seasons helping out at his brother's startup. One of his responsibilities was managing the engineers. He found himself "in awe of what they were able to do in a short amount of time." As he describes it, "I could be there at midnight thinking of something I wanted to add to my brother's company, and I couldn't do it. But an engineer – they could do it. It's a type of job where you can go from idea to actual tangible product – something that someone can touch and use – in the shortest amount of time." Inspired, Steven started teaching himself how to code before coming to Flatiron School.

Takeaway #1

According to Steven, the best way to learn to code is to “just get in there and figure out how to really build something. So build Twitter, build Facebook, whatever it is, but you'll learn more valuable knowledge on how to program if you're actually trying to build something and there's some sort of end goal of something that you'd like to build. So just get in there. Don't think. Just get in there.”

The Flatiron School Experience

After hearing from a friend in Flatiron’s first-ever class who “only had the greatest things to say about it,” Steven joined the second-ever Flatiron School class. Steven had been learning on his own but felt he needed structure to take it to the next level. “Flatiron School obviously provided that and a lot more,” he says.

His favorite part of the experience? “I loved Feelings Fridays,” he says, of Flatiron’s weekly group check-ins. “I would pour it all out there.” Steven notes that with no engineering background, it was a struggle to essentially retrain his mind to think like a programmer. But in these weekly meetings, “I could look to my left or I could look to my right, and for the most part, everyone was exactly where I was in the struggle – all working towards the same goal. I didn't feel alone.”

Takeaway #2

For Steven, the most important skill he learned at Flatiron was how to break down problems into their smallest parts. “Now that I'm more involved in the business end at StreetEasy, it's very helpful for me to be able to think that way. If we have a big goal, like make X amount of money next year on rentals, that's a *huge* goal – but if you break it down to its smallest parts like you’d do making an app, it helps tremendously.”

Life at StreetEasy

Though Steven was nervous before starting at StreetEasy (“you're afraid someone's going to find out that you're not as good as you think you are”), he quickly realized that being an active member of the team at StreetEasy involves a lot more than just having good technical chops: “Can you learn quickly? Can you be a good member of the team, and can you provide some sort of value?” Steven says. “No one's really expecting you, as a junior engineer, to be able to create everything and do all the bells and whistles that a senior engineer does, but they do expect you to be part of a team.” Steven thrived in the team environment at StreetEasy and worked his way up to Engineering Manager.

As Engineering Manager, Steven spends each morning running standups with StreetEasy’s engineering teams, including 30 offshore engineers, as well as the company’s customer support, product, and business teams. He will *always* take lunch, and recommends aspiring developers make sure to step away from their desks: “I never skip it. It's blocked off my calendar. I won't be successful in the afternoon if I can't kind of clear my head. The people that eat at

their desks or don’t have lunch at all – I don't think they're as effective in the afternoon.” For the rest of the afternoon, Steven works with engineers and the product team to align their communications and build out products – making sure the engineering team gets the requirements for products far in advance “so that the engineers have all the tools they need to succeed by the time they actually get the task.”

Takeaway #3

Steven credits his success at StreetEasy to his willingness to communicate and ask questions. “There's this concept of an engineer who's stuck in this rabbit hole,” says Steven. “I never really did that. I had my own rule, where if I couldn't understand a problem after 15 minutes, I would ask someone. Because if it was going to take me three hours to figure it out on my own and someone could help me in ten minutes – well that's going to save the team a lot of time. So don't get stuck in that rabbit hole. There's never a question too embarrassing or an answer too obvious, especially with the type of community we have around programming.”

What’s Next for Steven?

When asked about his career goals, Steven has a simple answer: “I want to be the GM of Zillow's New York office in the next four years.” Given his determination, that outcome wouldn’t surprise us at all.

VOGUE

Simone Hill: From Assistant Editor to Front-End Engineer at Vogue

Before Flatiron School, Simone Hill worked as an assistant editor at The Knot, “writing about all things weddings,” as she says. She’s now leveraging her creative instincts in another way: as a front-end engineer at Vogue.

Learning to Code

While working as assistant editor at The Knot, Simone realized what she really loved about her job was the tech product and the platform – a tool that brides and couples could use to plan their weddings. But, according to Simone, “I didn't know anything about how it was created or what would go into building something like that. So I had no technical background, but I really wanted to immerse myself in that and learn to build those things so I could have a skill that I could use to create tools, platforms, and apps like that myself.”

Takeaway #1

Simone’s advice for others starting on their coding journeys: “I would say just stick with it. It's really challenging, and that never goes away. Even when you graduate from Flatiron School, even when you get your first job, even once you get a promotion in that job, there's always going to be this feeling of, ‘should I be here? I've somehow tricked everyone up to this point. I'm way, way in over my head.’” One of Simone’s techniques for overcoming those feelings of imposter syndrome was to learn to ask questions. “You really cannot get anywhere in any field, but especially as an engineer, if you don't

… open yourself up to being willing to ask a question," she says. "Actual preparedness is knowing what you don't know and the questions you need to ask to keep learning."

The Flatiron School Experience

What Simone was seeking in a coding bootcamp wasn't so different from what she wanted out of her undergraduate education. "I really enjoy the liberal arts experience in broadening problem solving skills as opposed to teaching someone a very specific tool," she says. "I think one of the big critiques of a lot of bootcamps is that when this technology changes in five years, you'll be left behind." That wasn't the case for Simone at Flatiron School: "I learned problem solving skills that I still use as an engineer and built a great base through which to continue learning and teaching myself as well."

For Simone, the best part of the experience was how students presented their projects to the technical community. "I hate public speaking to this day," she says, "but it was a really great way to force me to articulate technical ideas in front of an audience, and I think that was a really big benefit to have, especially when you think about going into job interviews and things like that – it really prepares you for that kind of experience."

Takeaway #2

Simone's advice for coding bootcamp students is to cement your learning through teaching. "Even if you don't really know something that well, find somebody who knows a little bit less than you do and

… try to teach it to them," she says. "You can get a lot of that at Flatiron because you're working so closely with your peers. There might be a concept that clicked for you really quickly. Or even if it doesn't, find somebody and try to explain it to them. Even if they already know it, it can still be a benefit to you and to the other person."

Life at Vogue

Vogue hired Simone as a junior engineer. Following her passion for front-end challenges, she was promoted to front-end engineer. Still, according to Simone, working on a small team within the larger organization allows Simone to "see and interact with a lot of different pieces of our code base that's not necessarily strictly front end. No one is pigeon-holed." Plus, she notes that with the advent of React, what it means to be a front-end developer has changed drastically: "Being a front end engineer used to mean you're just writing HTML and CSS. Now we have this thing called React, where you're doing a lot of JavaScript logic in your view with your HTML, and there are some people that even bring CSS directly in there too, within React. I get to see and learn so much about the different pieces of our app."

Takeaway #3

Simone's top tip for bootcamp grads seeking a job is to take the initiative to keep on learning and building after graduation. "Try out a new technology or a new language that you haven't had experience with. Employers are really impressed when they see that you've

… gone beyond just what you learned in bootcamp to apply your learnings to something new." she says. "I think they really appreciate seeing that this isn't just about getting a job for you – you're looking for a career. This is a dedication, this is a passion."

What's Next for Simone?

Learning to code and being a software engineer was a very intentional step for Simone, and development isn't exactly where she plans to end up. "Ultimately I want to transition into a product facing role, lead a product team at a startup, or product at any number of companies," she explains. "I think what really hooked me into my desire to learn to code and my love of technology is working on a tool and thinking of features, creating this story of who the user is, and what they do, and how they interact with the site, and how we fulfill their need for them with a piece of technology."

BLACKROCK

Chris Guthrie: From Equity Analyst to VP of Engineering at BlackRock

Before Flatiron School, Chris Guthrie worked as an Equity Analyst, but he wasn't satisfied with the "volatile" life of a stock trader. He came to Flatiron School to turn his programming hobby into a programming career. In the years since graduating, he's worked his way up to becoming Vice President and Senior Engineer at BlackRock, the global investment management company.

Learning to Code

While attending college, Chris embraced the marketplace, trading stocks to support himself. Two days after graduating, Chris moved to New York City to find a job as a stock trader and ended up working for a proprietary trading firm for six years. “It was a very interesting experience,” Chris says. “I learned a ton about how our markets work, but, ultimately I decided that wasn’t the way I wanted to spend my whole life because it was a very volatile life – there’ll be some days where you would do very well and some days where you would do quite the opposite. It was a tough way to live.”

To some degree, Chris had been surrounded by code his whole life – his father was a developer. “I think I was like 3 or 4 years old,” Chris remembers. “My dad turned on his Radio Shack TRS80 and showed me how to program some stuff in BASIC. So from a very early age I was around code but not really actively coding. I just kind of knew it existed.” Fast forward to college, Chris began making small trading algorithms “so when I actually went to class, I could turn on a switch and the computer would trade for me.” But Chris never went beyond being a hobbyist until a good friend of his introduced him to Flatiron School.

Takeaway #1

Chris’ advice for those curious about programming: “If you ever want to learn to code, you can learn to code *that very moment*. That’s just a fact of life with all the resources that are online right now. It’s really incredible: you can go to a variety of different sites and just code interactively immediately, learn any language you really want to. So my advice for people who want to learn how to code is you should go learn how to code. It’s really that simple. If you have even the smallest bit of ambition you can get started

… immediately and if you have an even a greater interest then go check out a place like Flatiron School."

The Flatiron School Experience

Flatiron School caught Chris's attention after he saw how a close friend transformed her own career. "I was astounded and blown away how quickly that was able to occur," he says. "It was very clear the demand for developers in the market place is just explosive, and so deciding that I really couldn't pass up that kind of value – a three month investment; the return on that is just tremendous. The amount of stuff you learn in a short amount time is just incredible."

His favorite part of the experience was how Flatiron "didn't mess around. On the very first day, they sat us all down and told us we had to push code in like two hours. We all had our five minutes of panic before we kind of settled down and tried to reach back to the pre-work we had done to figure out how to actually use Git, how to actually come up with a few lines or HTML, and so on." Chris liked that the majority of the Flatiron program was actual coding, which is "really the only way you can learn how to code – by making mistakes, making them often, and figuring out the right way to do things."

Takeaway #2

Chris's advice for coding students: Learning Ruby and JavaScript at Flatiron School is a powerful foundation for your software engineering career, but don't be afraid to keep learning more

… languages: “Once you know one language or once you know one framework, it’s just documentation and syntax to go onto another one. Make sure you have an open mind about what people are programming in, make sure you have an open mind about the development community at large and trends that are happening.”

Life at BlackRock

Chris started at BlackRock as a developer, primarily repairing legacy code and deploying emergency bug fixes. But he quickly began to stand out – early on, he taught himself Angular on his own and rewrote one of BlackRock’s legacy apps in the framework. “Looking back at the code, it’s hilariously awful,” Chris says, “but any project that you work on, six months later you’re going to point at it and say how bad it really is. But you have to start somewhere. The cool thing was that I was able to do it and it impressed a lot of people. That kind of cemented my reputation of someone who can figure something out, especially on their website where BlackRock was seriously ramping up.”

After working his way up at BlackRock, he was recently promoted to Vice President and he’s had the opportunity to work on a truly impactful project. “The biggest project that I’m working on right now has to do with really safeguarding the retirement accounts and investment accounts of people all over the world and especially in America,” Chris says. “It’s really cool that I’m part of potentially helping somebody’s grandmother, or someone’s dad from getting taken advantage of from an unscrupulous financial advisor.”

Takeaway #3

To get that first software engineer job, “you’ve got to stand out a little bit,” says Chris. “So I really would suggest that if you’re going out to interviews, be over-prepared. Have your apps available and hosted somewhere, so once you talked about them you can say, ‘Hey, do you want to check out my apps?’ and you hand them your laptop. Every interview is going to be different – you might have a whiteboard session, you might have some variety of technical interview – but having something that you’ve actually done ready for an interviewer to see is very valuable thing.”

What’s Next for Chris?

Chris admits it’s hard to forecast years out considering the speed with which technology develops. Even the project he’s working on now is “going to be in a very different place a year from now, and so we’ll have to see what happens,” he says. “But it’s tremendously exciting to work on, and I’m very proud of the work I’m doing and the work my team is doing to protect people out there in the world.”

New York Public Library

Ashley Blewer: From Archivist to Applications Developer at New York Public Library

Ashley Blewer worked at the University of South Carolina as a Moving Image Archivist and Metadata Cataloging Manager – “which is essentially a long complicated title for Archivist slash librarian,” she says. She’s remained in the field she’s passionate about, but is poised to make a larger impact with her technical skills. She’s now a developer at the New York Public Library.

Learning to Code

Working in the University of South Carolina library, Ashley couldn’t help feeling there were inefficiencies in her job. “I've always been technically-oriented and I wanted to be able to do more and do my job better than I was already doing it,” she says. “I was working with a lot of catalog records and doing them one at a time and there was this ability to fix things in scripted batches. After realizing that that was so easy and possible, it made me want to be able to do even more for the kinds of roles and positions that I was working in.” It’s that drive that inspired Ashley to commit herself to learning to code more seriously.

Takeaway #1

Ashley’s advice for those curious about learning to code: “If you're thinking about learning how to code, 1) you should do it, but with the caveat of making sure you really want to do it and it's something you're really passionate about. 100 percent try it out. It's been incredibly beneficial for me and so many people that I know, but also

… know that if it's not for you, it's not for you and you don't have to learn how to code. Do it, also maybe don't do it if you really hate it."

The Flatiron School Experience

Ashley decided to attend Flatiron School because, as she says, "I really wanted to learn things in the right way. Flatiron school starts you at the bottom and builds you up – you learn everything the right way and the hard way and I really appreciated that."

Her favorite part of the experience was the people – collaborating with like-minded, passionate people pursuing the same goal. "I know people who have done other programs or just in general have been through this process," she says. "and there wasn't this sort of camaraderie that I was able to experience. I really appreciate that so much that we're all supporting each other and working towards this one goal together rather than working against each other or trying to one up each other constantly."

Takeaway #2

The most important lesson Ashley learned at Flatiron was that it's okay to be wrong. "I think so much of programming is trying things out until you get it right and that knowing that even the people teaching you are not necessarily going to be able to whip out this beautiful code the first time perfectly – you're sort of figuring that out as you go along," she says. "I think a lot of people who don't do programming at all assume that you just write out this perfect poem and it becomes a beautiful website, rather than having to work

… through it constantly. So, failure is what I learned, and how failure is okay."

Life at the New York Public Library

When Ashley first graduated from Flatiron School, she interviewed for a developer role at the New York Public Library – "we got pretty far along in the interview process and I ended up being their second choice and they hired someone else." She ended up working at a tech startup, but felt that it wasn't quite the right fit. "I was moving back into doing a lot of archiving work and I was doing consulting with these new skills I had developed. At the same time, New York Public Library got back to me and said, 'Oh, actually we have another job opening and this one maybe fits you a little better.' It was pretty lucky that this was the first job that I wanted after Flatiron School and I didn't get it right away, but with some time and patience, I ended up getting the job." Now, she's given a lot more power "which was something I wasn't used to as a regular archivist. It's fun transitioning into tech because I feel like I'm doing more."

Ashley has been continuing to develop her programming skills outside of NYPL as well. "I work on a couple of open source projects targeted towards moving image archivists called QCTools and MediaConch," she says. "I've been really proud and pleased with how much I've been able to contribute – not just using my skills as an archivist but also using my skills as a developer – and to see these robust applications being built even though I'm not a core developer on those projects."

Takeaway #3

Ashley's notes that for job-seeking bootcamp grads it can be hard to stay confident throughout the interview process. "Just try your best to really feel like you are awesome all the time, which I know is hard," she says. "And then, my second piece of advice – advice that Flatiron gave me – is that I think you should take the first job you're offered because you don't have to be at that job forever. It's a big field and you've just got to get keep learning – and the best way to do that is to start getting paid well to learn more."

What's Next for Ashley?

"I feel very grateful and fortunate to have been able to do so many things that I've wanted to do," Ashley says. Ashley is thrilled to keep learning and becoming a better developer at New York Public Library.

bouncex

Sam Owens: Working His Way to Senior Director of Product Development at BounceX

Sam Owens came to Flatiron School after a stint as a product manager in the fitness industry. After graduating from Flatiron School, Sam worked his way up at BounceX, going from software engineer to Senior Director of Product Development.

Learning to Code

Sam had a few reasons for initially learning to code. As he says, “I wanted to get closer to figuring out how things are actually built and getting to do some of that, and I really wanted to get into tech, a faster moving industry with more opportunities, more motivated people.”

Takeaway #1

Sam says new coders need to jump in with both feet. “It’s easy to have a kind of trepidation around it: you think about it, you do research, you think about it, you do research – without actually doing any coding. Once you know that it's the thing you want to do, just jump in and do it.”

The Flatiron School Experience

After deciding he wanted to learn to code and doing his research on his options, Sam felt Flatiron was a “no-brainer,” citing Flatiron’s focus on people and building a diverse, driven community of students. “Other schools seemed to be more about creating a rapidly scaling business model,” says Sam, “but I never really felt like I was a 1 or a 0 or somebody checking a box towards their machine scaling at Flatiron. I felt like it was a very human experience.”

Takeaway #2

For Sam, the most important thing he learned at Flatiron wasn’t technical – it was learning to be comfortable with being

… uncomfortable. “When you dive into the fully immersive world of learning to code, you constantly encounter impostor syndrome and ask yourself, ‘Can I do this? I don’t know what’s going on.’ I learned that being in that zone is actually where you're going to learn the most. There should never be a point where you think, ‘I've learned all the things. Time to just relax.’ You kind of have to always be putting yourself in that position, no matter where you are, and keep learning.”

Life at BounceX

For Sam, the biggest surprise he encountered at BounceX was how fast everything moves in tech – and “how much everybody *cares*.” When Sam joined the company, there were only 20 people on the team. “Now we’re in the 250-ish range,” he says. “So for me, it's been a pretty wild ride of seeing a ton of people work really, really hard to build something really, really cool, and I just never expected it to go like that.”

Sam’s transition from engineer to heading the product team was organic: as Sam slowly moved from coding 100% of the time as the company’s first junior engineer to managing a growing team of developers, BounceX’s CEO asked Sam to take charge of the rollout of a product. “I guess the way that I did that worked really well, and he was happy, so he kept handing me more projects. I got to a point where I was split between trying to do two different jobs, and I kept going further up the product life cycle chain.” BounceX’s CEO wanted him to move over fully to the product team, but Sam didn't want to just abandon the team he had built, so he talked to his boss about a transition plan. The result? “Now there's another Flatiron grad who's in charge of that team, which is cool.”

Takeaway #3

While Sam admits there was no “magic bullet” that helped him succeed, he thinks the biggest factor in his success was his ability to ask questions. “Go to your direct supervisor or your mentor and ask what you need to work on, ask what you need to learn. If you're a junior engineer and you want to drop the junior, ask whoever's the head of your engineering department what the difference is, what your gaps are. Don't work and hope somebody will notice. You should always be trying to figure out what the next steps are and work your way towards those things.”

What’s Next for Sam?

Sam says, “I've been on a pretty wild ride with BounceX, and I love it.” He aims to keep building and shipping products that drive undeniable value for BounceX’s customers. “That involves a lot of the same things I'm doing now, but also continuing to align our development and product processes with our growth trajectory.”

Mike Spangler: From Locations Coordinator to Software Engineer at NASA

Mike Spangler has had quite a unique career journey: after working as in film and television, he's now a software engineer at NASA's Jet Propulsion Laboratory. (Yes, *that* NASA.)

Learning to Code

Mike learned to code because he was seeking greater opportunity in his career. "I had an okay career," Mike says of his work doing logistics and locations in the film and television industry. "I was sort of happy, but I wanted to be *really* happy." Seeing how quickly the tech field was expanding, Mike realized learning to code could expand his horizons and "raise the ceiling on what was possible in my career."

Takeaway #1

Mike's advice to those curious about learning to code: don't wait. "There's a lot of opportunity right now," he says. " The startup scene is still hot. We still need new developers. If you're an engineer right now and you have a LinkedIn profile, they're beating down the door for you. If you're thinking about it, and you're pretty sure you're not going to be happy doing what you're currently doing, take the leap."

The Flatiron School Experience

Flatiron School caught Mike's attention after he saw a video of Dean

Avi Flombaum produced by the Bindery, a production company Mike worked for. “I decided to attend Flatiron school because I just loved Avi’s message of how coding was available to anybody who wanted to put in the effort and considered themselves creative and thoughtful – that it’s not just for math majors,” Mike says. “It’s something that can be learned by anyone.”

For Mike, what really made the experience of attending Flatiron School was being surrounded by incredible peers. “Everyone in my class, almost without exception, dedicated their life entirely to the immersive,” he explains. “They threw themselves in and almost none of them knew anything about coding. There was just this great sense of we’re in it together. This is hard and that’s okay. We’re going to teach each other. We’re going to build each other up, and we’re going to learn this thing. We’re going to change our careers and that’s going to allow us to change our lives. I think Flatiron really offers a sense of community. You have a social and personal stake in your own learning that goes beyond textbook. You want to do it for yourself and you want to do it for your classmates, too.”

Takeaway #2

The most important thing Mike learned at Flatiron School was how to be a lifelong learner. “I really keep that in my mind every day – being an open minded lifelong learner,” he says. “Because programming languages come and go, frameworks come and go, development methods like Agile, Scrum, whatever, those can all come and go. The one constant is your openness and ability to take on new information and use that to tackle the problem at hand. Flatiron did an amazing job of making that first principal of being a lifelong learner the most important first step to being a successful programmer for your whole career.”

Life at NASA

Working for NASA had always been on Mike's career wish list, but working in film and television, Mike says, "I never would have thought that that was somewhere I was going to be able to work or had any idea how I could work there in any capacity." Nevertheless, after moving to Los Angeles, he saw a job posting and submitted his resume and cover letter "like anywhere else, and just, you know, things happened."

Mike works on the search team at NASA's Jet Propulsion Laboratory. There are currently over 30 active missions; several of them are in flight on other planets or around other planets, including Juno at Jupiter, and the Curiosity Rover on Mars. Where does Mike come in? As he explains, "All of these missions generate a tremendous amount of data, whether it's project schematics or telemetry data, or even something as mundane as personnel details and records. What my job, as part of the search team is, is to parse through that data, make it searchable for our scientists and administrators to find and make actionable decisions on."

Takeaway #3

At NASA, Mike is surrounded by very well-educated colleagues – "bachelors, masters, PhDs in computer science, aeronautical engineering." Coming from a coding bootcamp, there's a lot of opportunity for impostor syndrome to take hold. But for Mike, what really surprised him was how "it doesn't matter if you've had 12 years of schooling or 12 weeks of bootcamp, you're still going to be on Stack Overflow a fair amount of times throughout the day. I guess I was surprised and relieved to learn that everyone is always learning

… and always figuring this stuff out. It's in some ways just as powerful to have the education that I got as the education of someone who went to a long degree program got."

What's Next for Mike?

Mike could see himself exploring product management in the future. "There are a lot of diverse stakeholders and a lot of different individuals giving input on what products should look like and what features it should have," he says. "I think it's a very interesting role to wade through these disparate opinions and figure out what are the critical functionalities and elements that an application should have."

FlatironSchool.com