SURVIVING THE CODING BOOTCAMP

From no coding experience to earning a six-figure salary

MOHAMMAD AZAM

To my beautiful wife, Naila

To my two monkeys Mehreen and Imaad

To my parents, who always believed in me

Hello Coding Bootcamps

I came to United States in 2002 as an international student. For the next 4 years, I spent every living moment either in a class room or working to pay off my tuition. I graduated in 2005 with a bachelors degree of computer science and went straight to work after graduation.

Looking back now, I realized that first one and a half years of computer science college were mostly useless. These years comprised of many courses that had absolutely nothing to do with computer science. It includes history, political science, astronomy, geology, western civilization etc. Those courses were only added just to extend the time a student spend in school so college can rake in more money and the student can go further in debt.

What if there is a school, which only focuses on skills you need to become a software developer. A school that will take 4-6 months to complete, instead of 4-5 years. A school that will cost a fraction of the price of a traditional college. A school that will have almost the same job placement rate as compared to a traditional college.

Welcome to the wonderful world of Coding Bootcamps!

Coding bootcamps are based on the principles of Ultra Learning. The main idea is that you will completely immerse yourself in learning a particular craft. This means you will go from crawling to walking to running in only a matter of few months.

Coding bootcamps specializes in training students on a particular stack. This includes web development, iOS development, machine learning, UI/UX, cyber security and many more.

Coding bootcamps also differs from a typical four year computer science degree, because they don't teach subjects that have no direct correlation with the work you will be doing as a software developer. This means bootcamps completely skip over the first one and a half years of college and provides relevant knowledge in a timeframe of 12-16 weeks.

Apart from saving, hundreds of thousands of dollars, you will also save a lot of time when becoming a developer through a coding bootcamp. Think about graduating 3.2 - 3.6 years earlier and starting your career as a software developer as compared to college.

Now that I got you pumped, I must confess that coding bootcamps are not for everyone. They are fast paced and everyday you will learn something new. Coding bootcamps gives you a drowning like feeling. You drown in information given to you by your instructors. Each new concept, framework hits you like a big wave. You only get enough time to swim to the surface and inhale once, then you drown again.

This book is designed for passionate people like you who wants to enter the wonderful world of software development. It does not matter whether you are attending a bootcamp or taking the self taught route. This book will provide you the guidance to land a job as a software developer.

In the last 5+ years, I graduated hundreds of new developers through my teachings. Some of my students are working for prestigious companies like Apple, JP Morgan Chase, Accenture etc. I am confident that with hard work, passion and perseverance, anyone can become a software developer.

Let's get started!

Types of Bootcamps

Coding bootcamps comes in many different shapes and sizes but the two of the most common are immersive and flex also known as full-time and part-time bootcamps.

Immersive bootcamps are an excellent choice for students who wants to completely dedicate every second of their next 12-16 weeks learning programming. These bootcamps are fast paced because there is a lot to cover in a short span of time. If you are planning to enroll in an immersive bootcamp then make sure you are not working on the side. Immersive, really means immersive.

Flex bootcamps are designed for students, who are unable to attend a full-time bootcamp. This may be due to their job or family commitments. Flex bootcamps usually run longer as compared to immersive bootcamps. Most of the flex bootcamps I have seen host classes 3 times a week for a total of 24-26 weeks.

The content covered in both bootcamps is mostly the same. The only difference is the pace and the length of the bootcamp.

Before joining a bootcamp, make sure you research, which type of bootcamp will fit your schedule and needs. If you have a job or taking evening classes then part-time bootcamp will work better, if you want to completely immerse yourself then a full-time bootcamp will be an excellent choice.

What makes a good coding bootcamp?

There are several components that makes a good coding bootcamp but the most important is quality instructors. Before joining a bootcamp you must perform a thorough research on bootcamp instructors.

This includes looking at instructor’s work experience, coding community involvements, open source contributions, apps, websites, speaking engagements, podcast appearances, YouTube videos etc. This will give you an idea about your instructor’s teaching style, communication skills and dedication to the online development community.

One of the other factors that can greatly effect the quality of teaching is your instructor’s range.

Range demonstrates how many different technologies, programming languages, frameworks your instructor is comfortable teaching. Usually instructors who have worked with many different platforms are better in explaining and breaking down the problems as they can always utilize knowledge of other platforms to explain concepts in a different way.

For example, over the course of 10+ years I have worked in several different technologies. I am comfortable developing and teaching in many different technologies including iOS, .NET, full stack web and even cross platform frameworks like Flutter.

Another thing to research is your instructor's community involvement. Every programming stack has a community, which comprises of passionate developers around the world. The community grows because of the contributions of its members. Check to see if your instructor is an active member and contributor to the community.

Platforms such as LinkedIn Learning, Pluralsight, Udemy and Skillshare can also help you to gauge instructor's delivery style. If you can find instructor's videos online then make sure to check them out. The videos will demonstrate your instructor’s teaching style, presentation skills and a lot more which can be beneficial for you when you are making a decision.

Depending on the class size, bootcamp may provide instructor with extra help in the form of teacher assistants. Teacher assistant’s (TA) job is to provide help and guidance to the students and also support the instructor. Good TA’s can create small practice activities, hold review sessions and support the overall class just like an instructor.

Another very important aspect you must consider before joining bootcamp is the class size. The smaller the class size, the more attention and one-on-one time you can have with your instructor and TA. I have found class size of 30 or less to be ideal. A class of this size can easily be managed by an instructor and 3-4 TA’s.

Some bootcamps also offer career support, which can prove to be an important service for a student to land their first job. Career support services can include resume writing workshops, interview preparation, portfolio design and review, crafting LinkedIn profile and much more. Although these things are not programming related but they are as important as coding skills because it helps you get a job.

I encourage you to spend time in researching a bootcamp before joining. The time you invest in researching a bootcamp will pay dividends in the future.

Cost

The average cost of a 16 to 24 week coding bootcamp ranges from $10,000 - $20,000. Data science bootcamps are priced higher as compared to web development bootcamps.

The average cost of college for a student living on campus at a public 4-year in-state institution is $25,487 per year or $101, 948 over the course of 4 years.

Apart from the astronomical cost of college, it takes an average of more than 4 years to complete a bachelors degree. A degree, which is not even required to land a job in the world of software development.

Few years back, I was a guest speaker at the computer science club at University of Houston. After my presentation, students inquired about the cost of a coding bootcamp. I told them that the cost is around $14K. One student asked me why is coding bootcamp so expensive. I asked him how much is he paying each year for his degree. He replied $20K and I giggled for a moment. I told him that not only he is paying more, but he is also paying in something that he can never earn back... time.

On average a student spends 4 years at a university, while coding bootcamp grads will be out in 4-6 months. They will get a head start of 3+ years. Think about all the experience they have already gained during that time over a computer science college student who is still in school learning about rocks (Geology) and the solar system (Astronomy).

No, I am not some evil villain who wants to ban all colleges. I am just against students spending money like congress, when attending colleges. If you can afford to attend college without going into debt then go ahead and do it. But if you are planning to make ugly Sally Mae your roommate for the rest of your life, then I would strongly advise against it.

In the next section, I will go over my own college experience and how much of knowledge I learned in college was applied in real life.

* Average cost of college

Is CS degree necessary to become a software developer?

I graduated from University of Houston in 2005 with a Bachelors of Computer Science degree. It was the toughest 4 years of my life because I had to work in order to pay for college. I remember working multiple jobs every day and even working till 2:00 AM on the weekends. Actually there was a running joke at my place of work, when anyone was not able to cover their shift they came straight to be, because they knew I will never say no to extra work.

All my hard work paid off, because I graduated with zero debt to start my career as a software developer. Fortunately, I had a job waiting for me at the time of my graduation. It was not due to my studies at the university but it was a directly related to the self study I did outside of the university.

Looking back at what I learned at the university, I can say with 100% confidence that I only use 0.0000000000000001% of that knowledge in real life. For me personally, I had to unlearn a lot of bad programming practices that were taught at the university. This included writing comments for every single line of code, more code means better code, unit tests are not necessary etc.

It was not surprising to me since most professors had absolutely no real life programming experience. Their path for becoming an instructor was more educational rather than pragmatic. Most professors completed their bachelors, enrolled immediately into masters and then finally completed their PHD program, which made them eligible to land a job at a university. I really hope one day new requirements will be set, where university professors have to show at least 3-5 years of real world experience. I strongly believe that working in an industry before teaching can help them become better teachers.

As an instructor at a coding bootcamp, I see students from many different backgrounds. I have taught programming to real state agents, scuba divers, welders, school teachers, medical professionals, dentists and many more. None of them went to school for a computer science degree but all of them are now working as a software developer.

Programming is one of those rare professions where a degree is not required. You can learn programming completely on your own through reading books, watching videos or attending a coding bootcamp. This is obviously not true about many other professions, specially medical. I will not be comfortable if my doctor never went to medical school and learn how to perform surgery by watching YouTube videos.

Another thing to point out is that many CEOs of technology companies including Tim Cook, Mark Zukerburg and Elon Musk have openly said that when they hire a new employee, they don't require them to have a degree. They just want to make sure that the employee can do the assigned work and be productive.

The college that you attend has no direct correlation with your success in life.

Having said that, there are companies that do require that you have a degree. These companies operate in categories which includes oil/gas, energy and even educational institutions including coding bootcamps. Don't get me wrong, I am not against going to college. I am just not in favor of pilling up massive debt to attend college which could have easily been avoided. If you can attend college and graduate without debt then you should definitely go to college. Let me repeat that again.

If you can attend college and graduate without debt then you should definitely go to the college.

Unfortunately, graduating with a four year degree without any debt is getting more and more difficult. I still believe you can graduate college without any debt. Here is some advice to lower or even avoid your college debt all together.

1. **CLEP**: You can take CLEP (College-Level Examination Program) exams to pass all the courses required for an associate degree. At the time of this writing a single CLEP exam cost $89. This means you can pass a course and get the credit for just $89. In order to pass and transfer a CLEP course to your designated college, you need to score at least C grade or above. Keep in mind that GPA of courses cleared through CLEP is not added to your commutative university GPA. CLEP is simply a pass/fail exam and has no effect on your final grade point average.

2. **Community College**: I am not sure why more students don't enroll into community college. Community colleges are a great way to complete your associates degree. This means you can take all the basic courses at community college and then transfer them to your main university.

Community colleges cost a fraction of price as compared to the university and provides an excellent education. I attended multiple community colleges during my college years and saved a fortune on tuition fees.

3. **Stay Off-Campus:** Students typically spend $10K - $15K on on-campus accommodation. This is insane amount of money and can be avoided by living off-campus or with your parents. I lived off-campus, walking distance from the university and saved tons of money. I used that money to pay my tuition.

Just across the street from my apartment, I saw university sponsored dorm rooms, which cost more than 4 times as compared to my apartment. I always wondered how those students were able to afford such a hefty price tag.. Oh wait I remember, good old student loans.

4. **Work**: Wait what! You mean you can work while going to school. Of course you can! I worked all four years of my college. I scheduled my classes on few days a week, which left other days wide open for me to work. I sold University of Houston t-shirts, I worked at a bowling alley, I worked at an administrative office at university center, I worked as a web developer for bowling website, I worked

as a C developer for satellite pool table management application and even worked as a boring librarian at an optometry library.

The best place to go when you need money is work

Working while in school will not effect your grades, actually it has been proven that people who work at least part time get better grades than people who don't work at all.

5. **Books**: I believe the prices of college books is the biggest rip-off perpetuated on poor students. I have seen college books costing as much as $200 per book. There are couple of ways to save money on college books. First you can share the cost of a book with your friend(s). This means you can do group study and pay only portion of the book instead of the full price. Second, you can buy an older version of the book which is almost 95% identical. You can even rent books through Amazon and Chegg instead of buying them at full price.

6. **Scholarships**: Did you know that each year billions of dollars of scholarship money remains unclaimed. This means that students never applied for scholarships. Make sure to start applying for

scholarships even before your college starts. I remember reading a story about an immigrant girl named Kristina Ellis who applied for 40-50 different scholarships and received over half a million dollars in return. She spent that money wisely and attended her college of choice and never took out a loan. Luckily, you can read about her experience in her book called "Confessions of a Scholarship Winner - The Secrets That Helped Me Win $500,000 in Free Money for College - How You Can Too!".

If you just follow the above advice, you can save considerable amount of money and you will be one step closer to graduating with zero debt from a university.

Bootcamp Grads vs Computer Science Majors

SwitchUp is an online coding and computing programming platform, which keep tracks of bootcamp ratings and reviews.

According to the data gathered by SwitchUp, it is clear that highly rated coding bootcamps go head-to-head with top technology schools. In some cases, bootcamps offer higher in-field employment rates as compared to universities.

It is also noted that coding bootcamp graduates are less likely to pursue further education as they are able to find in-field employment.

And finally, coding bootcamps lead tech employment rates over UC Berkeley, University of Southern California, Brown and even Princeton.

There are several other studies, which came to the same conclusion. Hopefully these studies will help you reach a decision, when choosing between a computer science degree at a university or a coding bootcamp.

* New data shows which bootcamps have higher tech employment rates than The Ivy Leagues

Bootcamp Reviews

Humans are weird, when purchasing a $20 item from Amazon, we will spend hours researching and reading the reviews to make sure we are getting the best deal and a quality product but the same level of hard work and research is not applied when enrolling in college or even bootcamps.

This is the main reason students graduate with useless degrees that does not provide return on their investments. Who knew that underwater basket weaving is not a high paying profession.

There are several websites that are dedicated to publishing coding bootcamp reviews. Two of the most popular websites are CourseReport and SwitchUp. Both websites allows users to search for a particular bootcamp and read reviews. CourseReport reviews can even be filtered based on location, published date and even city.

When reading through the reviews, make sure that you are looking at reviews for the actual program you are planning to attend and not the reviews for practice exams or pre-work. Some bootcamps offer free of cost introductory classes to their students to boost their ratings. This is the main reason you should closely looking at the reviews of the actual program rather than introductory classes or pre-work.

Another great way to gauge the quality of a bootcamp is to contact an alumni and ask for their opinion. LinkedIn is a great resource to track down bootcamp graduates. Always make sure to be kind, respectful and introduce yourself and explain the purpose of your communication. Bootcamp alumni are the best validator of a bootcamp. If they had good experience they will encourage you to take the leap, if they had a bad experience then they will warn you to keep away.

I encourage you to allocate some time in reading reviews. Bootcamps cost ranges from $10K - 20K so even if you spend couple of hours, researching and digging into the reviews it will be worth it.

Remote Bootcamp vs In-Person

There are several different types of coding bootcamps, but the most common are in-person and remote. I had the opportunity to teach at both kinds of bootcamps. Each comes with its own set of advantages and disadvantages.

Before pandemic, the bootcamp I taught was primarily in-person. This means students and instructor will physically be present at a designated location, where the classes will be held.

The in-person bootcamp played an important role by providing the social aspect to the students during the bootcamp. Students got to know each other better and even formed study groups. The in-person bootcamp increased collaboration between students and allowed them to share knowledge in a more seamless way.

One other benefit of in-person bootcamp is the environment of the offices. Our bootcamp was located in a shared co-working space. This allowed students to interact with other companies and even land jobs with companies located in the co-working space. I had several of my students accepted job offers from companies in the same co-working space and they are still enjoying their work at those companies.

In-person bootcamp does come with few limitations. The biggest limitation being in-person. This means that students needs to be physically in the class room. This can be quite challenging to students who are not located in the same geographic region. I had few students who had to rent an apartment to be part of the in-person class.

Another disadvantage of in-person class is the commute. Our physical bootcamps were located in Houston and Atlanta. Houston is known for its traffic issues and long commutes. Depending on which part of the town you live in, it can take you more than an hour to drive across town. Apart from long commutes, city is also susceptible to flooding, freezing and other natural disaster. All of these events can prevent the instructor and the students to attend in-person bootcamp.

At present our bootcamp has moved 100% remote and there are no plans of returning to in-person bootcamp. Personally, I enjoy remote bootcamp because I save a lot of time not commuting. Another great benefit of remote bootcamp is that students from all around the nation can join in and attend the bootcamp. With remote bootcamps, geographical limitations do not apply.

One main disadvantage of a remote bootcamp is that students are not able to socialize with each other. At least not in the same sense as when attending an in-person bootcamp. One way our bootcamp solved this problem is by opening satellite offices in several parts of the nation. This allowed students in that region to bring back the social aspect of the bootcamp.

Pre-work

Pre-work is the work you should complete before the first day of class. For a web development bootcamp, it can involve introductory lessons on Git and GitHub, Python and even HTML and CSS. Not all bootcamps offer pre-work but if your bootcamp does then make sure to follow through with it.

Pre-work allows students to get a taste of programming before the cohort begins. This is where you will get your feet wet before taking a deep dive. If you have never done programming then pre-work will allow you to get comfortable with the initial course load. Most bootcamps also only supports macOS operating system. This means that if you have been using Windows OS all your life then it will be a little bit of a learning curve to get comfortable with a Mac. Make sure to utilize pre-work time to get yourself familiarize with the designated operating system. If you are going to wait till the cohort begins then it will be too late. Bootcamps moves at a fast pace and it is best to focus on actual development during the cohort rather than getting comfortable with the operating system.

In one of my cohorts, several students did not bother to complete their pre-work. They also never took the time to get comfortable with their operation system. Not surprisingly they struggled throughout the whole cohort.

The bottom line is that pre-work is extremely important. You must complete the pre-work because if you don’t, there is a good chance you will face more hurdles during the class.

Bootcamp in Session

Congratulations, you are now enrolled in a coding school. Hopefully, you have done intensive research on various bootcamps and selected the one that best matches your interests and needs. Now you can sit back and relax for the next 16 weeks and after 16 weeks you will magically become a software developer.

Unfortunately, that is not how things work in real life. This is like saying that you can get a membership at a gym and you will automagically become ripped in few months. In order to see the results, you must work hard day and night. It will not be easy, but it will definitely be worth it.

You will spend considerable amount of time away from your family and friends. You may not be able to binge watch your favorite shows on Netflix or HBO Max. You are going to sacrifice a lot during your time in a coding bootcamp, knowing that your hard work is going to pay off in the future.

In this section, you are going to learn, how to graduate from a bootcamp as a top student and learn the necessary skills to become a successful software developer.

Let’s get started!

Programming is not an open heart surgery

Even though I have been teaching at coding bootcamp for a very long time, one thing I witness in each cohort is student's fear of writing wrong code. Students would rather jump off a building than write wrong code. I think this is deeply rooted to our education system, where different and unique answers are punished instead of being appreciated.

In one of the TED talks, Sir Ken Robinson talked about a young girl who was considered hopeless by her teachers. The teachers contacted her parents and told them that they believe that the child has learning disability and she is not able to concentrate in class.

Her parents took her to see a specialist. The specialist discussed talked to the parents about their child's condition. After a while, the doctor left the room with the parents leaving the kid alone. As they left the room, the specialist turned on the radio.

To everyone's surprise, the little girl started dancing to the music. The doctor turned to her parents and told them that the girl is not sick, she is a dancer and you should take her to dance school and they did.

That little girl was Gillian Lynne, legendary English ballerina, dancer and choreographer. She is responsible for some of the most successful theater productions in history and she is a multi-millionaire.

* <u>Do schools kills creativity?</u>

Like Gillian Lynne, we all have creative genes in our body. Unfortunately, this creativity starts to fade away as we enter schools and colleges. Students are punished for having creative thoughts and are told to follow the yellow brick road. Colleges focuses solely on how students can get good grades instead of a good education.

I always tell my students that writing code is not like an open heart surgery. If you make a mistake, nobody will die. The worst that can happen is your editor is going to give syntax errors, presented as a learning opportunity.

Nobody writes perfect code first time. Everyone makes mistakes. The most important thing is that you learn from your mistakes and improve yourself to become a better developer.

If you are afraid to be wrong, you will never come up with anything original

Sir Ken Robinson

Curiosity also plays an important role in becoming a good developer. Instead of following a predefined path, try to experiment what will happen if you change few lines of code or if you adjust the ordering of the lines. Don't get discouraged if the path you choose hits a brick wall. You didn't fail in solving the problem, you just learned several other ways your program does not work.

As a new developer your main goal is to get the program working. It does not have to be the best and the most efficient code, it just have to work correctly. After you get it working then you can make it better through the process of refactoring.

Refactoring is a principle of changing existing code to make it better without changing the overall functionality of the code.

Next time you are writing code, explore your curiosity. Don’t be afraid to write wrong code. You never know where a little curiosity will take you.

Micro tasks

The year was 2004 and I was taking a .NET web services class at University of Houston. My assignment was due in few hours and I was stuck. I just could not comprehend the scope of the assignment. Each time I begin implementing the solution, I got overwhelmed. I felt like standing in front of a huge mountain. Mountain so high that I cannot even see the peak.

I decided to take a break from coding. I knew that my method was not working and I need to try something different. I took out my notepad and wrote down all the steps I needed to take in order to complete the assignment. After that I focused on one single step at a time. Every time I complete a step, I crossed it out. Soon I was able to complete all the steps, which in turn completed the assignment.

The lesson that I learned from this experience is that one of the most important skills you can learn as a software developer is to break down a large task into micro tasks. Micro tasks represent a very small subset of a larger task. Once you finish all the micro tasks associated with a particular task, your primary task is completed. This technique allows you to move much quicker since you are focusing on a single problem.

New developers usually look at a programming problem and get overwhelmed by the sheer scope of it. Try to break it down into smaller manageable pieces and then only focus on the smaller piece.

Dividing a large task into micro tasks is a skill that you will develop as you gain more experience. My advice is to always write down your tasks on a piece of paper (Remember paper, that comes from trees) and then check them off as you finish them. This will visually show you which tasks are completed and more importantly it will give you the confidence to move forward and attempt other tasks in the list.

A task can be as simple as displaying few text boxes on the screen or even changing the background color of the screen. The main idea is to break down to a granular level that you are comfortable with and are able to complete it in a short amount of time.

Another question my students ask is that should they attempt the easier task first or hard. My recommendation is to always start with an easier task first. The reason is that once you finish the easier task, it will give you the confidence to tackle hard tasks. If you attempt the hard task first and get stuck then even the easier tasks will feel challenging. The confidence you gain from completing easy tasks will provide you the momentum to face and conquer hard tasks.

Next time you are about to write an app, write down all the tasks associated with your app. Then take each task and divide it further into micro tasks. Now you can start with each micro task and only focus on that one micro task. You will notice that using this approach you are going to complete tasks much faster and it will provide you the encouragement you need to move forward.

Step away from the computer

As a new developer, you will find programming a very frustrating profession. Sometimes even missing a single character while coding can result in your app not working. Syntax errors simply indicates that you require more practice.

Once you are comfortable with the syntax, you will run into errors related to the logic of the application. New students typically spend hours trying to resolve these issues. I had a student who spent 2-4 hours in front of the computer on a particular problem only to find a solution when he was doing his daily evening walk.

My advice is that if you get stuck on a problem for more than 35-45 minutes then simply stand up and walk away. Taking a small break is the best thing you can do to find a solution.

During my 15+ years experience as a software developer, I have encountered countless problems with complicated solutions. Whenever I got stuck, I took a break and stepped away from the computer. In almost all cases I found the solution when I was away from the computer. These solutions just came to me when I was taking a shower, enjoying a walk in the park and even swimming.

Next time you are stuck on a problem, step away from the computer. Go for a walk, make delicious coffee or take a shower. The solution will come to you when you are not in front of the computer.

Fitness

As a software developer, you will spend most of your time sitting in front of a computer and coding. Overtime this can take a toll on your body. No matter what anyone says, your health is the most important thing. I read it somewhere that sitting is now considered the new cancer and I completely agree with that statement.
When you wake up from a good night sleep, you have 100% battery. Your battery declines as the day progresses and as you use your brain power to solve problems. This is the main reason you feel tired at the end of the day, even though you did not perform any physical activity. The only way to recharge your batteries is to unplug yourself from the computer and plug into a physical activity.

You don't have to start lifting heavy weights, a simple 30 minute walk in the park can be a great start. I recommend exercising early in the morning when the weather is nice and parks and gyms are not too crowded. Physical activity is a great way to start your day. You will notice a surge of energy throughout the day, ready to take on the next challenge.

Fitness not only includes physical activity but also mental activity. You need to make sure your mind is getting enough rest. This is accomplished by a good night sleep. According to a study, 50-70 million US adults have a sleep disorder. That is a lot of people not getting enough sleep. Make sure to get at least 8 hours of sleep every day. This will ensure that your mind is fresh and it will help you think clearer and focus on the problems at hand.

In the end your health is the most important thing in the world. You can write the best code and implement the fastest algorithm but if you don't have health then you have absolutely nothing. Make sure to put your health on the highest priority.

Google and StackOverFlow

One of the main aspects of a coding bootcamp is to teach students to be independent. As a student you should take advantage of your instructor's knowledge but at the same time you should work towards becoming independent.

Information is now readily available at your finger tips. Every coding question you have has already been answered and just waiting to be discovered. You no longer have to drive to a local library and search through magazines and books for a solution. You can simply go to Google and type in your question and voila!

Apart from searching for your answer on a search engine, you should also be comfortable with asking for help. There is nothing wrong with asking for help when you need it. There are several different forums where you can post your programming questions.

I remember taking part on ASP.NET forums, while I was still in college. At that time I was learning web development so I posted a lot of questions on the forum. Later on, after I gained more experience I started helping people out. My contributions were recognized and I was awarded the Microsoft Most Valuable Professional (MVP) award in 2007.

Google also has similar programs (GDE), which are awarded based on developer's contributions to the community. If you are awarded MVP or GDE then it will certainly look good on your resume.

When asking questions on forums make sure to formalize it correctly. Don't just paste 200 lines of code and ask why it does not work. Questions like that are regularly downvoted on online forums and if you continue posting such questions without any context then there is a probability that your account will be suspended.

You need to explain the exact problem you are facing and what solutions you have tried. The more context you are going to provide, more experienced developers will be willing to help you.

StackOverFlow has prepared a document, which illustrates how to ask a good question. You can read that document here.

Never Copy Paste

As a new developer, all the programming related questions you will have has already been answered. They are just waiting to be found. Depending on your searching skills, you will be able to find the answer on different forums. But you should always restrict yourself from simply copying and pasting the code from online resources into your code editor.

When you copy-paste code, you skip over understanding the workings of the code. This is same as reading a whole book about coding without typing a single line of code or reading a cooking recipe book without ever heating up the pan.

Even with a decade of experience under my belt, I never copy-paste code directly into my editor. I always type it out and during this process, I try to understand what the code is doing. By typing out code one statement at a time, you understand it much better as compared to just pasting it and hoping for the best.

Next time you are stuck at a problem and you find a solution online then try to type it out instead of copy-pasting it. Using this approach you will learn better and you will be able to understand the inner details of the code.

Debugging

According to Google, Debugging is defined as a process of detecting and removing of existing and potential errors (also called as ‘bugs’) in a software code that can cause it to behave unexpectedly or crash.

As a student, you should pay special attention to debugging, as majority of your time at work will be spent maintaining and fixing bugs in an existing legacy codebase.

In one of my earlier cohorts, I was teaching debugging in web development using Chrome developer tools. I strongly suggested all my students to pay close attention to the debugging techniques discussed in the class. Some did and others did not. Few months after graduation, one of my students from the same cohort came back to thank me for teaching debugging. He told me that debugging skills that he learned in class helped him fix a lot of bugs and allowing him to solve complicated problems at his new job.

It does not matter what programming language or platform you are working with, all of them provides debugging features. Debugging might feels boring and it might even feel like a slow process but when applied correctly, you can detect bugs much quickly as compared to good old console logs.

My simple rule is to locate the problem or bug using logs for the first 5 minutes. If I am not able to find the source of bugs then I resort to debugging.

Pet Projects

One of the best ways to boost your learning during a coding bootcamp is to work on your side/ pet project. This project is not meant to be associated with the coding bootcamp graduation requirements. The main purpose of your personal project is to integrate all the skills you have learned during the coding bootcamp into your own application.

This means your side project should be something that you are passionate about. Your passion will drive the progress of the project. One of my students from an earlier cohort had passion for astronomy. He used the concepts discussed in class to implement a custom iOS application to keep track of Earth's distance from various sources. During the class, when we learned about how to consume a JSON API, he immediately went to NASA's website and found a relevant API for his needs. Using the concepts that he learned in class, he was able to integrate the API into his own iOS app.

It is always encouraging to see your project growing as you learn new skills. It also makes you confident that the knowledge you have learned during the coding bootcamp is practical and can be used to create real world applications.

So go ahead and start your passion project on the side. When you acquire new knowledge through bootcamp learnings, apply it to your project. It is a great feeling when you can see your project growing right in front of your eyes.

Once the project is mature enough you can include it in your resume. Employers are always interested in project based work from potential employees. You never know, the difference between you and other candidate might come down to your pet project.

Certifications

One of the most commonly asked questions is around the value of certifications in software development. In my opinion there is no harm in getting certifications. Certifications can give you a slight edge over other candidates during the start of your career but their value diminish as you gain more experience.

It is important to note that certifications only matter if you took the time to study the material and practice the concepts. I have seen developers, who passed the certification exam through the use of exam cram books but when the time came to practice those concepts they failed miserably. If you are planning to take certification exams then make sure your main goal is to gain knowledge and not just to pass the exam.

There are hundreds of certifications available in software development. The certification you choose depends on your area of interest. Some of the biggest certifications in software development are from Microsoft, Cisco and Amazon.

Sometimes your employer can also pay for your certification exam provided that you pass the exam. If your employer is offering to pay for an exam then I would definitely encourage you to utilize this benefit. I am a strong believer that you should take advantage of all the different benefits provided by your employer. This can be in the form of certifications, trainings and even college degrees.

5 Hour Rule

We live in an age, where information is easily available to us in many different formats. Unfortunately, even with access to unlimited information we tend to waste our time on social media and other meaningless activities.

I always encourage my students to follow the 5 hour rule. The 5 hour rule suggests that you should spend at least 5 hours a week reading, reflecting and sharpening your skills. A lot of successful people follow the 5 hour rule including Bill Gates, Warren Buffet, Mark Cuban and Oprah Winfrey.

Information can be consumed in many different ways, this includes books, blogs, audio books, podcasts and even videos. The main purpose is to form a habit of setting aside some time each week to consume knowledge. If you are reading a programming book then make sure to try out the code discussed in the book. If you are watching a video course then make sure to take a break and practice what you are learning.

The investment of your time is going to pay off in the future as knowledge compounds over time. You will always be up to date and learn a lot using this principle. And since most people don't invest time in consuming information, it will always give you the edge when moving up the corporate ladder.

One of the most important aspects of the 5 hour rule is reflection and practice. It means that after you read an interesting article or watch an informative video, make sure to try out things you learned from the resource. Once you try out things and apply the principles then you will understand it much better and it will resonate in your mind.

Also make sure that when you are applying 5 hour rule, you are free from all distractions. You have to be 100% committed and focused on the task at hand. Keep your phone away, close all social media websites and apps. I would even go as further as disconnecting yourself from the internet. This is your time to learn and reflect, don’t waste it.

Newsletters

A million years ago, people used Google Reader to subscribe to RSS feeds and read their favorite blogs. I was one of those people and loved reading my subscriptions in the morning, with a hot cup of coffee. Good Reader was later killed by Google and ended up in the Google graveyard.

Please have a moment of silence for respect...

Nowadays, even though you can curate your own feeds. Most people rely on newsletters to stay updated. I love newsletters and I have subscribed to several of them. The main idea behind newsletter is that someone else is in charge of curating weekly articles/videos on a particular topic. As a subscriber to the newsletter, you only need to provide your email address and that’s it.

Once a week, on a designated day you will receive a list of curated content through your email. All you need to do is open the email and go through the articles that interest you. Most newsletters take liberty to write an abstract for the article. This will give you a much better idea about the article contents and if you will be interested in reading it further.

No matter what technology stack you are using to build an application, you can always find a newsletter pertaining to your needs. I have personally subscribed to newsletters ranging from machine learning, React, iOS development and even Flutter. I highly encourage you to subscribe to a few newsletters to get started, you will be amazed at how easy it is to keep up with all the latest happenings in the industry.

Make sure you allocate weekly time to go through your newsletters. I go through my feeds during early morning on the weekends. By checking my newsletters on a weekend, I get the advantage of having all the updated feeds, since most newsletters are sent during the weekdays.

Searching for newsletters is quite simple too. Simply go to Google and search for your favorite topic i.e “React Newsletter”. You will be presented with several different options. Subscribe to the ones you find interesting and then sit back and enjoy with a hot cup of coffee.

Career Week

The purpose of coding bootcamps is not only to provide you with technical knowledge but also guide you in your career. Some bootcamps offers career week during the cohort. Career week is used to prepare students for the job market. This preparation includes resume, portfolio, LinkedIn profile, GitHub profile, mock interviews and more.

If your bootcamp offers career services then make sure to take advantage of it. Even though career week is not targeted towards learning technical knowledge, it is as important as technical skills.

The work that you do in career week can form the basis of your future employment. The bootcamp where I teach, a designated career support specialist will give sessions on how to write your resume, improve your portfolio and polish your LinkedIn profile. They will even validate and check your resume for errors and give suggestions on how to improve it.

The common denominator among my most successful students, who were able to land their first job before anyone else was that they made sure that their career week requirements were fulfilled.

If your coding bootcamp offers career week and career support then make sure to take full advantage of this benefit. I have seen great success among my students, who took advantage of this benefit and were able to land their first developer job.

Resume

Resume writing is an art and you may have to perform multiple iterations to get to a decent looking resume. Resume experts recommend that your resume should be at most 2 pages long even if you have been developing for a while.

For a junior developer like yourself, one page should be the limit. Your resume should highlight your current skills and the roles you are looking for. If you have previous work experience then make sure to highlight the experience relevant to the role you are seeking.

Resume come in many different styles. Some will have a summary section on the top, while others will simply have your contact information. Some will display experience as a short paragraph, while others will display it in the form of bullet points. If you want, you can make multiple copies of the resume with different styles. Some recruiters prefers one style of resume over the other.

RayWenderlich website offers few resume examples for iOS developers but the same structure can be applied to any programming resume. Make sure your resume looks good in a PDF and printed format. You should also always have an updated copy of your resume online as well as printed on a physical paper.

There are some online websites that provide professional resume writing services. These services are not cheap and can cost an average of $200-$500. If you do plan to use these services, make sure your resume writing expert is familiar with writing resume for software developers.

Some of these services are listed below:

1. ResumeSpice
2. ResumeWriters
3. Zipjob

4. Find My Profession
5. Resume.com

* Source (https://www.cnet.com/tech/services-and-software/best-resume-writing-service/)

Portfolio

Nothing demonstrates your work better than a portfolio. Portfolio allows you to showcase your skills to an employer and make an immediate and a long lasting impression. I always advice my students to feature their main projects on their portfolio.

Although you can invest your time in implementing the portfolio website yourself but my recommendation is to use a low cost template as a starting point of your portfolio. You can easily purchase a very good portfolio template from ThemeForest or similar websites for under $15. When purchasing a template, make sure to select a basic HTML, CSS template. This will allow you to easily change the contents of the template and publish it on all available hosting providers.

After completing the portfolio your next step is to deploy it and make it available to the rest of the world. A static website can be deployed to many different hosting providers including Surge, Netlify, GitHub Pages and infinite others. Apart from hosting, you should also invest in getting a custom domain name. Once again there are several different options. I recommend hover.com as they provide excellent customer service and don't upsell you on additional features.

Make sure to keep your portfolio up to date. As soon as you finish a project, add it to your portfolio. Don't wait months or years to update your portfolio. It will take double the time if you wait, do it while it is fresh in your memory.

Group Projects

One of the main reasons students join a bootcamp is to work in groups with other developers. During your cohort, you may work on several group projects. In some projects you will be responsible for picking your own team and in others a team will be randomly assigned to you.

Group projects are the ultimate test of your technical and communication skills. In each cohort I have witnessed that students who usually work exceptionally well on their own, face challenges in a group environment. Some of these challenges are based on using GitHub as a team but most of them are related to communication issues.

When students are paired randomly in a group, it is common to have strong and weak members in the same team. Sometimes weaker students don't contribute much to the team, which makes stronger team members annoyed and puts pressure on them to complete the project.

Here is my simple advice. If you are paired with team members who are technically stronger than yourself, then try to learn from them and make sure you still contribute to the project. These contributions can be working on the user interface, css, source control read me files, hosting ...anything. Find small tasks and complete them. This will give you the confidence you need to take on the challenging tasks in the future.

On the other hand, if you are a strong team member, then try to act as a leader and educate weaker members of your team. Help them achieve a goal and provide them guidance. If they still don't feel interested in contributing anything to the project then instead of getting upset, see the positive sides of this scenario. One positive aspect of this case is that you get to learn a lot more than originally anticipated. This will be very helpful when you are applying for your first job.

When I was a student in college and working on a group project with 2 other team members, none of my team members contributed anything to the project. I was upset but I looked at the brighter side of things. Since, I was the only person who worked on the project, I ended up learning more. This resulted in gaining more knowledge, which helped me land a job before any of my two team members.

Always try to see the positive side of the situation

Hosting Projects

Once you have completed your individual or group project, it is your responsibility to make it publicly available. This process is also known as deployment. You need to make sure that employers, recruiters and other decision makers can experience your hard work.

For projects that are composed of static HTML, CSS and JavaScript with no authentication, you don't have to take any additional steps. You can simply deploy them to any of the free hosting websites like Surge, Netlify, GitHub Pages and many more. Things get complicated when your project is server based and requires authentication. This means that in order for a user to experience your website they must register and login. Unfortunately, not many employers will be interested in creating a new account to witness all the amazing features provided by your project.

Fortunately, solution is quite simple. You must provide a "**Login as guest**" button on your website. When a user clicks on the button, they will automatically be logged in using a preconfigured account. Make sure that the guest account has some dummy data associated with it, so it looks like a real account.

Employers receive hundreds of applications for each job posting. This means that each applicant only get couple of minutes to make a good impression. If your website requires registration without any flexibility of a guest login then most of the employers and recruiters will simply move on to the next candidate.

In the end you never know, your time investment of few minutes in implementing the guest login may end up landing you that job opportunity.

LinkedIn Profile

LinkedIn is a network for professionals used for networking and career development. It allows job seekers to post their resumes and employers to post jobs.

It is very important that you invest time to polish your LinkedIn profile as it is the largest network for professionals in the world.

Here are few steps you can take to make sure your LinkedIn profile stands out.

1. **Profile photo**: As mentioned before, LinkedIn is a professional network so make sure your profile photo reflects that. Profile photo is your opportunity to make a good impression on the employer. If your photo is unprofessional, then the employer will ignore you even though you may have technical skills to do your job.

Make sure photo is not grainy or pixelated. You don't need any expensive camera to take a great photo. These days you can take a great photo using your smart phone camera. Also don't take selfies as your profile photo. Ask someone to take photo for you.

A great photo goes a long way and apart from adding it to your LinkedIn profile, you can use it on your online portfolio, social networks like Twitter, Facebook etc and even when submitting open papers for a conference.

2. **Headline**: This 220 character or less screen real-estate is one of the most important section of your LinkedIn profile. This is where you will communicate about your current job title and what you do at your work.

If you are looking for job opportunities then don't write "unemployed". Instead try to communicate what you can do. So instead of "Unemployed full stack developer", you should say "Full stack developer seeking new opportunities".

You need to make sure to utilize the most of 220 characters available to you. Here is an example of a great LinkedIn profile headline.

Steven Smith
Full Stack Developer| Seeking Full Time Software Job |JavaScript, Node, Express| July 2022 Graduation

The above headline is packed with a lot of information. This is a great way to take advantage of the limited characters provided to you.

3. **Experience**:

In the experience section, you will enter all the relevant experience ordered by date in descending order. If you want you can copy and paste this portion of your profile from your resume.

Make sure the experience reflects the position you are seeking. This means you don't have to include your flipping patties experience at McDonalds, if you are looking for a programming job.

You should definitely include all your projects you have completed at the coding bootcamp, along with their GitHub repositories and link to the live website.

4. **Skills**: The skills section on LinkedIn allows you to add up to 50 skills. These skills can include programming languages, frameworks, platforms etc. Make sure to add skills specific to the type of job you are looking for. This means if you are looking for an iOS developer position then add "Xcode, Swift language, Core Data, Objective-C"

etc. Be honest about your proficiency for each skill you add to your profile.

After adding your skills, you must also try to get your skills endorsed. Getting an endorsement from your peers allows your profile to rank higher on LinkedIn search results. One of the best way to gain endorsements is to endorse your connections and hope they return the favor. In my experience, most of the professionals on LinkedIn return the favor and endorse the endorser.

GitHub Profile

If there is such a thing as a resume strictly for developers then it would be your GitHub profile. GitHub profile hosts all your coding projects and allows prospective employers to evaluate your technical skills.

A great GitHub project profile explains the purpose of the project clearly. It also list all the different technologies and APIs that were used in the project including live link and tags. If the project was UI/UX based then it should also include screenshots of the website. If it is a framework then it should consists of documentation on how to use the framework and even code samples. If your project is based on effects and animation, then it would be wise to add a small video or even a GIF animation showing the effect in action. This way viewers can simply watch the video and get a better understanding of your animation library, without spending time to download and install the framework.

During bootcamp you may be submitting your assignments through GitHub. This means your GitHub will get populated with lots of different repositories. You can avoid repository overcrowding by pinning your most important projects, this will make sure that only your pinned projects appear on your GitHub home screen.

Polishing your GitHub repository is mandatory in development world. If your repository does not even have a decent README file then it will send message to your audience that you don't care much about your work.

Another things to keep in mind is not to publish your API keys or sensitive data in your GitHub repositories. Provided that your repositories are listed as public repositories, anyone can see and use your keys. Always make sure that your sensitive data is encrypted and not published online.

GitHub is your developer resume. You must take time to make it stand out. It is not surprising that most students in my class who landed job first, had a polished GitHub profile. One of those students was Beyza Kilickol and her GitHub profile can be found here. As you can see Beyza worked extremely hard on her profile and made it easy for employers to experience her projects. She even added screenshots, videos, guest account login to streamline the experience. I encourage you to follow her GitHub profile as a template and learn how she has laid out her repositories.

As developers we are always focused on coding and find these tasks meaningless. But in reality, they are very important and can be the difference between you getting the job or not.

Mock Interviews

You may be good in programming and solving complex problems but not comfortable during the job interviews. It really depends on your comfort level and whether you are introvert or extrovert. Nevertheless, you must take steps to conquer your fear of interviews. I believe anyone can become good in interviews with little practice.

When researching a bootcamp, try to find if the bootcamp helps with mock interviews. This will allow you to practice your interviewing skills before presenting yourself in front of an employer. There are some online mock interview services also, which you can also utilize.

There are several different online services available to prepare you for mock interviews. These includes:

1. Pramp
2. GainLo
3. CareerCup
4. interviewing.io

Pramp is the most popular among the ones listed above. My recommendation is to check out all of them and see which one best fits your needs.

Life after graduation

Congratulations!

You are now a software developer. You spent the last several months going through a rigorous bootcamp. You missed family gatherings, parties, outings with your friends and maybe even a chance to go to space... ok probably not the last one.

By now you have realized that this was not an easy task. If it was easy then your class would have 200 students and not 20. For most of you this might be the most difficult thing you have ever accomplished!

Take a few days off for yourself, you deserve it. This will help to recharge your brain. After taking your time off, get back on the wagon and start the next phase of your journey.

In this chapter, you will learn how to search for your first job, you will learn about evaluating and negotiating job offers. At the end of this chapter, you will have a clear idea on how to start the next leg of your journey.

Type of Jobs

Now that you have graduated from the bootcamp, your next step is to land your first software development job. Landing your first job can be a mammoth task and does require a lot of patience, perseverance and consistency.

Before you apply, let's talk about different types of jobs available for software developers.

1. **Full time employment (FTE)**: This is the most common type of employment. Full time employment refers to an employment in which a person works a minimum number of hours defined by their employer.

In most FTE, employee usually works on a single project for a long period of time (years). FTE usually also provides excellent benefits, including medical and retirement plans. The base pay depends on the company but I have personally seen that for non-technology companies, base pay is industry average.

I know people who have been employed for the same company for over 20 or even 30 years. I am not saying that you should do that but depending on the company and your interests, it is possible.

One of the main benefits of FTE was job security, unfortunately lately I have seen that this benefit is fading away, many companies will let go of their long time employees in a glimpse in hopes to satisfy the investors.

I don’t meant to scare you, I just wanted to present you with reality. In today’s climate, nothing is secure. You have to work hard to keep your place in the company. Everyone is replaceable, some people are harder to replace as compared to other. Later in this book, I will discuss techniques you can use to be good at your job and become an asset to the company.

2. **Consultant**: A software consultant is a person who works for a consultant agency. This means that the agency will send the employee to different clients. For example, you may work on an oil/gas project for 6 months and then move onto a retail project.

When I started my career in software development, I worked for a consultant agency for 3 years. This allowed me to work on many different projects in a short span of time. Another benefit of being a consultant is that you always have to be on your feet. You will have to learn things on the fly and find solutions to the problems quickly. This will help you in the long term with your problem solving skills.

Some consultant agencies requires travel. This means you might work for a client, outside your state or even country. For non-international gigs, you may travel from Monday - Friday and then start the process over again next week.

If you like fast paced environment and also like to travel, then being a consultant can be a great option. As you get older, travel can become a burden so make sure you enjoy it in your younger years.

3. **Independent Contractors:** Independent contractors are similar to consultants with one key difference, they are not employed by anyone. Independent contractors may take help from a recruiter to land a job but once they are at work, they are on their own. This means they need to manage their own medical and retirement plans and also factor in the cost of vacations. Independent contractors are usually very experienced developers and their rates reflect their experience.

Once you have accumulated enough experience in the industry, you can move towards becoming an independent contractor. This allows you to bill at higher rates and maintain your own work schedule.

No matter which path you choose, each type of job brings its own benefits. The most important thing is that you learn at work and absorb as much information as possible.

Landing an Interview

I often hear stories of students applying for more than 200 jobs and only getting a handful of calls back from the employer. If this is the case, then there is definitely something wrong with your approach and you should take appropriate steps to rectify the situation.

There are many different websites that you can use to to look for a job. Some of the top ones includes:

- Monster
- Indeed
- Career Builder
- LinkedIn
- GlassDoor

Apart from the above mentioned services, one of the lesser known service is Google Jobs. Google Jobs is an aggregator website, which displays job postings from many different sources. Simply go to Google and search for the keyword “**Google Jobs**”. In the search result, it will display a small section with some job postings based on your current location. You can click on **100+ more jobs** or similar link and it will take you to a full search pages for Google Jobs.

The jobs page will allow you to filter your results based on many different criteria including job type, location, remote etc.

Google has done an amazing job of aggregating data from the resources but keep in mind that it does not include internal listings that are only available on employer’s website. For those jobs, you will have to individually visit the website and apply using their platform.

Apart from applying for jobs through job listing websites, one of the best ways to land a great opportunity is through connections. I always advice my students to attend local user group meetings, whether physical or online. Most of the meet-ups starts with a round of introductions. This will be your opportunity to introduce yourself and also clarify your situation that you are a current bootcamp graduate and looking for opportunities.

It is advisable that you attend user group meetings during the start of your developer journey, instead of the end. The main reason is that if you are going to wait after graduation to connect with user group members then you might come off as desperate. It is important that your connections are as genuine as possible and not solely based on a business transaction of finding a job.

So, if you are just starting out with a coding bootcamp then try to join user groups in your area and start making real connections. Even if you are few weeks or months into the bootcamp, make the effort to find a developer community. The fastest way to land a job is through references and connections.

Types of Interviews

It is a great feeling when you land your first developer job interview. You worked incredibly hard to reach this point and you should be very proud of yourself. Your next step is to start preparing for the interview.

Unfortunately in the world of software development, there is no single way of conducting an interview. Each company employs its own practices of managing the interview process. Below you can find the most common interview techniques.

Algorithms White Boarding

Developers despise white board algorithm interviews with a passion. The main reason is that these kind of interviews don't really measure anything that is going to be useful for your actual job. I have been doing software development for the last 15+ years and the number of times I implemented a binary search or inverted a binary tree is exactly ZERO.

Whether you like it or not you still need to prepare for algorithms. There are lots of great resources that will help you get ready for the algorithm interview questions. Two of the resources that I always recommend to my students are **LeetCode** website and a book from Gayle Laakmann McDowell called "**Cracking the Coding Interview**".

Algorithms questions exercise different muscles of our brain. Those muscles are usually not flexed when you are building websites or apps. This is why you may have trouble getting started and solving even basic algorithms questions. Don't worry too much. It will improve over time and you will start understanding common patterns among the algorithms. Remember, practice makes perfect.

Take Home Project

One of the others ways companies evaluate their candidates is by giving them a take home project. A take home project is a small project with a very specific goal that the candidate must successfully finish in an allotted time. Most developers enjoy take home projects as it is close to what they will be doing at their regular job.

Although take home projects are preferred by developers, I have heard of some isolated cases where the company camouflage their actual product features or MVP (Minimum Viable Product) as a take home project and use developers to implement it. The probability of you running into those companies is very low, but still, something to keep in mind.

Having said that always take time to evaluate "Take Home Projects". If you feel like this project cannot be done during a weekend then communicate with the company to either limit the features or provide you with a different project. You really don't want to spend 2 weeks of your time implementing a job interview project.

Technical Questions

When I was interviewing junior developers, I always resorted to asking technical questions. These were not algorithm questions or take home projects but basic questions that tested the theoretical knowledge of a candidate. Here are some of the questions I asked for web and iOS developer positions.

- What will happen if you call setState inside the render function of a React component
- What is Virtual DOM and how is it different from HTML DOM?
- What problems you may encounter, if you perform a network request on the main queue in iOS?
- Should I push node_modules folder to the GitHub repository? Why or why not?

All questions were directed towards finding out the depth of candidate's knowledge and their understanding of the concepts.

As a junior developer, you should be prepared for any type of interview. Take appropriate steps to prepare for each type of interview. This will increase your chances of getting hired.

The Interview

Ring ring!

You picked up the phone, it was from one of the companies you applied for a job. They are interested in meeting with you and talking further about the potential opportunity. This is great news, you have made through the first door. Your next step is to make a good impression.

When showing up for an interview, it is extremely important to dress right. This means a nice suit for men and a pantsuit or formal clothing for women. Do not and I repeat do not go to an interview in your shorts and flip-flops. It is better to overdress than underdress.

Make sure you arrive at least 10-15 minutes before your scheduled interview. This will give you ample amount of time to get settled and relax. Also, make sure to have at least 2-3 printed copies of your resume. Don't assume that the employer will have your printed resume. I made the same mistake in one of my interviews, where I did not bring my printed resume. When the employer asked me for a copy, I was unable to provide it. It was an embarrassing situation, which could have been easily avoided.

During the interview, try to remain as calm as possible. Don't argue with the interviewer and be as polite as possible. If you don't know the answer then just say that you don't know, but you are willing to learn and find the answer.

If your interview is a white board algorithm challenge, make sure you are explaining each step to the interviewer as you write code on the white board. Apart from solving the problem, you must also showcase your communicate skills. Make sure to ask follow up questions. For example if you had to write an app to check whether a string is a palindrome or not, you might ask the following questions:

- Are we taking into account numbers?
- What about casing of letters? Is Mom same as mom and will it be considered a palindrome?

Asking questions is a great way to get clarification and it will get you one step closer to your solution.

Finally, don't get discouraged if you don't land a job after your first interview. Usually, it takes several interviews to get comfortable and get an offer. Learn from your mistakes and get better each time.

Good luck!

Thank you Note

I bought my first brand new car in 2008. It was a Honda Accord. I still drive that car and love it. Each year on my birthday, I receive a birthday card from the sales representative who sold me the car. I have no intentions of buying a new car right now but if I ever do, I know who to call.

A simple act of a thank you note can have big effects in the future. When your interview is over, make sure to send a physical thank you note to all the people who interviewed you. You don't have to write a 1200 word essay in a thank you note. A simple one sentence saying that you are thankful for the opportunity is more than enough.

You never know a 10 cent thank you card may be the deciding factor between you and other candidate.

Evaluating the Offer

As a recent graduate of a coding bootcamp, receiving a job offer from an employer means a lot. You have spent the last 4-6 months learning programming and solving problems. You have lost a lot of sleep and missed a lot of gatherings. You have been interviewed dozens of times and written who knows how many thank you notes. But receiving your first job offer, always makes it worth it.

Before you accept an offer, it is always a good idea to throughly evaluate it. Most new developers make the judgement call based on the salary alone. This is not a good idea since your salary is just one part of the total package. Let's take a look at few things that are commonly found in most job offers:

Base Salary:

Salary is without a doubt, most important part of your total package. Make sure you are comfortable with the starting base salary because once you accept the offer, your base salary moves up slowly. Before accepting the offer do online research on how much a junior developer is paid in your region and make sure the offer reflects it.

With everything moving remote, don't be surprised if the salary offer you receive is based on your current location. This means that for the same role, the base salary might differ on whether you are located in San Francisco, CA or Houston, TX.

Most companies offer salaries based on your location. The main reason is that cost of living is different in different cities and job offer reflects the cost of living adjustments.

Medical/Dental/Vision

Unlike most of the European countries, people living in United States do not have access to universal health care. This means employer or an individual must spend an exorbitant amount to be covered for medical expenses. There are lot of different components of a medical coverage but the two most important includes premiums and deductibles.

Premium is the monthly amount you will have to pay, whether you visit a doctor or not. Think of it as a regular expense like your phone or internet. If you are signing up for medical insurance and I highly recommend that you do then premiums are automatically deducted from your pay check.

Deductible represents the amount you must pay before the insurance company is going to step in and cover anything. For example, if your deducible is $2500 then it means that you will spend your own $2500, before insurance is going to cover anything.

Even if you are young and healthy, I still encourage you to get medical coverage

Usually, if you have high premiums then your deductible will be lower and vice versa. Some companies also offer HSA and FSA accounts. Both type of accounts can be used to use pre-tax money to pay for medical expenses. If you decide to have HSA or FSA account then make sure to keep a copy of your receipt as it can be required by the IRS.

When evaluating the job offer make sure to pay very close attention to your medical coverage and cost because if you don't then it can eat a big chunk out of your pay check.

401K retirement

You are responsible for saving up for your retirement. The decisions you make at an early age will pay dividends down the road. It is recommended that you save 15% of your income towards retirement.

Typically retirement plans are handled by a third party company like Vanguard, T. Rowe Price, Fidelity etc. Your employer will be responsible for picking the company that will manage the retirement plan.

The amount you will contribute to the retirement 401K plan is considered pre-tax. This means that you have not paid tax on your contributions. Don't get too excited because eventually, when you retire and take the money out you will have to pay taxes on it.

Some employers allows matching contributions. This means that if you contribute 6% of your pay check into a retirement account then your employer will also contribute 6% to your account. This is free money and under no circumstances you should ignore it. Let me repeat that if your employer is matching 8% then you should also contribute at least 8%. The remaining 7% can be invested in either the same 401K account or an IRA account, making your total investment to be 15%. The 15% amount comes out of your paycheck even if you have matching contributions from your employer.

When you switch jobs then make sure to rollover your 401K into a personal Traditional IRA or Roth

IRA account. If you move your 401K into a traditional IRA account then you don't have to pay taxes right away, since taxes are differed until you take the money out. However if you move your 401K into a Roth IRA account then this may incur taxes. It is usually a good idea to consult with a tax advisor before moving money between different retirement accounts.

It is very important that you contribute to your retirement. Only you and you only is responsible for your retirement. The earlier you start, the more wealth you can create and retire peacefully.

Paid Time Off (PTO)

Being a developer and coding 4-6 hours a day, can have a big impact on your mental state. Even though as developers, we are not performing physical work but after a while our brain will need to take a break. If you continuously ignore your brain signals then it can lead to burnout.

This is why PTO is extremely important for your well being. As a junior developer you may get 2 weeks of paid time off. Make sure to take every single day even if you have the option of cashing it out. Your physical and mental health is far more important than the amount you will receive giving up your vacation.

One of the trends that has been emerging, specially in startups is to lure developers by offering them unlimited vacation. Let me be very clear, there is no such thing as unlimited anything. Always clarify with management about the time-off you are planning to take in a given year. It has been proven times that when an employee is offered unlimited vacation then they end up taking less. The main reason is that employees are unclear and don’t understand the context of unlimited when it comes to vacation. So in the end they play it safe and take less vacation.

If you don’t agree with me then I have an open challenge for you. Take 8 months of paid vacation from your current employer. You should be able to take 8 months out of your unlimited vacation right! Once you are done with 8 months, take another 6 months out of your unlimited vacation. Let me know how it goes. I am certain that when you come back from your vacation, a pink slip will be waiting for you on your desk.

Salary Negotiations

Everything can be negotiated. Salary negotiations can be intimidating but as you grow in your career, you will find it necessary to negotiate. The general rule of negotiation is that whoever says the number first looses. This means you should always let the other party throw in the number first. If you blurt out your number then you may undersell yourself.

Some employers and recruiters will press you to provide them with the salary you are looking for. In those cases you can always give them a range i.e $75K - $85K. Make sure that you are comfortable with the lower end of the range. As I mentioned earlier once your base salary is set, it goes up quite slowly.

Apart from your salary, you can also negotiate several other things including PTO, signing bonus, remote option etc. In my decade of experience as a software developer, I have never been able to negotiate the start date of my medical coverage or 401K retirement contributions. If you can negotiate coverage start dates successfully then more power to you.

Let's run a salary negotiation scenario.

Employer: We are here to offer you $65K with 1 week of paid time off:

You: I think with my experience and skillset I am qualified to get $75K - $85K with 2 weeks of paid time off.

Employer: Ok. We can increase the offer to $78K with 2 weeks off.

You: Accepted

It is common to have salary negotiations over emails. Before you counter offer, write a small paragraph about how you are really excited to join the company and how your skills can be utilized to make their products better.

After you pick few items to negotiate, make sure to stick with them. In the exchange above you can see that the job seeker focused on the base salary and the paid time off. Once satisfied with the offer, don't include other demands. This will make you look greedy and employer can be turned off by this behavior.

Also don't just apply for a job so you can leverage a raise at your existing employer. These kind of tactics never work out in the long run and you will end up burning bridges with your current employer.

Salary negotiation is a skill and it requires time to develop. It is completely understandable if you don't negotiate your salary for your first job as a developer. But don't make a habit of accepting the first offer from a prospective employer. Remember, there is always room for negotiation.

Do you know what happens when you ask for more, you get more

- Unbelievable (Netflix)

Work

In one of the episodes of the show Seinfeld, Krammer spills coffee on himself and sues the coffee company on the basis that the coffee was too hot. The company was planning to offer him $50K and free coffee for life. Unfortunately, Krammer did not listen to the complete offer and immediately accepted when he heard free coffee for life.

Hopefully, unlike Krammer you have done your homework and selected the best offer available. In this chapter you will learn about what is expected from you during first few months on the job and what steps you can take to make it most productive.

SpongeBob SquarePants

SpongeBob is a character on a very popular show on Nickelodeon called “**SpongeBob SquarePants**”. SpongeBob lives underwater in a pineapple house and works at a burger joint. His best friend is Patrick, who is a starfish.

Since SpongeBob is basically a sponge with eyes, arms and legs he has the super power to absorb large quantities of matter. In the show sometimes he saves the day by absorbing water, solid elements and anything thrown at him.

For the first few months at your new job, it will be expected from you to behave like SpongeBob. This means you should absorb as much knowledge as possible. Unless you are working for a fast paced consulting agency, you will not be pushing code to production anytime soon.

This will be a great time to find a mentor at your work. A mentor can be a team lead or a senior developer. The main idea is that you will learn the building blocks of the application, work on small features, pair program etc.

Be mindful of your mentor's time and don't overburden them with millions of questions. They have a job to do and they are not responsible for holding your hand for every little task.

Try your best to solve the problems but if you are stuck on a particular issue for 30-45 minutes then consult with your mentor. This way you can explain to your mentor all the different paths you took which did not worked out.

As you gain more experience, you will learn that most of programming revolves around searching and researching. No one knows every thing, but we can always try to find solutions.

Do the Dirty Work

Dirty Jobs is a show on Discovery Channel where the host Mike Rowe performs difficult, strange, disgusting or messy jobs done by hardworking men and women allowing civilized life possible for the rest of us.

Dirty jobs exists in all occupations of our society, even in software development. Few years ago when I was working as a senior mobile developer for a very large retail company, we had a developer who would always rollup his sleeves and do the work no one wanted to do.

This included writing build scripts, updating documentation, setting up servers, creating deployment actions and more. Whenever anyone had any question about any of those things, he was the only one with the answer. By doing the dirty work that no one wanted to do, he made himself very hard to replace.

I am not advocating that you should always be doing the dirty work, but at the start of your career you can make a good impact by doing things no one else wants to do.

Understanding the Business

If you ask a developer working on a medical claims processing application on how to display data on the screen, most of them will be able to answer it. But if you ask the same developers to explain how a medical claim is processed then they will have no clue.

This is the difference between technical knowledge and domain knowledge. Developers are technical people and they can solve technical problems but in order to become a great developer you need to understand the business. How does business work behind the scenes? What goes on when filing and processing a claim? How is it determined that a claim will be accepted or rejected? These are all the questions related to the particular domain you are working in.

The first few months will be the most important months of your new job. I always recommend my students to work a little extra in those months so they get to know the lay of land. This means either you can come an hour early or stay up an hour late.

Make sure to capitalize on this time and dive into your assigned project. Become a sponge and try to absorb as much information as possible. This includes the project architecture, information flow and the most important, business domain.

It is extremely important that you take steps in understanding the core business and how the software helps the company in making revenue. Your first step would be to find the domain expert within the company. A domain expert is a person who understands the business and knows what happens behind the scene. Usually domain experts are non-technical people or with limited programming skills but they have deeper understanding of the business than anyone else.

So, let’s say you have located the domain expert. What will be your next steps? I suggest taking the expert out for a lunch. Nothing fancy! A sandwich would work just fine. Once, you have passed the casual conversation then you can start asking them about the business domain. Remember that you are trying to understand the business and not a piece of code. It is unlikely that your domain expert will be able to help you with your coding questions. If you encounter some business terms while reading the codebase then definitely ask the story behind those keywords. Once you understand the backstory and business meaning behind those keywords, you will be able to code with clarity.

I have always believed that most of the developers are easy to replace. I know this sounds harsh but it is true. Having said that a developer who understands the business is very hard to replace. This is because not many developers take the time to understand the business domain and those who do become a rare breed in the software development industry.

Be Kind

Mark Cuban once said that one of the most underrated and missing skill among people is being kind. I have worked with a lot of young developers throughout my career. One thing I noticed is that even with their limited experience they start complaining about certain aspects of the code. Here are some of the most common complaints:

- Why did we choose React, we should have implemented this project in Angular?

- Why do we have to follow this processes for checking in code? This waste a lot of time.
- Unit tests are slowing me down. Who needs to write unit tests these days?

As a junior developer, you should take the time to understand why those decisions were made. You will find out that your senior peers, looked at all the options and selected the best that fit their and company's needs and requirements.

Be kind and respectful to everyone. When you make a mistake, be the first one to admit it. It shows maturity and character and it will help you not only become a better developer but also a better person.

Raise and Promotion

Every company operates differently from every other company. This is also true when it comes to raises and promotions. I think that a one year mark is a good point to ask for a raise. Keep in mind that depending on the company, you may not be eligible for a promotion i.e new job title, but you can still ask for a raise.

When asking for a raise, keep your personal matters aside. Your raise negotiation should not be based on expenses or change in your family tree. This means don't ask for raise because you just bought the new Tesla car or don't ask for raise because you just had twins. Your raise should be solely based on your contributions to the company, which resulted in company making additional revenue.

It is always a good idea to keep a running log of all your contributions and accomplishments so in the end you are not trying to remember them from memory.

If you don't get the promotion then make sure to keep your emotions under control and ask the manager to setup a personalized goal plan for you. This will give you a better idea on what is expected from you so you can work towards your goals and get that promotion.

Two and Done

I read an article long time ago by an experienced developer, who talked about how he was managed to keep his job for over 20 years. One of the very important points he discussed was a simple rule called "**Two and Done**". According to this rule, he will suggest his solution/view to his peer two times and if they don't agree with his solution, he will not suggest it again.

I think this is a great way to handle conflicts in a team. Instead of getting stuck on your decision and not moving the needle, we can pitch the solution two times and if we fail to change the trajectory of the project then we should leave out our ego and go with the team's decision.

Don't Burn Bridges

Sooner or later you will change jobs. This can be for many different reasons but the most important factor will be how you handle this transition with your current company.

First of all, don't leave your current job unless you already have a job lined up. This can cause a lot of financial stress. Also leaving job because you did not get that promotion or bonus is never a good idea. These are just emotional reactions, which may cost you dearly in the future.

After you have received and signed the new job offer from the employer, it is only ethical that you give your current employer two weeks notice. Your current employer can use these two weeks to find your replacement, train an existing employee and ask you to document your work etc.

There are several different scenarios which can stem from your two weeks notice. Your current employer might offer you the same pay as the potential new employer. In most situations, it is never a good idea to play these cards. The main issue is that even if you take your current employer's offer, most probably you will be seen as the black sheep in the company. The sheep that almost left the herd.

This can scar your relationship with the current employer. I believe the best course of action in this scenario is to simply thank your current employer for everything and move on to the next opportunity.

The next two weeks, which are your last two weeks with your current employer are extremely important. Most people will start slacking off, coming in late and leaving early. They know they will be gone in two weeks so they stop... caring.

Make sure your last two weeks are extremely productive. Come in early, leave late and make sure everything is in order before you depart the company. Ask your current employer, if you can do anything to smooth this transition.

The main purpose for doing all the hard work during these last two weeks is to continue making a good impression in front of your current employer. If things don’t work out in the new company, you can always know you can go back. Employees who leave their current employer but return back to them in the future are known as boomerang employees. Boomerang employees are more common than you think, but remember you can only return to your original employer if you **don’t burn the bridges**.

Success Stories

I have been teaching at coding bootcamps for more than 5 years and have graduated more than 200 students. Many of those students went on to land their first development job and almost doubled their previous salaries.

This not only allowed them a fresh start but also completely change their family tree. Becoming part of someone's life journey is always a very rewarding experience, but the credit goes to my students who worked day and night and showed passion and perseverance to fulfill their dreams.

In this chapter, we will look at few success stories from my students. Hopefully, these stories will inspire you to take the plunge into software development. Remember, anyone can become a software developer. All it needs is passion, perseverance and plain old hard work.

Astrid Countee

Astrid graduated with a bachelors in psychology and masters in anthropology. After graduation, she started working as an analyst at a major oil and as company. It was during her work as an analyst, she discovered her love for technology and decided to join a coding bootcamp.

Astrid joined backend engineering program via Ruby and Rails offered by The Iron Yard. According to her, the program was very intense but it helped that the coding bootcamp was located in a working space, which allowed her the opportunity to immerse in a startup culture. She volunteered for several technology conferences, which helped her become familiar with the industry and ease the transition from an analyst to a software developer.

After graduation Astrid working as a freelancer immediately and in 3.5 months she was hired as a quality assurance engineer at Hewlett Packard (HP).

Astrid credits her career to the knowledge she learned during the coding bootcamp. It helped her transition from her previous role to a new role more aligned with her skills and talent. Astrid continue to make advancements in her career and now working as an independent researcher working in cryptocurrencies.

Joseph Frasier

Joseph graduated with a business management degree and before joining a coding bootcamp, he worked as a sales associate. After hearing about coding bootcamps he decided to join web development program offered at The Iron Yard.

Joseph describes his experience at The Iron Yard as “**life changing**”. According to him, the program was fast paced and each day he was learning new concepts. The key was to make sure to stay on top of your assignments and projects and also make sure to ask questions to your instructors.

Three months after graduation Joseph was able to land his first job as a software developer. According to Joseph, the new job came with an overall salary increase of whopping 80% as compared to his previous salary.

Since graduation Joseph has changed jobs with multiple other companies and each change came with higher pay. He is confident that the next job change could bring him an overall salary raise of more than 50% of his current salary.

Lorenzo Covarrubias

Lorenzo worked in sales and construction before joining full stack development program with Python and React offered by Byte Academy. Having no background in programming and no college degree, it was a nerve wrecking decision.

During the bootcamp, Lorenzo worked hard to make sure to take advantage of all the knowledge offered by the bootcamp. He also used all of his resources and connections, which proved to be instrumental in his job hunt.

Lorenzo's efforts paid off as he was able to land a job, just 2 months after graduation. The job was through one of his friends who referred him for a position. Recently, Lorenzo started his second position with a raise of $43,000.

Ricky Padilla

Ricky attended Galvanize Web Developer program in Denver, Colorado. He graduated with a political science degree and worked as an entrepreneur owning a coffee roasting business.

Ricky was able to land a software developer job immediately after graduating from the coding bootcamp. Ricky works as an iOS developer and increased his pay almost $20,000 as compared to his previous jobs.

According to Ricky, being a developer has allowed him to flex his creative muscles and at the same time provide great income for his family. He enjoys working on complicated problems and learning new things.

About the Author

Mohammad Azam is a veteran developer with over 15 years of software development experience. Azam has worked with many fortune 500 companies as a lead developer. He was also awarded the prestigious Microsoft MVP award for his contributions in the community.

At present, Azam is working as a coding bootcamp instructor at DigitalCrafts. Apart from that Azam frequently publishes video courses on Udemy and LinkedIn Learning and is an active member of the programming community.

When not programming, Azam likes to spend time with his family and take adventurous hikes.

You can follow Azam on Twitter at @azamsharp.