A few years ago, I was bouncing back and forth between landscaping jobs and restaurant jobs. I had just left college, and didn’t know what I was going to do with my life.

I had a lot of ideas, but no direction. During that time, I started teaching myself programming. At first it was a hobby. I thought it was cool to be able to build things using nothing but my brain and some code.

But then I started thinking about where my life was going, and saw this as a potential living.

At first, I put the idea out of my head. I couldn’t afford real education. I already dropped out of college once because of money, and if I went in for computer science, I’d have to start over.

I’d leave with 6 years of school and well over $50,000 in debt if I took that route. So that wasn’t an option.

Then I started thinking that I could teach myself web development well enough to get an internship.

My initial plan was to introduce myself to a few companies in my town, and ask if they would want to meet with me. I wanted to discuss the potential of me working with them while I was in school. That way I could pay for school and get some experience at the same time.

So I got serious about web development.

Instead of tinkering I started to build a real portfolio and document my skills.

I started becoming active on places like Stack Overflow.

I built a few practical applications and put them on GitHub. They were nothing fancy, but they demonstrated that I knew how to code.

One company didn’t offer me a part time job. They didn’t say to come back after I had my degree. They offered me a full-time job on a 6 month trial basis as their new Junior Developer.

I was over the moon. It turns out that once I got serious and started developing with a purpose, I taught myself quite a bit.

I was able to answer their questions. I was able to walk them through the modest applications I had built. And I was able to explain how my projects worked.

I stayed at that company for two and a half years, and then took a job as a web developer for the city I live in.

View yourself as a lifelong apprentice

An important part of my transition into a professional developer role was viewing the time I spent with my previous company as an apprenticeship. I learned as much as I could.

The real-world knowledge gained from working at a company is invaluable. Knowing how to work with clients, coworkers, and within constraints is essential. This is something you can only learn in the field.

While I may know more now than I did when I first started that job, I’ll never stop viewing myself as an apprentice. One of the requirements for being a great developer is the desire to continue learning. The minute we see ourselves as having mastered a skill is the minute we stop growing.

Hemingway said it best:

We are all apprentices in a craft where no one ever becomes a master.

He was talking about writing, but it applies to development as well.

The combination of teaching myself and working for a company has allowed me to learn so much (I’m even writing a book). I understand the technical practice of web development, and also how to go from a hobbyist to a professional. It’s a path that anyone can take, regardless of your time or abilities.

To give you some perspective, I was working two jobs at once — one of which involved getting up at 4 a.m. to drive a forklift around.

Learning to code as a busy adult takes determination, drive, and a stubborn persistence.

Making the transition from Hobbyist to Professional

Here’s a process that you can follow. The exact journey will be different for everyone, but there are steps you can take to get you on the right path.

1. Realize that you can do this

Anyone can teach themselves to be a developer. There’s this idea that being self-taught is something only a certain type of person can do. They’re right in a sense. You need to be self-driven and motivated by something other than immediate payment. But anyone can become this type of person.

There’s this idea present in our society that some people are born with certain traits and others aren’t. It’s detrimental to growth, and one of the reasons why so many people feel unfulfilled in life.

If you always felt that you either “had it” or you didn’t, it would be very easy to get discouraged.

I want to put that myth to bed right now. Anyone can learn to be self-motivated and teach themselves programming. Or start a successful business. Or achieve a long-term goal.

It’s not about catching a big break, or being born with the right traits. It’s about perseverance.

If you can put your head down, push through the hard times, and commit, you can do anything you want to.

That last part is super important, but I want to provide a warning before moving on.

People are often too quick to embrace their own successes and the successes of others. It’s known as survivorship bias.

There is an element of luck in everything. Sometimes things just work out. For example, I contacted a web development agency and was lucky enough that they happened to be looking for someone at that moment, and I happened to fit what they were looking for.

But what is luck?

Sure, I was lucky to get that job, but I never would have been lucky if I hadn’t made the decision to teach myself development. And then made the decision to apply to that job.

Luck does play a factor, but the myth is that it is all up to luck. You can increase your odds of getting lucky, you just have to be willing to put yourself out there.

But luck will never find you if you don’t commit to being great at something.

2. Commit to being incredible at your craft

One of my biggest weaknesses is that I get bored and distracted. I want to jump into the next project. This tendency will kill your success.

It feels like freedom. Being able to bounce between whatever project happens to suit your mood that day, but...

It’s a trap! If you take away nothing else from this article, let it be this:

The number one key to succeeding in becoming a professional developer is to commit. Commit and never stop until you make it happen.

This applies to everything.

People stress out about which framework to use. But what matters is picking one and sticking with it. You can transfer and learn new languages and frameworks later.

What matters is the problem solving skills you will gain when developing. The ability to think like a developer.

I taught myself programming using Laravel, but the company that hired me used CakePHP. It didn’t matter. They knew I could pick up the technical skills required to switch frameworks.

Pick a direction and see it through, no matter what. You have to remove the possibility of getting distracted by something else.

Few feelings can compare to the relentless pursuit of mastery of a craft.

It isn’t easy. Once you learn to ignore distractions, you will notice an increase in enjoyment of your work.

Mike Rowe is fond of saying that people shouldn’t start with finding their passion.

People are so unhappy because they look for the perfect career. They look for the one that they are passionate about.

But passion comes from an unstoppable desire to be incredible at your craft. Once you adopt that mindset, your abilities as a developer will take on a new life.

3. Start building things immediately

Aspiring developers can get stuck in the trap of reading too much without taking action.

Tutorials and books are great for learning the basics. The problem is that they instill a false sense of confidence in the developer.

Have you ever finished a programming book and gone to build something on your own only to realize you had no idea how to go about doing it? Then you know what I’m talking about.

The solution to this is simple, but not easy.

Start building.

Make something. Make an app that solves a problem you have in your own life, or that addresses an issue for someone close to you.

Make something for fun.

Make something and put it out there. Make it open source and put in on GitHub. You aren’t doing it for anyone else, it’s for you, so don’t worry about other people’s opinion of it.

Your code will be ugly at first. I look back at some of the code I wrote even a few months ago and want to vomit. But you can’t learn development without building stuff. Books are fantastic, and I am obsessed with reading as many as possible. Then you must apply that knowledge.

You’re going to run into issues and you’re going to struggle. That’s good. Those are the times we learn the most.

Start off by building things that solve problems, I’ll talk more about that in step 6 below.

4. Set up an online presence

As soon as you start building things, you’ll want to set up an online presence. Your GitHub account will be a great start.

This is where you’ll be able to house the projects you are working on and share them with the world.

But you want to go further than this. I recommend setting up your own portfolio site.

This site will do a few things:

- It will serve as a public place to tell potential employers about yourself

- It will be another place where you can showcase your work

- It will serve as your platform

- That last one is huge. Once you start building things, you should immediately start writing about them. Start a simple blog where you share what you are working on and teach everything you know.

This is one of the best ways to give potential employers a taste of who you are and what you can do. It’s a way to get your name out there and start building a platform for yourself.

This can lead to job opportunities and the possibility to make more income on the side by writing books or freelancing.

Your site should serve a very specific purpose.

Most people create an online resume, but you should do more. What is your specific goal? Your website should be designed and created around that goal.

If you want to get a job working on a certain kind of project or with a certain framework, put that in your site.

I recommend having 4 core areas for your site:

1. Home page

Your home page is the entry point to your site. It should provide a very brief overview about who you are and what you do. And should direct people to go where is most relevant for them.

For example, you could have two main buttons. One leading people to your writing section to learn more about web development, and one leading to a hire me page if someone is interested in hiring you.

2. Writing

This is where your blog and your tutorials will live. Write as much as you can here, and don’t be afraid to share it.

3. About

A simple about section that goes into more detail about who you are and what you do. Don’t make this a life story. Again, target this section to be relevant towards what you want to do.

Rather than talk about your personal life, talk about what led you to web development, your journey so far, and where you want to go. Mention some of your favorite projects and link to them.

4. Hire Me

An essential part of your site, this is where people will go if they are interested in hiring you as a developer.

Make sure to find the right balance between selling yourself and being honest. There may be some overlap between this page and your about page, but this page will be more specific about your skills and what you bring to the table.

This page should also have a contact form so people can get in touch.

In addition to your own site, start offering to write for other major publications. Then you can provide a link back to your site in the bio section.

5. Start teaching everything you know

Nathan Barry is a big fan of teaching everything you know. He tells the story of Chris Coyier, founder of CSS Tricks.

That site started out as Chris publicly writing about what he was learning so others could follow along. Now it’s one of the biggest web development sites out there.

The lesson here is that you don’t have to be the world’s greatest expert to start writing about something and teaching it.

In the online business world, there is this idea of the relative expert. It’s the idea that everyone is an expert in something relative to someone else.

I have my issues with this, especially when it is used by someone to justify selling something that maybe they shouldn’t be selling. But it is a useful comparison to make.

What bothers me is the use of the word expert. I don’t think there is anything wrong with teaching what you know, and even potentially selling that information if it is valuable to someone else.

But calling yourself an expert may be taking it too far. So when you write your content, approach it honestly.

I prefer the term learning in public.

There are many people who got their start by simply being a public learner. They were learning a craft and documenting what they were learning along the way.

This is the perfect way to approach teaching everything you know. As you learn more and more, you build up your content, and become a better writer in the process. Over time, others in your industry will start to see you as an authority in your space.

This will be invaluable both when it comes to finding a job and if you ever want to strike out on your own.

6. Build to solve problems

One of the most important aspects of becoming a professional developer is doing everything with a specific intention.

It’s one thing to build random apps for fun, it’s another to build apps and sites that solve specific problems.

Web development shops aren’t really in the business of coding, they are in the business of solving problems. The coding is just their preferred tool to make that happen.

Read any marketing or copywriting book and they will tell you to sell the benefits of your product, not the features.

Web developers should market their apps to show how they effectively solve a customer’s problems. And then back up their claims with specific metrics. Customers are generally more responsive to this approach than if the developers talk about the bleeding edge technologies they use.

You will be a very attractive prospect for employers if you can demonstrate your programming skills as well as your ability to code with the specific intention of solving problems and making meaningful applications.

Think about benefits vs features when you are communicating with potential employers or clients, and when writing the content for your site.

Of course, you should also mention your coding proficiency, but most people spend all their time on this. Mention it briefly so potential employers know what you do. If you have a portfolio of effective applications, your coding skills will mostly speak for themselves.

7. Take on an apprentice mindset

The day you think you have mastered something is the day you stop learning.

Adopt the mindset of a lifelong apprentice. There is always more to learn and always more to improve on.

This is especially important in the early stages of your career. If you get a part time job or internship or land a role as a junior developer, you need to immediately get in the mindset of learning and growing as much as possible.

You should really be doing this right away, even before you have an actual ‘mentor’.

In his book, The Art of Work, Jeff Goins talks about the 21st century version of the apprentice-master relationship.

Back in medieval times, the relationship was very formal. A master would take on an apprentice for years, and they would slowly start to master their craft until they reached the title of master, at which point they would take on an apprentice.

The relationship has changed, but it is still very important to consider yourself an apprentice. The main difference is that now you have to keep an eye out for potential mentors and learning opportunities, and there will be many throughout your journey.

In the world of web development, we are constantly on the internet, so this can come in various forms.

Books, tutorials, courses, forums, and other forms of learning are all valuable. I think, however, the most valuable form of apprenticeship comes from learning from someone who is currently in the position you want to be in.

This is why it is so important to be eager and willing to learn. Getting your first development job is not the end of the journey, it’s the beginning.

That is when you will really start learning and exponentially growing your knowledge.

8. Learn to collaborate

One of the biggest differences between coding as a hobby and coding for a living is learning to collaborate with people.

You’ll need to interact and work with peers, bosses, colleagues, clients, partner companies, and all kinds of personalities throughout your career.

Learning how to effectively work with other people is important.

In the field of web development, communication is key. When a company comes in and tells you what they want, and you aren’t clear about exactly what that looks like, it can cause a lot of problems and headaches in the future.

Likewise, if you can’t communicate with the people you work with, your work will suffer and you won’t be able to do your job as well.

While you’re still learning, there are a few great ways to do this.

Part of this will come when you start teaching everything you know. People will interact with you, sometimes negatively, and you’ll learn how to deal with those situations.

I also highly recommend contributing to open source projects. This gives you a taste of what it is like to collaborate on a project where different people may have different ideas on the best way to do things.

Contributing to open source projects can be intimidating, but it will do wonders for your development career.

Check out this site to get started.

Get out there and make a living

Being a web developer is hard. It means a life of non-stop learning and adapting to new technologies. It’s one of those careers where you need to be well-versed in not only technology, but business and communication as well.

It’s an extremely rewarding path. You get to make things that solve people’s problems and make their lives easier, while making a fantastic living at the same time.

There are countless resources to help you learn to code, many of them completely free, but there seems to be a lack of resources helping people make that transition into professional developer.

I hope this short guide provided a good road map for you to get started becoming a professional developer.

Remember, nothing will happen unless you take action. Build a simple portfolio website, email a few potential employers, write some posts on Medium. Just start doing something.

The more you put yourself out there, and the more you do, the sooner you’ll make the shift from amateur to professional.