LEARN TO CODE IN 2023, GET HIRED, AND HAVE FUN ALONG THE WAY

Andrei Neagoie

V1.07

HEEELLLOOOOO!

I'm Andrei Neagoie, Founder and Lead Instructor of the Zero To Mastery Academy.

After working as a Senior Software Developer over the years, I now dedicate 100% of my time teaching others valuable software development skills, help them break into the tech industry, and advance their careers. Over the past few years, **over 1,000,000 students** from around the world have taken my courses and many of them are now working at top tier companies like **Apple, Google, Amazon, Tesla, IBM, Facebook and Shopify**, just to name a few.

This guide provides step by step instructions on how to become a web developer from having zero knowledge… **for free**. By putting in the work, you'll have the opportunity to take control of your life, work in an exciting industry with infinite possibilities and live the life you want.

Happy Coding!
Andrei

Founder & Lead Instructor, Zero To Mastery
Andrei Neagoie

P.S. I also wrote a book called Principles For Programmers. You can download the first five chapters for free here.

Preface

In 2015, I found myself in the situation many of you probably find yourselves in. I was bored with my career in the travel industry and heard about the great lifestyle and pay within the tech industry.

So I decided I was going to learn how to code and get hired as a web developer. But I already spent 4 years in University and didn't have the time or money to go back to school so I knew I'd have to teach myself.

Spoiler alert: I made it happen and got multiple job offers within 5 months.

So in 2017, I shared the first version of step-by-step guide which was exactly what I used myself as a way to help others break into tech as well. To my surprise, it went viral.

So every year since, I've created an updated version of this guide to ensure it contains the best modern free resources and the best ways to learn to code and get hired as efficiently as possible in the current year.

100,000+ of people have now used this guide to learn to code for free & get hired as a fullstack web developer, from scratch. You can too.

A lot has changed since last year's edition.

But there's one thing that has never changed, and that is the focus on efficiency: *learning the right topics that are in-demand right now, so you can get hired as soon as possible.*

This guide has the exact steps that you should take if you want to learn to code in 2023, change your career, and get hired as a Web Developer (or get into the tech industry).

If you're a complete beginner, you call yourself a junior developer, or you're curious about this industry, then this post is for you.

If you're an established developer, you will still find some useful links in here as I list the best free resources to level-up your skills. But I also wrote a post on how to become a senior software developer that may be more useful to you.

Due to popular demand, I also wrote 2 additional follow-up posts that I'd highly recommend you read *after* you've finished this guide:

> **Part 2: Don't Be A Junior Developer: The Roadmap From Junior to Senior**. Seriously, don't be a junior developer. These are the steps and topics to learn to become a senior developer.
>
> **Part 3: How to Interview, Land a Job, and Get a Raise**. The strategies and tactics I used to get multiple job offers after following the steps in this guide.

If you find this post too long, skip over and start from **The 5 months: step-by-step** section. You'll hurt my feelings… so know that you'll have to live with that guilt.

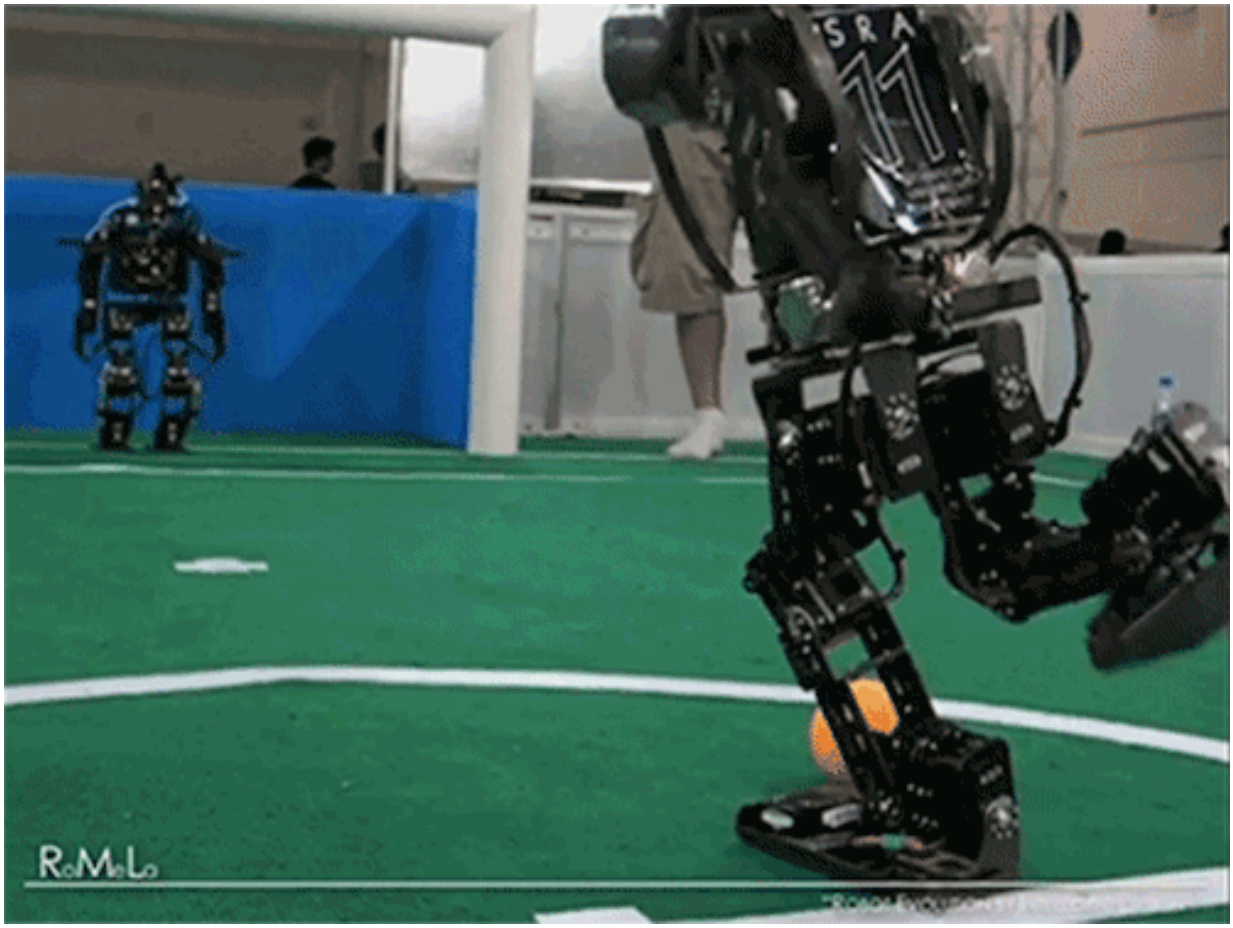
> ***Ok you're still here. Great! I like you already. Let's keep going…***

Using only free online courses, tutorials, and tools, you can gain a valuable skill that will allow you to be employed in a great industry that is rewarding, challenging, and with a lot of options to move around the world (more on this later).

The best part? You don't need a college degree or an expensive bootcamp. And you definitely don't need to give away part of your income once you get hired which some new schools are doing via something called an ISA (Income Sharing Agreement). This sounds great until you have to start giving away some of your paychecks.

Important note: This post may seem like it's just a step-by-step guide of what to do to become a developer. But if you look more closely, it's also a strategy you can apply to any sort of learning.

Why coding?



One day you can build the best soccer goalie in the world…

Before we get into the steps you can take to become a developer, we must first dive into why you would want to go down this path.

Every decision that will require a significant time of your life should be justified. Time, after all, is the most important resource we have:

A. You want to be working in an industry where there is a high demand for the skill and many possibilities to be in important roles at the top of the food chain.

B. You love being location independent. You want a skill that allows you to go anywhere in the world and still be able to find a job easily. If you decide to move

to Iceland tomorrow, you want to make sure that you won't have issues finding a job.

C. You've noticed the difference between 2010 and 2023 and how much technological progress we have made in only ~10 years. You want to be at the forefront of an industry that is impacting the world.

D. The industries that have seen the largest growth in the last couple of years have been in the areas of:

- artificial intelligence & machine learning
- bio tech
- VR
- autonomous cars
- blockchain/crypto/web3

What is essential to all of these industries? Programming (aka coding or development).

We interact with these technologies every day and you don't want to be left behind as these take over our future. Becoming a web developer is a great foot in the door to these industries.

E. You think change is good and that learning should never stop. So why not do something new?

> ***But I don't have a computer science degree and I don't even know how the internet works! Don't worry, we will use that to your advantage. Keep reading…***

When choosing a new career path here are some must-haves and nice to-haves:

1. It must be relevant for the next 10+ years. This skill should be valued for many years in the future, guaranteeing your job security.

2. Demand for people with this skill must be higher than the supply. The smaller the available pool of skilled workers in the industry, the more control you can have over your job and the companies you choose to work for.

3. Ability to have a high salary regardless of your number of years in the industry. You don't want to spend many years climbing the corporate ladder before you're able to make a decent living.

4. An industry that doesn't require a specialized degree from a college or university. You don't want to spend the next 4 years getting into debt and going to a graduate program before you start making money. And yes, if you want to spend some money to hold yourself accountable and not have to learn alone, there are better alternatives than paying $8,000+ to go to an expensive coding bootcamp.

5. Ability to catch up to the top performers in the industry in the shortest amount of time. Can little experience still get you employed? And can you close the gap as fast as possible to be considered a senior or an expert in the field?

6. It must allow you to build foundational skills that will give you multiple career options no matter what the future holds. For example, by learning to code, you're able to better understand new upcoming technologies like distributed applications, data science, machine learning (AI), and cloud computing. This allows you to quickly and easily choose which field you want to jump into next.

7. Have fun. This is the most important one. You have to enjoy what you're doing. Can you see yourself doing this 40 hours a week for a long time? If you're still not sure if coding is for you or what a developer even does, check out this post and try the 21-day coding challenge.

In my experience, coding hits every one of the points above. Warning: Your mileage may vary.

One of my favourite books is called So Good They Can't Ignore You. The author argues that passion is a myth.

You shouldn't go into the travel industry because you are "passionate" about

travel. Most people find passion by struggling and working hard to master a skill. Once people start acknowledging your valuable skills and you feel respected for these skills, that's when you develop a passion for what you do.

> ***Still with me? I haven't scared you off? Ok, we shall keep going then…***

IMPORTANT POINT, READ IT: The first 2 months will feel like you are climbing an insurmountable mountain.

Every tutorial, course or lesson you do will make you feel like you are the only person in the world that doesn't know this stuff.

This is called Impostor Syndrome. The feeling of complete self-doubt.

Stay strong.

You will get there and you will have more and more 'AHA!' moments as time progresses.

Rest assured, we all feel this way when we learn something new. This is good. This is how you know you are stretching your boundaries.

It's kind of like doing a really tough workout and your muscles are sore for the next couple days. You know you're pushing yourself and growing when you feel uncomfortable.

What you will learn at the end of this is that being a good developer isn't just memorizing a whole bunch of documentation. It's about learning how to solve problems using all of the tools that are available to you.

It's about being a problem solver and getting from a state of not knowing, to knowing. This guide will help you get those skills.

Who are you and why should I listen to you?

Wow, you're direct, but I guess that's a fair question.

First off, I'm a Senior Software Developer that has worked in various locations including Silicon Valley and Toronto at some of the top tech firms.

I've been very fortunate in my career and over the past few years I've taught over 1 million people around the world how to become developers from scratch. Many of those Zero To Mastery graduates now work at companies like Google and Amazon.

But I wasn't born a computer wiz. I didn't graduate with a Computer Science degree. I am completely self-taught.

P.S. This part is all about me, so if you don't care (totally fair point), just skip this section. I'll get over it eventually.

It all started many years ago, 2015 to be exact... I wanted a career change and decided to teach myself computer programming.

Unlike what most people do, I spent the first month *avoiding* any coding tutorials or programming courses or books. Instead, I spent this month figuring out what would be the best way for me to learn code and get hired as efficiently as possible.

I didn't want to waste my time learning outdated technologies that most potential employers didn't care about or learn things that I would never actually use and just forget after a month.

I studied other people's experiences, looked at job postings, spoke to established developers, reviewed online courses, looked at bootcamps, and even read articles by futurists on where we will be with technology in 20 years.

Based on all of that research, I created a curriculum for myself focused on efficiency: **The critical amount of learning in order to be employable in the shortest amount of time.**

If you love the works of **Tim Ferriss** as much as I do, you're going to love this. The curriculum isn't focused on doing the least amount of work. Instead, it is focused on working really hard at the things that matter most in order to be employed in the most optimum way.

This doesn't mean doing the bare minimum and being hired as a junior developer. If you can work hard and skip the line by jumping straight into an intermediate developer role, that is a better outcome.

Luckily for you, I have already sifted through everything you need to make that happen.

Although I spent one month planning my studying instead of actually studying, it was a benefit in the long run because I wasn't running blind. I knew where I was going, and I had a clear roadmap to the finish line. You will too.

So yes, I have been where you are and I know what it takes. When I was getting started, I wish there was something like this that outlined things for me step by

step.

I also found many tutorials were taught by people with a lot of technical knowledge but without being able to properly teach a beginner. Alternatively, some courses were taught by people who took advantage of beginners not knowing much about the industry and selling them a course that sounds great but doesn't actually teach you how to succeed (we call these superficial skills).

I've read and studied every single video, tutorial and course that time permitted, and I still continue to do so to try and find the most efficient path to succeed.

I'm obsessed with the art of learning and even developed a system around efficient learning.

Since then, I have consulted for Fortune 500 tech companies, ran coding workshops, consulted on published tech books, given technical talks, and I have helped those with zero experience in programming get jobs in just a few months.

I am now in a position where I don't have to work for anybody. I love this career and I think many people would enjoy it and benefit from it as well. So I'm on a mission to help others who want to make this jump no matter what their economic situation, age or background.

I'm sharing this guide with you as a way to give back to the community and also because I think bootcamps and colleges completely overcharge you.

Don't worry, you can learn to code for free, get hired, and have fun as you will see below.

What programming language are we going to learn?

Yep, this one up here.

Javascript. You are going to become a Javascript ninja/ninjess/ninjother for the following reasons:

→ Javascript is everywhere. Every company that has a website or an app needs someone with Javascript knowledge. This language is a requirement for a TON of

job postings (If you don't trust me, search for Javascript in your area on LinkedIn).

→ With the introduction of Node.js, you can use Javascript to create a full-stack app (English = you can use Javascript to build your entire project).

Using tools like Electron, React Native and many others, Javascript allows you to build a desktop app, a mobile app, a web app, and even VR apps.

You can even control robots by using something like Jonny-Five.

Want to be in Data Science or Data Analysis? Maybe AI and Machine Learning? Great, you can use Javascript to do all of that too.

→ If you didn't read the first point and you were thinking about something else, let me reiterate: Industry demand for Javascript experts is HUGE.

It is the most in-demand language by employers and growing *(TypeScript is part of the JavaScript ecosystem so you can consider them as one)* and one of the most popular.

It is not uncommon for developers to get contacted by recruiters and head hunters multiple times a week with job offers.

→ The Javascript community is growing at a crazy fast pace. There is a lot of new developments in the community. Some people complain about Javascript fatigue since more and more new tools are being developed every day.

However, we can use this to our advantage. You will be learning these new technologies at the same time as people who have been in the industry for years.

→ But what about Python? I heard that it is all the rage? Python is definitely great for things like Data Science and Machine Learning, but you don't get that immediate satisfaction that Javascript gives you because it isn't really used to

build websites (which is the easiest way to get excited about coding).

Both languages are in demand, but the path of a JavaScript developer is more defined and focused to getting you hired most quickly. You can learn Python later in your career. Many people follow the JavaScript --> Python path, myself included.

Trust me, it is a great community with a lot of demand.

What is the 20% that will get me 80% of the results?

Most people have an idea that you need to get to 100% before you can move on to the next step. However, for most skills, including programming, the closer you get to 100%, the longer it takes to get there.

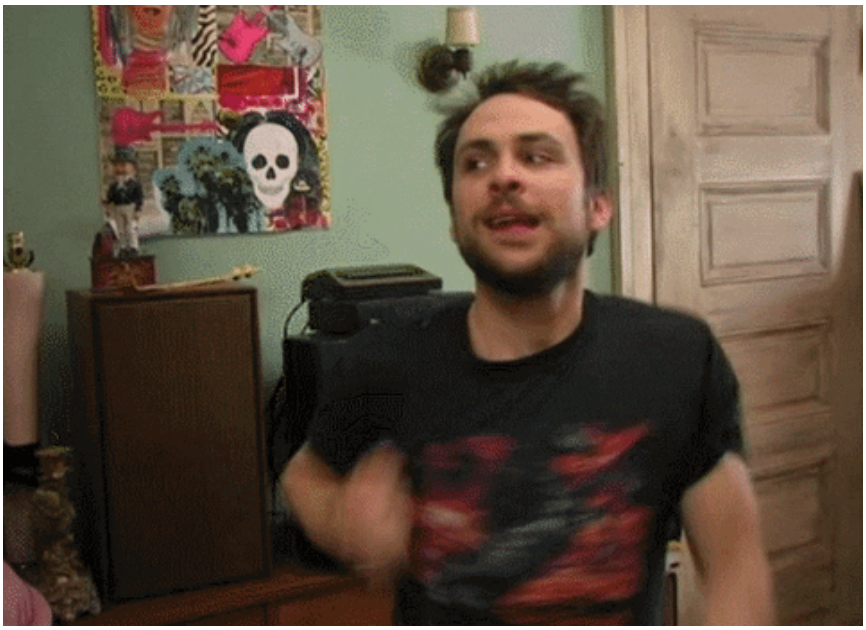
You only have 5 months.

The last 20% will be better achieved by actually working in teams, on real projects (and getting paid). So we are only focusing on getting 80% of the knowledge to use our time efficiently.

So enough jabber, let's get started. Below you will find what I believe are the best resources for you to get the most out of your time.

By the end of 5 months, you should be able to learn to code for free and land your first real non-entry level programming job. No coding bootcamps. Just you, this guide and your determination.

The 5 months: step-by-step



Get excited!

We will be focusing on the most employable and in-demand skills in 2023.

We have no time for outdated technologies like Wordpress, PHP or jQuery.

There is nothing wrong with them and I have total respect for them. But based on many of the emails I have received over the years from students, a lot of people are in financial need and have families that they have to support.

Time is important to you and you want to be employable as soon as possible. To do that, you need to learn the most modern skills with the best job prospects. So that's exactly what we're going to do.

1st Month: The Big Picture

Big question we will answer: *How do computers, the internet, and websites work? How can I build my first website?*

- Before we get to coding, make sure to understand the Feynman Technique and the Trunk Method so that you are truly *learning* over the course of the next 5 months instead of just using your short-term memory. This will help us learn faster and more efficiently.

- Throughout the months, you will be building lots of projects. In order to help you, I have compiled a list of assets like free images, templates, icons, logos, etc. that you can use to make your projects look nice. Bookmark this list of free resources.

- How does the internet work? How does the web work? Watch the first 40 minutes of this crash course (we'll come back it to it a couple times later this month) and then watch this computing networking playlist.

- Get an overview of Computer Science: this is the best one to start with and then watch this Map of Computer Science.

- Watch the real web developer roadmap to understand the full web developer industry so that you understand how all the skills fit together. When you first watch this, it will seem very confusing. At the end of the 5 months, watch this again and you will see how everything makes sense now. Although this is from 2020, 99% of the fundamentals are there for 2023. We will also explore some of the more modern technologies in the later months of this guide.

- Follow this CS50 Harvard course on YouTube (you don't need to do the exercises). This is just pure gold from probably the best computer science instructor there is and you don't need to pay $50,000/year in Harvard tuition. They do a new one every year but this one from 2017 is the best version of the lectures since it focuses on the web. If you have time, you can watch this year's lectures as well... but I still recommend the 2017 one.

- [ ] Read this to learn how to use the command Line (by Zed Shaw). Don't worry that it has Python in the title. Command Line is a topic that applies to all programming languages including the web development field.
- [ ] Next up... Watch this YouTube playlist to learn how to build a website, get a domain, and have it up and running.
- [ ] How does HTML + CSS fit together? Back to the crash course from earlier. Watch from this timestamp until you get to Flexbox (~2 hours... from 41:00 - 3:06:00). Make sure to actually pause the video to code along and try to figure things out and not just watch the video.
- [ ] Learn to build websites with Bootstrap. Start with this Bootstrap 5 tutorial then go to the Bootstrap 5 documentation and add components you see there to a sample website. Understand the benefits that it provides vs. writing CSS yourself. BUT... don't get stuck on this because we want to focus more on the next two which are more employable skills 👇
- [ ] Flexbox & CSS Grid. Learn how to use Flexbox and CSS Grid by watching of the rest of the crash course we've been following starting from this point onwards. Once you're done, go practice Flexbox here and also on Flexbox Froggy and then practice CSS on CSS Grid Garden. Here are two good additional resources to learn Flexbox and to learn CSS Grid.
- [ ] Now go build your own website layout from scratch.
- [ ] Understand how to use templates to build websites using free themes and templates.
- [ ] If you have time, you can do a few of the courses on the HTML and CSS Responsive Web Design sections from freeCodeCamp. It's 300 hours long so I wouldn't say this is the best use of your time but a nice thing to skim through.

THIS IS IMPORTANT: Don't try and memorize all the HTML and CSS properties and tags. This is a mistake I made as well.

You want to start learning Javascript as soon as possible, which is the main part of being a web developer. No matter how “unready” you feel or incomplete your knowledge of CSS is, just move on to the next part as you will still be using and learning HTML and CSS throughout the rest of the months.

Trust me on this.

2nd Month: Javascript

Big question we will answer: *How does Javascript make machines do what you want?*

This is where most of your focus will be over the next few months. Before we dive in, let's do a quick recap and see where Javascript fits into the picture.

Let's look at a house as an example.

HTML is like the blueprint (content). It outlines the structure and layout of the house (ex: the locations of the walls, doors, windows, and rooms).

Just like a blueprint, HTML tells the web browser where each piece of content should go.

CSS is like the design (style). It controls the appearance and style of the house (ex: color of the walls).

Just like interior design, CSS helps you make your website look nice.

Javascript is like the electrical system (behaviour). It adds functionality and interactivity to the house (ex: being able to turn the lights on and off).

Just like an electrical system, Javascript lets you add dynamic features to your website, like buttons that do something when you click them, or forms that validate input.

- [ ] Do this free in-depth course and start writing little programs in Javascript to make your website(s) from Month 1 behave in a certain way.

- [ ] Learn about DOM manipulation. Learn to inject `<script>` tags in your html to run Javascript files. And then do these free DOM lessons from my web developer bootcamp.
- [ ] Read this great article about programming.
- [ ] This is a long series that you won't finish (and don't need to) but make sure to bookmark it and use it as a reference anytime you encounter something you don't understand in Javascript.
- [ ] Learn the new ES6, ES7, ES8, ES9 and ES10 features with this tutorial. If you don't get everything in here, don't worry, we will go over another resource next month on the topic of "Asynchronous". Understand the difference between JavaScript and ECMAScript by reading this and understand how JavaScript gets updated every year. Then stay up-to-date with the latest JavaScript features that are coming up using this website.
- [ ] Learn Git and Github with this 40 minute tutorial (yes, that's me 🙋‍♂️). Also use this Git Explorer to practice and then learn more about Git Branching here.
- [ ] Create a GitHub profile. Don't waste a bunch of time here but at least make it look decent. Follow the steps in the video below to get setup and create a nice looking profile in under 20 minutes. Once your done, start making commits every day.
- [ ] Start developing a website. Use Github pages to put your website online for free. This will be your portfolio.
- [ ] Use this tool to learn Terminology & Jargon.

3rd Month: Javascript + NPM + Building Your Website

Big question we will answer: *Can I build a professional looking website and understand the entire process?*

- Google Developer Tools → learn how to debug your programs and websites using Google Chrome. Follow this short little course.
- Start networking by attending local (or virtual) meet-ups on coding and Javascript.
- Learn the difference between synchronous and asynchronous Javascript.
- What is the event loop? → Once you have a good grasp of Javascript event loop, this talk will be a game changer. Hands down the best talk on Javascript ever given. Watch this video every month for the next 3 months. Then watch this free video that I made to make sure you understand how Javascript works.
- Learn about Promises and Async Await in ES7 here.
- Learn about the history of modules in Javascript here.
- Download Node.js and NPM. Download lodash from NPM and use browserify to use CommonJS imports. Learn about it in this tutorial. Not many people use browserify anymore, but it's important for you to get the historical context of why it was used. Understand why NPM is such an amazing tool for developers. Now learn about why we no longer use Browserify, and learn about native imports and exports by watching this.
- Now using the above knowledge, learn a skill often overlooked by beginners: how to read documentation. Learn to use Parcel by reading their documentation and see how it bundles your code.

4th Month: ReactJS

Big question we will answer: *What problem does React solve?*

I'm biased. I love React.js. So much that I teach it to others and run workshops on it. So just trust me on this one.

React dominates the industry when it comes to job demand. In 2023, this trend is even stronger.

There is also Svelte, Angular and VueJS as main alternatives, but you want to stick with React for the best outcome. For example, check out the average salary of a developer that knows React.

- Learn React from scratch:
  - Do these in order: one, two.
  - Then head on over to the official documentation and read through it.
  - If you have the time and you want an even more in-depth tutorial on React, here it is for free in text form.
- Optional: Learn Redux → Watch this course. Don't let your head explode. Then read the documentation for it as well. Learn why managing state is a big problem that all large applications need to solve.
- Build a sample React application using create-react-app. Create-react-app will blow you away. It will open up a new world for you. Command Line Interfaces (CLIs for short) are now becoming common practice with all frontend frameworks and lets us set up a project quickly.
- Deploy your React app on GitHub pages. In the future, you should deploy all your projects on Github pages to show off in your portfolio.
- Deploy your React app on Netlify.
- Start reading through some of the articles here in the Must Read category.
- Sign up to these email lists to keep in touch with what is happening in the industry: Javascript Weekly and Web Developer Monthly.

- [ ] Create your resume. Make it simple and concise. One page. To help you out, watch this video and this one. Then check out this blog, this video and this one. As a bonus, check out this workshop... but this post is already getting too long and you're starting to give me evil eyes. Don't spend a lot of time here.
- [ ] Next, update your LinkedIn profile with similar details. Add some skills to your profile and then get some friends and family to help endorse your skills. Follow the steps in this video. You can also join our group to help endorse skills with other ZTM students.

5th Month: Servers, Databases and Connecting the Dots

Big question we will answer: *Where do servers, databases, and raspberryPis fit into all of this?*

- [ ] HTTP, JSON and AJAX. Learn how these allow you to communicate with servers.
- [ ] Go a step further and master Node.js and Express.js here. Learn how to build an API server.
- [ ] Once you are done with this, use a fun API found here to build a simple app. This is my favourite one.
- [ ] Subscribe to the computerphile YouTube channel and watch their videos as they come. Even though some topics will be difficult, it will introduce you to some amazing things.
- [ ] Optional (not "free"): What is a Computer/Server/OS? Buy a raspberryPi. Look up different projects on YouTube that you can do with your raspberryPi. Build a simple script that makes lights attached to your raspberryPi blink. Follow this course. Host your website on the raspberryPi.

Be amazed at how cool you are. I know I said this is supposed to be all free, but when I was starting out, doing this was one of the best "Aha" moments I had. It could be for you, too.

- [ ] Learn basic Web Architecture concepts by reading this article.
- [ ] If you have the time, spend a day building this chat application using React hooks and sockets. Add this to your portfolio (you did this in Month 2 right?! That website you created and hosted for free on Github pages).
- [ ] Learn the difference between server-side rendering, client-side rendering and pre-rendering (static site generator).
- [ ] Start practicing for interviews by trying to answer all of these questions. If you get something wrong, learn why you made that mistake and learn from it. Learn about behavioural interviews here.
- [ ] Spend one day each on the subjects below. You don't need to have a good grasp on them. Just learn why they are there and what problems they are solving: Testing (TDD), Machine Learning Basics, Time Complexity (Big O), SQL, TypeScript, UX/UI, Continuous Delivery, Basic Data Structures (you should be able to explain what a data structure is. Hint: Arrays and Objects are two popular Javascript data structures).
- [ ] Build a small project using a database you create here. Go a step further and create an app using firebase as the database and use firebase to set up user login/logout.

> ***I can already hear people screaming at me based on these suggestions. "Are you out of your mind?! You don't think X topic is important? Only 1 day for each of these topics?" Hear me out.***
>
> ***Yes, there are many important topics to learn to be a great developer. But we are focused on efficiency: 1) build your foundation and 2) get you hired.***
>
> ***Most job postings you will be applying for won't mention skills other than what I'm telling you to focus on. You might as well spend more time learning these additional skills once you are on the job (and getting paid to learn them).***

- [ ] Read and follow the steps I outline in this post: How to Interview, Land a Job, and Get a Raise. You should also read this coding interview guide.

REMEMBER: Your goal is to get employed in the most efficient way. Your learning will grow exponentially once you're on the job so we just need to get to a solid foundation.

Let's Recap

By the end of the 5 months, you should have the below requirements completed:

1. Learn HTML and CSS.
Then, buy a domain and hosting from a place like BlueHost or HostGator. Get the cheapest option, make a website, and put it online. My personal favourite is Netlify. If you don't want to pay, you can use Github Pages which is free. But if you can afford it, buy one of the above hosting platforms so you understand how they work.

This is going to be your portfolio from now on. Learn how to update it and make edits. As you learn new things, continue to make it nicer and nicer. Don't spend too much time on this. Just enough to show that you're able to put something online and make it look nice.

Focus on having 1-2 really good and big projects in your portfolio instead of 30 small ones that anyone can build in a day (since employers won't find this impressive).

2. **Start learning Javascript.**
Now how can you make your website interactive? Go through the above resources and see what Javascript does.

3. **Start pushing your little projects to GitHub.**
Employers will look at your GitHub profile and how active you are on there. Try to make commits 5 times a week on your personal projects.

Also, read through this article and try contributing to some open-source projects like freeCodeCamp or Zero To Mastery Open Source.

We set up the projects here so that you can participate no matter when you join or what your level is. You can read the getting started guide here.

4. Learn how to ask questions and where to find answers on your own.

Learn to Google and use StackOverflow when you run into problems. 99% of the problems you will encounter when you start out can be found online.
You can also join a Discord or Slack server for developers and ask questions when you are stuck and can't easily find the answer on Google or StackOverflow. Here is a Slack group list where you can talk with other developers.

The key is to practice solving your own problems rather than just constantly

following tutorials (aka tutorial hell) and watching somebody else answer your questions. The best learning comes through the struggle.

5. **Become comfortable using a command line to do things.**
Always have it open when practicing and try using it instead of the GUI (graphical user interface).

6. **Learn the newest language features and trends in Javascript, and learn to solve problems with them** (i.e. Promises, ES6, ES7, ES8, ES9, ES10, ES2020, ES2021 functional programming techniques).

Also keep an eye out on the state of Javascript survey every year to see what is trending in the industry.

7. Build your first project using React.
If you haven't already built a project using React, make sure you have one in your portfolio by following this crash course.

8. **Attend local (or virtual) meet-ups and start talking to people.**
You will be overwhelmed and confused by all of the things you don't know. Don't worry, this is natural. Just start meeting other programmers and coders so you're surrounded by the lingo and jargon.

9. **Keep up with the industry.**
Do this by reading my monthly industry newsletter and by listening to the Javascript Jabber podcast and/or the SyntaxFM podcast.

This will get you familiar with the jargon so when interview time comes, it doesn't overwhelm you. The first few times you listen, you will have no idea what they are

talking about. Don’t lose hope. Eventually it will all make sense.

For a more advanced podcast, but probably the best on software, check out Software Engineering Daily. This is a podcast you will appreciate a lot more later on in your career.

I'm not going to mention YouTube here because we all know it. Search YouTube anytime you want to learn quickly about a certain topic. There are so many more options out there now compared to when I first started years ago! You can also check out the Zero To Mastery YouTube Channel.

10. **Create a simple resume and make your LinkedIn profile look nice.**
Don't spend too much time on your resume. Make it one page, make it concise, and write down all the skills you've learned in the previous months. Use a prebuilt template like this.

Being self-taught shows a lot of courage. Remember that your resume is just to get you an interview, after which, they are as good as paper towels… ok bad analogy because paper towels are very useful. I spent less than 2 hours on my resume.

What makes you different than other developers is the fact that you come from a different field and background. Think about and highlight how your background and experiences differentiate you.

Once you have your one-page resume, update your LinkedIn profile with similar details. Follow this guide.

11. **Start applying to recruitment agencies.**
We are going to use them as practice. Most of these have practice interviews with professional coders so they can rank your skills. You are going to leverage these to practice programming questions and ask these experts any questions you want, for free!

12. **Start applying for jobs that you are way under-qualified for.**
You probably won't feel like you're ready, that's totally normal. But you'll be surprised when you actually get some interviews. Start off by applying for jobs/companies that aren't #1 on your list.

The more jobs you apply for, the more interviews you get to do and the better you will become at doing them.

Then start applying for the jobs/companies you want most.

You should never settle for a job (ex: the first offer you get). Aim higher and don't just settle for the lowest junior dev role.

See part 2 for more detail on this.

13. **Interview and be amazed at how employable you are.**
Not all of them will go well, but then again, not many developers learned everything in the last 5 months. It shows ambition.

ONLY apply to jobs on LinkedIn. Don't waste your time on the mass job board sites like Craigslist, Kijiji, or Monster.com.

Spend the rest of your time networking to get referrals and directly emailing, messaging or calling the company you want to work for. I go into more detail on this here.

Finally, you can check out this handbook for some technical interview advice.

14. **At the end of 5 months, watch this roadmap again to understand the full web developer landscape and industry (still 99% relevant even though I created this in 2020).**

With everything you've learned, you should now be able to understand everything in the video and how all the skills fit together. This should give you confidence moving forward.

Biggest takeaway from all of this

Technology is always changing. This is especially true with web development. Things are moving so fast right now that it is impossible to know every single library, syntax, or framework.

What you do need to know is how everything fits together, what each technology is trying to solve and the foundations. Most importantly, you just need to know what exists so you can easily look into it and figure it out when the time comes on the job.

Programmers are problem solvers. Learn to solve problems with the tools available to you. Most of us (yes, even senior developers) spend a lot of time on sites like StackOverflow and looking things up on Google.

Once you build the foundation of your knowledge, you can go anywhere. You just need to know how to look for answers and ask questions.

Conclusion

Focus on efficiency. The reason most of us give up on a goal is because we don't see results. By focusing on the things that matter, it makes learning fun.

And it doesn't end here. Learning never stops. Our first goal is to get employed as a developer as soon as possible so that you get to keep learning while getting paid.

Coding gets more and more fun with each passing day and it's even better when you are getting paid every day to solve problems and develop your skills. The real growth happens when you start working on real projects with real teams.

That's why I strongly believe that you want your initial 'study' period to be as short as possible. And you get to avoid going into debt and increase your time in the best environment for learning: working in teams.

I wouldn't even recommend freelancing to start off. You want to surround yourself in an environment where everybody is smarter than you and you are working everyday with them. From there, be a sponge and absorb all of the information.

We're building your tree trunk of knowledge. When that trunk gets big and strong, and the roots are all put into place, your rate of learning new things will be exponential. You'll form branches and leaves of knowledge faster and faster with each passing day.

> ***Make 2023 the year that you took a risk, you learned a highly in-demand skill, you were terrified, you had new experiences, and you received new opportunities. Good luck!***

One last thing...

I created an online course - The Complete Web Developer - where I personally walk you through all of the entire steps I mentioned above.

You just convinced me I could do this for free, why should I pay for your course?

You're 100% right. But this is an option if you want everything in one place, extra help with your questions and want to be part of our thriving community of thousands of students & developers who are also going through the course and helping each other out every day.

It's over 200 HD videos and 40+ hours of content. It took an insane number of hours to make (and keep updated), but I'm really proud of it. I strongly believe it is better than any coding bootcamp (and much more affordable) or any other online course out there.

And the testimonials speak for themselves.

Or maybe you just want to support my work by taking some of my intermediate and advanced courses once you use this guide to get hired.

Don't forget to get your company to pay for it. This is another perk of getting hired as a developer... most companies will give you a "learning budget" that you can use to keep developing your skills!

Update for 2023: You can now watch the first 6+ hours of my Coding Bootcamp using this link.

Good luck and most importantly…have FUN!

If you enjoyed this guide and found it helpful, all I ask is that you pay it forward and share it with someone else you think could find it useful.

Due to popular demand, I also wrote these additional follow-up posts to help you go from Junior to Senior:

Part 2: **Don't Be A Junior Developer: The Roadmap From Junior to Senior**. Seriously, don't be a junior developer. These are the steps and topics to learn to become a senior developer.

Part 3: **How to Interview, Land a Job, and Get a Raise**. The strategies and tactics I used to get multiple job offers after following the steps in this guide.