**The Definitive Guide to Becoming a FullStack Developer (2022)**

**Introduction, Background, & Disclaimers**

[⠀](https://www.simplilearn.com/ice9/free_resources_article_thumb/full_stack_banner.jpg)

The post is finally back! I've posted this guide to [GitHub here](https://github.com/aaltarazi98/fullstack-guide-2022) in case it gets taken down again.

THIS IS NOT A PROMOTION! None of the resources listed here belong to me, they are ALL FREE! I am not trying to promote myself or anyone else, just wanted to provide a resource for everyone. Thank you.

Hello everyone, in this post I will be writing a detailed guide on how to get a full-stack engineer job the self teaching way. This will include a more efficient version of what I did so you don't waste time. I will be going over what you need to learn, resources, and what you need to do after. It is critically important that you take EACH section to heart

A little background about me, I have been a construction engineer for a year when I decided construction was not for me and I wanted to go somewhere else. I took quizzes on what I should become, I landed on fullstack development and I haven't looked back since. Since then, I have learned a lot, built great projects, made connections, worked a contract, and landed a full time job. This process took me 8 months and it may take more or less time for you depending on who you are.

I want to preface this by saying, this is NOT the only way to learn full-stack development and there are many other stacks you can learn. This guide is focused on MERN & PERN which are very popular in the USA. For instance, [the Odin Project for JavaScript](https://www.theodinproject.com/paths/full-stack-javascript?) is a great alternative.

Do NOT be overwhelmed with the sheer amount of content here. It is a lot, but it will all become secondhand knowledge with time. Take it one section at a time and do what you can. Now without further ado, let's get started.

**Roadmap**

Here is a general roadmap of how your process should look like, I will provide you with resources and guidance at each step.

* CSS & HTML
* JavaScript
* Git & GitHub
* Build a project with Git, vanilla CSS, HTML, and JS
* Node.js & NPM
* React.js
* TypeScript
* Build a project with React.js in TypeScript
* Express.js
* MongoDB & PostgreSQL
* Build a full-stack project with either MERN or PERN (or BOTH)
* Bonus material, and projects with bonus material
* Build your portfolio & resume
* LC & Sending out Applications

Always remember that you need to tailor some things to what works for you. This is by no means a size fits all approach, but it will work if you follow it as closely as possible.

A VERY IMPORTANT NOTE ABOUT PROJECTS: You need to build something UNIQUE and OF YOUR OWN DESIGN/STRUCTURE. Do NOT look up easy examples of projects because they will NOT get you far. You must hold yourself up to a standard. This will give you a better understanding of full stack development and systems design which is critical for a lot of jobs.

Now, let's break down each section.

**CSS & HTML**

CSS & HTML are the bread and butter of every website. They determine the overall structure, content, and looks of every website. Here are the only things you need to cover them so make sure to follow along the course:

* [HTML Crash Course](https://www.youtube.com/watch?v=UB1O30fR-EE&t=1s)
* [CSS Crash Course](https://www.youtube.com/watch?v=yfoY53QXEnI)

Great, now you know how to build a basic website. Let's move on to JS.

**JavaScript**

If you are a full stack engineer, this can be the only language you ever need to know, Thus, it IS critical that you come to learn it DEEPLY and understand how it functions. It is always up to you how you want to learn but I will recommend this e-Book which is FREE and EXHAUSTIVE and will contain all the info you will ever need on JavaScript as a vanilla language. You need to go through both Parts 1 & 2 to understand JS as a language and how it interacts with the browser.

Not all of the book will make sense to you now, but I promise you will use its information once you move on to React, Node, and LeetCode. Furthermore, watch the event loop video which is important to understand JS in the browser and will allow you to do some cool stuff.

* [The Modern JavaScript Tutorial](https://javascript.info/)
* [What is an Event Loop Anyway?](https://www.youtube.com/watch?v=8aGhZQkoFbQ&t=1s)

Congrats, you now understand HTML, CSS, and Vanilla JS

**Git and GitHub**

Git is a version control system that allows you to manage your projects and code via versions. Furthermore, it will allow you to post things to GitHub and host them online. GitHub, which I'm sure you've interacted with at this point, is an online platform where you can share and post your code on the internet. It is crucial for hosting websites and servers. Git Bash is a CLI for Git that will allow you to execute Git commands in the terminal.

* [Git Bash](https://www.educative.io/edpresso/how-to-install-git-bash-in-windows)
* [Git & GitHub Course](https://www.youtube.com/watch?v=SWYqp7iY_Tc&t=1077s)

Now that you've learned these two. Let's move on.

**Build a Project**

Now that you learned Git, HTML, CSS, and JS, you will be building your first project. Use git init to start a project and take it from there. I will leave the details to you.

For each and every single project step in this process, you NEED to THINK of what YOU WANT to build and build it! Since this is your first project, be realistic with what you can accomplish but CHALLENGE yourself. What you have learned so far will NOT be everything you need to make this project happen. Google is your friend as you will need to visit MANY websites to learn how to make a certain thing work.

Here's a big hint: there are a lot of great free API's online that you can use for your project (Star Wars API, Weather API, Google Maps API, the list goes on).

Furthermore, you have to make your projects dynamic and mobile friendly. Look up CSS media queries as a starter on how to do that.

Challenge yourself, prepare to be humbled, learn, and build an AWESOME first project. Start strong!

**Node.js & NPM**

So far, we've made JS run in the browser, but how can we run it on our computer? That's where Node.js comes in. Node is a JavaScript runtime which allows your computer to understand and run JavaScript. All you really need to understand is that.

Node Package Manager (NPM) will allow you to install and manage packages via node, which allows you to customize your project with pre-built packages and services. This one is fairly straightforward and you will naturally pick it up as you're building projects.

* [Node Crash Course](https://www.youtube.com/watch?v=fBNz5xF-Kx4&t=1s)

**React.js**

Congrats, you've reached the big boy stuff. React is the single BIGGEST JS framework and the most widely sought out skill if you are looking for either a front end or full stack job. It is CRITICAL that you become REALLY good with React. Thankfully, this scrimba course IS A PERFECT FREE LEARNING ENVIRONMENT for React. Go through it step by step as the instructor says. This is how I learned React and became VERY good at it.

* [Scrimba Learn React for Free](https://scrimba.com/learn/learnreact)

**TypeScript**

TypeScript is a superset of JavaScript that adds static typing to data. What does that mean? It means that your IDE will know exactly what data type each constant/variable will be and will make your life A LOT easier. TypeScript's power comes when you're building a project as it builds a structure where you will KNOW if your code will run. Anyone who built a JS project knows how many times you will run into runtime errors.

It's important to understand that TypeScript does NOT actually run in your browser. It gets compiled down to Vanilla JS when it's being run. It is fairly straightforward and you will mostly pick it up just by using it.

* [TypeScript Crash Course](https://www.youtube.com/watch?v=BCg4U1FzODs&t=1s)

Let's move on.

**Build a project with React.js in TypeScript**

Now that you know TypeScript and React, build a React TS project using the same general guides for your first project (unique, ambitious, and awesome). Again, I will let you decide what you want to build for yourself. Make it a front end only, don't worry about servers and databases for now. Some resources to help:

* [React & TS environments](https://www.youtube.com/watch?v=Z5iWr6Srsj8&t=23s)
* [Create React App TS](https://create-react-app.dev/docs/adding-typescript/)

Once you've built a project you're happy with, let's move on.

**Express.js**

Express is a Node.js framework which makes running a server/API REALLY EASY for any project. Understand that when building your projects, your front end and back end will run on DIFFERENT ports. For instance, I like to run my react apps on 3000 and express apps on 4000. Now, let's learn some Express:

* [Express Crash Course](https://www.youtube.com/watch?v=KgnJNJk9-to)

**MongoDB & PostgreSQL**

MongoDB is a NoSQL database, which means each data type is unrelated to other data types and it uses it's own query language. That's not to say these schema do not interact with each other. PostgreSQL, on the other hand, is a SQL database which means it uses Structured Query Language (SQL) to work and the different tables can interact with each other. You should definitely learn both, but it doesn't hurt much if you just learn one. Some jobs will look for SQL others will look for Mongo, up to you but I recommend both.

* [MongoDB crash course](https://www.youtube.com/watch?v=-56x56UppqQ)
* [Mongoose crash course](https://www.youtube.com/watch?v=DZBGEVgL2eE)
* [Get started with PSQL](https://www.youtube.com/watch?v=fZQI7nBu32M)
* [SQL for Beginners](https://www.youtube.com/watch?v=5hzZtqCNQKk&t=8218s)
* [PG Node](https://node-postgres.com/)

You should learn PG node if you want to use PSQL in your node environments.

**Build a full-stack project with either MERN or PERN**

Congratulations, you now know everything you need to build your first full stack project. As with the other two, build something UNIQUE TO YOU. You will be putting these projects on your portfolio, be proud of them. You have two options here:

* Build a PERN or MERN project.
* STRONGLY RECOMMENDED: Built 2 different projects with both (one MERN one PERN).
* [Here is an EXCELLENT tutorial project](https://www.youtube.com/watch?v=-0exw-9YJBo&list=PLillGF-RfqbbQeVSccR9PGKHzPJSWqcsm), again from Traversy Media. You don't need to use every technology he uses, but they are covered in the Bonus Material section so you should try to learn them.

**Bonus Material**

This is incredibly important if you want to stand out, here is some extra stuff you can learn to take your full-stack projects to the next level.

* [Material UI](https://mui.com/) - A library of components that makes building frontend projects easy and uniform. Highly sought after in candidates and I use it on each and every single one of my projects.
* [Redux & Redux Toolkit](https://www.youtube.com/watch?v=iBUJVy8phqw) - A state management library that makes managing global state in your projects really easy. Strongly recommended.
* [React Router](https://www.youtube.com/watch?v=Law7wfdg_ls) - A library that helps manage pages on your apps.
* [JWT & bcrypt](https://www.youtube.com/watch?v=mbsmsi7l3r4) - Straightforward packages that help secure your backends:
* [Socket.io](https://www.youtube.com/watch?v=9HFwJ9hrmls&t=429s) - Websocketting is a powerful alternative to traditional REST API's. This establishes a two way connection between your server and frontend where the server can send information to the client at any time! It allows you to build things like multiplayer games, chat apps, streaming services, and more!
* [Next.js](https://nextjs.org/) - A powerful React/Express framework built on top of React Router. It allows your website to be statically served by the server (SSR). Improved performance and overall security!

Whatever you decide to learn (I recommend all) you MUST either build a project with all these technologies or implement them in your old projects.

**Build your Portfolio & Resume**

To build your portfolio, you will need to host your projects online. To do so, you must get familiar with Heroku; where you will host your servers, and Netlify; where you will host your websites.

* [Heroku (Backend)](https://www.heroku.com/)
* [Netlify (Frontend)](https://www.netlify.com/)
* For databases, there are a lot of options for SQL, but you should use MongoDB Atlas for MongoDB.

Now that you're familiar with these two, push all your projects to GitHub and use Heroku & Netlify to host them as needed. Pin your projects on your GitHub, make a clean readme for each one, and a readme for your profile to stand out.

For your resume, you will highlight your projects and all the skills you learned. [Here is mine as an example](https://www.reddit.com/r/resumes/comments/tk4r5h/hey_all_im_a_self_taught_full_stack_developer/)

**LC & Sending out Applications**

You may or may not need to LeetCode to land a job, however I strongly recommend it because it will teach you a lot on how to improve as a developer. Sure, it gets a lot of hate from the dev community but it's part of the game you need to play to get a job. Better to learn and work than to complain about it. Here are the only resources you need:

* Go through [this repo](https://github.com/Chanda-Abdul/Several-Coding-Patterns-for-Solving-Data-Structures-and-Algorithms-Problems-during-Interviews) pattern by pattern and look up how to solve problems of each pattern until you understand them and can solve them.
* [A very helpful and structured guide for LC problems](https://neetcode.io/)

Build your LeetCode experience and solve problems as much as you can.

Now that you have a resume, GitHub, projects, and LC under your belt you can start applying. I won't get too much into this because it is beyond the scope of what I'm trying to convey so you will need your own research. Build a strong LinkedIn and AngelList profile. Apply to companies on both, email them, call them, sell yourself. You NEED to hustle on the jobs you REALLY want if you want to get them. After enough applications, you will land something. Each failure is a learning experience for you, so your soft skills better be sharp as a knife. Good luck.

* [Referrals are King](https://www.reddit.com/r/cscareerquestions/comments/uczkay/referrals_are_king_a_shithead_guide_on)

You can still land a job by cold applications, and that's what I did. There are plenty of guides on this section online, I'll leave that research to you.

At this point, your projects and the knowledge you've built while working on them will CARRY you through your interviews. Believe in yourself and what you've accomplished.

**Closing Remarks**

This by no means is a one size fits all, and you will likely deviate from it a little bit and that's completely okay. I intentionally left a lot of details out because you will need to be comfortable running on your own, be ready to do LOTS AND LOTS of research to get what you want.

Wishing you all luck on your journeys. Stay strong, ambitious, patient, and hungry my friends. Please let me know in the comments if you have any questions or input and I will be glad to answer.

EDIT: Thanks to everyone for the feedback, I will be updating this list to be better. I plan on keeping it up-to-date as much as I can so it can always be a go-to on Reddit.