The Path to Becoming a Front End Developer in 2019

10
Shares

JANUARY 15, 2019

8

s 2019 kicks off, there's always a flood of New Years' resolutions.

ose weight. Eat healthy. Get a job as a front end developer.

ou know. Little, easy things. Nothing crazy.

n kidding, of course. These big life goals are never easy. *Simple*, maybe – go to the ym every day, choose salad for lunch, practice coding every evening – but we know om experience that actually *doing the work every day* is not easy.

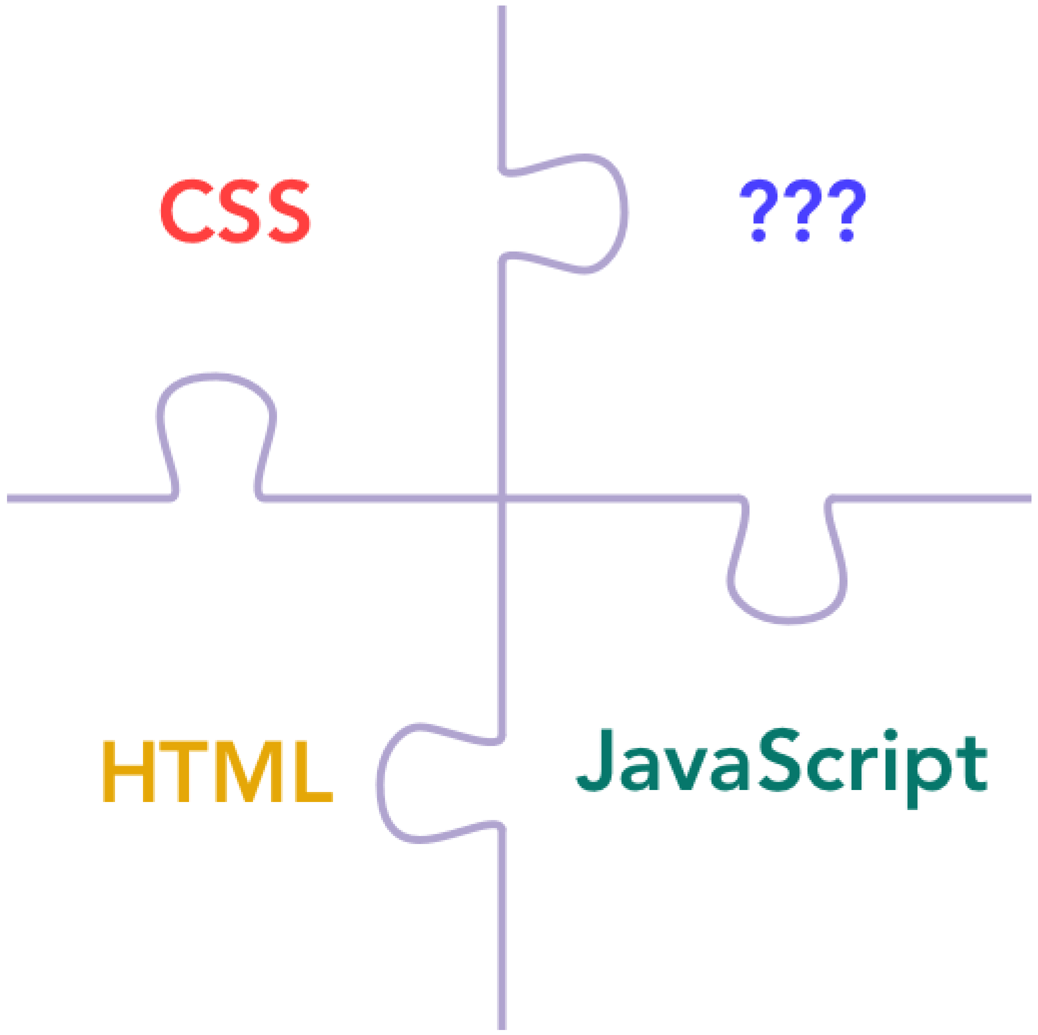
And the results never come as quick as we want. Day-to-day, sometimes it feels like a slog. Honestly, it *is* a slog sometimes.

And yet, if we want to make progress on our goals, *real* progress, some amount of daily effort is a big help. Daily effort quickens the pace.

The big decision, then, is **what to focus on**. What are the critical skills you need to master, and what order should you tackle them?

I'm going to try to give you some perspective and a concrete plan of action.

Learning is a Puzzle



10
Shares

8

If you're putting together a puzzle, you probably wouldn't try to assemble the top row, then the second row, then the third. Puzzle pieces don't tend to fit into "rows." Except in that image I drew, because it was easier to draw that way.

More likely, you'll start from the corners and the edges, and work your way in. Start with some pieces you're *sure of*.

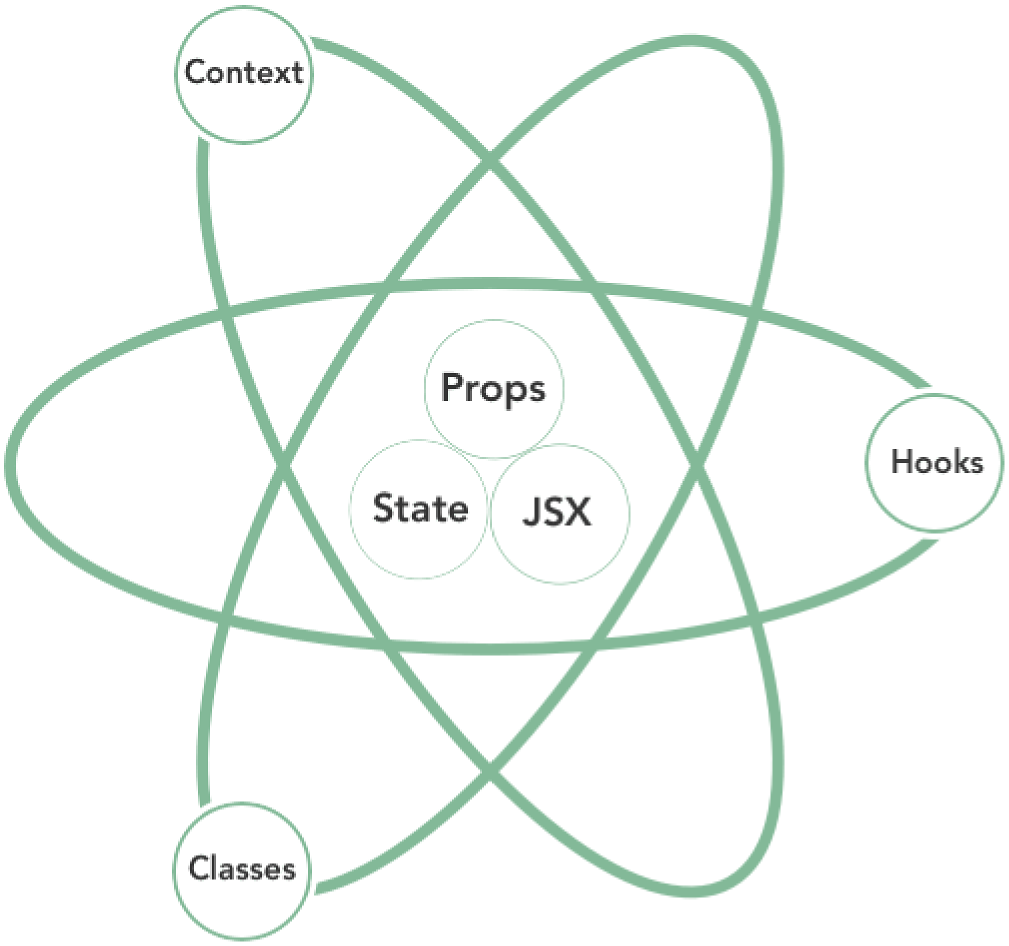
Along the way you might discover a handful pieces that fit together in a little clump. 2, 3, maybe 5 pieces. Then you'll wonder "where does this clump fit?", but you might not find an answer until much later in the puzzle.

Learning web development is like this.

You are collecting little clusters of knowledge. Some of those clusters you'll use every day, and some of them you'll use rarely (but they'll come in super handy one day, like when the server crashes at 2am and you remember that you used `grep` once, and start grepping through logs to find the problem.). They're all part of the massive puzzle we call "web development."

Learn in Isolation

Whenever you can, break down the subject you are trying to learn into the smallest atoms possible – individual skills.



Then, learn each skill all by itself, even if that means taking a diversion and creating a small sandbox project to focus on that one skill. This will make the learning go much faster, and you'll retain more of it.

If there was one "hack" I could give to every person learning web development, it would be this: the ability to see a problem or project not as an *indivisible thing*, but as a cluster of skills which can be broken apart, learned separately, and then recombined.

React can be deconstructed into just a few core pieces: JSX, props, and state.

You'll learn them piece-by-piece in this 5-day course. Sign up below to get your first lesson right now!

10
Shares

8

Your email address

Start Learning

reak it down: Full Stack React App

ere's an example. Let's say you want to learn all the parts of building a roduction-level app with React, so you can build your own (or get hired to build ıe).

An app like this is made up of many parts.

There's the front end. It's written in JavaScript, probably. It uses React. Maybe Redux or MobX. There might be a Webpack config. There's definitely JSX. Maybe CSS or Sass, or maybe CSS-in-JS. There are probably HTTP requests with `fetch` or `axios` . There might be GraphQL.

The back end might be written with Node + Express, or Ruby on Rails, or Elixir and Phoenix, or any number of things. And that back end probably talks to a database, which might be SQL-based (PostgreSQL or MySQL) or document-based (MongoDB).

All of that is *waaaaayyy* too much to learn simultaneously. If you take it on as one big project, intending to learn it all as you go, it turns into a big jumble in your head and it's hard to remember which pieces go where. Even if you just try to isolate the front end part, it's still an overwhelming amount of stuff.

So ask yourself, could I split it up by technology, and learn one at a time?

Learn Each Skill on Its Own

Could you learn JavaScript by itself? Sure, there's the excellent You Don't Know JS ries (free online), and plenty of other resources.

10 Shares

ould you learn React by itself? Yep, there's a book focused on just React (I wrote !).

8

X looks a lot like HTML, so it's probably easier to figure out how HTML works efore going too far with React + JSX.

ou could write a janky-looking app without even touching CSS at all, so you could efinitely learn React without CSS. Or CSS without React. And you could wait to arn styled-components or some other CSS-in-JS lib until after you figure out how SS rules work.

Redux is an add-on to React, so you could learn Redux by itself. (once you know a bit about React).

If Redux feels like overkill, you could learn the React Context API, which is built right into React, and solves many of the same problems as Redux. I have a course on egghead.io all about how to use Context for state management.

Webpack configuration is orthogonal to the goal of getting a React app on the screen; you could use Create React App now and figure out Webpack later.

External data, no matter whether it comes from plain a REST API or GraphQL, is another complication. For the purposes of learning how React works, you can use static data initially, by copying the JSON response data from the API and storing it in a variable – open up DevTools, Network tab, pick a request, copy-paste the response! Then learn how to asynchronously fetch the data later.

The back end can be broken down in the same way.

Always question the assumption that a problem or project must be learned or built as a whole. See if you can break it down. Tease out the individual parts or layers. Learn those parts on their own when you can. Sometimes you can’t do that, and in that case, still: strip out as much unnecessary stuff as possible.

Web Development Learning Plan

ll that said, real-life you needs some real-life direction right now.

ere’s the path I suggest, linearized the best I can. Keep in mind the idea of “just in me learning.”

actice each new thing as you learn it. Reading blogs & watching tutorials is great, it your brain won’t remember it for long without practice. Devise your own kercises if none are given. Here are some ideas for how to come up with your own eact practice projects, for instance.

or each of these things, learn *just enough*, then move on.

You do not need an encyclopedic knowledge of every HTML element, CSS selector, JavaScript feature, or command line tool. That’s what Google and StackOverflow are for. Most of us who’ve been coding for years can tell you about recently learning some fairly basic thing and being amazed by it. (right this moment: omg there’s a `dialog` element?)

A Bit of Command Line

Often times the first step to getting started on a project is cloning a repo from Github, or creating a blank project with `create-react-app`. Then you’ll need to run `npm` or `yarn` to install packages, and be able to navigate your project on the filesystem.

A little bit of command line knowledge can help make all of this feel less like typing magical commands into a magical box.

You don't need to go crazy here, but it's worthwhile to understand the basics of navigating the fileystem, displaying files, and that kind of thing. Learn Enough Command Line to Be Dangerous by Michael Hartl (of Rails Tutorial fame) is a nice intro, and it's free to read online. This article by Max Antonucci is also good: The Shell Introduction I Wish I Had.

10
Shares

8

ersion Control with Git

u know that problem where your code is working great, and then you change one ny thing, and then for some reason it breaks?

nd *then*, when you undo that change, it's **still broken**? wtf!!

is is literally the worst.

ersion control solves this problem, and I wish I had learned about it earlier than I d. (I also wish someone had framed version control as a *benefit* to me rather than *chore*. Because the benefit is HUGE, and Git makes it pretty easy.)

My introduction was this explanation of how Git works in story form, The Git Parable. Give it a read, it's great. Even if you understand Git commands I bet you'll learn something new.

Git is just one of many different version control systems, but it's the current reigning champion, popularized by Github.

Also, fun fact: Git and Github are not the same thing. Github is a hosting service for Git repositories, but Git existed long before Github and it can be used independently. You can use Git to manage your code locally without even having an internet connection, which is part of what makes it so awesome.

So once you've got a bit of command line knowledge under your belt, install the `git` command. Then whenever you create a new project directory, do this:

```
git init .
git add .
git commit -m "Initial commit"
```

And then every time you get the code into a working state, or before you make a change that might break something, commit your code:

10
Shares

```
git add .
git commit -m "Saving this before I break it."
```

8

ıch commit is like a checkpoint. You can go jump back in time to any previous ›mmit (back when your code was working). This all exists locally on your ›mputer. If your computer goes up in flames, your code is gone. That's what ithub is for (and offsite backups, I suppose).

you want to learn more about Git, like how to checkout previous commits to get ıck to your working code, push to Github, and more, go through this Learn ıough Git to Be Dangerous tutorial. It's not too long and at the end you'll even nave a webpage deployed to Github Pages. Which is a great sandbox for…

The Languages of the Web

To build software for the web, you need to know HTML, CSS, and JavaScript. At least enough of each to get by.

You can write HTML without any CSS or JS. You can't do much with CSS unless you have an HTML document to style. So you'll want to learn some HTML before you learn some CSS.

JavaScript's utility is multi-pronged, though. It can be added on top of an HTML document, to make an interactive app… or it can be used outside the browser with Node.js to do any number of interesting things – from writing servers, to command line apps, to controlling IoT devices and doing machine learning.

What seems more fun? Writing JS to do stuff in a browser, or learning it in isolation with little coding exercises in Node? No right answer here. It depends on your goals.

If you're leaning toward front end development, I'd suggest learning JS in the browser from the beginning, starting with "vanilla" JS, without React and Webpack builds and all that stuff.

10 Shares

nd in all of these, remember, the point is not to *master each skill to completion* efore moving on. Just learn enough, until it feels like you can tackle the next skill.

8

l probably get yelled at for saying this, but I don't believe you need to master anilla JS and the DOM before you move on to something like React, if what you ally want to do is use frameworks. I *do* think it's good to get at least a bit of ractice with them, and know that they exist, and be able to look up the specifics n MDN, for example) when you need them.

earn to Debug

ebugging is a learnable skill. Luckily, if you're like me, or most other people getting started, you'll have plenty of practice running into errors and needing to solve them.

The easiest solution, of course, is to copy and paste your error into Google.

Sometimes, though, Google is no help. Maybe the error is too specific to your code, like a syntax error.

And then sometimes, Google finds almost no related results. This is *almost always* a sign that it's a simple, silly mistake somewhere. Like "I forgot to save the file" or "I forgot to restart the server."

One time, I hit upon an editor bug in some flavor of Eclipse where the Save function *stopped working*.

I'd change the file. The title bar would show `AbstractFactoryObserverPatternImpl.java *` (with the little star, showing it had been changed). I'd click "Save" and mash Ctrl+S, but the little star remained. "Maybe it's a UI bug," I thought. So after saving I opened up the file in `vim` – and sure enough, it hadn't been changed.

This was (a) ridiculous, because text editors basically have only two jobs, editing text and *saving it*… and (b) a nice reminder to always check assumptions. Even crazy nes.

o the next time you run into a weird problem, break down the problem into layers, nd check every assumption in the stack. Brainstorm possible reasons for failure.

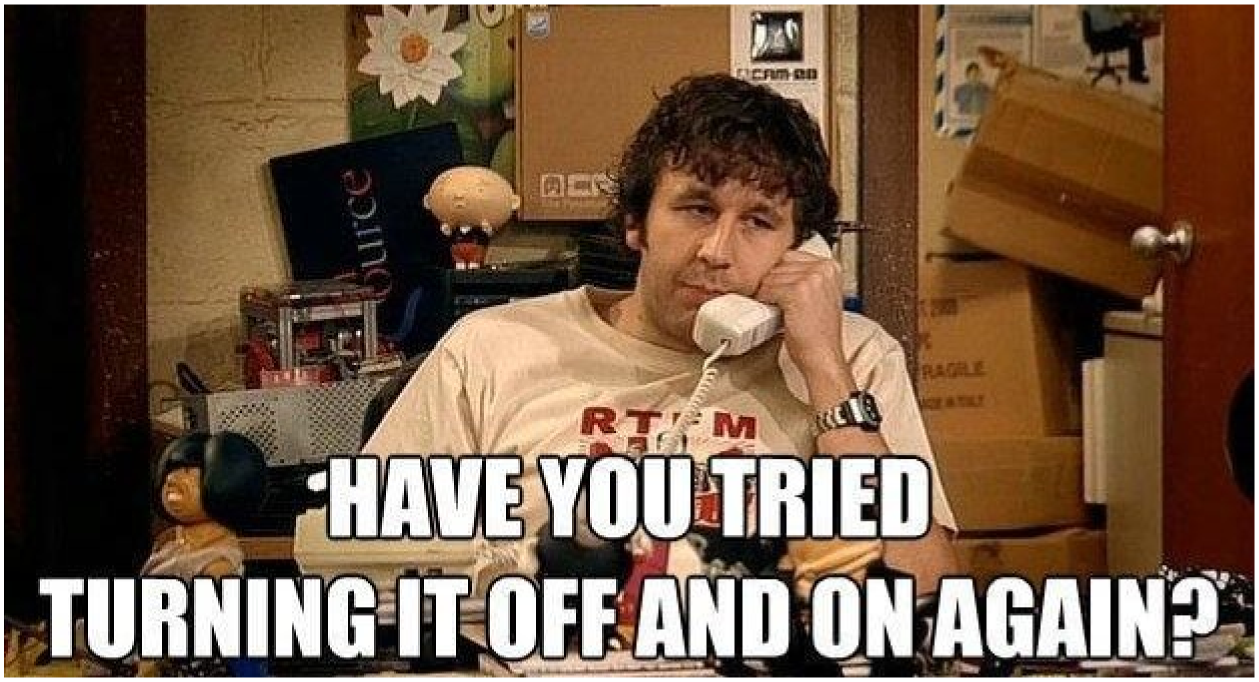
or example, with the "I changed the code but it didn't do anything" problem:

- Is that section of code running at all? Can you add a `console.log` and see it printed?
- Is the automatic build working? (or, if it's manual, did you run the build?)
- If it's part of a running server, has that server been restarted?
- Are you hitting the right server?
- Are you changing the right file?
- Are you changing the right *project*? (e.g. maybe you made a copy, `myproject-working-for-real-this-time-7`, but your editor is still open to `myproject-working-for-real-this-time-6`. Also: stop doing that and start using Git ;)
- Is the file actually changing? Did the Save function in your editor stop working?

After you ask all those questions, and you check everything, and everything seems correct but it's still broken… it's time to start restarting things. Trust nothing and nobody.

10
Shares

8



ɜmember how I mentioned earlier how learning web dev is like building little usters of interconnected skills? Debugging time is when you draw upon all those ːills. The better you understand *why* things work the way they do, including the ːterconnections between different parts of your app, the better and faster you can ɜbug problems when they arise.

ıou'll begin to see problems and solutions in high resolution. Instead of just "the app is broken," you'll see "the server threw an exception while preparing part of the data" or "the JS code failed to parse the JSON and stopped running."

Learn How The Web Works

It will be a big help to understand how the web works in general. Having the big picture in mind will help not only during development, but also help a TON when debugging.

Learn what happens when you visit a web page. There's a lot going on behind the scenes! Most of it can fail sometimes, too. If you know all the steps you can diagnose whether, for example, the server is down, or the DNS entry is wrong, or the server *machine* is up but the webserver process is not, or an adblocker prevented your icon font from loading, or whatever other weird thing might happen.

The Developer Tools are a huge help in learning how this works. Open up the devtools in your browser, look at the Network tab, and refresh the page. Look at all the requests that went out. Look for failures. Click into them and see the data they returned. This, by the way, is a great way to discover and use an undocumented "API" from a site you want to build an app around!

Learn a Framework

10 Shares

eact is very popular right now, and it benefits from a ton of good resources online r learning. The job market is also good for React developers at the moment.

8

ne official React Tutorial is a great place to start. It's well-written, and will help ou get from setup to a working app. I also put together a nice little free 5-day eact course for learning the basics, and if you want something more long-form, I rote a book, Pure React, that goes deep on *just React* with plenty of exercises and xamples to make it all stick.

you try React and don't like it, check out Vue.js. It's a popular alternative to React nd a lot of people love it.

Begin (or continue) Today!

If learning web development is a goal of yours this year, I encourage you to take action on it *right now*. Reading and wishing and hoping won't get you much closer. Putting hands to keyboard and writing code will.

Even though it's impossible to cover everything and provide a path for everyone's own unique starting point, I hope this guide has given you some direction in your web development learning journey. Leave a comment if it helped, or if you have done something specific that helped *you* move forward!

Learning React can be a struggle -- so many libraries and tools!
My advice? Ignore all of them :)
For a step-by-step approach, read my book Pure React.

Loved it! Very well written and put together. Love that you focused only on React. Wish I had stumbled onto your book first before I messed around with all those scary "boilerplate" projects.
— John Lyon-Smith

Learn the basics of React in 5 days

10
Shares

8

Finally understand how React works! You will:

🎉 Get something on screen
🔒 Write dynamic components
🏃 Make it interactive
😎 Fetch real data
🚢 Put it online

5 days, 5 emails. Walk away with the basics and a plan!

Get Lesson 1 right now 👇

Your email address

Send Me Lesson 1

I respect your email privacy. Unsubscribe any time.

5 Comments **Dave Ceddia** **Pauly C**

♡ **Recommend** 2 Tweet Share **Sort by Best**

Join the discussion…

zhaofeihao • 3 months ago
wow, that is really cheer me up, thx very much
1 ^ | ˅ • Reply • Share ›

Moonformeli • 2 months ago

Thanks for the good contents! Can I translate this post into my language, Korean, and upload in my blog with the back-link to here?

• Reply • Share ›

Joaquin Fox • 2 months ago

Thanks for the great post, insights, and resources. Pure React is fantastic and so are Michael Hartl's materials you recommend here.

• Reply • Share ›

10 Shares

Dave Ceddia Mod Joaquin Fox • 2 months ago

Thank you!

• Reply • Share ›

8

Long Tran • 4 months ago

oh my ghost. it's too long post

• Reply • Share ›

SO ON **DAVE CEDDIA**

ow to Use the useEffect Hook

omments • 7 months ago

atar Dave Ceddia — Hah, yeah... async worked when I first tried this. I saw that it's not supported anymore, but you can still call ...

How Redux Works: A Counter-Example

2 comments • a month ago

Avatar Dave Ceddia — Thanks! Awesome to hear :)

A Complete React Redux Tutorial for 2019

5 comments • 3 months ago

Avatar Chad Lare — Thanks for this. I already understood the general idea of Redux, but I was kind of fuzzy on some of the ...

How to Export a Connected Component

1 comment • 6 months ago

Avatar shrek — I am with `export default`

Subscribe **Add Disqus to your siteAdd DisqusAdd** **Disqus' Privacy PolicyPrivacy PolicyPrivacy**

Previous Next