Jose Aguinaga Follow

Web Engineer. Previously @numbrs, @plaidhq, currently @getflynt. Javascript, #people, startups, fintech..

Oct 3, 2016 · 12 min read

How it feels to learn JavaScript in 2016

No JavaScript frameworks were created during the writing of this article.

The following is inspired by the article "It's the future" from Circle CI. You can read the original here. This piece is just an opinion, and like any JavaScript framework, it shouldn't be taken too seriously.

Hey, I got this new web project, but to be honest I haven't coded much web in a few years and I've heard the landscape changed a bit. You are the most up-to date web dev around here right?

-The actual term is Front End engineer, but yeah, I'm the right guy. I do web in 2016. Visualisations, music players, flying drones that play football, you name it. I just came back from JsConf and ReactConf, so I know the latest technologies to create web apps.

Cool. I need to create a page that displays the latest activity from the users, so I just need to get the data from the REST endpoint and display it in some sort of filterable table, and update it if anything

changes in the server. I was thinking maybe using jQuery to fetch and display the data?

-Oh my god no, no one uses jQuery anymore. You should try learning React, it's 2016.

Oh, OK. What's React?

-It's a super cool library made by some guys at Facebook, it really brings control and performance to your application, by allowing you to handle any view changes very easily.

That sounds neat. Can I use React to display data from the server?

-Yeah, but first you need to add React and React DOM as a library in your webpage.

Wait, why two libraries?

-So one is the actual library and the second one is for manipulating the DOM, which now you can describe in JSX.

JSX? What is JSX?

-JSX is just a JavaScript syntax extension that looks pretty much like XML. It's kind of another way to describe the DOM, think of it as a better HTML.

What's wrong with HTML?

-It's 2016. No one codes HTML directly anymore.

Right. Anyway, if I add these two libraries then I can use React?

-Not quite. You need to add Babel, and then you are able to use React.

Another library? What's Babel?

-Oh, Babel is a transpiler that allows you to target specific versions of JavaScript, while you code in any version of JavaScript. You don't HAVE to include Babel to use ReactJS, but unless you do, you are stuck with using ES5, and let's be real, it's 2016, you should be coding in ES2016+ like the rest of the cool kids do.

ES5? ES2016+? I'm getting lost over here. What's ES5 and ES2016+?

-ES5 stands for ECMAScript 5. It's the edition that has most people target since it has been implemented by most browsers nowadays.

ECMAScript?

-Yes, you know, the scripting standard JavaScript was based on in 1999 after its initial release in 1995, back then when JavaScript was named Livescript and only ran in the Netscape Navigator. That was very messy back then, but thankfully now things are very clear and we have, like, 7 editions of this implementation.

7 editions. For real. And ES5 and ES2016+ are?

-The fifth and seventh edition respectively.

Wait, what happened with the sixth?

-You mean ES6? Yeah, I mean, each edition is a superset of the previous one, so if you are using ES2016+, you are using all the features of the previous versions.

Right. And why use ES2016+ over ES6 then?

-Well, you COULD use ES6, but to use cool features like async and await, you need to use ES2016+. Otherwise you are stuck with ES6 generators with coroutines to block asynchronous calls for proper control flow.

I have no idea what you just said, and all these names are confusing. Look, I'm just loading a bunch of data from a server, I used to be able to just include jQuery from a CDN and just get the data with AJAX calls, why can't I just do that?

-It's 2016 man, no one uses jQuery anymore, it ends up in a bunch of spaghetti code. Everyone knows that.

Right. So my alternative is to load three libraries to fetch data and display a HTML table.

-Well, you include those three libraries but bundle them up with a module manager to load only one file.

I see. And what's a module manager?

-The definition depends on the environment, but in the web we usually mean anything that supports AMD or CommonJS modules.

Riiight. And AMD and CommonJS are…?

-Definitions. There are ways to describe how multiple JavaScript libraries and classes should interact. You know, exports and requires? You can write multiple JavaScript files defining the AMD or CommonJS API and you can use something like Browserify to bundle them up.

OK, that makes sense… I think. What is Browserify?

-It's a tool that allows you to bundle CommonJS described dependencies to files that can be run in the browser. It was created because most people publish those dependencies in the npm registry.

npm registry?

-It's a very big public repository where smart people put code and dependencies as modules.

Like a CDN?

-Not really. It's more like a centralised database where anyone can publish and download libraries, so you can use them locally for development and then upload them to a CDN if you want to.

Oh, like Bower!

-Yes, but it's 2016 now, no one uses Bower anymore.

Oh, I see… so I need to download the libraries from npm then?

-Yes. So for instance, if you want to use React , you download the React module and import it in your code. You can do that for almost every popular JavaScript library.

Oh, like Angular!

-Angular is so 2015. But yes. Angular would be there, alongside VueJS or RxJS and other cool 2016 libraries. Want to learn about those?

Let's stick with React, I'm already learning too many things now. So, if I need to use React I fetch it from this npm and then use this Browserify thing?

-Yes.

That seems overly complicated to just grab a bunch of dependencies and tie them together.

-It is, that's why you use a task manager like Grunt or Gulp or Broccoli to automate running Browserify. Heck, you can even use Mimosa.

Grunt? Gulp? Broccoli? Mimosa? The heck are we talking about now?

-Task managers. But they are not cool anymore. We used them in like, 2015, then we used Makefiles, but now we wrap everything with Webpack.

Makefiles? I thought that was mostly used on C or C++ projects.

-Yeah, but apparently in the web we love making things complicated and then going back to the basics. We do that every year or so, just wait for it, we are going to do assembly in the web in a year or two.

Sigh. You mentioned something called Webpack?

-It's another module manager for the browser while being kind of a task runner as well. It's like a better version of Browserify.

Oh, Ok. Why is it better?

-Well, maybe not better, it's just more opinionated on how your dependencies should be tied. Webpack allows you to use different module managers, and not only CommonJS ones, so for instance native ES6 supported modules.

I'm extremely confused by this whole CommonJS/ES6 thing.

-Everyone is, but you shouldn't care anymore with SystemJS.

Jesus christ, another noun-js. Ok, and what is this SystemJS?

-Well, unlike Browserify and Webpack 1.x, SystemJS is a dynamic module loader that allows you to tie multiple modules in multiple files instead of bundling them in one big file.

Wait, but I thought we wanted to build our libraries in one big file and load that!

-Yes, but because HTTP/2 is coming now multiple HTTP requests are actually better.

Wait, so can't we just add the three original libraries for React??

-Not really. I mean, you could add them as external scripts from a CDN, but you would still need to include Babel then.

Sigh. And that is bad right?

-Yes, you would be including the entire babel-core, and it wouldn't be efficient for production. On production you need to perform a series of pre-tasks to get your project ready that make the ritual to summon Satan look like a boiled eggs recipe. You need to minify assets, uglify them, inline css above the fold, defer scripts, as well as-

I got it, I got it. So if you wouldn't include the libraries directly in a CDN, how would you do it?

-I would transpile it from Typescript using a Webpack + SystemJS + Babel combo.

Typescript? I thought we were coding in JavaScript!

-Typescript IS JavaScript, or better put, a superset of JavaScript, more specifically JavaScript on version ES6. You know, that sixth version we talked about before?

I thought ES2016+ was already a superset of ES6! WHY we need now this thing called Typescript?

-Oh, because it allows us to use JavaScript as a typed language, and reduce run-time errors. It's 2016, you should be adding some types to your JavaScript code.

And Typescript obviously does that.

-Flow as well, although it only checks for typing while Typescript is a superset of JavaScript which needs to be compiled.

Sigh… and Flow is?

-It's a static type checker made by some guys at Facebook. They coded it in OCaml, because functional programming is awesome.

OCaml? Functional programming?

-It's what the cool kids use nowadays man, you know, 2016? Functional programming? High order functions? Currying? Pure functions?

I have no idea what you just said.

-No one does at the beginning. Look, you just need to know that functional programming is better than OOP and that's what we should be using in 2016.

Wait, I learned OOP in college, I thought that was good?

-So was Java before being bought by Oracle. I mean, OOP was good back in the days, and it still has its uses today, but now everyone is realising modifying states is equivalent to kicking babies, so now everyone is moving to immutable objects and functional programming. Haskell guys had been calling it for years, -and don't get me started with the Elm guys- but luckily in the web now we have libraries like Ramda that allow us to use functional programming in plain JavaScript.

Are you just dropping names for the sake of it? What the hell is Ramnda?

-No. Ramda. Like Lambda. You know, that David Chambers' library?

David who?

-David Chambers. Cool guy. Plays a mean Coup game. One of the contributors for Ramda. You should also check Erik Meijer if you are serious about learning functional programming.

And Erik Meijer is…?

-Functional programming guy as well. Awesome guy. He has a bunch of presentations where he trashes Agile while using this weird coloured shirt. You should also check some of the stuff from Tj, Jash Kenas, Sindre Sorhus, Paul Irish, Addy Osmani-

Ok. I'm going to stop you there. All that is good and fine, but I think all that is just so complicated and unnecessary for just fetching data and displaying it. I'm pretty sure I don't need to know these people or learn all those things to create a table with dynamic data. Let's get back to React. How can I fetch the data from the server with React?

-Well, you actually don't fetch the data with React, you just display the data with React.

Oh, damn me. So what do you use to fetch the data?

-You use Fetch to fetch the data from the server.

I'm sorry? You use Fetch to fetch the data? Whoever is naming those things needs a thesaurus.

-I know right? Fetch it's the name of the native implementation for performing XMLHttpRequests against a server.

Oh, so AJAX.

-AJAX is just the use of XMLHttpRequests. But sure. Fetch allows you to do AJAX based in promises, which then you can resolve to avoid the callback hell.

Callback hell?

-Yeah. Every time you perform an asynchronous request against the server, you need to wait for its response, which then makes you to add a function within a function, which is called the callback pyramid from hell.

Oh, Ok. And this promise thing solves it?

-Indeed. By manipulating your callbacks through promises, you can write easier to understand code, mock and test them, as well as perform simultaneous requests at once and wait until all of them are loaded.

And that can be done with Fetch?

-Yes, but only if your user uses an evergreen browser, otherwise you need to include a Fetch polyfill or use Request, Bluebird or Axios.

How many libraries do I need to know for god's sake? How many are of them?

-It's JavaScript. There has to be thousands of libraries that all do the same thing. We know libraries, in fact, we have the best libraries. Our libraries are huuuge, and sometimes we include pictures of Guy Fieri in them.

Did you just say Guy Fieri? Let's get this over with. What these Bluebird, Request, Axios libraries do?

-They are libraries to perform XMLHttpRequests that return promises.

Didn't jQuery's AJAX method start to return promises as well?

-We don't use the "J" word in 2016 anymore. Just use Fetch, and polyfill it when it's not in a browser or use Bluebird, Request or Axios instead. Then manage the promise with await within an async function and boom, you have proper control flow.

It's the third time you mention await but I have no idea what it is.

-Await allows you to block an asynchronous call, allowing you to have better control on when the data is being fetch and overall increasing code readability. It's awesome, you just need to make sure you add the stage-3 preset in Babel, or use syntax-async-functions and transform-async-to-generator plugin.

This is insane.

-No, insane is the fact you need to precompile Typescript code and then transpile it with Babel to use await.

Wat? It's not included in Typescript?

-It does in the next version, but as of version 1.7 it only targets ES6, so if you want to use await in the browser, first you need to compile your Typescript code targeting ES6 and then Babel that shit up to target ES5.

At this point I don't know what to say.

-Look, it's easy. Code everything in Typescript. All modules that use Fetch compile them to target ES6, transpile them with Babel on a stage-3 preset, and load them with SystemJS. If you don't have Fetch, polyfill it, or use Bluebird, Request or Axios, and handle all your promises with await.

We have very different definitions of easy. So, with that ritual I finally fetched the data and now I can display it with React right?

-Is your application going to handle any state changes?

Err, I don't think so. I just need to display the data.

-Oh, thank god. Otherwise I would had to explain you Flux, and implementations like Flummox, Alt, Fluxible. Although to be honest you

should be using Redux.

I'm going to just fly over those names. Again, I just need to display data.

-Oh, if you are just displaying the data you didn't need React to begin with. You would had been fine with a templating engine.

Are you kidding me? Do you think this is funny? Is that how you treat your loved ones?

-I was just explaining what you could use.

Stop. Just stop.

-I mean, even if it's just using templating engine, I would still use a Typescript + SystemJS + Babel combo if I were you.

I need to display data on a page, not perform Sub Zero's original MK fatality. Just tell me what templating engine to use and I'll take it from there.

-There's a lot, which one you are familiar with?

Ugh, can't remember the name. It was a long time ago.

-jTemplates? jQote? PURE?

Err, doesn't ring a bell. Another one?

-Transparency? JSRender? MarkupJS? KnockoutJS? That one had two-way binding.

Another one?

-PlatesJS? jQuery-tmpl? Handlebars? Some people still use it.

Maybe. Are there similar to that last one?

-Mustache, underscore? I think now even lodash has one to be honest, but those are kind of 2014.

Err.. maybe it was newer.

-Jade? DustJS?

No.

-DotJS? EJS?

No.

-Nunjucks? ECT?

No.

-Mah, no one likes Coffeescript syntax anyway. Jade?

No, you already said Jade.

-I meant Pug. I meant Jade. I mean, Jade is now Pug.

Sigh. No. Can't remember. Which one would you use?

-Probably just ES6 native template strings.

Let me guess. And that requires ES6.

-Correct.

Which, depending on what browser I'm using needs Babel.

-Correct.

Which, if I want to include without adding the entire core library, I need to load it as a module from npm.

-Correct.

Which, requires Browserify, or Wepback, or most likely that other thing called SystemJS.

-Correct.

Which, unless it's Webpack, ideally should be managed by a task runner.

-Correct.

But, since I should be using functional programming and typed languages I first need to pre-compile Typescript or add this Flow thingy.

-Correct.

And then send that to Babel if I want to use await.

-Correct.

So I can then use Fetch, promises, and control flow and all that magic.

-Just don't forget to polyfill Fetch if it's not supported, Safari still can't handle it.

You know what. I think we are done here. Actually, I think I'm done. I'm done with the web, I'm done with JavaScript altogether.

-That's fine, in a few years we all are going to be coding in Elm or WebAssembly.

I'm just going to move back to the backend. I just can't handle these many changes and versions and editions and compilers and transpilers. The JavaScript community is insane if it thinks anyone can keep up with this.

-I hear you. You should try the Python community then.

Why?

-Ever heard of Python 3?

Update: Thanks for pointing typos and mistakes, I'll update the article as noted. Discussion in HackerNews and Reddit.