I am still in love with JavaScript, even more

As you may not know

As a senior back-end software engineer at LinkedIn, one stereotype people may think of me about is that I am an outsider of front-end world. Well, it's not this way. Actually JavaScript is one of the programming languages I learned earliest. Also, my github has been occupied by JavaScript - you can see my first ever pull request typicode/lowdb: add the Promise option to lowdb - Yes! It's about JavaScript, too. That's a wild old time when Promise was not widely supported and adopted.

My early years with JavaScript

I have fallen in love with JavaScript since late of my freshman year at college, when Java was the main programming language used at school courses. It all started with my curiosity about building something visible fast - no compilation, instant change, and numerous packages to use and learn from. Java, by contrast, was clumsy to me at that time when I tried to build some tiny interesting projects. In addition, though we were taught Java AWT and JSP, it didn't look like an beautiful tool to build interactive UI. That's how I started exploring other options. I surely also spent a lot of time with Python, but JavaScript, combining vanilla JS in browser, and Node.Js in back-end, grasped most of my interest and it also suited my use cases very well. That's also how I touched the concept of full-stack.

I then had been intensively using JavaScript for almost everything I worked on.

- Used AngularJS to demonstrate Dijkstra Algorithm and find shortest paths
- Used Vue to build an interactive teaching tool that combines mind map, presentations, student-teacher interactions
- Used React for personal projects - I love its concept of immutablibility, it's also let me know there's Functional Programming out in the world.
- Used Node.js for all server-side work (I used Node.js with Docker too), and immersed myself in async (Thanks to it, it's easy for me to understand Promise in Java at work)
- and so on.

Take a break, and explore distributed systems

Looking backwards, I think it's **Node.js** that bridged me back to the back-end world. I have been fascinated by how its event loop unlocked the potential of I/O, but afterwards, in my time at Carnegie Mellon University, I found an issue - it's easy to be fast enough with no specific configuration, but it's hard to tune it in a short time to be the fastest.

We had a course called Cloud Computing where we need to build Twitter-like distributed system , and its grade is based on the rank of server performance (especially READ QPS) across all the teams in the class. So it means super fierce competition. Every team has been crazy about every possible bottleneck - database schema, partition, indexing, cache, server structure, etc.

I started with Node.JS, and soon I bumped into some issues:

1. We were tested and graded by the performance of both MySQL and Apache HBase, and at that time, even though Node.JS already had a decent client for MySQL, its support for Apache HBase was not optimized yet.
2. It's very simple to get a great performance with one thread of Node.js, but it's not enough in order to get top scores. I had to make full use of all CPU cores, but Node.js multi-processing made it difficult for state sharing

sharing.

3. TypeScript was not the first popular choice at that time, but we had a team of 3 people and it also involved remote collaboration, and it means that without strong types, it's too error-prune, and hard to fix them during extremely intensive and short development period.

To be honest, all of these issues have solid solutions nowadays, but back to the school time when we were competing in the final 2 weeks, we just didn't have enough time to dive into each of them and came up with a perfect answer.

I then turned to Java, to be more specific, Vert.x, for its extensive database support, great performance, and strong types. It solved all our issues then and I got A for this course.

After graduation, I joined LinkedIn, and I have been dealing with all kinds of back-end and distributed-systems ventures since then.

Resume - serverless and more

Recently with the surge of Generative AI, I am attracted to start building something around it. For instance, I'm interested in building a platform to analyze podcast audio content, generate all transcripts, summary, break-down, and enable users to jump to specific time point by clicking on the corresponding part in the transcript. This question quickly came to my mind: I need to build something fast, visible, and easy, for both front-end and back-end..., and that's JavaScript.

It's also noted that within all these years, Next.js and NestJS has gained so much popularity that they're among the first choices when people think about building websites and services. On top of that, Vercel, Cloudflare Worker, fly.io are so convenient, powerful, and cheap (equally important!) for building server-less applications. JavaScript has been the first-class member in server-less.

"Any application that can be written in JavaScript, will eventually be written in JavaScript." — Jeff Atwood

There're so many great programming languages out there (Go, Rust, etc.), but JavaScript has always been so vibrant. I love it.

This post was inspired by Simlon Willison (co-creator of the Django Web Framework)'s What to blog about