How Condé Nast Used Node.js and NodeSource to Unify Brands, Create Customer Faith

NODESOURCE®

Contents

3 A Two-Headed Start

4 Inside the Node.js Shop

5 Along Came NodeSource

7 Advantages Gained

8 Lessons Learneds

Condé Nast was in search of a means to unify its iconic brands on a single platform and create heightened customer faith through new consumer experiences. Here's how they did it.

Condé Nast is an iconic umbrella company for twenty of the world's most loved media brands, including Wired, Vogue, and The New Yorker. While each brand is renowned for its stellar content and excellent customer service, that reputation is hard-earned every single day. In 2012, the challenge was at its zenith as different brands were using different systems, tech stacks, and programming languages, a situation that strained the IT teams, both culturally and technically.

"We wanted to unify that onto a single common platform," explained Ed Cudahy, CTO at Condé Nast. "At the end of the day, a lot of what we're doing is creating consumer faith in experiences. Even though it's in its early stages, Node.js fit well with what we were aiming to do."

Back then, like many organizations, Condé Nast operated on the legacy model: frontend and backend engineers, languages and stacks. Java was used on the backend and JavaScript, HTML, CSS on the frontend. The mix led to both a cultural disconnect between different groups and between members on any given team. Even with a typical team comprised of only five people, reconciling differences bordered on impossible.

A common language for frontend and backend, across and within teams, "was a powerful motivator for why we wanted to give Node a shot," said Cudahy.

A Two-Headed Start

Ultimately, the decision to use Node.js was based on external factors as much as internal needs. Like many iconic companies, Condé Nast, a 100-year-old company, was locked into yesteryear, with legacy applications and the mindsets and work routines that go with them. This situation was counterproductive, both to innovation and to attracting new talent. Node.js was hot in the developer community. Cudahy thought it could easily move the company off its path to stagnation.

“Cudahy wanted to use a new technology to inject some energy,” explained Paul Fryzel, principal engineer at Condé Nast.

“We could've picked Ruby or PHP or Python or Java and those would've been perfectly good choices, quite frankly. But they wouldn't have really given much hype or much novelty to our company,” Fryzel added. “And choosing languages that were a little bit more powerful, yet a bit esoteric or academic in nature, like OCaml or Haskell, you would've gotten the hype but maybe not have been able to recruit vigorously for it.”

The start of the Node journey at Condé Nast was mounted on dual goals: unifying the platform for all brands to improve internal efficiencies and customer experiences; and attracting top, innovative talent to the company to ensure it remained competitive in a fast-changing marketscape.

> “*We chose Node.js pragmatically, because it's fast enough and powerful enough for what we need. Secondarily but arguably just as importantly, it was going to be kind of our hiring PR magnet.*”
> Paul Fryzel, Principal Engineer, *Condé Nast*

Forbes ranks Node.js as the #2 job skill in terms of demand. In early 2017, the magazine reported an increase of 2,493% in jobs requiring this skill. But Node.js is no passing trend, given its popularity in current and emerging technologies such as containers, analytics at the edge and machine learning, making it a good choice for the long-haul at Condé Nast.

Inside the Node.js Shop

Technically speaking, Condé Nast is now a “universal, or isomorphic, JavaScript on the server, JavaScript on the client, server-side rendered React” shop, according to James Socol, Director of Engineering.

“Everything we do, from our test running, to APIs or proxies that we have to build to the actual client-side JavaScript, which is where we spend a lot of our time, all runs in Node.js,” he said.

Team members find it easier to move from frontend to backend and back again in their work because it all runs on the same language. This significantly lowers the barrier of entry for engineers in transitioning to work on server-side projects.

While Node.js has attracted talent to the company, the need to train engineers internally continues, not only in the use of Node.js, but in ways to problem solve outside of Node.js experiences. For example, the concurrency model is not always the best choice for various types of load and application use profiles. Finding workarounds is easier if the engineer has experience with other approaches.

“Because we have people who maybe have not worked heavily outside of JavaScript, or haven't worked in languages that approach that in different ways, they don't necessarily know when something is JavaScript or when it's Node, or other options that could be valuable for various types of workload,” Socol said.

Overall, he said, the problems encountered are more tied to JavaScript than to Node.js. Engineers and developers with experience in more than one programming language tend to resolve issues and leverage Node.js strengths better.

Along Came NodeSource

When the process of migrating all of the Condé Nast portfolio brands to Node.js began in 2012, teams faced some pain points common to early adopters of new technology. Although the team had access to APM tools that could capture data from legacy applications, at the time there wasn’t a solution on the market that could provide the same visibility into apps built on Node.

"If something went wrong, you run into some kind of major issue — performance issue or memory issue or whatever — it would take a lot longer to solve it," said Cudahy. "That was just the reality."

This meant that identifying and resolving memory leaks, performance bottlenecks, and other issues as they popped up could be a time-consuming task for the team.

By the time NodeSource was founded in late 2014, Paul Fryzel was already familiar with team members through their contributions to the Node.js project. NodeSource's deep ties to the open source Node.js project and technical expertise gave Fryzel confidence to try N|Solid at an early stage in the company's history. For Condé Nast, the leap of faith is still paying dividends.

"If you start cold with Node.js thinking 'I'll just point my APM tool at it,' you'll probably find that you don't get the same quality or granularity of information that you were getting previously," said Cudahy. "You really need to think through how you're going to operationalize as well, especially if you're transitioning from a different, mature infrastructure."

The introduction of N|Solid provided granular process-level visibility, addressing a key pain point for the Condé Nast team. N|Solid provided insight that ultimately helped reduce resolution times, because engineers could quickly identify the root cause of issues. Customizable thresholds and notifications help the team remediate emerging issues before they escalate far enough to have a major impact on performance.

> *"I feel very confident running N|Solid in production."*
> Paul Fryzel, Principal Engineer, *Condé Nast*

The New Yorker was among the last few brands to complete the migration to Node.js, but was the first brand to adopt containerized infrastructure. On launch day, the team encountered a troubling snag: a few critical errors in the application powering The New Yorker caused it to repeatedly shut down and restart, taking services offline in the process.

"Thanks to NodeSource N|Solid we were able to watch the processes in real time and it was a very helpful tool for debugging," said Hudson. "We were able to keep track of what was going on and monitor the health of the system as we were fixing these bugs."

> *"There's nothing else that's really at the maturity of N|Solid in the industry right now. I can't imagine a company like Condé Nast ever running Node.js in production without it."*
> Paul Fryzel, Principal Engineer, *Condé Nast*

Condé Nast completed its migration to Node.js in mid-2017, and N|Solid remains critical to ongoing monitoring, operations, and security for their production Node.js applications, says Fryzel. "There's nothing else that's really at the maturity of N|Solid in the industry right now. I can't imagine a company like Condé Nast ever running Node.js in production without it."

Advantages Gained

While security is a top concern for Condé Nast, as it is to all companies these days, the steady push of security announcements, and keen attention to emerging threats from the Node.js community are much appreciated. This attention to security drives more advantages when using Node.js.

"Thanks to Node's continued evolution, we've been able to adopt promises, maps, generators, and other features without having to depend on modules that polyfill these features," says Matthew Hudson, who is also a Director of Engineering and focuses his work on five brands: The New Yorker, Vanity Fair, Condé Nast Traveler, Bon Appétit, and Epicurious.

The flexibility within a unified platform that Cudahy and team initially envisioned has materialized within Condé Nast.

The company has also taken advantage of Node.js as a technology choice by building an engineering team with strong JavaScript skills; members of the engineering team can now work on projects throughout the stack.

“This dramatically lowers the effort, or ability to contribute, to full-stack development,” says Hudson. “Engineers may not be server or client experts, but those with stronger understanding of one or the other are still able to look at both server and client and quickly grasp what’s going on.”

Lessons Learned

Fryzel says working extensively with Node.js has revealed its sweet spot: it’s a bridge between the frontend traditional developer and things they need to do to get their job done. Beyond that, it really shines in the web services/web applications sphere.

“In my opinion, Node.js’ sweet spot is really anything that you need to do that's basically close to the web — and ‘the web’ being defined as things that fit into common web components,” he said. “Things that are going to be serving out websites, mobile applications, progressive web applications, etc. Things that are going to be doing HTML, CSS, JavaScript, frontend rendering, and so on. APIs that need to transmit data to those, or any sort of data proxy tier for your web appliances.”

As to lessons learned specifically to using and managing Node.js, Hudson says the following are top of mind at Condé Nast:

- It is very important to lock dependency versions. Otherwise, your build artifacts will lack consistency. What you developed locally and tested in your QA environment may not be what you deployed to production at 1:01PM.
- Familiarize yourself with Node.js core contributors and popular module developers. Their thoughts and code can help you understand how the ecosystem evolves, and provide directional feedback on your own approach to challenges.
- Git URLs in package.json can be a helpful shortcut for testing / deploying / qa-ing experimental / risky branches of work.
- Use some sort of a version manager. This is necessary when switching among multiple modules / applications that are using different versions of Node.js .

Having learned all this and perfected their Node.js game through launch and beyond, what's next for Condé Nast?

Given the usefulness of Node.js in myriad applications from containers to edge computing apps, there are virtually no limits on what the company can innovate.

"Now that all the brands are on a single platform, it opens the doors to a lot of different things," says Cudahy. "We've got the data APIs, we've got the content APIs, we've got the consumer APIs. All of these things are now consolidated. That allows us to create new interactive product experiences and to use that connectedness between our brands in ways that were impossible before."

ABOUT THE NODE.JS FOUNDATION

The Node.js Foundation's mission is to enable widespread adoption and help accelerate development of Node.js and other related modules through an open governance model that encourages participation, technical contribution, and a framework for long term stewardship by an ecosystem invested in Node.js' success.

Learn more at nodejs.org

ABOUT NODESOURCE

NodeSource helps organizations run mission-critical Node.js applications with products and services that support success in the modern digital era.
The NodeSource platform empowers teams to deliver peak performance by offering greater visibility into resource usage and enhanced awareness around application behavior and security.

Learn more at nodesource.com