The Dawn Of Enterprise JavaScript

JavaScript Finds Its Place In Enterprise Software Development

by Michael Facemire
with Christopher Mines, Jeffery S. Hammond, and Nathaniel Fleming

WHY READ THIS REPORT

Adoption of JavaScript — and the Node.js runtime environment in particular — sets the stage for the biggest shift in enterprise application development in more than a decade. When building a technology platform to power their customer-facing web presence, enterprise development shops have generally chosen between two options, Java or .NET. The demands of mobile, particularly speed and scale of delivery and deployment, are fracturing this duopoly. New examples appear regularly of the retail site that was overloaded on Black Friday or the streaming app that stuttered when it went viral. This report examines how JavaScript addresses the scalability challenge, how it's changing both enterprise architectures and programming models, and what application development and delivery (AD&D) leaders should focus on during this transformative time.

BACK-END JAVASCRIPT? WHAT'S WRONG WITH JAVA AND .NET?

JavaScript — a simple development language — will change the way business technology is created; a change on par with Java's impact in the late 1990s. JavaScript isn't new, it's been an integral part of browser code and the modern web for quite some time. But JavaScript in enterprise back ends? Like tattoos on back ends, no self-respecting organization wants any part of that . . . right? Wrong. Without it, your digital business future is bleak.

One of the first questions we hear when discussing back-end JavaScript is simple: Why? Why move away now from languages companies have used to build systems of record for years? The reasons span technology and organizational performance:

- **Today's time-to-market demands are nearly real-time.** Meeting customer demand means delivering more software-based experiences quickly, regardless of which digital channel they use. This forces development shops to respond by changing how they build and update software.

- **Mobile traffic volume will bury existing systems of record.** Scale is changing how that same software is deployed, while also changing parts of enterprise architecture that are commonly thought to be a commodity. Wal-Mart didn't have a single second of downtime on 2014's Black Friday. What's unique about their model? Every transaction was delivered through a Node.js infrastructure!

- **Java and .NET container startup and shutdown are onerous.** Instead of spending time waiting for traditional containers to initialize, developers on Node platforms are using that time to innovate new business solutions. Technologies such as Docker are language- and platform-agnostic replacements for the platform-specific containers of the past.[1]

- **A four-tier engagement architecture requires loosely-coupled services.** Many service deployments on Java and .NET stacks are large, monolithic implementations.[2] Back-end JavaScript serves to fragment these monoliths dynamically into small, composable microservices that are readily consumed by mobile devices and their users.[3]

- **JavaScript developers align personal and business success.** The majority of developers are concerned with their own success — your (often altruistic) hope is that this personal success aligns with the business. Node.js developers, on the other hand, actively want success for both themselves and the organization! Adopting cutting-edge modern technology is a risk for developers — they're willing to take that risk to expedite solutions that drive business success.

Non-Blocking I/O, Or How Santa Scales His Delivery Business

Assume for a moment that Santa is real — anyone with kids at home is adept at maintaining this illusion. His challenge is delivering the right toy to every child on the planet in a single day. Consider two approaches to meeting this challenge:

- **Serial request and fulfillment.** If Santa were to fly around and meet each kid, ask what they wanted, wait for the response, and then go back to the North Pole to make them, he'd never be finished by Christmas morning. Traditional web and application servers operate this way; maintaining connections while answering incoming requests. As the request comes in, a socket is opened and is not closed until the application generates a result and that result is sent back to the requesting client. These persistent server connections require CPU and memory to maintain; if these resources run short, every follow-on request has to wait, leading to a painful delay for your customers.

- **Non-blocking I/O.** If instead Santa were able to craft every possible toy and simply throw them over a wall, where each child's parents would catch the one that's meant for their children, the delivery process would be much more efficient. Web servers that utilize non-blocking I/O change how client requests are serviced at the back end in this same manner. Instead of dedicating a system thread per resource request, non-blocking I/O solutions have a single thread to service all requests. That thread does the minimum work possible, throws it over the wall to an event handler, which passes it off to the appropriate service(s), and shuts down the connection. This breaks the bind between incoming requests and physical hardware, resulting in much more headroom in server-side scale. This is the heart of asynchronous, event-driven, digital architectures.

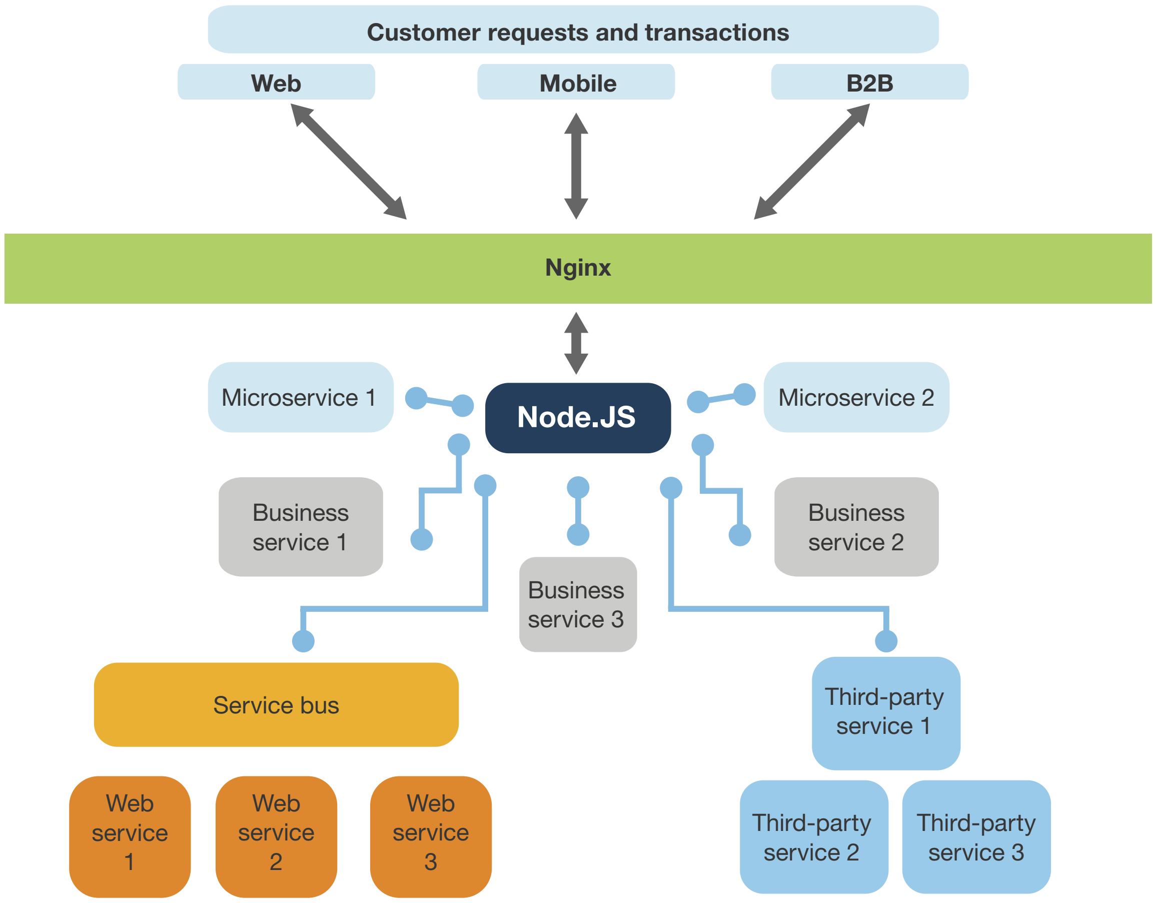
Nginx And Node.js, Santa's Elves For Digital Delivery

Just as Santa has elves that help him with his worldwide gift delivery operation, the digital world has a new pair of helper elves for data delivery (Nginx) and data composition (Node.js). Increasing customer volume requires a new asynchronous, event-driven architecture; therefore it's not surprising that these new tools commonly appear together. Data produced by Node.js is often served by an Nginx web server. Nginx also often acts as a load-balancing reverse proxy or SSL termination endpoint (see Figure 1). These two technologies are increasingly handling the back end of mobile and web transactions as:

- **Web servers switch to Nginx.** The primary function of a web server is to route requests to the proper application and then deliver the resultant response. For nearly 20 years the web server choice was primarily between apache and Microsoft IIS, but in July 2013, Nginx became the most popular among the top 1,000 websites in the world.[4] These websites must scale elastically to handle wildly varying magnitudes of traffic; Nginx is the choice to power such high-traffic sites because of its asynchronous event-driven nature, providing hyperscale without requiring racks of often-idle hardware.

- **Application servers evolve toward composition.** The challenge for app servers is to compose a response based on data from one or many back-end data sources and deliver that response to the web server for final delivery.[5] Yet response time from the back-end data sources is unbounded; maintaining a connection to these sources tie up hardware resources as outlined above. Node.js maximizes the available computing power by asynchronously requesting the individual service responses and responding to the web server with a federated response.

Figure 1 Node.js And Nginx Comprise The Modern Enterprise Aggregation Tier



120686 Source: Forrester Research, Inc. Unauthorized reproduction or distribution prohibited.

WHERE DOES JAVASCRIPT FIT?

Java and .NET are alive and well; for most companies, JavaScript platforms are not replacing these as the foundations of enterprise architectures. Instead, view JavaScript as a must-have tool in the software development and delivery toolkit of modern companies. Already invested in a service-oriented architecture implementation or enterprise service bus? Deployed proprietary line-of-business systems? Making use of third-party software-as-a-service (SaaS) solutions? Node.js does an outstanding job of weaving such existing services together while providing a platform for incorporating new services at any time.

Companies with long-running processes and legacy workflows will find they still operate best on legacy Java and .NET stacks. But for those with customer-facing, web-scale systems of engagement, the parallel and lightweight nature of the JS stack will prove superior to traditional stacks.

Three Technology Entry Points For Enterprise JavaScript

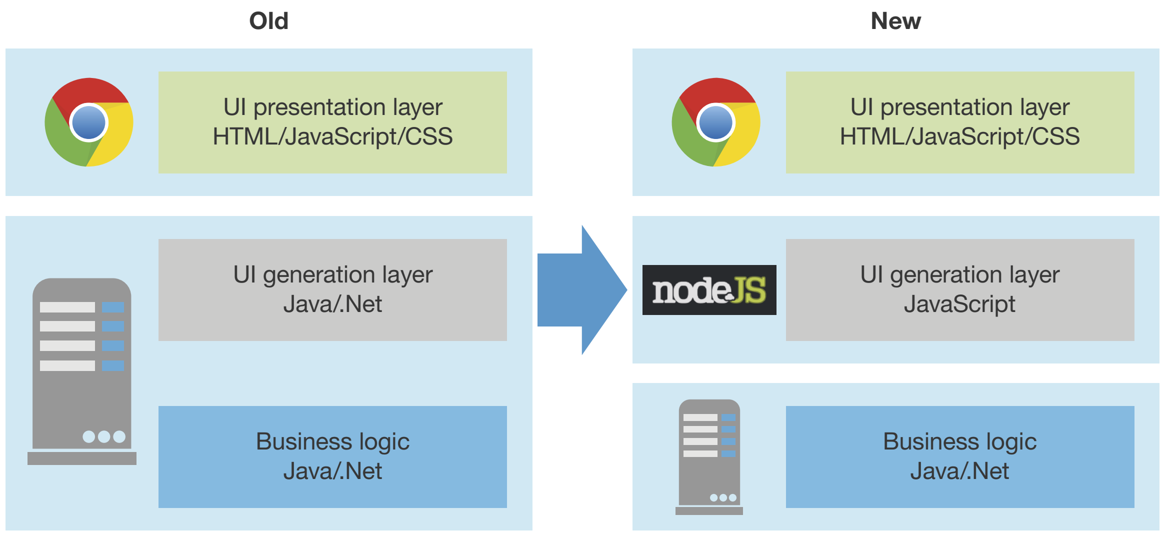
While JavaScript and associated platforms aren't obviating the traditional stacks, it is augmenting in three important areas:

- **API hosting.** Application programming interfaces (APIs) are the core of tomorrow's digital business model. As such, they must always perform and scale, making a JavaScript platform the ideal hosting option. Many companies are already using this model today for new RESTful APIs that build on existing SOA infrastructure.[6] Mobile devices often don't need the full set of SOAP data that is generated from web services interfaces. Instead of replicating each of these APIs for a mobile client, JavaScript event-handlers can intercept incoming mobile requests, make an underlying web service request, pull out the necessary data, create a lightweight JSON object, and pass that object back to the mobile client.

- **Full-stack JavaScript.** Using a common language throughout the technology stack drives development efficiency and thus expedites time-to-market. As JavaScript becomes the de facto front-end language for the Web, using solutions such as Node.js and React (from Facebook) allow a single developer to impact both ends of the stack.[7] The attraction here is simple: front-end developers (web, mobile, and otherwise) are spending more time in JavaScript and becoming proficient. If large back-end changes are needed, they can work with the server-side team to implement them, but for the small changes that appear daily, asking front-end developers to understand Java or .NET doesn't make sense (see Figure 2).[8]

- **Real-time web applications.** Streaming web solutions make use of Node's ability to handle large volumes of requests at scale. Streaming data over web sockets, MQTT, WebRTC, and the like can be done using a standard application server, but legacy app servers tend to introduce a fair amount of per-request or per-session overhead that limits the scalability of these solutions. Node was built for these types of scenarios and therefore fits in well, especially when using socket.io, a popular Node.js module.[9]

Figure 2 Monolithic Back-End Systems Are Being Decomposed


Old
New
UI presentation layer HTML/JavaScript/CSS
UI generation layer Java/.Net
Business logic Java/.Net
UI presentation layer HTML/JavaScript/CSS
nodeJS
UI generation layer JavaScript
Business logic Java/.Net

120686 Source: Forrester Research, Inc. Unauthorized reproduction or distribution prohibited.

The Business And Organizational Benefits Of Node.js Are Equally Compelling

One of our clients recently complained that, "it takes two days to get a simple copy change into our traditional SpringMVC application." Simply updating some text on a website took two days — in a time when innovation is at a premium, spending more than a few minutes updating commodity copy cannot slow this down! How can modern JavaScript address this innovation challenge?

> "Innovation is no longer just a differentiator. It's now a means of survival." (Joe McCann, founder, NodeSource)

Let's look at the modern corporate battle cry of having individual departments "act like a startup." What does that require? Quite often it's "get a functional application deployed as quickly and for as little investment as possible and see if it's a viable offering." Startups are doing this by building with JavaScript; non-startups should be leveraging this approach as well for the following reasons:

- **Node developers aspire to greatness.** In a recent Forrester survey, 48% of node developers stated that "wanting to manage one or more development teams and having responsibility for overall project success or failure" describes them completely, while only 19% of developers who don't use node hold the same sentiment (see Figure 3). These are the developers that will lead your company into the next generation of digital experiences; your organization has a better shot at excellence if the developers have an equally aligned inner desire for excellence.[10]

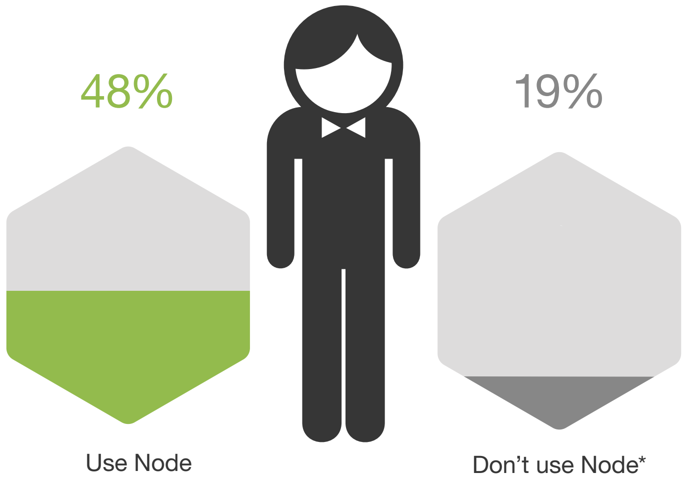
- **Innovation flourishes when deployment delays disappear.** Bringing up and tearing down Node.js environments happen in microseconds, not minutes. Unfortunately, Java and .NET developers spend days doing the same thing. The traditional stacks have led many shops to create large monolithic binaries that are very difficult to start, stop, and modify, both during development and in production. Remove this 1 to 2 minute delay (which often happens more than 10 times per day) and developers have much more time to innovate.

- **New features are just an "npm install <xyz>" command line statement away!** Node.js is powerful due to its pluggable nature; the node package manager (npm) allows developers to easily add third-party functionality to applications built on node. The express web application framework, the gulp source code build framework, and the socket.io streaming framework are common examples. This software composition model is a direct result of the development need for speed.

- **System-wide updates can be done once.** The asynchronous, event-driven model also addresses the common challenge of portfolio-wide application updates. Organizations often get new governance or regulatory changes that affect all digital properties; audit requirements are a typical example. Instead of updating every application to add or update audit support, why not simply chain a new "audit" event to each call to existing applications or services? That's exactly what Node.js-driven architectures allow — simply build a new "audit" event handler that gets fired at the same time (asynchronously) as the API call to the business logic service to audit each action.

Figure 3 Node.js Developers Are More Committed To The Overall Success Of The Business

"Please indicate how much 'I want to manage one or more development teams and have responsibility for overall project success or failure' describes your attitudes"
(Percent selecting "describes me completely")



Base: 42 global developers who do use Node
*872 global developers who don't use Node

Source: Forrester's Business Technographics® Global Developer Survey, 2014

120686 Source: Forrester Research, Inc. Unauthorized reproduction or distribution prohibited.

New Platforms Come With New Challenges

Enterprise JavaScript drives faster innovation, makes for happier developers, and allows business to scale to the needs of today's users. What's keeping every shop from dropping everything and jumping into this new model? As always, new development platforms and paradigms bring new challenges as well, so be aware of the following:

- **New features are just an "npm install <xyz>" command line statement away!** Yes, this capability creates business value, driving rapid innovation as outlined above. The challenge is that, if unchecked, developers will include third-party software packages to solve small problems, introducing potential software bloat, unknown security vulnerabilities, and open source violations with a simple command line statement. Fortunately, companies can maintain their own private npm repository, ensuring that only vetted packages are used by development. Maintaining these repositories isn't trivial, so vendors such as npm, Salesforce Heroku, and NodeSource have hosted private npm solutions.

- **Node.js is maturing as fast as those that build on it.** Four of the five primary contributors to Node.js recently forked the codebase, under the new name of Io.js. This type of change in low-level platform software is unheard of in the enterprise, but it's a precursor to the new norm in the development world.[11] Fortunately, Joyent established a foundation for Node, which will (hopefully) provide a formal governance model and ensures a stable growth trajectory. Joining Joyent in this foundation are IBM, PayPal, Microsoft, Fidelity, and The Linux Foundation.[12]

RECOMMENDATIONS

ASYNCHRONOUS EVENT-DRIVEN ARCHITECTURES ARE THE NEW NORMAL

The growth of mobile and connected customers make digital performance at scale a prerequisite for business success. For companies that have been building enterprise applications for more than 12 to 15 years, the advent of JS and node is similar in scope to the change from C/C++ to Java. At that time, naysayers held that no one would build "real" software on a platform that used automatic garbage collection. Today that same mindset will state that no one would build applications on JavaScript. Both statements are wrong. With that in mind, application development and delivery leaders should prepare for the change by doing the following:

1. **Become familiar with Node.js and where these technologies fit best.** Your organization may not be using server-side JavaScript today, but it will soon. Unfortunately, thinking in JavaScript is completely different than thinking in object-oriented languages — it's much more than a change in semicolons. Ensure that your development organizations have native JavaScript thinkers: those that can solve problems using event handlers with context input, as opposed to those that assume environmental context as was the norm in the OO days.

2. **Investigate Amazon Lambda and similar platforms.** Understand how to create standalone functional entities. This will derive from the first recommendation, as these entities will often be written in JavaScript, but familiarity with writing, packaging, and delivering these bundles will drive changes within each of the development, test, and operations teams.

WHAT IT MEANS

NODE.JS WILL CHANGE CORPORATE TECHNOLOGY JUST LIKE JAVA BEFORE IT

In February 1997, enterprise development teams' toolkit changed significantly when James Gosling and Sun Microsystems released version 1.1 of the Java Development Kit (JDK). Suddenly a single language could be used to write code on a variety of dissimilar systems; Java applets revolutionized web experiences, Java servlets and Java Server Pages changed how web back-ends were created, Java user interfaces appeared on Windows desktop applications, and Java Database Connectivity (JDBC) allowed developers to connect to data from any operating system. Java was everywhere and allowed us to do everything! From a very small start, Java accelerated the adoption of object-oriented programming started by C++ and became one of the two de facto technology stacks driving most businesses today.

JavaScript, particularly Node.js, is currently at that JDK 1.1 inflection point and will make a similar revolutionary change to how companies build software. In addition to the back-end supernova already underway, Node also drives modern web experiences, both on the desktop and mobile (using the node-webkit module). Command-line interfaces for many popular tools and platforms are being built on node — PhoneGap and Gulp are excellent examples of this phenomenon. Within 18 months, we'll see hardware platforms, previously the domain of highly optimized machine-level code, move to a Node.js implementation due to its superior performance characteristics.

SUPPLEMENTAL MATERIAL

Survey Methodology

Forrester's Business Technographics® Global Developer Survey, 2014, was fielded to 1,716 business and technology decision-makers located in Australia, Brazil, Canada, China, France, Germany, India, New Zealand, the UK, and the US from SMB and enterprise companies with two or more employees. This survey is part of Forrester's Business Technographics and was fielded from February 2014 to April 2014. ResearchNow fielded this survey on behalf of Forrester. Survey respondent incentives include points redeemable for gift certificates. We have provided exact sample sizes in this report on a question-by-question basis.

Each calendar year, Forrester's Business Technographics fields business-to-business technology studies in 10 countries spanning North America, Latin America, Europe, and Asia Pacific. For quality control, we carefully screen respondents according to job title and function. Forrester's Business Technographics ensures that the final survey population contains only those with significant involvement in the planning, funding, and purchasing of business and technology products and services. Additionally, we set quotas for company size (number of employees) and industry as a means of controlling the data distribution and establishing alignment with IT spend calculated by Forrester analysts. Business Technographics uses only superior data sources and advanced data-cleaning techniques to ensure the highest data quality.

Companies Interviewed For This Report

IBM	npm
Kinvey	Oracle
Mediafly	OutSystems
Nginx	StrongLoop
NodeSource	Whogloo

ENDNOTES

1 For details on the common uses of Docker, see the "Brief: Why Docker Is All The Rage" Forrester report.

2 For additional detail on this architectural direction, see the "Mobile Needs A Four-Tier Engagement Platform" Forrester report.

3 For additional detail on composing experiences versus writing code, see the "From Application Design To Application Composition" Forrester report.

4 For detail on this ranking, read Web Technology Survey's blog post. Source: Matthias Gelbmann, "Nginx just became the most used web server among the top 1000 websites," W3Techs, July 3, 2013 (http://w3techs.com/blog/entry/nginx_just_became_the_most_used_web_server_among_the_top_1000_websites).

5 For additional detail on composing experiences versus writing code, see the "From Application Design To Application Composition" Forrester report.

6 For more detail on building RESTful APIs on existing SOA infrastructure, see the "How To Design APIs For Mobile" Forrester report.

7 JavaScript isn't limited to just the web for mobile front ends — native apps are now joining the parade! Native iOS apps can now interface with the JavaScriptCore API which provides full access to Objective-C or Swift objects from a JavaScript interface. Android provides similar functionality through the WebView.addJavaScriptInterface API call.

8 There are many additional benefits to full-stack JavaScript, starting with development-time optimizations. Developers start up and shut down their application server or containers within that app server many times a day. This cycle time may be in the 10s of minutes, whereas doing the same with Node.js is most often done under a minute. This delay is compounded when that traditional server has errors on startup.

9 For more detail and best practices around socket.io, see their website. Source: Socket.io (http://socket.io/).

10 For more detail on high performance development organizations, see the "Best Practices: Building High-Performance Application Development Teams" Forrester report.

11 Details of this fork can be found here. Source: Wired (http://www.wired.com/2014/12/io-js/).

12 Details on this are found here. Source: "Joyent Moves to Establish Node.js Foundation," Joyent press release, February 10, 2015 (http://www.joyent.com/about/press/joyent-moves-to-establish-nodejs-foundation).