G00262641

# Improve Mobile App Development and Integration With JavaScript Technologies

**Published:** 30 April 2014

**Analyst(s):** Jason Wong

Enterprise mobile strategists are hungry for ways to make mobile app development and integration easier and faster. JavaScript and its related technologies have emerged as viable tools that can help address these challenges and must be factored into an enterprise's mobile architecture and strategy.

## Key Challenges

- Developing native apps across device platforms is expensive and time-consuming due to lack of code reusability across projects.
- Back-end integration for mobile apps is complex and costly because RESTful APIs are not readily available for most enterprise systems.
- Enterprises are challenged with finding skilled mobile developer talent, especially for development across multiple device platforms.

## Recommendations

- When reusability and portability are high priorities for mobile apps, choose cross-platform development tools with extensible JavaScript libraries over native development.
- Although there are many approaches to build RESTful interfaces, evaluate Node.js to create an agile API layer to optimize mobile integration.
- Build up internal developer JavaScript skill sets to support front-end and back-end mobile app development.

## Strategic Planning Assumption

By 2017, JavaScript will be the most in-demand language skill in mobile application development (AD).

## Introduction

Developing mobile apps using native SDKs, such as Xcode for iOS and Android SDK, is often the desired approach to build mobile apps because of assumed superior user interfaces (UIs). However, with the high costs associated with native development and the increasing need for multiplatform device support, developers must balance UI needs with the need for reusable and extensible development tools. Enter JavaScript and the rise of hybrid mobile app development combining HTML5 and native code.

It was not too long ago that JavaScript was considered a toy language not capable of enterprise-class applications, but that has changed thanks to advances made to support the language by technology stalwarts such as Google, Yahoo, Facebook, and Twitter. Today, JavaScript and JavaScript-based technologies (see Glossary) are not only ubiquitous in client-side Web and mobile app development, but increasingly they are being adopted on the server-side with the enterprise shift toward cloud (see "Predicts 2014: Application Development").

Enterprises have a plethora of technology options for front-end and back-end mobile app development and integration, but there are efficiencies to be gained by adopting a common and uniform set of technologies. This research outlines best practices for mobile strategists to evaluate and deploy JavaScript technologies throughout the entire mobile stack for enterprises that want a modern mobile app architecture to bridge the needs of rapid development and agile integration.

## Analysis

### Choose Cross-Platform Development Tools With Extensible JavaScript Libraries Over Native Development

Native mobile app development is costly in time and resources because specialized developer talent is required and the effort needs to be duplicated across multiple device platforms. A large percentage of mobile apps (ones that don't require intensive graphics, large transaction volumes, or pixel-perfect native UI) can benefit from employing a cross-platform development approach that leverages JavaScript for native-level device integration and rich UI and functionality (see "Containers Have a Major Impact on HTML5 Mobile Application Development Projects").

JavaScript is ubiquitous on the Web because it is a browser-agnostic scripting language, but it has now become a critical language for mobile app development. Keys to this trend are its portability across multiple device platforms (which lends itself to reuse) and its use to bridge the HTML5 mobile app to device-level functions through native plug-ins called via JavaScript (such as calling an infinite scroll list control). This combination makes JavaScript a natural fit for developing hybrid mobile apps to address the ever-growing number of mobile devices.

There are many mobile application development platform (MADP) solutions such as IBM Worklight, Sencha, Kony and Appcelerator that rely heavily on JavaScript and JavaScript-based technologies to build mobile apps with access to native device controls. Additionally, a wide variety of open-source JavaScript frameworks, such as Backbone.js, AngularJS and jQuery Mobile, coupled with the Apache Cordova container, provide a familiar and powerful combination for Web developers to make the leap to mobile apps. Although JavaScript is a strategic technology, different frameworks are better suited for certain projects and there is no single one-size-fits-all, strategic solution among JavaScript frameworks. Enterprises should also exercise caution in building apps with complex JavaScript code without a framework.

Note that hybrid apps using lots of JavaScript also present usability issues. Slow app performance is often cited as the largest concern, and this is especially true when running hybrid apps on older devices and OSs. On the positive side, new native device extensions and functionality can be readily added and exposed via JavaScript libraries by an expanding mobile developer community. In the constantly changing mobile environments, this extensibility is critical. JavaScript, in conjunction with HTLM5 and CSS3, needs be part of any enterprise's client-side mobile development strategy to ensure flexibility and portability across devices.

## Evaluate Node.js to Create an Agile API Layer to Optimize Mobile Integration

Enterprises are realizing that the middleware and back-end data access standards used with desktop Web apps don't typically apply to the needs of mobile apps. Mobile apps consume and interact with data differently, requiring different data formats, payload sizes, and transaction handling. Moreover, the data often needs to be delivered asynchronously to an app that runs on an intermittently connected wireless device.

For mobile integration, enterprises must re-evaluate their service-oriented architecture (SOA) strategies to include more agile, lightweight RESTful services layer that can be developed and changed quickly for specific mobile requirements. In this REST model, JavaScript is starting to play an important role on the server-side and is gaining prominence for integrating disparate systems and creating open APIs — in particular, using Node.js (see "Hype Cycle for Web Computing, 2013").

Created in 2009, Node.js has quickly been adopted by tech-savvy companies such as Walmart, Groupon, PayPal and LinkedIn (see Case Study section) as an optimal solution to connect mobile apps to many different types of data sources, whether behind the corporate firewall or in the cloud. Node.js has an event-driven, nonblocking input/output (I/O) model, which translates into enterprises potentially needing fewer server resources than with other server-side programming languages. By building stateless back-end services designed to maximize throughput using asynchronous events, enterprises can ensure their mobile apps can scale for unpredictable usage across potentially millions of mobile users.

Recognize, however, that Node.js is a new paradigm that is foreign to most Web developers and, operationally, Node.js is still immature in areas such as difficulty in managing and debugging complex code. It also can't just be plugged into existing back-end environments, since various bridges need to be developed to extract data from existing systems — although vendors like StrongLoop, Appcelerator and MuleSoft are working to solve this problem. Data extraction for mobile app consumption requires normalizing a myriad of back-end systems to a common, reusable, and portable format. By using and working with Node.js, enterprises can create an open RESTful API layer using a common language so that frontend developers can also code the back-end integration of the mobile app or work more closely with back-end developers, increasing efficiency and productivity.

JavaScript technologies, in particular Node.js, are now the primary technologies supported by vendors offering emerging mobile back-end-as-a-service (MBaaS) space such as StrongLoop, Appcelerator, Kony, and FeedHenry. It has also been adopted on the majority of platform as a service (PaaS) offerings, including salesforce.com's Heroku, Microsoft Windows Azure, Amazon Web Services (Elastic Beanstalk), and Cloud Foundry. Mobile strategists need to review their enterprise back-end architectures to align with this growing and important server-side movement.

## Build Up Internal Developer JavaScript Skill Sets to Support Front-End and Back-End Mobile App Development

One of the biggest challenges for enterprises engaged in mobility is finding great mobile developer talent. As an alternative or complement to native app developers for iOS, Android and other OSs, enterprises should hire developers with JavaScript experience and expertise, or train existing Web front-end developers on JavaScript. As indications of its rising popularity, the job website Indeed.com reports 150% growth in JavaScript job postings since 2005, and JavaScript was ranked as the most popular language on the GitHub project-hosting site in 2013 (see Table 1). Even though mobile developers will be working in other languages, evidence shows that JavaScript is playing a larger role in most projects across front-end and back-end development.

Table 1. GitHub Historical Count of Project Repositories Showing the Rapid Rise of JavaScript to the Top

| Language | 2009 | 2010 | 2011 | 2012 | 2013 | CAGR 2009-2013 |
|---|---|---|---|---|---|---|
| JavaScript | 6,076 | 25,161 | 76,282 | 434,875 | 475,969 | 197.50% |
| Ruby | 13,013 | 24,631 | 76,921 | 469,734 | 385,347 | 133.28% |
| Java | 7,987 | 21,768 | 103,625 | 393,124 | 302,174 | 148.01% |
| PHP | 3,238 | 13,668 | 78,733 | 258,943 | 230,402 | 190.44% |
| Python | 9,784 | 17,862 | 61,235 | 254,448 | 187,910 | 108.34% |
| C++ | 2,110 | 7,807 | 36,779 | 146,666 | 140,845 | 185.83% |
| C | 5,895 | 15,913 | 45,316 | 271,788 | 127,429 | 115.62% |
| Objective-C | 1,011 | 2,509 | 7,481 | 59,664 | 67,715 | 186.08% |
| C# | 948 | 3,565 | 11,549 | 63,877 | 65,257 | 188.04% |
| Shell | 760 | 14,962 | 12,641 | 93,732 | 57,710 | 195.20% |
| CSS | 205 | 285 | 2,179 | 5,744 | 110,004 | 340.78% |
| Perl | 6,056 | 9,945 | 17,478 | 65,110 | 30,033 | 49.23% |
| CoffeeScript | 64 | 606 | 5,392 | 22,301 | 24,158 | 340.78% |
| VimL | 370 | 1,871 | 2,803 | 23,534 | 13,547 | 145.99% |
| Scala | 2,530 | 1,588 | 7,668 | 19,912 | 15,344 | 56.93% |

Source: Gartner (April 2014)

Enterprises should consider JavaScript as a viable long-term language for developing rich mobile apps. Specifically, IT organizations that adopt hybrid or cross-platform mobile app development strategies must have developers with a mastery of JavaScript in order to succeed (see "IT Market Clock for Programming Languages, 2013"). Although more seasoned JavaScript developers (those with five or more years of experience) with mobile expertise could be harder to find, novice Web developers can be quickly trained since JavaScript has been shown to be a relatively easy language to learn. Several large companies in retail and banking have indicated that their front-end Web developer teams have extended their JavaScript skill set into building hybrid apps and even started developing back-end integration using Node.js.

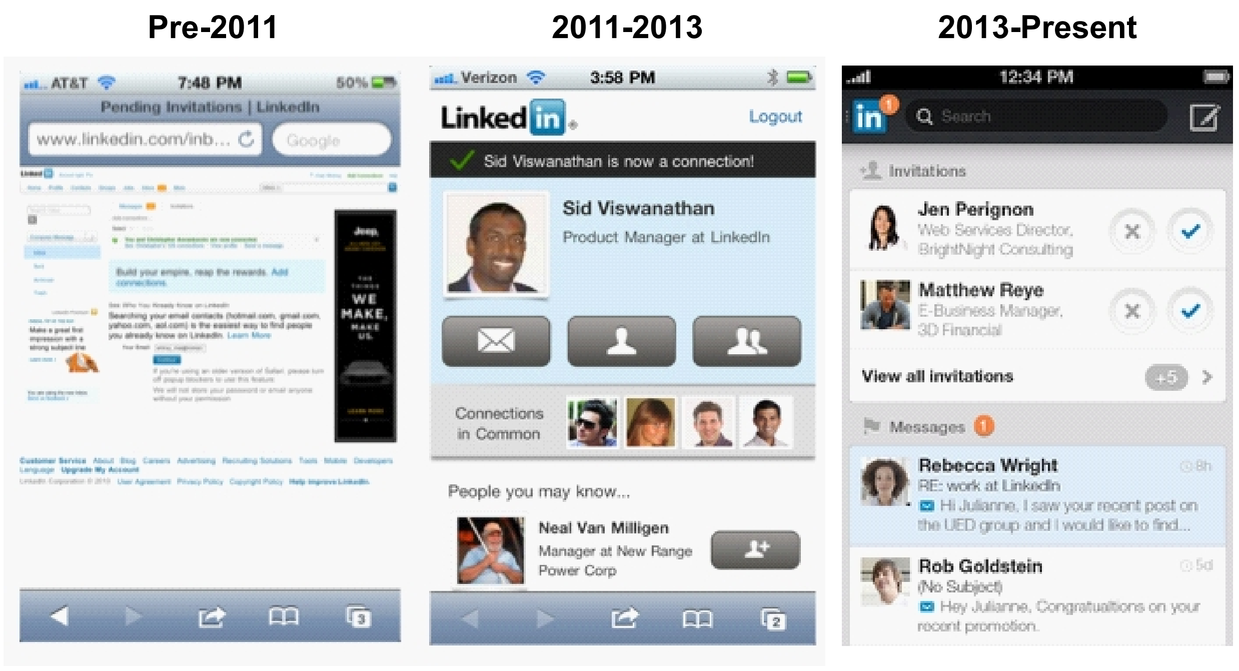
Similar to the Web development world, JavaScript is now an optimal language for mobile app development because of the vibrant developer community that has blossomed over the past few years, particularly around Apache Cordova, jQuery Mobile and Node.js. Developers contribute reusable libraries and framework extensions that further the JavaScript influence across the mobile stack. The proliferation of these JavaScript mobile libraries and frameworks enables developers to write less low-level code and ultimately build and integrate apps more quickly and efficiently.

## Case Study

### LinkedIn Leverages JavaScript Technologies to Build a Better App

First launched in August 2011, LinkedIn's hybrid mobile app was a big leap from the Web app it replaced. According to LinkedIn, the hybrid app achieved two to 10 times faster performance on the client side than its predecessor, and on the server side, it used a fraction of the resources. However, by 2013 LinkedIn's user base nearly doubled and the percentage of users accessing the mobile app had nearly tripled to about 27% of all traffic, and expected to reach 50% by end of 2014. Because of demands on the mobile app, LinkedIn has had to re-evaluate their client-side strategy, but what they had implemented on the server side continues to serve them well (see Figure 1).

Figure 1. LinkedIn's Three Major Mobile App Releases


Pre-2011
2011-2013
2013-Present
AT&T 7:48 PM 50%
Pending Invitations | LinkedIn
www.linkedin.com/inb...
Google
Verizon 3:58 PM
LinkedIn
Logout
Sid Viswanathan is now a connection!
Sid Viswanathan
Product Manager at LinkedIn
Connections in Common
People you may know...
Neal Van Milligen
Manager at New Range Power Corp
12:34 PM
Search
Invitations
Jen Perignon
Web Services Director, BrightNight Consulting
Matthew Reye
E-Business Manager, 3D Financial
View all invitations
+5
Messages
Rebecca Wright
8h
RE: work at LinkedIn
Hi Julianne, I saw your recent post on the UED group and I would like to find...
Rob Goldstein
5d
(No Subject)
Hey Julianne, Congratulations on your recent promotion.

Source: Gartner (April 2014)

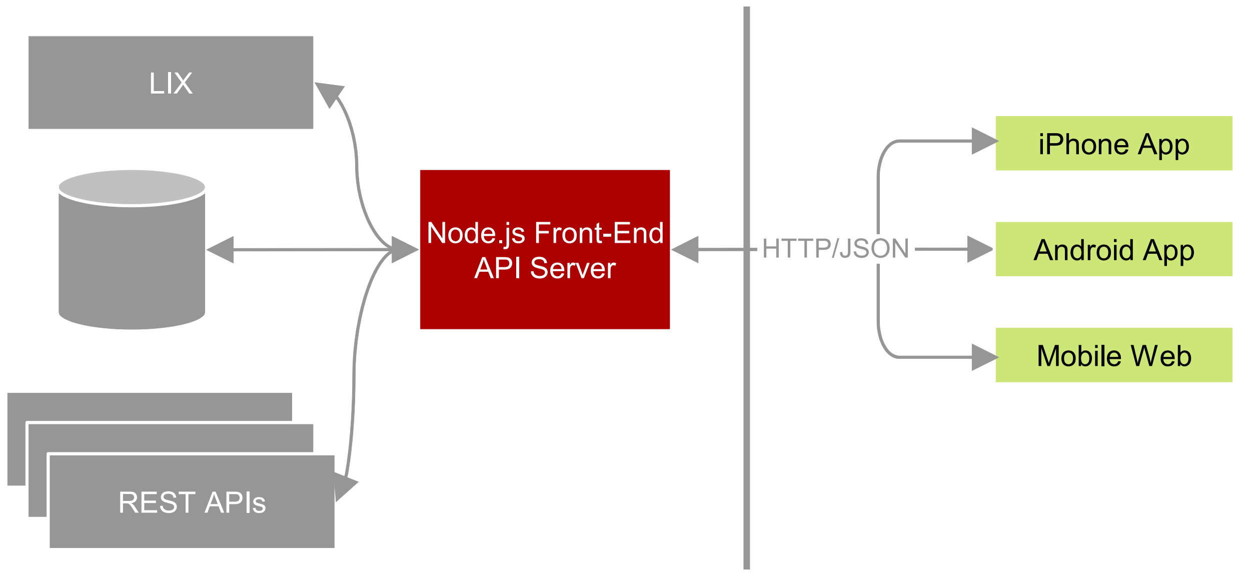
A key reason for the original hybrid app's development gains was due to the way the LinkedIn team reused HTML5 from the Web app in the native apps for iOS and Android. They utilized HTML5 Web views in the native app for displaying Web-based content, and where HTML5 has functional gaps, such as a scrolling infinite list, they leveraged native controls. The mobile app also relied on critical JavaScript technologies including the Backbone.js framework and Underscore.js libraries.

The hybrid app worked really well when mobile usage made up less than 10% of traffic. Issues started creeping up as more users spent more time in the app. It wasn't performance issues in the app, like speed or rendering, but memory issues as users used the app more. Another important issue centered around some of the animations and the difficulty in achieving a native-level smoothness and feel. By April 2013, the LinkedIn team decided to deploy fully native iPhone and Android apps to solve these two challenges. As a mobile app becomes a more strategic part of the business, enterprises need to change course accordingly — in LinkedIn's case, moving from mobile Web to hybrid apps to native apps.

On the server side, LinkedIn has had a smoother experience since the 2011 launch. A big part of the scalability improvements was due to their switch from Ruby on Rails to Node.js, helped by expert engineers. In 2011, LinkedIn used the project as a hiring incentive, because great developers want to work on leading-edge technology. They hired engineers with experience in Google's V8 JavaScript engine, and the company's existing Ruby on Rails developers converted to Node.js and JavaScript.

Their entire mobile server stack was completely built in Node.js for two important reasons: scalability, and the ability for Node.js to easily talk to other services. The LinkedIn mobile app has to interface to many platform APIs and databases, and Node.js enabled significant performance gains compared to what was used before with Ruby on Rails. They went from running 15 servers with 15 virtual servers on each physical machine to just four instances that can handle double the traffic.

Figure 2. LinkedIn's Current Mobile App Architecture



Source: Gartner (April 2014)

Today, LinkedIn's native iPhone and Android apps, as well as their mobile Web apps, all access a tier of Node.js servers that provide a common set of APIs accessible over HTTP. The Node.js API servers broker data from multiple RESTful services, such as the LinkedIn recommendation service.

*Additional research contribution and review was provided by: Van Baker, Adrian Leow, Richard Marshall, Nick Jones, Mark Driver, David Smith, Danny Brian, Ray Valdes and Ian Finley.*

### Acronym Key and Glossary Terms

| | |
|---|---|
| **Node.js** | A software platform for scalable server-side and networking applications. Node.js applications are written in JavaScript. |
| **AngularJS** | An open-source JavaScript framework, maintained by Google, that assists with running single-page applications. Its goal is to augment Web-based applications with model view controller (MVC) capability, in an effort to make both development and testing easier. |
| **Backbone.js** | A JavaScript library with a RESTful JSON interface and is based on the model view presenter (MVP) application design paradigm. |
| **Undersccore.js** | A JavaScript library that provides utility functions for common JavaScript tasks. |
| **jQuery Mobile** | A touch-optimized JavaScript library or mobile framework. |
| **Apache Cordova** | A mobile development framework that enables software programmers to build applications for mobile devices using JavaScript, HTML5, and CSS3, instead of device-specific languages such as Objective-C. |

## Gartner Recommended Reading

*Some documents may not be available as part of your current Gartner subscription.*

"IT Market Clock for Programming Languages, 2013"

"Hype Cycle for Web Computing, 2013"

"Predicts 2014: Application Development"

"JavaScript: Past, Present and Future"

"HTML5 and Web Technologies Are Becoming Major Forces in Mobile Application Development"

"Containers Have a Major Impact on HTML5 Mobile Application Development Projects"

"Modern Web App Architecture"

"Build Great Mobile Apps With HTML5"

## Evidence

"Why Walmart Is Using Node.js"

"Node.js at PayPal"

"I-Tier: Dismantling the Monoliths"

"LinkedIn Refreshes Mobile App"

"Why LinkedIn Dumped HTML5 and Went Native for Its Mobile Apps"

"Exclusive: How LinkedIn Used Node.js and HTML5 to Build a Better, Faster App"

"LinkedIn's New Mobile App Is So Gorgeous, You'll Actually Want to Use It"

"Building Dynamic Personalized Onboarding Flows for Mobile"

"LinkedIn Hits 300 Million Users Amid Mobile Push"

## More on This Topic

This is part of an in-depth collection of research. See the collection:

- Cloud/Client Computing Changes the Nature of Applications

**GARTNER HEADQUARTERS**

**Corporate Headquarters**
56 Top Gallant Road
Stamford, CT 06902-7700
USA
+1 203 964 0096

**Regional Headquarters**
AUSTRALIA
BRAZIL
JAPAN
UNITED KINGDOM

For a complete list of worldwide locations,
visit http://www.gartner.com/technology/about.jsp