Mobile app development, JavaScript everywhere and "the three amigos"

By Jerry Cuomo, IBM Fellow
Vice President and Chief Technology Officer, IBM WebSphere

IBM

Every 12 years or so, there is a renaissance in application development. In the early 2000s the "web boom" sparked an extreme focus on the building of web applications. Now we see another renaissance with the advent of mobile, social, big data and cloud. The application boom has reached unprecedented peaks. People use these apps every day, for everything from making a hotel reservation, to checking bank accounts, to ordering a pizza. And with more than 1.5 million apps available in the Apple App Store and in Google Play, consumers have an almost-unlimited number of options from which to choose.

In previous articles about the industry's web-app renaissance, I have coined the term "engaging enterprise." This term describes a style of application that promotes the use of "systems of engagement" to drive new transactions to "systems of record." This pattern, where interactions drive new transactions, is at the center of the new age of engaging applications. Paypal alone expects more than USD20 billion in mobile transactions in 2013.[1]

IT solutions in the early 2000s better catered to the developer. In the mid-2000s, attention was more diversely focused on catering to a broader set of IT roles. Specifically, the systems-administration function became the prime beneficiary of technology advancements in virtualization and automation. This reality is appropriate, considering the fact that the systems administrator is often the person who is required to manage all the newly built web apps. Today's "app boom" is shifting the focus back to the developer with movements such as DevOps.

Business leaders cannot underestimate the influence that mobile app developers possess today.

DevOps is a software development method that stresses communication, collaboration and integration between software developers and information technology (IT) professionals. As consumers around the world embrace apps with vigor, DevOps helps create business success because the method is focused on "breaking down the walls" between developers and systems administrators. The method helps to simplify the development process from start to finish. And now we have arrived at the subject of this paper: simplifying development of engaging applications.

JavaScript pervasiveness creates an opportunity: you can simplify radically the lifecycle of applications that are interactive, web facing and transactional.

Concept count and the "JavaScript everywhere" movement

It is astonishing to consider the number of technologies, application programming interfaces (APIs) and techniques that an enterprise application developer must master these days. I like to use the notion of a "concept-count" to represent the number of concepts that a reasonable developer needs to keep in their heads to accomplish a task. For example, the task of building even a simple enterprise web application carries a high concept-count, and requires the understanding of Java APIs (JSE), JEE APIs, JavaScript, HTML5, CSS, XML, JSON, SQL and more. So, in this case, the concept-count is at 8+ for even a simple web app. I usually visualize this as a developer possessing eight or more O'Reilly technology books (the ones with the clever animal covers) stored on a Kindle device at any given moment. The "JavaScript everywhere" movement holds the promise of dramatically reducing the concept-count for developers. Let me explain why.

JavaScript is currently listed in many language studies as the most-used computer language. For example, JavaScript is ranked at number one by The RedMonk Programming Language Rankings for January 2013.2 Good reason exists for this ranking, since JavaScript is everywhere—literally. The JavaScript language is in all major web browsers. JavaScript exists on all the major smartphones (as part of the device's operating system). The JavaScript language can be found on the server-side (such as Java-based Mozilla Rhino and Node.js open source software) and can be found in the NoSQL databases (such as MongoDB). With the pervasiveness of JavaScript, an opportunity exists to radically simplify the lifecycle of engaging applications that are interactive, web facing and transactional—including development, operations and hosting.

JavaScript-based app servers, client libraries, mobile libraries and databases do the best job of exhibiting the attributes of systems that are "born on the cloud." The concept of "JavaScript everywhere" lowers the concept count for web development, and the "JavaScript everywhere" reality reduces the need for context switching between disparate languages. JSON (JavaScript Object Notation) is a major contributor to the lowering of the developers concept count. With JSON, a single JavaScript-centric data format is pervasive throughout the client tier, the server tier and the database tier.

JSON is a lightweight data-interchange format that is based on the object-literal notation of JavaScript. The format is programming-language neutral but uses conventions from languages that include C, C++, C#, Java, JavaScript, Perl and Python.

Let me explain more about why I have so intimately linked JavaScript and JSON. Business leaders cannot underestimate the influence that mobile app developers possess today. Specifically, the programming model that has evolved on the mobile client (and that is emerging on the server and database) *assumes* an intimate linkage between JavaScript and JSON. The intimacy is enforced by APIs, which are ingrained into JavaScript and which have been perfected over the years. These realities serve to maximize the efficiency of developers. The JSON text format is syntactically identical to the code that is used for creating JavaScript objects. In short, JSON evaluates to JavaScript Objects, and this linkage is both simple and powerful.

When I look at the last three corporate acquisitions made by the WebSphere team, including the IBM® Worklight® mobile application platform, the IBM Cast Iron® cloud integration products and the IBM Business Process Manager platform, I can see that the "JavaScript everywhere" approach predicted the future. Worklight software, Cast Iron products and Lombardi software all include Rhino for server-side JavaScript. Each of these WebSphere solutions supports a mantra of *(Java)Script-first, code later (if ever)*.

Reduce your concept count with "the three amigos"

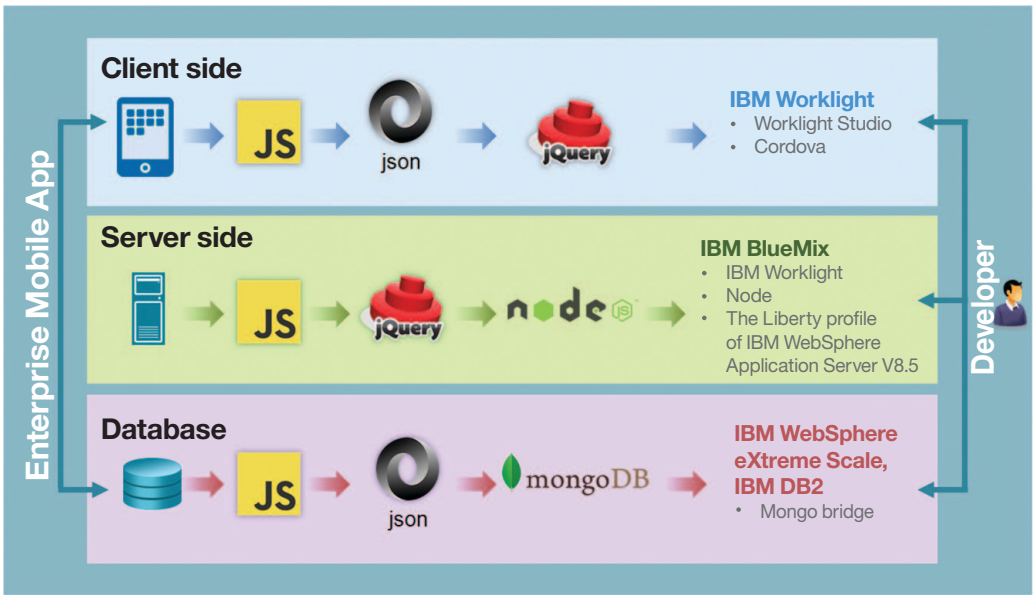
Looking across the three major compute tiers, three technologies stand out as uncontested leaders when they are identified using Google trend searches. Vibrant development communities form a strong foundation for these technologies and provide an ecosystem for support and custom extensions. I am interested to see that these three technologies are often used together in the creation of engaging mobile applications, and I have begun to call them the "three amigos," which is Spanish for "the three friends." These three technologies act as a developer's three best friends, because the technologies lower the concept count for the act of building mobile apps. The "three amigos" are jQuery, Node.JS (with Google V8) and MongoDB.

jQuery. The jQuery client library is fast and concise. The jQuery approach simplifies HTML document traversing, event handling, animating and Ajax interactions. The result: rapid web development. The jQuery library starts small and makes it easy for you to interact with the DOM elements, easy to add effects, and easy to execute Ajax requests. The jQuery community adds tens of thousands of plug-in modules of all functional varieties; this addition of plug-in modules makes it possible for developers to quickly enhance the simple base to a more-functional client. When it comes to jQuery, the IBM team is taking action to bridge JavaScript and native-device APIs.

Node.JS (often simply called "node") is an open source software system that is designed for writing highly scalable Internet applications, notably web application servers. Programs are written in JavaScript, using event-driven, asynchronous input and output (I/O) to minimize overhead and to maximize scalability. Node.JS consists of Google's V8 JavaScript engine, libUV, and several built-in libraries. Node.JS has attracted a vibrant community of dedicated developers, who have built a large collection of additional value-added libraries and have contributed these libraries for open use.

MongoDB is an open source document-oriented database system. MongoDB is part of the NoSQL family of database systems. Instead of storing data in tables as is done in a "classical" relational database, MongoDB stores structured data as JSON-like documents with dynamic schemas (MongoDB calls the format BSON), making the integration of data in certain types of applications easier and faster. With MongoDB, JavaScript can be used in queries, and aggregation functions (such as MapReduce) are sent directly to the database to be executed.

Within the IBM Software Group, the motto is "embrace and extend" open source. The "three amigos" are also becoming honorable friends as the Software Group mobilizes the IBM MobileFirst set of capabilities.



Embracing and extending jQuery: bridging JavaScript and APIs

As I said earlier, every dozen years or so, a renaissance occurs in application development, and we are all working within one now. The browser platform comes with limitations, and JavaScript is everywhere. If your application use cases require capabilities that cannot be achieved using browser engines, investigate an approach called hybrid application development. You get the benefits of web development with JavaScript and support for bridging technology. Your JavaScript application code can "call out" to make use of native-device operating system capabilities, or your code can make use of native libraries that are not supported directly by the browser.

Apache Cordova is the most popular open source library for this sort of bridging. You can package a JavaScript web application as a real native app that can be distributed through mobile app stores. Device APIs and other services can be used directly. This is important, because the APIs are usable on older browsers. And with Cordova, only the code for your app's relevant features must be loaded.

The **IBM Worklight** mobile application platform is the primary IBM product offering that directly supports this hybrid application style of app development. Worklight builds upon Apache Cordova, and adds Cordova plug-ins for enterprise app features. Worklight supports the use of any JavaScript library and integrates with software development kits from native-device vendors.

Today's enterprise application developer must master a range of technologies, APIs and techniques that is, quite frankly, astonishing.

Embracing and extending Node.js

These days, developers choose programming languages as platforms upon which to develop applications quickly. This ability to choose, coupled with modern Platform-as-a-Service (PaaS) environments such as the IBM Bluemix implementation of IBM Open Cloud Architecture, helps you to build, manage and deploy applications repeatedly, at the scale you require.

The polyglot mix is becoming the "new normal" for developers.

Emerging platforms using scripting languages are becoming more popular in developer communities, and these platforms are increasingly used in cloud environments. A wide variety of frameworks, tools and languages are being employed, since modern applications typically use multiple languages to deliver web and mobile applications.

IBM is embracing Node.js as the first in a series of languages in this polyglot environment. Enterprise leaders will take systems that are running in production today and will expose more of these existing systems to application developers. Leaders will accomplish this by making available high-value services—and then providing access to those services using new APIs, enabling mobile apps and creating new patterns of engagement with these existing systems.

I believe that JavaScript and Node.js will help bind these new services together in a way that is easier for apps to consume. Leaders are rapidly adopting Node.js, and Node.js will evolve to become "enterprise-ready." I expect that several functions will be added to the ecosystem, including internationalization for National Language Support (NLS), built-in security, trusted repositories and "no-need-to-compile" node-packaged modules with C/C++ code. Enterprise concerns will drive these changes quickly.

Typically, if you create a Node.js application, you will follow proven methods to manage issues with security and load balancing—and the IBM team can help you with that. With enhanced tooling and enterprise-level support, Node.js becomes an ever-more-viable solution for IBM clients. Node.js remains a valuable "tool in the toolbox," and enterprise teams are learning how they can benefit from an approach that is robust, nuanced and well-supported.

Embracing and extending MongoDB

IBM can add value to these open source communities by creating a collection of (elastic) services that are 100 percent compatible with the technologies found within these communities, yet differentiated in the areas of analytics, transactional-integrity, and internet-level scale. For example, my IBM team has started an incubator project that allows MongoDB apps (coded to the MongoDB APIs) to run 100 percent unchanged against IBM DB2® database software and the IBM WebSphere® eXtreme Scale data grid. This is achieved in part by implementing the MongoDB wire protocol within DB2 and eXtreme Scale. My team calls this the "elastic data service."

When a developer points a MongoDB app at the elastic data service, the developer gets additional capabilities and qualities of service. For example, you can use an extension library that supports extended query capabilities such as joins. The IBM team is also looking at providing other value-added features for developers, including running MongoDB apps against older versions of their data—even when the "older" version was created just hours before. This capability represents a powerful feature for debugging, simulation testing and analytical studies.

In June 2013, analyst firm IDC named IBM a leader in their Marketscape analysis of worldwide enterprise mobility consulting vendors.[3]

The importance of JavaScript for front-end development

These days, your teams have moved from building websites to building apps. When they were building websites, your site was served by the application server to send an HTML page to the browser. Your team has layered in Cascading Style Sheets (CSS) and JavaScript to add behavior and to provide a more-interactive experience such as dynamic menus. On websites, most of the site's navigation logic continued to route through the server.

When you build apps, most of the navigation logic shifts to the client side that is running on the device. The result: increasingly larger amounts of JavaScript are being used for the front end of the application. In classic application architecture, your website backend would be where the Model View Controller (MVC) resided (the controller is responsible for navigation and mediation between the business model and the HTML user interface code). Now, in apps, the MVC is running in the client device, within a desktop browser or within a mobile browser (often embedded within native application code)—or the various MVC tiers might be split amongst the back end and devices. This creates a situation in which larger amounts of JavaScript code is now shifting over to become the responsibility of the front-end app developer.

The trend toward libraries and tools for building large applications within the JavaScript ecosystem parallels the trends that occurred in other language environments in the past, such as Java. Building sustainable applications that have very large amounts of code requires modular architecture, which must possess the ability to dynamically load only the minimal amount of code (modules) necessary for the parts of the application being used—and only when the code is needed. Modular JavaScript systems have emerged over time. An example is RequireJS, which is based on the Asynchronous Module Definition specification (AMD). Using a modular approach to constructing larger JavaScript applications helps you to better maintain the application over time; these module systems can safely load jQuery alongside many other JavaScript packages that are needed to build apps.

There are many open source JavaScript libraries that can be used together in a modular fashion for front-end development. Examples include jQuery, Dojo Toolkit, BackboneJS, EmberJS, AngularJS, Polymer and many others. At this time, the industry has not settled on a single JavaScript library or MVC approach; however, formal standards are emerging (with shim or polyfills or a combination of shim and polyfills, for these APIs on older browsers); examples include World Wide Web Consortium (W3C) WebComponents, HTML Templates, MDV, Shadow DOM, Web Animations and Pointer Events. As specifications are refined, I expect them to help bring together what are currently disparate programming models, with native support for these specifications built into modern browsers. And I expect this to happen in the near future.

When you consider what constitutes the "correct" package of granularity, remember Eric Raymond's Rule of Modularity**:**

Developers should build a program out of simple parts connected by well defined interfaces, so problems are local, and parts of the program can be replaced in future versions to support new features. This rule aims to save time on debugging complex code that is long and unreadable.

I believe that the embrace of modularity has had a significant impact on the ecosystem that is growing up around JavaScript (on client and server) and contributes to the GitHub phenomenon.

It is more important than ever that robust quality controls and governance can be applied across the growing number of JavaScript packages that you will be using from open source. When you build high-quality packaged applications that will stand the test of time, it is vital that you determine quality characteristics and maintain consistency across the entire set of applications that are being developed, maintained and used over time by your organization. Here are just some of the important aspects of quality which require understanding the compliance of each JavaScript package being used:

- JavaScript language compliance
- Browser platform support
- User input handling approaches (mouse, keyboard, touch, pointer)
- Accessibility standards compliance
- Internationalization standards compliance
- Proven methods for security
- Legal licensing and pedigree standards
- Packaging and versioning standards
- Documentation standards

The IBM team is embracing modern JavaScript libraries and emerging standards with techniques that work for large-scale, multichannel-ready applications. The team puts a special focus on technologies that help maintaining quality over time. With years of JavaScript experience and proven methods learned from building thousands of web applications, the IBM team delivers the kind of knowledge that can help you to "shine the headlight" on changes that could occur in your environment—whether mobile, browser or desktop. If you are concerned about data behind firewalls, or if you must treat the back end, involve the IBM team. You can "mix and match," you are not locked into IBM systems and you get full-spectrum support.

Meet the "fourth amigo": MQTT

With the propagation of various smart devices, the internet will evolve to include what some are calling an Internet of Things. Billions of interconnected smart devices will measure and move all the bits of data that make up daily life. Smart devices will even act upon these bits of data in an independent manner. The Internet of Things is a realm in which the digital meets the physical, and from this Internet of Things a "fourth amigo" is rising: the MQ Telemetry Transport protocol (MQTT). This open-device messaging protocol is lightweight enough to be supported by the smallest devices, yet robust enough to ensure that important messages get to their destinations every time.

The openness of the "JavaScript everywhere" environment is reflected in the wide range of support options for open source MQTT implementations. MQTT for JavaScript is called Paho, an MQTT client that provides scalable, open source implementations of messaging protocols that are aimed directly at existing and emerging applications for the Internet of Things. With the IBM approach to MQTT, devices such as smart energy meters, cars, trains and satellite receivers can communicate with each other—and can communicate with other systems or applications.

Conclusion: "JavaScript everywhere" holds true promise

The concept of "JavaScript everywhere" holds the promise of reducing the concept count for developers who build engaging applications, and the IBM team has a clear view of how to work within the context of the key JavaScript communities.

Expect IBM offerings to add value in the form of a collection of elastic services that extend these technologies. Expect differentiation in the areas of analytics, transactional integrity and internet-level scale.

The great thing about a renaissance is that innovation reigns. The speed of creativity matches the speed of change step-for-step, and the best minds are set free to explore. As a wide range of open source implementations comes together with communities of innovation and easier tools, I see remarkable victories every day. To me, the fact is undeniable: developers possess the power to change the way that companies engage the world.

Acknowledgements

Thanks to my IBM colleagues for their help in the creation of this white paper.

Christopher Mitchell, IBM Software Group, Application and Integration Middleware Software STSM, IBM Multi-channel Strategy, Front-end Architecture and UI Integration

Andrew Low, IBM Software Group, Application and Integration Middleware Software STSM, JavascriptEverywhere Technical Lead, Mobile Elastic Services, Master Inventor

John Duimovich, IBM Software Group Distinguished Engineer

For more information

To learn more about how you can rapidly take mobile, WebSphere and cloud technologies to the next level, please contact your IBM representative or IBM Business Partner, or visit the following website:
ibm.com/developerworks/community/blogs/icap/entry/home?lang=en

Additionally, IBM Global Financing can help you acquire the software capabilities that your business needs in the most cost-effective and strategic way possible. We'll partner with credit-qualified clients to customize a financing solution to suit your business and development goals, enable effective cash management, and improve your total cost of ownership. Fund your critical IT investment and propel your business forward with IBM Global Financing. For more information, visit:
ibm.com/financing

IBM Corporation
Software Group
Route 100
Somers, NY 10589

Produced in the United States of America
October 2013

[1] *Bloomberg TV, April 24, 2013*

[2] *RedMonk.com*, The RedMonk Programming Language Rankings: January 2013

[3] IDC MarketScape: Worldwide Enterprise Mobility Consulting 2013 Vendor Analysis

Please Recycle

WSW14215-USEN-00