Mainframe Integration Using JavaScript Tools

By Russ Teubner
HostBridge Technology

This three-part series describes developer productivity gains, fast SOA/cloud deployments by real companies, and more top 10 benefits of JavaScript tools for mainframe integration.

“3... 4... 25 Times More Productive”: Web Services Programmers Talk JavaScript Tools

Here’s some very good, very timely news for organizations resolved to do more with the mainframe, e.g., by putting CICS applications and mainframe data to work in an SOA or other integration architecture.

Web services developers and development teams report significantly higher productivity when they use a JavaScript-based web services development tool.

I know, I know.... I too have heard all the “ifs,” “ands,” and “buts” – JavaScript isn’t a server-side technology, JavaScript isn’t a full-fledged programming language.... Fact is, these just aren’t true anymore. JavaScript has matured, particularly on the server side, and, thanks to the efforts of advocacy groups like CommonJS, the language, libraries, and new extensions have been and continue to be standardized by ECMA. As for productivity, you don’t have to take my word for it.

Listen to real web services developers who are doing CICS integration for large enterprises. Three were asked about productivity using a JavaScript tool for mainframe web services compared to other, more common tools. Their answers may surprise you.

Services Programmer, Life Insurance Company
A veteran contractor-developer working for a large life insurance provider said, “I’ve written lots of COBOL in my career. If I put a number on productivity using a JavaScript tool, I’d say 3 to 1 – I am at least 3 times more productive using a JavaScript tool for creating web services from legacy mainframe applications.”

That’s a good testimonial. But this individual’s productivity really improved when doing web services that adhere to the insurance industry’s ACORD XML standards. “For customizing the output of a web service to produce ACORD XML, I am 25 times more productive. ACORD XML is very tedious with something like COBOL and very easy with a JavaScript tool.”
“25 times more productive.” I’ll let that quote speak for itself.

**Application Delivery Manager, Bank**
An application development manager at a regional U.S. bank had similarly positive answers. “I have a lot of experience with FEPI and HLLAPI interfaces, and I’d say that this isn’t really a fair comparison. There’s no contest. As a novice with the JavaScript integration tool, I was at least 4 times faster at developing web services.”

He went on to offer another compelling perspective, that of a long-term project. “In one of our projects, I was able to develop a ‘production-sturdy’ (not quite ready for production) integration application in just 2.5 months. The same application took someone else using conventional methods almost 3 years.”

If you’re contemplating mainframe web services, accelerating deployment by 90% and achieving your business goals much, much faster is another good outcome. You might even agree that this is one of those improvements marketers justifiably describe as “dramatic.”

As our next case demonstrates, the productivity benefits of a JavaScript-based integration tool also extend far beyond the individual programmer.

**Production Control Manager, Bank**
As our next case demonstrates, the productivity benefits of a JavaScript-based integration tool also extend far beyond the individual programmer.

“You’re asking the wrong question,” he said. “It’s not about the productivity of a single individual. Using a JavaScript tool, we now have entire groups of developers that have the skills and are excited about writing and supporting web services for the mainframe.” The JavaScript tool, he added, resolved a long-standing issue – how to maintain high job satisfaction among the mainframe web services team and keep his skilled, experienced people on the job.

I won’t belabor the point since these developers and team leaders make an authoritative case. I’ll simply close by reiterating a few salient points: JavaScript is a familiar and widely known programming tool, probably within your own team. JavaScript tools make it easier and faster to transform legacy applications into web services. And JavaScript tools make developers and integration teams quantifiably more productive.

Next time, I’ll share more about the flexibility and unique capabilities of JavaScript in the mainframe integration world.

* * * * *
JavaScript Tools for Mainframe Integration: 10 Top Reasons

Previously, we heard testimonials from web services developers and team leaders responsible for mainframe integration at large enterprises. They reported productivity gains ranging from 4 to 25 times when using a JavaScript tool versus more common tools.

This time, I’d like to explain how and why these kinds of improvement are possible. If you are resolved to do more with CICS and mainframe data, and spend less, here are 10 quick reasons – from familiarity and flexibility to unique mainframe integration capabilities – to look at JavaScript tools.

10. **JavaScript is widely known.** Look around your own integration team. I bet many of your services developers already know JavaScript. I’ll even double down – I bet more know JavaScript than any of the most common mainframe programming languages – COBOL, REXX, you name it. Choose JavaScript and a much broader set of developers – .NET, Java, web – are ready to help you reach your integration objectives.

9. **JavaScript is easier.** With its straightforward rules and procedures, JavaScript is easier to master and use than almost any other programming language, especially of the mainframe variety. JavaScript requires no extra development tools, and the latest versions, with built-in generators and iterators, make scripting easier than ever.

8. **JavaScript costs less.** In the exciting new math of this article, 10 + 9 = 8. Less programmer training plus faster development will save you money. Or from a business perspective, more programming resources plus quicker deployment might just make you money sooner. With JavaScript-based integration, you can do more, spend less, and accelerate your Return on Integration.

7. **JavaScript is flexible.** Because it’s a scripting language, JavaScript unlocks those old proprietary silos – instantly. It “speaks” all the industry-standard integration languages – HTTP, XML, web services – and it enables you to bring together existing and emerging technologies – from 3270, IMS, and COMMAREA to SOAP, REST, and Atom. You name it, JavaScript integrates it.

6. **JavaScript is standardized.** Netscape had the right idea when they submitted JavaScript to ECMA, the European Computer Manufacturer’s Association, for standardization years ago. The latest ECMA-262 specification – ECMAScript Edition 3/JavaScript 1.8 – ensures consistency and compatibility as the language grows and libraries evolve. Recently ECMA also defined the ECMA-357 standard for E4X, ECMAScript for XML, which extends standardization to the wide world of XML-based integration.
5. **JavaScript is for servers.** Many still think of JavaScript as a client-side technology. *Au contraire.* Powerful server-side engines/interpreters have proven their mettle, and new ones are introduced regularly. With these advances, JavaScript is fast becoming the programming language of choice for dynamic cross-system interaction and integration. On this point I recommend a recent article by Kris Kowal in the December 2009 *Ars Technica* entitled “CommonJS effort sets JavaScript on path for world domination.” Reasons No. 6 and 5 borrow from his article. Here’s a link (but feel free to wait to click till you’ve finished reading): [http://arstechnica.com/web/news/2009/12/commonjs-effort-sets-javascript-on-path-for-world-domination.ars](http://arstechnica.com/web/news/2009/12/commonjs-effort-sets-javascript-on-path-for-world-domination.ars).

4. **JavaScript’s integration power has been dramatically enhanced by HTTP request capabilities.** Though the capability emerged more than five years ago, it is catching on with mainframe integration developers slowly. It’s an opportunity untapped. (I considered this for reason No. 1, but I haven’t gotten to the mainframe yet.) JavaScript-driven HTTP requests are the pivot of the pendulum that is swinging mainframe integration toward a brighter tomorrow. Simply put, JavaScript-driven HTTP requests make mainframe applications interactive and dynamic, and they enable the mainframe to integrate with anything – faster, easier, better, cheaper.

3. **JavaScript’s object-oriented methodology is taking hold in the mainframe world.** JavaScript is a good citizen of the mainframe world. Your programmers will find it easy to integrate mainframe functions using any service-oriented technology – SOAP and REST, integration objects, situational mashups, or full-blown SOAs. Using JavaScript in compiled form, developers can script interactions and save the reusable scripts on the mainframe to be invoked as services. And just a cursory glance at the latest System z product releases and development directions shows that IBM is fully in sync with the object-oriented paradigm.

2. **JavaScript simplifies mainframe integration.** Now that the vast majority of mainframe applications and distributed systems can call and respond to scripts, integration programming is infinitely easier (and “infinitely” is only a slight exaggeration). Ask developers or IT managers. It’s much easier and much, much safer to modify scripts than to rewrite mainframe code.

1. **JavaScript is ideal for integrating CICS transaction micro flows.** It’s a given that reusing CICS data and application logic brings tremendous benefit. But integrating terminal-oriented CICS applications, which happen to be the most common, has been a very tough challenge. *Problem solved.* JavaScript is uniquely capable of orchestrating and executing the most complex terminal-oriented micro flows, involving dozens or even hundreds of screen transactions, as a single service. So interactions that used to require dozens or hundreds of HTTP requests/responses across the network can now be
completed with one request/response when a JavaScript engine resides on the mainframe. For high-performance, high-fidelity integration of terminal-oriented applications, JavaScript tools are emerging as the No. 1 choice.

There you have the JavaScript Top 10. Real cases to follow....

* * * * *

Applications, SOAs, and Clouds: Using JavaScript Tools for Mainframe Integration

In my first article, I shared testimonials from web services developers reporting productivity gains of 4 to 25 times when using a JavaScript tool for mainframe integration. Next, we counted down 10 top reasons why more integration developers/architects are turning to fast, easy, flexible JavaScript.

Now I want to close out the series with three brief cases of companies using a JavaScript-based engine to develop and deploy web services to integrate CICS applications and mainframe data with numerous distributed systems.

The three companies – an auto maker integrating applications, a credit union doing SOA enablement, and a health insurer bringing its mainframe to the cloud – are gaining tactical advantages by rapidly developing reusable, standards-based services ready for any environment. At the same time, they are building toward the goal of leveraging the mainframe as an integral part of SOA strategy.

Auto Maker: Application Integration

The finance division of a top ten auto manufacturer runs a CICS lease management application and stores lessee and program data in VSAM files. Customer service reps use these resources all day every day. When the division implemented Siebel CRM, they needed a practical way to integrate their mainframe into the new customer service system.

With a JavaScript-based integration tool, the company developed, tested, and deployed 40 web services in just 35 man days. Also, because their JavaScript engine supports the full range of services types, and both their systems reside inside the firewall, they were able to employ a lightweight RESTful services model (HTTP GET/POST to an ESB and Siebel CRM) and avoid the heavy lifting associated with formal SOAP/WSDL services.

They chose their JavaScript tool specifically because of the inherent flexibility of its standards-based approach and because many of their developers and integration architects already knew JavaScript. Shorter development cycles enabled the auto maker to achieve...
their objective and initial ROI in just over a month. This first tactical implementation also fit in seamlessly with a larger SOA vision.

**Credit Union: SOA Enablement**

The largest credit union in the world has 7,000 employees serving more than 3 million members deployed around the world. The credit union standardized on a JavaScript integration tool several years ago, and their approach to SOA is a model case of tactical implementations building toward strategic goals. Today, their SOA is the backbone for most member- and employee-facing systems, including online banking, ATM machines in all 24 time zones, a Universal Agent Desktop, and teller applications. Thanks to their JavaScript integration tool, all these user-facing systems integrate flawlessly with Fidelity banking apps and other CICS applications.

Among other benefits, the credit union found that JavaScript is ideal for orchestrating and automating the complex and challenging transaction sequences that are common with CICS terminal-oriented applications. The mainframe-resident JavaScript engine is uniquely capable of orchestrating a micro flow – even if it involves hundreds of transactions – and delivering it as a single web service with the highest possible fidelity to the source applications.

The JavaScript development and runtime engine made it easy for the credit union to manage each services-based integration as both a discrete technical objective and an architecture component supporting their larger SOA vision. Currently, the credit union is prototyping another new solution that will use the JavaScript tool to generate lightweight web services linking mainframe events to event management and reporting systems. Stay tuned....

**Health Insurance Provider: eHealth Cloud**

A health insurer in a European nation is a founding member of a consortium taking a creative approach to a recent government mandate requiring electronic health records for all residents. To fulfill this directive, the consortium is building a private eHealth cloud. The insurer is using a JavaScript-based development tool to merge its mainframe in the cloud.

After initial rollout for founding members, the cloud’s *Platform-, Software-, and Data as a Service* components will be made available nationwide, potentially to millions of residents, healthcare providers, and insurers. The cloud has two core application modules – an online health records and community interaction module and a full-service insurance module.

JavaScript-based web services connect the insurer’s CICS resources – CA Ideal applications and CA Datacom database – with the cloud platform. Here’s a quick and simple illustration. When a user requests policy or claims information, the request makes its way through the cloud infrastructure via standard protocols and transports to the insurer’s web server and
on to the mainframe-resident JavaScript engine, which accesses the requested application and data. The JavaScript engine then POSTs the response back to/through the cloud to the user.

The JavaScript tool provides the insurer with a particularly clean, fast way to develop its web services. The JavaScript tool has a XML engine that makes it very easy to see, navigate, and create services for CICS screens and transaction processes. And by delivering mainframe data through standard transports in standard languages (HTTP, TCP/IP, XML), the JavaScript tool affords an easy way to make the mainframe a good citizen of the services-based cloud.

Among other benefits of offloading integration functions and user interfaces to the cloud provider, the insurer expects to reduce TCO for user-facing systems by 75%.

Whether you’re developing composite applications, mashups, SOAs, clouds, or any services-based solution, I hope you’ll agree (or at least agree to entertain the idea) that a JavaScript tool offers a way to harmonize your IT resources to achieve tactical objectives on the way to strategic goals.