JOURNAL OF OBJECT TECHNOLOGY

Online at http://www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2007**

Vol. 6, No. 10, November-December 2007

Programming the World in a Browser *Real Men Don't Do JavaScript Do They?*!

By **Dave Thomas**

1 AND THE WINNER IS JAVASCRIPT

The mainstream professional developer community has never taken JavaScript seriously but soon they will have no choice. JavaScript is ready to move to center stage as the development and delivery technology for Web 2.x applications. In the past, most enterprise and product developers flocked to Java or C# while web developers moved to PHP, Perl, Python and more recently Ruby, with most ignoring the web based scripting language called JavaScript. At best it has been considered something to spiff up one's HTML pages. To make matters worse, incompatible implementations of JS and the DOM have tormented developers and made JS very unpopular with many. Until Ajax and Web 2.0 Douglas Crockford seemed to be the only advocate for JavaScript as a reasonable programming language. He pointed out that JS was really a Scheme like language with a prototype-based object model layered on top of it. I'm sure that popular author David Flanagan never dreamed that he would be best known for his book Definitive JavaScript.

While many smaller companies had built quality widgets and applications in JS it is was the entry of Yahoo Widgets, and more importantly the that of Google Mail, Calendar, etc. that laid the commercial foundation for the Ajax revolution with a plethora of frameworks and tools. Rather than bulky and complex web standards, more and more of these toolkits support simpler Restful style services that use JSON rather than SOAP.

JavaScript's ubiquitous browser availability makes it the frontrunner in that environment and this will no doubt ripple to servers and appliances. JavaScript is headed into the limelight once promised to Java, then later to Flash. It will be a must know language for everyone within 3 years. If JavaScript does cross over from the browser to other platforms it could inflict collateral damage on these languages down the road.

Web 2.0 – The Push for a Faster, More Robust JS in the Browser

The Browser as a platform for Web 2.0 and a software as a service application delivery system requires a robust JS that supports the execution of complete applications. It is well known that the initial implementations were not designed as high performance VMs and that the GCs were problematic. The race to improve JS was kicked off by the Mozilla SpiderMonkey implementation, which demonstrated that JS performance could be improved. Several popular Web 2.0 sites quickly showed that the IE implementation performed poorly, making Firefox a more attractive web platform. This of course had MS racing to put new legs on its JS implementations, which were announced in March at the Mix conference.

2 JS ICING ON THE CAKE - UNIVERSAL LIGHT WEIGHT RUNTIMES

Silverlight and the Dynamic Language Runtime

Silverlight is the recently announced lightweight CLR implementation with an associated subset of the .NET framework that runs in the browser. Silverlight extends the reach of the MS CLR beyond IE and Windows. More impressively, it allows MS Visual Studio to be used to develop and debug applications running in non MS browsers running on non Windows platforms. MS has demonstrated IronPython and IronRuby implementations of these popular scripting languages as well as JS implementations.

Recently, there has been discussion of a dynamic language runtime (DLR), which is based on Jim Hugunin's work on IronPython, rumored to include Python, Javascript, Visual Basic and a version of Ruby which will use the CLR rather than managed C code. This promises to finally give dynamic languages first class status at Microsoft. So far the most impressive feature is the demonstration using MS Visual Studio to debug a Safari web application running on a Mac.

Scripting Language Execution on the JVM

At last it appears that Dynamic Languages may finally be worthy of consideration by the Java community. Efforts by researchers and implementors at Sun have demonstrated the viability of the approach and are adding the machinery to improve Java execution.

The Sun Finally Shines on Dynamic Languages

In Java 6 Sun has introduced two JSRs specifically intended to improve Java as a scripting platform. JSR 223 makes it easier for Java based scripting languages to interact with existing Java classes. This will improve JS execution with Rhino, the popular JS implementation used on the JVM. JSR 292 introduces a new bytecode invokedynamic that supports efficient and flexible execution of dynamic method sends. The good work by the JRuby team at Sun demonstrates that dynamic languages such as Ruby can be implemented quite efficiently on the JVM by using a combination of compilation and proven VM implementation techniques such as inline caches. Through the use of tricks used in Smalltalk and Self as well as type inference and selective compilation it is now possible to script server side applications using JS or Ruby.

Google Reviving Rhino?

Google has recently been an active contributor to Rhino, the open source JS implementation on the JVM. Although it is doesn't support the current standard perhaps renewed efforts will bring it in line. It should be enabled through Java JSR 223.

More impressively, Google has recently demonstrated the implementation of a Rails like framework Rhino-On-Rails using Rhino JS. A recent blog post by Steve Yettes has spiked rumors about Rhino at Google. However, if Google does do something it would seem more likely to leverage its Smalltalk, Self and Hot Spot expertise given Google's VM wizards such as Jeffrey Dean, Urs Holtz and Lars Buk, etc. This would provide a better client and browser side story that didn't mandate a JVM. Perhaps there is a Hot JS on the horizon, when it comes to Google anything is possible.

IBM Project Zero?

Earlier this year, IBM introduced Project Zero, which promises many things but comes up short on delivery, at least initially, and seems to position IBM out of the running for a serious dynamic runtime. The current offerings include a subset of PHP and some sort of cooperation on Groovy, which is itself well behind the work on JRuby. IBM clearly has the expertise in its OTI Lab VM team but Project Zero seems to be an independent effort that doesn't appear to leverage that expertise. IBM has bet so heavily on Java runtimes that it will now have difficulty reacting to a world where Javascript is king. Unfortunately it isn't easy to have zero overhead nor zero complexity when resting on top of an infrastructure such as the JVM[1]. Also, the more JS remains a dynamic language the more problematic a compile to Java based implementation will become.

Rather than leveraging the true open source community that made IBM a mainstream player with Eclipse, IBM seems to be playing bait and switch with bogus community style licenses for Project Zero and Jazz. This has the potential to take IBM from OSS leader to loser, especially if in doing so they alienate the good will and support of the Eclipse community, which has supported IBM in its tooling and runtime efforts.

[1] JVMs in a "Software as A Service" Environment? It is interesting to point out that the JVM community is very silent about memory footprints of Java versus C based JS, Ruby, PHP or Python in a hosted software as a service environment. I'll bet Sun and IBM are very busy trying to figure out how to get Java starting a lot faster and sharing a lot more code!

3 ECMASCRIPT 4 – JUST SAY NO!

Quietly, while no one was watching or caring, Adobe and Mozilla hijacked ECMAScript and are now driving forward with a monster disguised as ECMAScript 4. The open reference implementation is a nice idea, but it is hardly a proven approach for defining a language standard, especially with a C based implementation[2]. One may argue that it worked for C# and JS however these were already defacto standards and they were not designed by committee.

Just browsing through the wiki shows a language which has prototypes, classes, multi-methods?, static types, dynamic types, etc, etc. This reminds an old guy like myself of other large design by committee languages such as PL/I, Algol 68 and ADA. These ambitious efforts all had smart people involved in the design and implementation but were unfortunately far too complex and came to the market too late. JS is intended to be a language for the people, not another language that only technical wizards can understand. If you are an Ajax developer or care about dynamic languages I suggest that it is time for you to speak up and help put ECMAScript 4 on a much less ambitious path than is currently being charted. Less is truly more when it comes to languages.

4 JS AS A PLATFORM!

There are numerous JS UI frameworks which enable developers to "target" JS in the browser for delivering applications including Google Web Toolkit (GWT), Yahoo Widgets and various Ajax frameworks. A more ambitious approach is used by Morfik's JST, which compiles applications developed using their UI builder and Basic, C#, Java or Pascal into JS Ajax.

A similar project, JSC is an experimental project to compile C# to JS.

Unfortunately, JS is not without its problems, one of which is the security risk exposed in XMLHTTP and JS/DOM interactions. These problems are due more to the DOM and Browser however. The Browser in particular is larger than many operating systems!

But surely no one would seriously consider compiling real applications to a native JS Platform. You clearly can't do that with JavaScript! Well, if you have not been watching your RSS feed you need to read about the bleeding edge research at Sun Labs and Microsoft Live Labs.

Microsoft Live Labs Volta research project led by Erik Meijer, the father of LINQ, compiles MSIL to JS. The main goal of the Microsoft Live Labs Volta experiment is to delay irreversible decisions when building Web 2.0 applications until the last possible responsible moment. Volta allows today's MS tools such as Visual Studio, C# and Visual Basic and applications to leap into the browser and cross

[2] It can be very difficult to develop tools or independent implementations if the syntax and semantics are encrypted in C. PHP and Ruby are just two recent examples where it has proven difficult to understand the grammar and semantics from the reliable and very useful de facto reference implementations.

platforms with zero deployment cost, optimizing for whichever execution environment (JavaScript, Silverlight) is already available on the client. Volta explores simple ways to build applications which span the internet cloud from user to data source using declarative tier-splitting refactoring.

Sun Lively is billed as a WebOS in JS. Lively leverages the impressive Squeak Morphic graphic framework to deliver applications on a JS + SVG platform. Lively is inspired by Dan Ingall's work on Smalltalk and Squeak and no doubt by Dave Ungar's work on Self. It provides an open, live programming experience in which the running code can be edited on the fly. The use of vector graphics enables rich new UIs that go beyond classical widgets. This brings to mind Sun NeWs, which pioneered the use of programmable vector graphics based UIs using Display Postcript and was used heavily in NextStep.

JavaScript – The World's Most Popular Dynamic Language

JavaScript is growing up very quickly from a little scripting language to a full blown application delivery run-time. It enables a new generation of easy to use Web 2.0 tooling, such as IBM QEDwiki, Yahoo Pipes and Microsoft PopFly, for rapid application development for the Programmable Web. With luck Web 2.0 may allow many of us to stop fighting the middleware and get back to the joy of building applications!

It also appears that we may finally be moving away from platform-dictated UIs and windowing systems to richer UI experiences tailored to the needs and expertise of the user. I look forward to the IT community waking up to find that while they were lost in the complexity of SOA, end users are already using their own enterprise application integration and workflow orchestration as a restful and easy to use Mashup!

About the author

Dave Thomas is cofounder/chairman of Bedarra Research Labs (www.bedarra.com), www.Online-Learning.com and the Open Augment Consortium (www.openaugment.org) and a founding director of the Agile Alliance (www.agilealliance.com). He is an adjunct research professor at Carleton University, Canada and the University of Queensland, Australia. Dave is the founder and past CEO of Object Technology International (www.oti.com) creator of the Eclipse IDE Platform, IBM VisualAge for Smalltalk, for Java, and MicroEdition for embedded systems. Contact him at dave@bedarra.com or www.davethomas.net.