JOURNAL OF OBJECT TECHNOLOGY

Online at http://www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2008**

Vol. 7, No. 8, November-December 2008

Real Men Do JavaScript!
Programming the World in a Browser

By Dave Thomas

JAVASCRIPT – THE PERVASIVE DYNAMIC APPLICATION LANGUAGE

JavaScript is the most widely used dynamic language in the world and is becoming increasingly important as an application programming language. While many hard core developers are still in denial, Web 2.0 application developers from small and large companies are developing increasingly complex applications that run close to the user.

Flex is a popular and productive tool for building rich client web applications which are scripted using ActionScript, a dialect of standard ECMAScript. Increasingly RESTful protocols leverage the JavaScript Object Notation (JSON) a simple concise portable object serialization format. Rhino enables JS to be run on the server side for applications compiling JS to the JVM. There is even a version of the popular Rails framework called Jails. For the near term, JS dominance and major applications will be on the client such as GoogleApps and Thinkature.

Recent announcements of more robust and significantly higher performance runtimes, including Google V8, Mozilla SpiderMonkey and TraceMonkey, Safari SquirrelFish and MS Jscript for the dynamic language run-time (DLR), are moving JavaScript far from being just a way to spiff up your web page to being both a fun and serious programming language. Google V8 is designed to support large applications such as Gmail. The MS Volta research project compiles the CLR directly to JS to allow MS CLR applications to run in any browser without requiring even the DLR. Competitors are rushing to show that they have the best performing engine. These efforts show how seriously the major players see the language as a platform for the future.

JAVASCRIPT IS A REAL LANGUAGE

In JavaScript "The Wrrld's Most Misunderstood Programming Language" Doug Crockford articulates the major features of JS. In his recent book [1] he provides a balanced perspective of the good and the bad of JavaScript. JS in many ways

resembles Scheme[1], the clean lexically scoped dialect of Lisp, which is still widely taught in top CS programs. It has first class functions, lists and lexical closures, the key functional building blocks, although it lacks the continuations and tail recursion that are important to many functional programmers.

In the spirit of dynamic languages like Smalltalk, Ruby and Python, the values rather than the variables are strongly typed. Unlike these languages, however, objects are cloned from prototypes rather than instantiated as instances of classes in the spirit of the pioneering prototype based language Self. Prototypes can have state (properties) and behavior (methods) modified on the fly. In this regard, JS is much more dynamic than its class based cousins. It is still very straight forward to do class based OO programming in JS and there are patterns and libraries for applications where this is appropriate. JS has a basic but sufficient set of base classes. Unfortunately, there is no explicit mechanism for dealing with name scopes so this means developers have to provide their own name space management. It isn't that it isn't possible to write packaged code, rather that there are problems when one includes code from a large number of sources where one can get name conflicts, especially for selectors.

Like most OO languages, input and output is done through a library, which for the browser is the DOM. While the language isn't without its share of small but irritating syntactic and semantic anomalies, it is more than good enough for serious web work. JS has been plagued by the DOM[2] and the associated browser dependence and defect levels. Fortunately, browser standardization and improved test suites for HTML, CSS and DOM practices have significantly improved cross browser programming though it is still frustrating at times.

ECMAScript - The JavaScript Language Standard

Standards are an important way to ensure portability of programs. Recently, the ECMAScript (ES) 3.1 and 4 efforts were merged with the near term focus on the delivery of the more modest ES 3.1 standard. ES 3.1, expected out in 2009, is being designed to ensure that existing web pages don't break.

Major features tentatively approved include:

- Getter/Setter properties with both syntactic support in Object literals and programmatic support via new property definition functions
- The ability to query and set property attributes such as "readonly" and "dontdelete"
- The ability to "seal" objects in order to prevent modifications or additions to their properties
- Ability to query the prototype of an object
- Function to create an object with a specified prototype
- The ability to use previous reserved words as the names of object properties and to access them using "dot notation"
- Array-like indexing for access to individual string characters

[1]Unfortunately current implementations don't guarantee optimization of tail recursive calls but there is also nothing in the language standard which prevents these optimizations.
[2] Dumb in Dutch, the Damn DOM to many

- Built-in support for JSON
- The “array extra” functions first introduced in Mozilla’s implementations and subsequently widely copied by others
- A function to trim whitespace from strings
- Support for parsing and creating ISO format date strings
- Ability to query the declared name of a function
- The ability to “bind” the “this” value or arguments of a function object to specific values
- The ability for a programmer to opt-in to using a “cautious/strict” subset of the language that performs additional error check and restricts use of some error prone or insecure features of “ES3”
- Numerous specification bug fixes and clarifications intended to improve interoperability among implementations

ES 4 was very ambitious and advocated the addition of a lot of complex Java-like machinery such as manifest types, generics etc. which have already proved difficult for professional Java developers. One hopes that the ES efforts beyond 3.1 will focus on small critical issues like proper name spaces and reducing language ambiguities rather than complicating the language. It is important to note that many issues such as the DOM, security, and potentially even name spaces are at least in part outside the scope of the ES language standard.

SERIOUS PROGRAMMING IN JAVASCRIPT

One of the major factors limiting developers was a lack of the industrial strength programming environments they had come to love. It is important to know that early JS developers assembled a toolkit that is still in wide spread use today. JSLint and JSUnit are two notable stalwarts in this toolkit. Agilists may even argue that the lack of tools forced JS developers to apply test first programming[3] rather than leaning on their IDE and its debugger. While there has been a lack of good tools for many years, this has been improving substantially of late. Eclipse based tools such as Adobe Flex Builder, JSDT, MyEclipse etc., and MS Visual Studio provide IDE support for developers.

Since the arrival of Google Mail and Ajax there are some good libraries such as DoJo, jQuery and more recently the highly polished EXT JS library. These libraries provide for Class OO programming; support for XML/HTML; support for Ajax XMLHttp, REST or, JSON; UI widgets and numerous utilities for common tasks. Prototype + Script.aculo.us, DoJo, jQuery, MochiKit, YUI and Ext JS etc. eliminate the need for building such things oneself. OpenAjax has over a hundred members collaborating to harmonize libraries and their component models.

Occasionally disconnected operation is one of the major challenges for browser based applications, which have been until recently dependent on an always available Internet connection. Mobile applications in particular are constantly disconnecting and connecting. Gears is a first attempt by Google to enable simple web programming

[3] “Debugging Sucks, Testing Rocks” http://googletesting.blogspot.com/

on the client using an embedded web server and SQL DB. Microsoft and others are working on more elaborate Sync frameworks which would enable syncing application and server data.

Sun Lively - Using JavaScript as a Real Programming Language

Lively is billed as a WebOS implemented in JavaScript. Lively leverages the impressive Etoys Squeak and Morphic graphic framework to deliver applications on a JS + SVG platform. It provides an open live programming experience in which the running code can be edited on the fly. Unlike traditional web pages, lively pages require no cryptic coding CSS or HTML; rather they are live JS programs. Lively includes a Squeak style IDE and an interface to CVS for team development. Independent of whether one likes the back to the future IDE, this large application seems like an amazing engineering feat for those who have been burned by even a small amount of JS. The authors' account [2] of building Lively in JS provides a glimpse at how one can approach a large JS effort with a small team and the challenges of JS.

GOOGLE OPEN SOURCE V8 CHALLENGES THE INDUSTRY

With V8 Google has thrown down the gauntlet to the major platform and VM vendors, challenging them to improve their offerings. For many years industrial strength dynamic VM technology has been proprietary to major corporations such as IBM, MS and Sun. Google V8 will be developed in the open under a BSD license ensuring that researchers, hobbyists and competitors can watch and benefit from the V8 code base. Of particular interest to dynamic language advocates are the engineering design decisions which strongly support JS as a dynamic language. The benchmarks in particular bring smiles to the faces of many savvy Smalltalk developers.

APPLICATION DEVELOPERS – TIME TO TOOL UP FOR JS

JavaScript, like most commercially successful languages, has lots of gotchas and irritations; it definitely isn't perfect but it is certainly good enough for serious client side web application development. We anticipate that it won't be long until a server side application engine running V8 appears, allowing JS developers to use their language anywhere in the cloud. Professional Developers and Educators alike need stop sticking out their tongues and embrace the ubiquitous dynamic language. It can be both fun and productive! For fun try programming webapps your iPhone in JS and you may soon be twittering your friends.

REFERENCES

[1] Douglas Crockford, JavaScript: The Good Parts, O'Reilly

[2] Tommi Mikkonen and Antero Taivalsaari, Using JavaScript as a Real Programming Language

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.