JavaScript creator ponders past, future

Brendan Eich created JavaScript, the popular scripting language being used to liven up Internet applications. Coupled with XML, JavaScript has become part of the AJAX (Asynchronous JavaScript and XML) technique popular in Web development. InfoWorld recently met with Eich to talk about JavaScript: where it has been and where it is headed. Eich, who serves as chief technology officer at Mozilla, also commented on other languages and about working with Microsoft in developing standards.

[Take a tour of today's Internet-oriented dynamic scripting languages.]

InfoWorld: As I understand it, JavaScript started out as Mocha, then became LiveScript and then became JavaScript when Netscape and Sun got together. But it actually has nothing to do with Java or not much to do with it, correct?

Eich: That's right. It was all within six months from May till December (1995) that it was Mocha and then LiveScript. And then in early December, Netscape and Sun did a license agreement and it became JavaScript. And the idea was to make it a complementary scripting language to go with Java, with the compiled language.

InfoWorld: What's the difference between ECMAScript and JavaScript, or are they one and the same?

Eich: ECMAScript is the standards name for the language. Its number is ECMA-262 and it's a name that the standards body came up with because they couldn't get anybody to donate a trademark that was agreeable to all parties. So there's an issue with marketing the programming languages.

JavaScript would have been the ideal name because that's what everyone called it and that's what the books call it. Microsoft couldn't get a license from Sun so they called their implementation JScript. So ECMA wanted to call it something and they couldn't get anybody to donate or they couldn't get everybody to agree to a donation of the trademark, so they ended up inventing ECMAScript, which sounds a little like a skin disease. Nobody really wants it.

And so you have this funny [situation] where you have a standard with a funny name or number and then various implementations that have trade names. And the trade names don't have a huge value, except JavaScript is the common name. It's in all the books, it's what people say. It's what people refer to on "Saturday Night Live."*[Editor's note: JavaScript once merited mention during a skit on the show.]*

InfoWorld: What was your main goal in developing JavaScript?

Eich: The idea was to make something that Web designers, people who may or may not have much programming training, could use to add a little bit of animation or a little bit of smarts to their Web forms and their Web pages.

So it's 1995, the Web is very early. HTML was 3.2, I think, or something like that. People did not have

much programmability. Java was coming along at the same time but it required you to use a high-powered programming language and then run a compiler and put your code into a package that became an applet that was part of the page but it was in a little silo. It was kind of walled off.

And it was hard to do -- it was for professional programmers. It was for the high-powered real estate virtual tour or something like that. Whereas JavaScript was just a little snippet you could write, you could copy somebody else's, you could learn as you went. You didn't have to learn the whole language to use it and you could buy it by the yard.

That idea was very strongly held by Marc Andreessen and myself. Bill Joy at Sun was the champion of it, which was very helpful because that's how we got the name. And we were pushing it as a little brother to Java, as a complementary language like Visual Basic was to C++ in Microsoft's language families at the time. And it took off. We got it out in time.

[Get inside the minds of the leaders of open source with InfoWorld's open source roundtable.]

InfoWorld: Why do you think JavaScript has become so popular for Web programming and did you ever think that it would become this popular?

Eich: Because of the reach of the Web. The Web is the biggest, broadest platform, maybe not the best platform and people sometimes have to reach for the Flash Player or Microsoft's putting out something called Silverlight, which is competing with Flash. But the Web is what everybody wants to use. You get a browser and an Internet kiosk, you can use it. So if JavaScript is the scripting language of the Web, it has the most reach.

And as computers got faster and JavaScript got better and other parts of the browser programming model -- the so-called libraries or APIs got better, the document object model got better -- people found they could do things they hadn't dreamed of.

I will say that we didn't think it would go this far because we were underinvesting in it at Netscape. I was kind of the lone gun for too long. But even in 1995 there were these great demos that would have held up today as sort of early AJAX almost. They didn't have all the APIs for doing background I/O to a server. They had the reload pages but they would do it in the hidden frame and you could still do background communication that way. This was a proof that JavaScript could be used by itself with HTML and images to do amazing apps and only a few knew it was very esoteric.

And then, of course, it was much, much later that we had things like Google Maps and Google Mail and all the other AJAX apps people know about.

InfoWorld: I often hear criticism that JavaScript is kind of hard to work with. Would you agree with that or why do you think people feel that way?

Eich: There's a general bad odor. [lingering from the browser wars and Internet Explorer's Document Object Model (DOM) being different]. The language implementation between the browsers agrees generally and pretty well compared to the library code, the so-called document object model. Other

browsers' implementations tend to agree [with the DOM] because they came later and had a standards orientation.

I had an early version of the DOM that was somewhat proprietary; not all of it got standardized in the W3C and they never changed it for many years. IE6 was in 2001 and IE7 wasn't for five years, I guess. And so there was this big gulf between the browsers, and when people come to JavaScript to program the browser they want the same APIs and they don't get them.

Now what helps make this less painful are these JavaScript libraries, which are pure JavaScript code that makes this higher layer of code that masquerades and makes all the differences go away. And those are becoming quite popular. Those are the so-called AJAX libraries.

InfoWorld: How would you compare JavaScript to some of the other scripting languages -- Ruby, Perl, PHP, most of which are server-based whereas you're client-based. What were the other major differences?

Eich: I'd say, especially Ruby and Python and Perl have had great community, open source community development over the years where they don't have to worry about being in every browser and working on arbitrary Web pages across billions of Web pages. Instead, they can say, "Here's the new version, port your code." They can bind very tightly to the operating system, which are the server APIs. So they can be really tight and useful and very powerful in their domains, but their domains don't have to have code-sharing in the same ubiquitous way that Web pages are shared.

JavaScript's the only language that is so widely interoperable. When you take a Python script written for Linux, it uses certain system calls or certain library code to do process management or input/output. Porting that to Windows means changing some of the code. You don't have the same API in all operating systems. Whereas with JavaScript it's got to be the same everywhere because browsers don't care about operating systems.

InfoWorld: Have you heard of ARAX (Asynchronous Ruby and XML) and APAX (Asynchronous Python and XML), which will enable you to use Python and Ruby in the browser instead of using JavaScript? Is that something you would support?

Eich: Yes. I think that's great, and I think we will support other programming languages. The problem is that if other browsers don't all jump on the same bandwagon, then the developers won't be able to count on it in this broad Web content way that reaches all browsers.

Some of these techniques, like HotRuby, actually translate Ruby into JavaScript. So they use JavaScript as kind of an underlying assembly language, if you will. And HotRuby is the source language you write your ARAX app in but you don't have to have any browser extension. It just gets translated and runs its JavaScript and it runs pretty fast. So there's hope for multiple language support one way or another.

What will be interesting will be to see how it's standardized because until it's really in all browsers, developers can't count on it.

InfoWorld: I think APAX and ARAX are just for Silverlight. That's the only thing they're useful with at this point.

Eich: You could use HotRuby for ARAX, so that technique is possibly applicable to Python too.

InfoWorld: When will JavaScript 2 be out and what are the main improvements in it? Are JavaScript 2 and ECMAScript 4 the same thing?

Eich: Let me answer the last one first. We in our new fourth edition of the ECMA standard are specifying what the version numbers people use on various designators mean. We're saying that JavaScript 2 and ECMAScript 4 mean the same thing. There's no value-add, there's no feature sneak. JavaScript 2 and ECMAScript 4 are equated.

Let me go back to the first one. When will it be out? We're not sure. We'd like to standardize after we implement because we don't want to just rubber stamp a standard. We'd like to have several different implementations in some kind of open source [format] in which we build community-tested quality that we can evaluate to see if the specification led them to interoperate. So that means that we should get the implementations going as we're writing the spec and the spec is being constructed right now. It's getting done, it's getting more solid. So we'll have implementations I think by the end of this year that people will be able to play with. They won't be in Firefox 3. They won't be in probably the 3.1 that we've talked about doing, but they might be in our [nightly] builds, our trunk builds. It'll be like a draft version of the spec, so we might call it JavaScript 1.9 or 1.99. We don't want to get people to confuse it with what becomes the final spec, but we have to be able to test it with real programmers and get usability feedback. That's critical.

InfoWorld: So what are some of the features in JavaScript 2?

Eich: We're trying to address programming in the large, because no one thought JavaScript would be used at the wide scale it is. Not just reaching lots of people on the Web, but large applications like Gmail.

To write large code, you don't just want this little snippet language that I made easy for beginners to start buying by the yard. You want strong APIs, ways of saying -- this is my module and this is your module and you can throw your code over to me and I can use it safely.

To do that, you need more than is in the current third edition of the language. You need some kind of a way of talking about the types of values, some way of talking about interfaces or the arguments that can go into a function in the result and come back. Can they be of any type? Can they be of the wrong type? Or can they only be of certain types?

We'd like to be flexible about this and not make things painfully static in a fixed way like Java does. We'd like to say, "Here's the patterns people use in their code today. Here are the latent types they use for their data. Their data types look like these JSON objects, these trees of data." We can write down very concise descriptions of those in the fourth edition using the [type] system, and then you can have the implementation automatically check that the shape of the data matches. So you can make sure that

somebody isn't sending you the wrong data by accident or maliciously -- you can make sure that certain names can't be changed to mean something other than what they should mean.

Generally we're improving the integrity of the programming language because you've heard of mash-ups. JavaScript is very mutable, it's very flexible. You can rename things. That's good except when it's bad, and so there should be ways for programmers as they figure out what their modules are and they know what their contract with their neighbor is, to lock things down.

It doesn't mean everything gets locked down; it's still JavaScript. You can start buying by the yard, you don't have to declare the type of any variable. You can just start writing code and snippets.

InfoWorld: So is this the first major upgrade in what -- 13 years?

Eich: Nine years.

InfoWorld: What is this dispute with Microsoft over ECMAScript 4 and JavaScript 2?

Eich: Well, we're trying to work together in the ECMA working group and it's going OK but we have a split committee. So part of the committee is focused on what's being called ECMAScript 3.1 and the idea there, at least the idea that I think everyone in the committee agrees with, is that it would be a small improvement to the third edition, the last edition from 1999. That it would fix known bugs, it would maybe add a few standards that are already implemented in three out of four browsers, maybe it would add a few more things. But it has to be pretty small because, for one thing, it's supposed to be a subset of the fourth edition. It's not supposed to have anything in it that's not in the fourth edition. And it's also supposed to be done sooner, so if they keep adding things to it, it'll never get done.

I'm hopeful that it does come through because it would be an improvement. In many ways in Firefox we've already moved way beyond what's in there. There are a few things in there that might be good to add to Firefox, so I'm not saying we thought of it all before, but some of it is just based on work we've already done. And so we don't want to take a step backwards and do only 3.1 -- that's why we're doing 4. Microsoft seems much more focused on 3.1, and that's their choice.

InfoWorld: What is Project Screaming Monkey? Apparently it's some kind of a scripting engine for Internet Explorer?

Eich: Yes. Internet Explorer is a very flexible platform, and you can add scripting engines. They made it possible to add Visual Basic script and so they allowed other people to add Perl and Python; ActiveState did that. So we've commissioned Mark Hammond who worked at ActiveState to do active scripting glue for Tamarin, which is the Adobe-donated virtual machine for ActionScript, an ECMAScript implementation, that's in the Flash Player.

It's possible that if you're a developer and you get this, or if Adobe were to distribute this active scripting glue that Mark wrote, that IE would be able to support JavaScript through the Flash Player. It wouldn't need to have native support for JavaScript 2, it would get it just for free because Flash is

widely distributed. Now I don’t know if Adobe will do that. It’d be good if they did, in case Microsoft does not ever get around to supporting JavaScript 2.

And frankly, if Microsoft does a great job on JavaScript 2 and knocks it out of the park, Screaming Monkey doesn’t need to exist. It’s really just a way of getting browsers, starting with IE, to be uplifted to JavaScript 2. Because a lot of people worry -- well, if Mozilla and Opera say, "Do JavaScript 2, but we don’t know when Apple is going to do it and Microsoft says they won’t," then how can anyone ever use JavaScript 2?

One answer is -- you can see this already on a Web site called ECMAScript4.com: Someone has just released a translator that takes draft fourth edition JavaScript 2 code and translates it into JavaScript that works in today’s browsers. That’s one tool you could use to use the new language soon. On those browsers that don’t have support for it natively, you translate to JavaScript. Those that do, you just ship the primary source straight through, say, Firefox. The other way to do it is Screaming Monkey and that could be applied to other browsers than IE, but IE is the one that most people use. So if we can uplift IE to support JavaScript 2 without Microsoft’s cooperation, and it’s part of their platform to support other scripting engines, then why not?

RELATED TOPICS

-