Issues in making CF9 ECMAScript and XML Compliant

Posted At : June 24, 2008 2:02 PM | Posted By : Elliott Sprehn
Related Categories: technical

Lately there's been a lot of talk about what to do in CF9 and what features developers really want.

For some time there's been this push to make CFScript more powerful, and fill in it's deficiencies compared to CFML. Now that we close on CF9, and developers have had a taste of the new operators in CF8, I'm starting to hear about making CFScript "ECMAScript Compliant" from quite a few people, or even more extreme, implementing AS3 instead of CFScript. Another request I'm starting to see pop up is making CFML "XML Complaint." These requests seem to stem from a significant amount of confusion about what those compliances really mean.

So lets start with CFScript...

CFScript can **never** be ECMAScript compliant. ECMAScript is a very long, and quite specific standard. It specifies everything from how math should work, to how expressions are evaluated, to how parsing should happen, and quite a bit more.

So what would it take to make CFScript actually ECMAScript complaint?

Well that would involve a completely different prototype based object model, different casting rules, different math rules, different function names, different binding rules, different operators (+ for strings, & is not valid), different type names, different syntax for structures... which wouldn't be structures, but rather Objects, a completely different closure and context system in the runtime, anonymous functions, optional semicolons, different regexp syntax, ... the list goes on.

All of this is totally at odds with CF's current runtime and how it works, and what CF code compiles down into. If it was done then every existing CF application would break. So what if we did it by doing something like embed Rhino? While this would give us JavaScript, it could never replace CFScript. And assuming we did do this, now you'd have two radically different object and language models to deal with. The CF one, and the ECMAScript one.

Okay, so what about XML?

XML compliancy would require very significant language changes as well. When many people think of "XML Compliant" they think of the short hand close tags (the NET feature) that XML uses. This is really a quite small set of what XML really is though.

So what would it take to make CFML XML compliant?

First we'd need to fix cfif, cfelse, cfelseif and cfreturn so that we weren't putting expressions in the

tag itself, but rather in attributes. This would make the new struct and array syntax rather odd, and make assigning strings even more odd:

<cfset expr="variables.string = "Why do you want XML?&quot"/>

Doing this is quite against best practices for XML schema design, so something like <cfset name="variables.string" string="Why do you want XML?"/> might work better, but now we're really deviating from old style syntax.

The next issue is that XML documents all need a root element. So we'd need something like <cfpage> that wraps every cfm file (cfcomponent would work for cfcs), to meet that requirement.

Then we need to deal with cfimport and prefixes. These are namespaces in XML, so we'd need to alter the way you import tags significantly. Essentially removing cfimport and replacing it with xmlns attributes on the root.

Now there's issues of dealing with code that uses > and < in script and expressions, so we'd probably be required to wrap all script code in cfscript in CDATA blocks. Worse is that if we really wanted valid HTML entities, they'd need to be double escaped to something like . We'd also end up with & to concat strings, provided we didn't ditch that syntax and go with the ECMAScript redesign discussed above.

Conclusion

I hope this makes it a little more clear what reformulating CFML and CFScript into these standards would mean. The result would be far from the CF we know (and love) today, and not even remotely backwards compatible. I'd imagine it would take a significant engineering effort to totally rewrite CF like this, and what we'd end up with wouldn't really be CF anymore, but some kind of new language entirely. I'm not going to argue if the language would be good or bad, but it certainly wouldn't be anything like CF8 on almost every level from syntax, to semantics.

So here's my big question... what's so wrong with CF as it exists now? :)