Contribution : Position Statement to TC39-TG1 Regarding the Evolution of ECMAScript

Author : Microsoft

Date : November 7, 2007

ECMAScript is the official name of the programming language, commonly known as JavaScript™, which is implemented by all major web browsers. ECMAScript is arguably the world's most widely used programming language with millions of web pages containing ECMAScript code. Although the overall programming experience for the web platform consists of the Document Object Model (DOM) and associated APIs as well as the programming language and runtime, this document only discusses the language portion of the platform.

The emergence of Ajax has renewed interest in the evolution of ECMAScript. Originally ECMAScript was primarily used to implement simple dynamic behaviors within web pages such as changing the label of a button when it is clicked. More recently, with the growing popularity of technologies such as DHTML and Ajax, ECMAScript is increasingly used to implement complex interactive user interfaces within web browsers and to implement application frameworks that simplify the creation of such user interfaces. This change in usage is being used by some TC39-TG1 participants as a justification for changes that alter the fundamental nature of ECMAScript. The proponents of such changes are developing a complex language that is arguably more similar to Java than it is to the original JavaScript language. This new language is commonly referred to as "ECMAScript 4" (alternatively "JavaScript 2" or "ES4"). Microsoft agrees that it is time to evolve the ECMAScript standard, but we do not see any need or justification for a radical redesign of the language. We disagree with the priorities currently guiding "ECMAScript 4" development and as an alternative offer a set of principles that we believe should guide the evolution of ECMAScript.

Guiding Principles for Evolving ECMAScript

Microsoft believes that the evolution of ECMAScript should be guided by four prioritized tenets:

1. **Ensure Stability** of the existing web platform ecosystem. The evolution of ECMAScript must give highest priority to ensuring continuing function of existing web content and applications. Over the last 15 years, the web has become the primary global computing platform tying together hundreds of millions of sites. The highest priority consideration for evolving any core web technology such as ECMAScript must be ensuring the continuing validity and function of existing web content that uses ECMAScript, and allowing that content to continue to be used in mashup applications, etc. In particular, the continuing operation of the deployed base of ECMAScript 3rd Edition (ES3) scripts, script libraries/frameworks, tools, and browser implementations must not be sacrificed.
2. **Ensure Interoperability** among web browsers. Strong interoperability across web browsers is essential to the World Wide Web. The developer of an ECMAScript-based web

application does not get to choose which ECMAScript implementation will run their application; the user implicitly chooses that when they choose their preferred browser. Web developers need to have confidence that ECMAScript code that works with one browser will work identically on all other browsers including those embedded within mobile devices. The definition of ECMAScript has been stable for eight years, and language implementation differences and incompatibilities between the most popular web browsers have either been eliminated or are at least well understood by the user community; this stability was an essential enabler for the recent emergence and broad adoption of Ajax technologies. However, the current ECMAScript standard does not document any of the *de facto* interoperability conventions reached during this period of ECMAScript stability. Multiple independent vendors must be able to create interoperable ECMAScript implementations based purely on the specification. We believe that a new version of the ECMAScript standard should codify the *de facto* interoperability conventions that have emerged and should resolve remaining ambiguities where inadequate specification has resulted in poor interoperability among browsers.

3. **Enhance Performance** of ECMAScript programs. Performance issues of deployed applications cannot be fixed by adding new language features. The usage patterns for ECMAScript have significantly changed since the publication of the ECMAScript 3rd Edition specification and browser implementations are exhibiting performance problems when they are used to execute complex Ajax frameworks. We believe that improving performance does not require radical changes to the ECMAScript language. Instead, the implementations of the language within browsers need to be modernized. Browser implementations can radically improve their performance if they apply well established and widely known dynamic language implementation techniques. Major language changes that speculatively promise improved performance for future applications should not be considered unless such implementation techniques have been proven inadequate. It is also important to avoid introduction of new language features that degrade performance.
4. **Enable Innovation** in the language. ECMAScript needs to embrace innovation as it evolves to better support new applications on the web. New ECMAScript features must be technically sound and should enable innovative solutions to web application development problems. Unfortunately ECMAScript was not designed with language innovation or even evolution in mind. This makes it very difficult to introduce new language features without destabilizing the web. Microsoft believes that after resolving existing stability, interoperability, and performance issues top priority should be given to ECMAScript enhancements that will support and enable the stable evolution of the language. Once such a foundation is in place ECMAScript should be able to accommodate a regular stream of innovation.

In the interest of the web, Microsoft believes that any ECMAScript evolution must follow these principles.

Current Status of ECMAScript Standardization

The ECMAScript specification is published by the ECMA International standards organization. The most recent official version is the *ECMAScript Language Specification, 3rd Edition* published in December 1999. The ECMAScript specification is authored by the ECMA TC39-TG1 Task Group, currently chaired by Brendan Eich.

In the eight years since the release of the 3rd Edition specification, ECMAScript has continued to grow in importance as one of the key technologies of the web. During this period ECMA TC39-TG1 has developed several ancillary specifications relating to ECMAScript. However during this period the task group has not maintained or enhanced the actual ECMAScript specification. Known errors have not been corrected; ambiguities have not been clarified; conflicting interpretations have not been resolved; new features introduced by implementers have not been added to the standard. In many cases the resolution of such issues needed to achieve interoperability among implementations have been obtained by informal understandings among language implementers or simply by copying the implementation decisions of the most widely used implementations. The result is that there is no complete specification for ECMAScript as it is most broadly used on the web.

Rather than resolving interoperability issues and codifying widely used enhancements into a new version of the ECMA standard, the major focus of TC39-TG1 for the last several years has been the development of a major redesign of the ECMAScript language. This new language design is generally known by the unofficial name "ECMAScript 4" (or "ES4") because its supporters hope that its specification will ultimately be published by ECMA as the official 4th Edition of ECMAScript. As of October 2007, ES4 is still under active design and development. The language design remains incomplete and still has many unresolved issues. ES4 is essentially a new language that incorporates many innovative but unproven features. It is a complex language that includes a Java-like class and interface based object model; a hybrid type model that combines dynamic typing with an a optional static typing system that incorporates nominal, structural, and generic types; both Java-like packages and XML-like namespaces; multi-methods; and many other new features. "ECMAScript 4", as currently envisioned, is a radical departure from the language defined by the current ECMAScript 3rd Edition specification.

Reservations Regarding "ECMAScript 4"

Microsoft believes TC39-TG1's current effort to create "ECMAScript 4" is seriously flawed and will not be beneficial to the web. Replacing ECMAScript 3rd Edition with the current "ECMAScript 4" design would be a major disruption to the web that would destabilize existing web application, create significant new interoperability issues, and delay performance improvements that would benefit existing web applications. We believe that the root problem is that the task group in creating "ECMAScript 4" has not followed the evolutionary principles of: ensure stability, ensure interoperability, enhance performance, and enable innovation. While the task group generally acknowledges these principles, it has prioritized them much lower than the goal of redesigning the language to support large scale programming projects and complex applications.

ES4 will destabilize the web by breaking some existing applications. Even though the proponents of "ECMAScript 4" have made a good faith commitment to fully supporting existing ECMAScript code and applications we believe that the history of programming languages adequately demonstrates that that this effort will initially have unsatisfactory results. The biggest challenge for any language designer is dealing with the interactions that occur between language features. Often such interactions are unintended and remain undiscovered until a language has had significant usage. If "ECMAScript 4" is introduced as a replacement for ECMAScript 3rd Edition it is inevitable that there will be unanticipated feature interactions between legacy ECMAScript 3rd Edition features and the large number of new ES4 features. These interactions will in many cases have a destabilizing effect upon existing ECMAScript code. Some of these destabilizing interactions will occur even if a web page or ECMAScript library uses none of the new features of "ECMAScript 4". Even though a specific program uses no new features, the implementation of each language feature must try to take into account all possible feature interactions whether or not specific features are actually used by a particular program. These implementation considerations can easily result in subtle behavior changes that break existing programs.

ES4 will be detrimental to interoperability. New or significantly enhanced languages inevitably have significant initial interoperability problems between independent implementations. This occurs because no language specification is ever complete enough or sufficiently unambiguous to guarantee that every language implementer will produce identically performing implementations of all the features and subtleties of a language. Because of the unintended feature interaction problem, large languages tend to have many more initial interoperability problems than simpler languages. These sorts of problems occur with any new language implementation and typically take several years and multiple revisions to resolve. If "ECMAScript 4" replaces ECMAScript 3rd Edition it will take many years for web browser to again achieve the level of interoperability that exists today.

ES4 will not improve the performance of existing ECMAScript applications. Many of the new features in "ECMAScript 4" have been justified on the basis that they will enable programmers to create higher performance applications. Even if this assertion is correct, it provides no immediate performance benefit to the massive body of existing ECMAScript code. To gain the hoped for benefit, existing code would first have to be rewritten to utilize the new performance enhancing features. Microsoft believes that the adoption of "ECMAScript 4" is likely to delay improvements to ECMAScript performance. As ECMAScript implementers focus on implementing and debugging the new "ECMAScript 4" language, their attention will be diverted away from making improvements that address the performance issues of deployed applications written using the existing ECMAScript 3rd Edition language.

ES4 contains too much unwarranted and/or unproven innovation. The ES4 effort is attempting to design a new language to replace ECMAScript 3rd Edition. In many ways, this new language is intentionally more similar to Java or C# than it is to ECMAScript 3rd Edition. Portions of the language design, such as its hybrid type system, are still topics of active computer science research. While there are certainly some developers who might prefer such a language others are equally happy with the existing ECMAScript language foundation. Regardless of such preferences, there is simply no current web application development issue that necessitates such a wholesale redesign of the language or justifies the inevitable disruption of the web that would result from its deployment.

Refocusing the 4th Edition of ECMAScript

ECMAScript's importance to the web is greater than ever. The web needs a 4th Edition of the ECMAScript Specification that codifies conventional usage on the web, corrects known issues, and provides innovative solutions to real web development problems. It needs to fulfill these goals without destabilize the web, decreasing interoperability, or interfering with ongoing performance improvements.

Microsoft does not believe that TC39-TG1's current "ECMAScript 4" design can meet these requirements. It is taking ECMAScript in the wrong direction. Instead, we believe that TC39-TG1 should adopt the four guiding principles of stability, interoperability, performance, and innovation and follow them as it refocuses its efforts on the development of an ECMAScript 4th Edition specification that can be broadly deployed this decade.

The first step in this redirected effort should be to create a draft specification derived from the current ECMAScript 3rd Edition specification that corrects all known errors and captures the consensus interoperability contract of the existing web browser implementations of ECMAScript.

The next step should be to incorporate into the draft specification high utility features that have already been proven as non-standard extensions to existing implementations. Examples of features that should be seriously considered include Mozilla's "array extras" and JSON support. Historically ECMAScript was not designed to be "future proof". It is very difficult to add new features to the language without potentially disrupting at least some existing programs. In the next edition, top priority should be given to new features that enable the future non-disruptive addition of more innovative, features.

Finally, the task group should look for and consider for inclusion targeted new features that support known best practices or address critical web development concerns such as mashup security and code modularity.

Participation in TC39-TG1 needs to be broadened to better represent the actual users of the ECMAScript programming language. The current membership of the task group predominately consists of language implementers. There is very little participation by the developers of major websites, complex web applications, or the frameworks and libraries that support such applications. Microsoft strongly encourages all stakeholders, and in particular web framework and application developers, to join ECMA International and participate in the ECMAScript task group as it develops the specification for the 4th Edition of the ECMAScript Specification.

Conclusion

Microsoft recognizes the key role that ECMAScript plays for the World Wide Web. We believe that careful stewardship of the ECMAScript specification is critical to the web. To this end, we have enumerated four key principles that we believe should guide the on-going evolution of ECMAScript. We believe that following these principles are essential for any meaningful effort to create a new edition of the ECMAScript specification. Unfortunately, the current "ECMAScript 4" effort significantly strays from these principles. That effort needs to be refocused on achievable goals and to adopt the four principles as its guidance. We have presented an alternative plan that would accomplish this.

Microsoft is committed to the future ECMAScript and intend to work diligently within the ECMA process to ensure that ECMAScript evolves in a manner that is consistent with these principles.