Minutes of the: ***Ecma TC39-TG1***

held in: ***Mountain View, CA***

on: ***27-28th of September 2007***

Table of Contents

Logistics

- September 27, 1200-1700 PDT, with dinner at 1830, at Sundance Mining Co. (regrets only)
- September 28, 1000-1700 PDT
- Mozilla Foundation, building "S"
  - 2121 Landings Drive, Mountain View, CA 94043 (map link)
  - `<script>` conference room

Attendees

- Igor Bukanov, Mozilla Foundation
- Douglas Crockford, Yahoo! (Thursday only)
- Jeff Dyer, Adobe Systems
- Brendan Eich, Mozilla Foundation
- Cormac Flanagan, UCSC (arriving ~1pm Thursday due to teaching)
- Lars T Hansen, Adobe Systems
- Waldemar Horwat, Google (invited by Convenor)
- Pratap Lakshman, Microsoft
- Chris Pine, Opera Software

Agenda

- Finalize ES4 proposals

- We propose that the proposals pages (including discussion pages, clarification pages, etc) are frozen after this meeting, except that it is allowed to add references to Trac tickets at the top of the proposal pages. All issues should be logged in Trac and cross-referenced.
- Unresolved proposals. Each one of these should have a smiley at the end of the meeting, or else be deferred until ES5.
  - maintenance of es3, a proposal to refocus TC39-TG1 on the Maintenance of the ECMAScript, 3rd Edition Specification
    - JScript Deviations from ES3
  - generic functions, a facility for multimethods with type dispatch
  - bound this, a proposal to allow this to be bound to methods on plain objects
  - self type, a proposal to allow the type Self (exact name TBD) in structural object types
  - oopKoolAid, an extension to the Map proposal for object-oriented protocols
  - `like` types and the `wrap` operation. These ideas are described in solution 9. A core implementation is at `tests/micro/option9.es` (under monotone and here) and the underlying ideas are formalized in The ValleyScript writeup.
- Tickets categorized as "proposals". Two categories, substantive language issues and simple features. The following are the subtantive ones; each one should be moved to spec or refimpl by the end of the meeting, or otherwise resolved. The feature proposals are flagged as "minor" in the Trac and do not have to be acted on now.
  - #10: Where does C.prototype.constructor.name get set?. A proposal to have a "name" property on functions, probably should be rejected in favor of the meta-object proposal.
  - #53: Should we clean up the grammar to make string and regexp literals more similar?.
  - #84: A standard "length" property on classes?.
  - #90, #205: A proposal to allow the use of `public`,`private`,`protected`, and `internal` to be used in constructing `Name` objects, and to be included in explicitly constructed namespace sets, which are then used in privacy-obeying iteration and in the reflection protocol.
  - #103: RHS of 'is', 'to', and 'cast' as a general expression. Previously discussed and rejected but reopened after new use cases came to light.
  - #121, #127, #162: open type system issues that are probably resolved by the introduction of \`like\` and \`wrap\`
  - #163: drop 'default' from 'switch type'. A simplification.
  - #165: Simple sugar for allocation of structurally typed record data. Previously discussed but no conclusion has been recorded.
  - #166: Private constructors are useful, common, and need to be supported. A proposal to allow namespaces other than "public" on constructor functions to support common idioms like singleton objects and static factory functions.
  - #167: When are redefinitions allowed?. A clarification issue on redefinitions of fixtures.
  - #210: Allow interface methods to have other namespaces than "public". A proposal to remove a future-proofing restriction inherited from AS3.
  - #213: Update Unicode requires distinguishing between ECMAScript string data and encoded data.
  - #214: Catchall semantics.
  - #215: Minimum requirements for proper tail calls. Situations in which any conforming implementation must honor tail calls.
  - #218: Union types should be unordered and should require an exact match
- Proposal walkthrough. The following are major issues that came up during a pre-meeting walkthrough. We should probably go through all the proposals if we have time, but the following are highest priority.
  - #211: Abandon ByteArray.
  - #212: Abandon JSON support.

        - #216: Abandon "use numbertype" except for "decimal"; adopt uint-specific arithmetic operators instead. A late proposal to remove some complexity.
- Spec issues.
    - Block access to the spec wiki because there's too little that's current?
    - Review of library draft spec
        - A rough draft of the library spec is available in spec/library.pdf in the monotone repository. It is currently undergoing review by Brendan and will be subject to bugfixing over the next couple of weeks. The plan is then to release it to the es4-discuss readership for preliminary review. Your thoughts on this matter are solicited.
- Schedule / how to do the spec
    - Orientation by Convenor/Editor on the schedule for the spec work.
    - How to write it?

Minutes - 27 September

Wiki issues

- Wiki shutdown / move to Trac
    - Proposal: wiki can link to trac, but no other edits
    - BE: Wiki prone to data loss, most things should be small now. Should be OK to block wiki edits.
    - BE: We'll compensate if wiki R/O turns out to be a problem, but ok rule of thumb
    - LH: Work flow: we'll move to Trac/Spec/RI; these have equal weight if they differ; once we go to ECMA TC39 the spec becomes normative by itself
    - ® We do this: the wiki is R/O except for new links into the Trac.

- Block non-member access to the wiki
    - BE: We should close just the "spec:" namespace, the "proposals:" and "clarification:" should stay open
    - BE: Dave will change permissions
    - ® lth will email dherman

Library draft spec

- LH: there's a draft, please review (esp on the formalism used, but on whatever you like) and comment to me, it will go to the mailing list eventually
- JD: will it replace the proposals?
- LH: only after it's been reviewed against them and against various other documents (like ES3)

Finalize proposals

Unresolved proposals

- Maintenance of ES3 (also see a separate section further down, from the late afternoon session)
    - JD: What do we do with the JScript delta document?
    - PL: Things in the doc: where JScript deviates, where the ES3 spec is unclear, and de-facto agreements amongst implementations. If the browsers are in de-facto agreement in the latter category we should consider these areas for bugfixes, and in any case it would be useful to tighten up the language where there's a lot of leeway. This benefits interoperability.
    - DC: Pratap's document has a major role in informing the ES3.1 work. The 3.1 proposal is not yet ready for release, however.
    - BE: Some of these things are good to codify, some (maybe more) are not.
    - DC: In particular the implementation's ability to add arbitrary properties to any user object is a problem.
    - BE: Adding R/O properties in the prototype is also a problem because of [[CanPut]], and globals can be problematic.

- LH: Maybe good to file specific requests for tightening as tickets in Trac?
- BE: It's time to decide whether 3.1 will happen under the auspices of TG1 or not. We have 4.0 already mapped out in terms of features and timeline, and TC39 is aware of that. We are the ones who get to decide what to do. If we don't agree we may need a new activity in TC39.
- JD: It is not clear what the purpose of 3.1 is
- DC: We're including errata and tightening things, and deprecating some things. We'll propose it as the new 4.0. We'll be done in the next couple of months.
- BE: If we had a vote now then 3.1 would not survive, so it would be better if we could agree on how to handle this.
- JD: It's disconcerting that no draft of 3.1 has been forthcoming, we can't schedule or decide based on nothing.
- DC: You've seen everything we have to date. Now we have to write the spec.
- PL: Most pieces for the spec are available now, ES3 spec, errata, this deviations document.
- PL: Our spec is an edit of the ES3 source document itself
- PL: We'd love to have similar documentation related to spec conformance from other vendors.
- BE: We don't have the time to document what we've done so as to achieve interoperation for the benefits of better market share, this is extremely time consuming work.
- BE: JS has better interop than DOM and CSS
- DC: I agree, JS may be top dog in the world on that count
- BE: One thing that might be useful to adopt is the string.prototype.eval method, but we're trying to close proposals now.
- JD: How useful is it to go along for another year and then have two competing specs?
- DC: Depends on how willing you are to drop stuff from your spec that we don't need
- JD: This does not seem like a productive dicussion, we've been here before
- BE: Another thing to fix (adopt from 3.1) is automatic semicolon insertion following "return", this is a language bug, all sorts of code wants to put the return value on the next line.
- BE: I've been talking to ECMA about possibilities for including errata into ES3, maybe that's the way to go
- JD: Or an ECMA technical report consisting of the errata or the edited document
- BE: Pratap's document with very little changes could go as a TR
- BE: A TR without trying to make a new edition would be good. It's in everyone's interest not to go to war or to (ab)use the standards process to force a decision in the market. It would be better to do something lightweight like a TR to revise 3rd Ed, as long as it does not compete with ES4, but ES4 will go forward.
- JD: The current makeup of the group will likely endorse the ES4 work, so you must be aware of that, but nothing stops people from working on 3.1.
- BE: Have you had success in enlisting other organizations in participating in the 3.1 work?
- PL: Not yet.
- BE: That probably reflects the realities of the marketplace, and groups joining TG1 now must first prove their worth in the peer group before being able to pull any weight. It's unlikely that TC39 will be very happy about having two groups working on the same subject matter.
- JD: The eval method on string needs to be presented as a proposal, probably.
- JD: What do we do about the proposal "Maintenance of ES3"?
    - BE: Maybe if we could agree that conditional on agreement from ECMA about allowing something like a TR or a spec called "3.1", and provided 3.1 is not in conflict with ES4, then TG1 could endorse the work and the 3.1 work could go on within TG1.
    - JD: Well it has to be a TR or an errata sheet
    - BE: Or something new, but not something bound for ISO
    - JD: We need to either smiley-face it or axe it. We'll do it this pm.
- ® continues below, late afternoon same day

- "bound this"
    - LH: I now understand what it does. How useful is it – very or just a little?

- BE: Good question. People reinvent this as a “bind” operator all the time. The pattern is common.
- CF: It’s good for the type system too. And it’s simpler than “self type”
- LH: Three objections: it’s ugly, it’s contextual (you can’t talk about what you’re binding), and you can’t bind something arbitrary, only the current value of “this” (or “this-for-binding-bound-this” in constructors).
- ... discussion ...
- LH: I move we defer, “bind” solves enough of the problem
- JD: It’s future-proof
- BE: Resolved.
- CF: We should defer “self type too”.
- ® both proposals deferred

(Coffee break.)

- oopKoolAid
  - LH: I’m not defending it
  - BE: I think we’re done with it
  - GH: I don’t personally care
  - ® proposal deferred
- Multimethods
  - (long discussion)
  - BE: Strongly in favor
  - LH: Ditto
  - JD: Fine with me.
  - ® proposal accepted
- “like” and “wrap”
  - LH: What are the open issues?
  - CF: Some small things to resolve around when we check types and when we check values, for “like”. “wrap” is really best thought of as an operation `(x wrap T)` with some sugar, eg as annotations.
  - LH: A wrapper is still a suite of getters and setters?
  - BE: Yes
  - JD: Custom extensions as meta static functions?
  - LH: Or as generic global functions? If wrap is an operator then why should there not just be a global generic function to hang custom conversions onto?
  - JD: I’m already on the record for not liking the prefix syntax.
  - CF: “like” is a type constructor. And `like` can therefore be used in type definitions, and can be abstracted.
  - CF: the type system is a little wacky, with “like” type making strange, weak guarantees that can’t be used directly by optimization, it’s like the optimizer has a different type system
  - LH: “like” is not premature optimization, it occupies a nice middle ground between static and dynamic typing, and it’s a pleasant way of programming
  - GH: I can’t deal with the syntax either... I also don’t think it’ll be used enough.
  - (long discussion about whether like/wrap make sense, and whether “:” should mean “wrap”)
  - BE: I’m in favor (Solution 9 elaborated). Let’s implement.
  - LH: Totally.
  - CF: For sure.
  - ® proposal accepted

(Another coffee break!)

Proposal walkthrough

- #211: Remove `ByteArray`, add `byte`

- We agree `byte` is needed for usability, otherwise a thousand wrapper classes will proliferate
  - It needs to be unsigned
  - It shouldn't be called `ubyte`!
  - ® reject `ByteArray`
  - ® adopt `byte` as new unsigned number type
- #212: Remove JSON
  - JD: JSON does not seem quite done yet
  - BE: Various implementations differ substantially in how they do filtering
  - JD: What about ES4-style names (eg) that won't be serialized
  - DC: JSON wants to stay independent
  - JD: What about a simple model that throws an exception if it finds something it can't encode? No filtering.
  - BE: Skipping stuff it can't encode is a little worrisome. It's too much like silently not updating a read-only value.
  - LH: Without filtering you don't have anything
  - DC: The code on json.org has a whitelist on encoding. If something on the whitelist is not encodable it's silently dropped.
  - JD: Could we throw when something can't be encoded?
  - DC: It's possible, though often it's not what you want.
  - BE: Postel's law suggests throwing exceptions...
  - LH: It seems to be a problem that a decision of whether to encode something is made in the container, not by the object itself (which would know how to encode itself in typical OO style, or tell the client that it can't be encoded). One possibility is to throw an exception from the encoder when a value can't be encoded (a la StopIteration), or return eg undefined (not a string).
  - BE: How do we get a solid spec?
  - LH: I'll do a draft spec, and communicate with Doug, and then we'll circulate it
  - ® unresolved until more work has been done
- #216: Abandon "use numbertype" except for "use decimal"; adopt uint operators
  - BE: We don't really want these ugly operators...
  - LH: But you wanted decimal so we could use operators...
  - LH: I hate function call syntax here
  - GH: And ++ is so common it should be supported, it's super ugly otherwise
  - WH: some of the overflow checks can be optimized, but not all
  - One resolution: we agree the pragmas are hard to control, so "use uint" etc should go
  - LH: note I'm proposing that you can't put "use decimal" whereever you want, only at the "beginning of the program"
  - GH: There is no sensible "beginning of program", so restricting pragmas to that location does not make sense
  - (long discussion that lth needs to summarize, no real conclusions)
  - BE: Maybe get Mike C on the horn tomorrow?
  - ® unresolved; start over tomorrow

Unresolved proposals: the sequel

- Maintenance of ES3 again, with WH present
  - BE: Hard to see how the proposal fits in with the ES4 work
  - JD: Default is to defer the proposal
  - LH: Or give it some other, new form of status: "Related work", on the main page; work under the auspices of TG1 for the time being
  - BE: A critical issue is whether 3.1 will be compatible with ES4;
  - DC: We think special status [as "Related work" of ES4] for now is fine.
  - PL: We do not support or agree to the current ES4 proposal, either in whole or in part. We intend to continue to work with interested TG members to develop a proposal for a more incremental revision of the current specification.

  - ® The maintenance work moved to top-level node “es3update” under auspices of TG1 for the moment, this is no longer an open proposal, the work on ES4 continues.

Minutes - 28 September

Finalize proposals

Proposal walkthrough

- #216: “use decimal” etc
  - LH: I propose I take this back to the lab and think about it and circulate some email, and we can get back to it later
  - ® We do this
- #10: C.prototype.constructor.name
  - LH: Nothing like this in ES3, the meta-objects system takes care of it
  - BE: WONTFIX
  - ® Make it so
- #53: Unify escape characters in regexes and strings
  - Breaks backward compatibility
  - JD: Not very important
  - LH: WONTFIX?
  - ® Make it so
- #84: Automagically add “length” as static field?
  - BE: People doing meta-programming will be surprised when they fall off the builtins cliff. And it simplifies the language (even if it is brain damage).
  - Straw vote: 3 for special case, 4 for general functionality.
  - LH: I’d be happier just dropping it, but barring that I’d like to be consistent in the builtins, at least.
  - WH: What if we add length if the class object is callable?
  - ® Make it so
- #90, #205: namespaces
  - #90: General consensus is that one should be able to write just `new Name(private, ‘‘foo’’)` for example, and not have to use `namespace private` to reference that namespace as a value. This has backward compatibility issues for `internal`, but c’est la guerre.
  - ® Make it so
  - #205: Are namespaces for security/integrity, or are they just an organizational mechanism?
  - LH: If they’re not, then they don’t pay for themselves
  - WH: Tend to agree
  - GH: But even as organizational mechanism it has value
  - WH: What if you can’t extract a namespace from a Name, only ask if it has a particular namespace?
  - ... presentation of NamespaceSet, NamespaceSetList, “namespaces”, ...
  - LH: I suggest we implement it, spec it, and let the world try to rip it apart
  - ® Make it so

(Coffee break.)

- #103: rhs of ‘is’, ‘cast’, ‘wrap’
  - JD: intro
  - BE: only hardship is having to write “type” in front of structural type expressions
  - ... long discussion, WH thinks it’s bad to overconstrain the operator and that it would be better for future-proofing to escape to the value language ...
  - LH: It’s possible to use the meta-objects system to accomplish this.
  - ® Make it so – no change to the spec.
- #163: drop default from “switch type”

  - JD: default is weird because it has no binding
  - BE: switch type is different in other ways too
  - LH: can I leave out the ":*"?
  - JD: why not?
  - ® make it so
- #121, #127, #162
  - #162 ® is redundant/fixed now
  - #127 ® `*` is not the same as (undefined,null,Object!), because for compatibility we need to distinguish them. Thus `*` is the natural top type.
  - #121 ® is redundant/fixed now
- #165: sugar for allocating records
  - JD: I don't much care for the positional parameters
  - BE: We want positional parameters!
  - WH: So now the order becomes significant... not very future-proof.
  - BE: It already is (for-in, notably), alas.
  - CF: But if we upgrade a type to a class this is very pretty. Unlike t-at-the-end.
  - CF: But it has no data hiding.
  - LH: It's not opaque like constructors are
  - JD: What about `new RGB({r: 10, g: 20, b: 30})`?
  - LH: Yuck!
  - JD: Provided too many or too few arguments is a run-time error, I'm fine with it.
  - LH: And strict mode must catch the error if there are argument mismatches, or type mismatches
  - ® Make it so.
- #166: need to make constructors private/namespaced
  - ® It's fine to make the constructor private/protected/internal
  - ® Rules for accessibility are subtle, see the ticket
- #210: non-public interface methods
  - WH: Interfaces - Eewh!
  - JD: AS3 interfaces were as simple as possible
  - No problem with this proposal, really (except AS3 compatibility)
  - ® Make it so

(Late lunch.)

Spec discussion

- We agreed on a tentative schedule, see wiki
- We discussed the structure of the language spec, a doc is forthcoming from Lars
- The goal is to nail down the structure and some ground rules and then parallelize as much as possible

Finalize proposals

Proposal walkthrough

- #167: Redefinitions
  - Lots of discussion
  - No resolution yet
- #213: Unicode
  - See notes in the ticket
  - ® Must be resolved by next meeting
- #218: Union types
  - GH: The types T and (T) are the same thing
  - WH: What about converting to eg "String", which is really (String!,null)

- GH: Probably need some specificity ordering of conversions if we want conversions to work
    - BE: How about we handle (T!,null) as special cases for the builtin types int, uint, double, decimal, byte, Number, boolean, Boolean, String, string?. A different way of phrasing is it that if you have a union of C with null then try to convert to C.
    - ® General agreement
- #215: Tail calls
    - ® We believe it
- #214: Catchalls
    - ® We need to reflect on this a little longer