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

held in: ***Redmond (Microsoft)***

on: ***21st – 23rd March 2007***

Attendees

- Jeff Dyer, Adobe Systems
- Dan Smith, Adobe Systems
- Brendan Eich, Mozilla Foundation
- Graydon Hoare, Mozilla Foundation
- Mike Shaver, Mozilla Foundation (Thursday)
- Douglas Crockford, Yahoo! (Thursday and Friday)
- Chris Pine, Opera Software
- Lars Hansen, unaffiliated
- Allen Wirfs-Brock, Microsoft
- Pratap Lakshman, Microsoft
- David Simmons, Microsoft (Friday)
- Dave Herman, Northeastern University (Thursday and Friday)
- Michael O'Brien, mbedthis (Thursday)
- Chris Wilson, Microsoft (Thursday)
- Scott Isaacs, Microsoft (Thursday)
- Mike Cowlishaw, IBM (Thursday) (on the phone)
- Brian Crowder, Mozilla (on the phone)
- Cormac Flanagan, UC Santa Cruz (Thursday morning) (on the phone)

Agenda items

- proposals:documentation: we have two competing proposals and several loose ends
- proposals:meta_objects: uses structural types for some data, not obvious that this is a good idea
- Implementation of type checks
- Tracking refimpl and spec completness
- Tracking bugs

Day 2

- Discussion on the proposal for incremental evolution of ES3 (forenoon).

Day 3

- Spec issues
  - informative vs. normative
    - can SML be pretty-printed as normative spec language?
    - typographic and presentation separation ideas
  - generate HTML+CSS in addition to Word
    - or first, then to Word?
- Schedule/Venue

    - face to face calendar
    - bug tracking
- Detailed Work Items
    - ListExpr vs. AssignExpr for (let,yield,function) expression body-forms
    - expression closure syntax, otherwise
    - Dictionary
    - MOP function naming, at last
    - ISSUE from iterators
    - enumeration type
    - versioning
    - catchalls
    - name objects

Discussion - Day 1

Morning

Issue: There is no way to indicate certain attributes of the MOP using structural type, as specified

Resolution: we will rewrite them using interfaces rather than structural types

Issue: the MOP allows for reflection of subtypes

Resolution: this was discussed and thought to be removed. If desired implementations can extend the MOP as defined

Issue: are nominal types always named?

Resolution: yes, but what we need to be aware of future extensions

Issue: the name NominalType maybe obscure for some uses (on the left side of the bell curve of user ability)

Resolution: none yet

Commentary: Brendan notes that we are targeting users in the center of the bell curve at the middle of the graph of usability

Issue: documentation comments need to be better defined

Resolution: we have higher priority things to think about. feature is deferred

Issue: implementation of type checks. graydon is looking for guidance on how to implement runtime type checks. We want to wait until Dave and Cormac are in the room. At the Feb-27 meeting there was some discussion, but the details are not clear.

Resolution: None, yet. we will discuss with use cases in mind and dave and cormac in the room.

Afternoon

ISSUE: what are the nullabliity, default values and finality of ‘string’, ‘String’, ‘Boolean’ and ‘boolean’?

RESOLUTION:

string < String boolean < Boolean

string and boolean are final and have the obvious default values

String and Boolean are not final, nullable and unsealed (for compat with ES1-3)

Catchalls are used to simulate the ES3 behavor of read and writes to primitive values.

Number.prototype.valueOf returns a Numeric value

type Numeric = (int,uint,double,decmial)

ISSUE: how do we type prototype slot. The rough idea is that they have the instance type of the class that owns it.

RESOLUTION: Lars and Brendan say Object

ISSUE: does `instrinsic` mean final. AS3 says no, but Lars wonders what its for then.

RESOLUTION: `intrinsic` is just a namespace. It allows for static type checking without shadowing their prototype counterparts. It is a builtin namespace and a reserved word so that it cannot be redefined.

ISSUE: we need to work through the builtins to see if we have valid/compatible subtyping of the intrinsics

One particular issue is that there are cases where overrides have one more param than the base method. e.g.

class Object { toString() : string } class Number extends Object { toString(...args) : string }

RESOLUTION

We want/need to allow for additional parameters to be allowed in overrides

ISSUE: what are the names of the special functions

```
function get p () {}
function set p (...) {}
function C() : x=y, super(x) {}
function get * (string)
function set * (string,*)
function call * (...)
```

What about:

```
function convert
function invoke
```

We need it for classes and instances. Write it like this:

```
meta static function invoke meta
function invoke
```

Graydon asks if we need to expose this to users like this. Brendan and others say yes, there are time when users want to define them.

RESOLUTION: But `construct` is not needed for implementing non "normal" behavior of `new`. This only occurs with functions, which does something specific, so we don't need syntax for it. function construct

ISSUE: we need to cover all these named things in the MOP

RESOLUTION: none

Discussion - Day 2

Brendan's agenda (annotated)

(Evolved over time)

- Introductions
- What users / use-cases: charter issues
  - Number of tasks?
  - Dynamic
    - risk of static misreading by implementers
  - Scripting (defined as load-and-go, play-and-learn, ...)
  - Evolving to address users
  - Disruptive due to size or static addition
    - rushing implementations
    - rushing users
    - vendor value of standard
  - Profiles
    - "ES3.1" → ES4
    - ES5
- Principles to discuss
  - backward compatibility
    - bug risk
    - footprint
    - web standards lag web workloads
      - cross-browser compatibility w ES3 – fixes to ES3
        - errata
        - ambiguities
        - lacunae
        - deprecation
      - stability (?)
        - MTBF
        - clarity, predictability
      - security
      - performance
  - type system
  - generous features / API additions
- Other issues
  - ES3 chapter 16 "Errors" + RFC 4329 vs Script#/GWT/Lazlo/Links/HaXe/etc vs syntax vs @if or equivalent

List of things not in ES3:

- security
- networking, etc APIs (graphics)
- classical and other OOP inheritance
- integrity

- confidentiality

(this started as a list of things that Scott thought should be priorities in maintaining ES3, but the list evolved during the discussion)

Discussion around the agenda

Brendan: Two languages not really credible for small systems; backwards compatibility within growth is a better choice.

Doug: Major change to the "major language of the web" is a hazard to the business model of the web; standards must catch up with the application, cross-browser compatibility must be improved, security is important (eg mashups). ES4 should be a maintenance release on ES3. There is another language that we need to consider for the future.

Scott: Forward compatibility is a big deal too. Code in the new language should parse in the old implementations (example, defineGetter et al in Mozilla as opposed to the "get p" syntax).

Brendan: Every edition of ES added syntax.

Doug: New syntax does us no good at all, the web takes many years to change.

Scott: We want something that aids web programming in the context of current implementations, syntax evolution is secondary.

Shaver: We don't *have* to tightly couple what the programmer writes with what the browser executes. One can still write to ES4 which is postprocessed into something current browsers can handle (by version-dependent inclusion or translation). Mozilla has had sophisticated code for this to allow scripts to ask for eg JS 1.2.

Scott: We don't do this any more.

Shaver: But it is isomorphic to what you're asking for.

Brendan: Note, ubiquity is not and option, revving the web neither. There will always be some conditional hacks, and that cost will always be present.

Shaver: consider eg `function window.fnord()`, which will be with us for a long time.

Scott: Code generators are not good for performance and browser workarounds, and Windows Live does not use them.

Doug: Stability is very important, and standards provide stability. "Stability" is the ability to be able to predict what is going to happen with a script when it's loaded in a browser, and the current situation is not good.

Shaver: What would improve stability in ES3?

Doug: Clarity can help, specs are misinterpreted. For example, order-of-execution problems in Mozilla relative to MSIE.

Brendan: I disagree about the example, the spec is pretty clear. But there are other aspects of the spec that could be clearer because they're implementation dependent (date parsing, the meaning of leading zeroes in numbers). Many of these problems have been addressed in bug fixes.

Doug: I'd like to explicitly deprecate some dangerous features.

Graydon: Conceivably we could adapt the reference implementation to be a lint for things like this.

Dave: Of course this doesn't solve anything, unless the code goes away.

Scott: Lots of this stuff in HTML.

Brendan: And none of it is going away. (But for translators we can't provide a low-level language, either.)

Scott: There are lots of problems with web programming that are vastly more important than inheritance – every serious developer has solved that problem. But security etc are important. And the world does not distinguish ES from web programming.

Graydon: We do not have good proposals for security, it's a hard research problem.

Scott: And then there's JSON, and other things. We need to be able to fix pieces of this without taking the whole ES4 package.

Allen: Our approach has been to take a simpler language and introduce some features (static typing), like into VB.NET. It was a painful experience.

Brendan: But we're not going that route, we are maintaining compatibility. Among other things we're trying to capture the type of things already in the language, both in the ES3 spec and in latent in user code.

Scott: Acid test – can the language replace the inheritance we already have?

General consensus: would be good to work through that. **Update**: scotti's live.com framework is extremely dynamic, but its integrity would be helped by types to make objects and namespaces unhackable, and explicit structural types for loosely-coupled APIs instead of code to check latent structural types.

Pratap: hard to reason about security in ES3. ES4, which is "bigger" and has to retain backwards compat with ES3 makes it even harder to reason about security.

Brendan: but the type system can be used to provide certain security guaranteees (invariants, immutability).

Brendan: Things like security are hard to do now.

Brendan: Let's move to how type systems help programmers build reliable programs.

Jeff: We've found that programmers become much more rigorous when they have eg class facilities.

Doug: ES is very popular, and its strengths are underrated. Types are not needed. High-value features lacking might be some of the Array methods (eg map), and the typeof brokenness must be worked around by most authors. The libraries will converge soon, into two or three libraries.

Jeff: There's no reliable way for that convergence to occur without a type system – no integrity.

Scott: Convergence is ongoing, people read each others' code. And cost can be low.

Michael O'Brien: Cost is high when the libraries are written in JS.

Doug: DOM cost dominates.

Graydon: Two caveats: Current apps are not all that big, and the DOM interface is bound by marshalling cost, it can't get much faster. Static types might however let us improve DOM performance.

Brendan: And don't forget the cost of dynamic dispatch.

Shaver: And types (and things like tail-call optimization) allow developers to write to a predictable performance model.

Allen: But should all web development be done in ECMAScript?

Brendan: Everyone who's been working in TG1 share a common understanding of the user as somebody who wants to communicated about his code with types. Maybe that consensus is not shared by Microsoft.

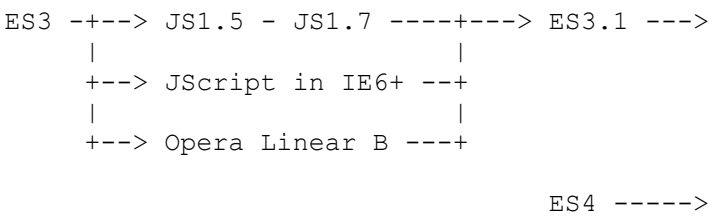
Brendan: ES4 may be disruptive, but people are scrambling to learn alternative technologies (GWT, Lazlo, ...) and that's disruptive too. But Lazlo and HaXe can converge to ES4, and translate to ES3. We need to evolve ES to respond to what users do. ES4 is such a response. It is an attempt to serve many existing users (JScript.NET, ActionScript) by merging dialects and evolving the language, rather than evolving different languages for different user groups. My interpretation of the charter is to evolve one language, and not to stick with bugfixing.

Allen: Then how can we evolve a technology carefully that might replace ES3 in the longer term? We're questioning the replacement of ES3 by the proposed ES4 at this time and would like to see an "ES3.1" called ES4, with some "ES5" called something else that does not confuse users.

Brendan: But there's scarcity of resources, and making the ES3 spec solid (in the sense of the current ES4 spec) is a lot of work (for little gain). Another aspect is the public nature of the current work – there is a lot of buzz about ES4 in the community and a lot of excitement. If MS and Yahoo! want to force a pullback from that, it will be negative for MS and Yahoo!.

Michael O'Brien: the broader base of the users is much more comfortable with class-based OOP, and it's important to cater to these.

Brendan: what's being suggested is something like this:



Divergence is easy here. The only sound footing for ES3.1 is the type system we're doing for ES4. So the argument is that even if there's an ES3.1, ES4 must come in sequence after it (by the same group) and the work ongoing for ES4 would necessarily form the basis for the work on ES3.1.

Jeff: Several practical problems with a 3.1 spec. It's hard to circumscribe 3.1 features in the reference implementation because it would want to hide things that are not hidden in ES4.

Brendan: Will a 3.1 spec – which is a lot of work – make sense? If based on the technology we do for ES4, the spec will be very different from the ES3 spec.

Dave: Can we factor the spec?

Brendan: Factoring is not free, and there are risks of divergence.

Jeff: We must evolve the language, but we should make concessions to needs for bugfixes.

Brendan: But it would be bad if MSIE 7.5 reverse engineered JS1.7 and implemented that.

Allen: The purpose of a standards body must be to define agreement and convergence among implementors.

... other discussion

Outcomes

We agreed in addition to continuing work on finishing the definition of ES4 as described by the current proposals, we will investigate the possibility of defining a smaller set of changes that encapsulate the features of current browser implementations of ES3 based languages, certain well known bug fixes (TBD) and possibly some feature deprecation. We tentatively call this ES3.1. The ES4 specification would be a superset of the ES3.1 specification. The ES3.1 specificaton should follow the form developed for the ES 4 spec and reference implementation and generally leverage completed ES 4 work. This is important to take advantage of the greater rigor and testability of the ES 4 specification and reference implementation, but also for guaranteeing compatibility with the ES4 language. There is not yet consensus on how either the ES3.1 or the ES4 specification would named or published.

The initial owners of this proposal (maintenance of es3) will be Doug, Pratap, and Allen.

Open issues:

- whether or not it is possible to define such a subset language given the resources available
- if we do define an maintenance subset, whether or not it would be issued as a separate document or as an integral apart of ES4
    - Microsoft still believes that the relationship between ES3.1 and ES4 is similar in concept to the relationship between C and C++ and they should be named and positioned accordingly.
- in either case, if we agree to define such a subset, whether it is a distinguished version of the standardized MIME types as defined in versioning, i.e. something browsers may implement.

Discussion - Day 3

Spec issues

Issue: Informative vs normative bits. Jim Miller asserts that SML code can't be normative; it'll be overspecified. But we want to test that, and we can prettyprint the SML code to be English-like if the standards bodies require it.

Resolution: Jeff will follow up on this.

Issue: Generate HTML+CSS instead of doc, and then generate doc from HTML+CSS, or generate both from the wiki? The advantage of web content is that it's interactive and can be viewed in various ways (@media).

Resolution: Francis / Jeff / Lars will look into this.

Issue: Licensing – what do we do?

Resolution: Dan should talk to ECMA, then TC members should discuss this with their legal depts.

Venue

Issue: Motion to meet in Newton in April. Brendan's in favor (along with Lars, Dave, and Chris). For the next months it's good to spread the travel around.

Resolution: April in Newton, May in Oslo, June in San Jose, July in Mountain View. We'd like to be off in August but that won't be possible if we have an October 1 deadline (see below). Tentatively we would meet in Hyderabad in September (open for revision).

Schedule and worklist

Issue: Schedule: Brendan thinks we're not schedule-driven yet, but we should cut things that seem broken beyond reasonable rescue. Come June, things that are too hard will probably have to go.

Resolution: A goal is to have the ES3 builtins in testable shape by the April f2f, and the reference implementation should be functionally complete by June, for a public release. We'll start internal nightly builds now.

Issue: Milestones and worklist. Jeff has a draft (from meetings:meetings). "Functionally complete" means there are tests to run. Beyond that there's QA, reading, and writing. But how do we construct a test suite? One possibility is to evolve a reference test suite that we share and make "official".

Resolution: Brendan will tackle the test suite issue, and talk to the various representatives, with an aim toward unforking the currently forked Mozilla test suite (as an ES3 basis for an ES4 suite).

Issue: Deadline for ECMA TC39.

Resolution: Probably present a final draft for TC39 in October, so our deadline is October 1.

Issue: Bug tracking.

Resolution: We'll try Francis's structure for a while, see how it works out.

Detailed items

Issue: ListExpr vs AssignExpr for let, yield, and expression closure. Should the bodies of these expression forms allow for the comma form or not. One important use case is for the expression closure in an argument list, which generally has to be parenthesized, which feels painfully noisy. The following are not currently allowed:

```
f( a, function (x) e, c )
f( a, yield e, c )
f( a, let (x=...) e, c )
```

Resolution: We leave things alone.

Issue: Expression closure syntax, these were proposed on es4-discuss:

```
(\ x ) e
( x ) => e
```

Resolution: We will not do these, there are no really good arguments for them.

Issue: Dictionaries – should we have a dictionary class? The only real argument in favor of putting these into the language is if they have weakly held keys (because that can't be done in the language). But a weak mechanism, even if magic, is a tax on the implementation.

Resolution: We may write up the code and some prose, but it will not be in ES4 (goes into the deferred namespace with a hope that implementations can agree on it as an extension, if they want it – but it's strictly informative).

Issue: MOP function naming. We need names for magic functions like [[Call]] in order to model the language in the language (the builtins) and to support user expansion:

```
        invoke
static  invoke
static  convert
        get*
        set*
        call*
        has* ?
        get p
        set p
```

The issues are surface syntax as well as reflection of these methods in the reflection API, and secondarily implementation complexity (magic names must all be handled specially).

The case for `has*` is that `get*` can define names algorithmically, so `has` should be available to mirror that (idempotently, unlike `get*`).

Resolution: For `invoke` and `convert` we put them in a system-defined namespace called `meta`.

Resolution: For the catchalls we get rid of the `*` and use `meta function get(id)` etc.

Resolution: We will support `meta::has`, but its meaning and its interaction with `meta::get` and with [[HasProperty]] needs to be defined carefully (part of the catchalls proposal). Brendan's job; see work item below.

Resolution: Name and signature of the method that is invoked by the `to` operator shall be called `meta::from` and its return type, if present, must be the name of the class within which the method is defined.

Resolution: The reflection interface to a class has a dedicated method for getting the constructor.

Resolution: The operators on a class reflect as static properties with their natural names (eg "+"), since their being static prevents there from being a confusion about that name. (Any dynamic object can have a "+" property, but it would never be "static".)

Issue: Should `iterator::get` be predefined on `Object.prototype` or should the iterator protocol, when looking for that method, pick up a global value if the method is not found? Brendan thinks this is a philosophical point.

Work item: Lars needs to reread the proposal and ponder this.

Issue: enumeration type

Resolution: we defer this proposal.

Issue: proposal:versioning

Resolution: accepted, with future discussion about ordering of version numbers (Brendan, please elaborate).

Issue: proposals:catchalls: Dave wonders if there are ways in which a fixture can bottom out to a getter or a setter. This has to do with esp the length property on Arrays.

Resolution: Several problems here having to do with how catchalls interact with array trunctation and element storage.

Work item: Brendan must clean up this proposal, including `meta::has` and maybe even `meta::delete`.

Issue: proposals:name objects

Resolution: No objections registered.