Minutes of the: *17th meeting of Ecma TC39*
held in: *Redmond, WA, USA*
on: *28 - 29 July 2010*

1 Opening, welcome and roll call

1.1 Opening of the meeting (Mr. Neumann)

The meeting (hosted by Microsoft at their premises in Redmond, WA) was opened by **Mr. Neumann**, Chair of TC39 at approximately 10:15 AM on 28th July 2010 (2010/029: Venue for the 17th meeting of TC39, Redmond, July 2010).

1.2 Introduction of attendees

A roll call of delegates present and over the phone was held.

Ecma participants present at the TC39 meeting:

John Neumann – Ecma – TC39 Chair

Brendan Eich – Mozilla

Sam Tobin-Hochstadt – Northeastern University

Andreas Gal – Mozilla

Waldemar Horwat – Google

Cormac Flanagan – UCSC

Allen Wirfs-Brock – Microsoft

Jeff Dyer – Adobe (partly, on the phone)

Erik Arrvidsson – Google

Douglas Crockford – Yahoo!

Sam Ruby – IBM

Adrian Bateman – Microsoft

Arjun Guha – Brown University

Shiram Krishnamurth – Brown University

Jungshik Shin – Google

Nebojsa Ciric – Google

Mark Miller – Google

Dave Herman – Mozilla

Travis Leithead – Microsoft

Istvan Sebestyen – Ecma-International – TC39 Secretary

Tom van Cutsem – University Belgium (partly, on the phone)

Addison Phillips – W3C / Amazon (partly, on the phone)

Mark Davis – Google (partly, on the phone)

1.3 Host facilities, local logistics

Mr. Wirfs-Brock welcomed the delegates on behalf of the host, Microsoft and provided information about the meeting logistics.

Mr. Horwat volunteered to take the technical notes of the meeting

2 Adoption of the agenda (2010/030-Rev2)

The meeting agenda (TC39/2010/030 Rev. 2) was approved by the meeting.

The following documents were presented and discussed:

Ecma/TC39/2010/029 Venue for the 17th meeting of TC39, Redmond, July 2010

Ecma/TC39/2010/030 Agenda for the 17th meeting of TC39, Redmond, July 2010 (Rev. 2)

Ecma/TC39/2010/031 Iterators

Ecma/TC39/2010/032 Enumeration

Ecma/TC39/2010/033 Array comprehensions

Ecma/TC39/2010/034 Generators

Ecma/TC39/2010/035 Generator expressions

Ecma/TC39/2010/036 Simple modules: external modules and scope

Ecma/TC39/2010/037 Extended Object API

Ecma/TC39/2010/038 Unicode and Internationalization

Ecma/TC39/2010/039 Const Functions

Ecma/TC39/2010/040 Ecma Liaison Report to ISO/IEC JTC 1/SC 22 for 2009–2010

Ecma/TC39/2010/041 Final erratum for ECMAScript, 5th edition (Rev. 1)

Ecma/TC39/2010/042 Ecma comments on ECMAScript, 5th edition (Rev. 1)

Ecma/TC39/2010/043 "Array comprehensions" by D. Herman

Ecma/TC39/2010/044 "Enumeration" by D. Herman

Ecma/TC39/2010/045 "Generator expressions" by D. Herman

Ecma/TC39/2010/046 "Generators" by D. Herman

Ecma/TC39/2010/047 "Iterable proxies" by D. Herman

Ecma/TC39/2010/048 "Module scoping and linking" by Sam and Dave

3 Approval of minutes from May (2010/028)

The minutes of the May 2010 TC39 have been approved without changes.

4 Report from the Secretariat

4.1 Status of ES5

(see report under 5.1)

4.2 Report from the Ecma GA / Status of Software License

Mr. Sebestyen has reported about the GA meeting on June 16, 2010 in Tokyo. He said that the GA has approved as new Ecma member Brown University, who is participating in TC39 work.

He has also reported about the approval of the TC39 experimental software copyright policy, and gave again a brief introduction about it.

4.3 TC39 joint meeting with W3C in November 2010

There was a discussion about a possible meeting with W3C again. Clearly there was a TC39 preference not to go for a joint meeting in Europe in November. It was then discussed if an additional meeting with W3C members the day before the TC39 meeting in September would not be a better solution. Sam Ruby and John Neumann will follow up that possibility.

4.4 TC39 possible relationship with Khronos (WebGL WG and Typed Arrays)

Mr. Neumann has received interest from Khronos in the Webgl and Typearrays work. There is currently no formal relationship between Ecma and Chronos. There is certainly also some TC39 interest in the co-operation with Khronos. **Mr. Sebestyen** should follow up how to formulate the formal relationship between Ecma and Khronos.

4.5 Technical Report on interoperability/conformance tests

The work on ES5 conformance testing is progressing. Microsoft announced – now that the software copyright policy is in place - that they have formally released their contribution to Ecma (has not arrived yet…). They have encouraged Google to do similar thing with their testing programs.

The timeframe for the Technical Report on interoperability / conformance test appear to be slipping. **Mr. Neumann** warned the group that even if the Technical Report has to be approved by the June 2011 GA, even then there is not so much time left, because then all work to be approved have to frozen by March 2011.

There was also some discussion if the Technical Report should also cover Harmony. It was agreed that this is a living document and will undergo evolution to include Harmony in the future.

5 Progression of ES 5

5.1 Approval of Errata

Mr. Wirfs-Brock explained the progress made since the last meeting. The Ecma comments on ES5 were very close to be finished. The submission to JTC 1 has to be made latest on August 6, 2010. (This has been achieved…).

The proposed changes have been approved by TC39 and the Editor and the Ecma Secretariat has been instructed to submit the ES5 Ecma comments to the JTC 1 fast-track process.

The changes are both editorial and some technical. Very much depends on what other comments will be submitted to ISO/IEC JTC 1/SC 22, and if SC 22 decides to hold a BRM or not. This has to be seen. Anyway, it is clear that after the approval of ES5 in JTC 1, Ecma TC39 has to harmonize the JTC 1 and the Ecma versions again. The wish of TC39 is that the harmonization should not lead to a new ECMA-262 Edition. A solution like ES5.1 (Edition 5.1) would be ideal. **Mr. Sebestyen** said, that he has to consult that with the CC, as this is not the usual Ecma practice (just to progress the edition numbers with 1).

Depending on the progress in the the JTC 1 approval, the Ecma approval would be either December 2010, or June 2011.

It was also brought up if for the new Edition the Software Copyright policy can be applied. **Mr. Sebestyen** said that this was a “new software project” and since JTC 1 does not have a software copyright policy, simply they have to be consulted, if they saw a problem with that.

Unrelated to the approval TC39 has reviewed the draft report of the SC 22 Chairman on the co-operation with Ecma TC39 and useful comments have been made.

6 Discussion of ES harmony (technical contributions are available and can be found on the ES wiki)

6.1 Iterators

http://wiki.ecmascript.org/doku.php?id=strawman:iterators (2010/031)

6.2 Enumeration

http://wiki.ecmascript.org/doku.php?id=strawman:enumeration (2010/032)

6.3 Array comprehensions

http://wiki.ecmascript.org/doku.php?id=strawman:array_comprehensions (2010/033)

6.4 Generators

http://wiki.ecmascript.org/doku.php?id=strawman:shallow_continuations (2010/034)

6.5 Generator expressions

http://wiki.ecmascript.org/doku.php?id=strawman:generator_expressions (2010/035)

6.6 Proxy.[[Construct]] reform

6.7 Simple modules: external modules and scope

http://wiki.ecmascript.org/doku.php?id=strawman:simple_modules (2010/036)

6.8 API for Internationalization

6.9 Extended Object API

<http://wiki.ecmascript.org/doku.php?id=strawman:extended_object_api> (2010/037)

6.10 Unicode and Internationalization

http://wiki.ecmascript.org/doku.php?id=strawman:unicode_support (2010/038)

6.11 Const Functions

http://wiki.ecmascript.org/doku.php?id=strawman:const_functions (2010/039)

7 Date and place of the next meeting(s)

September 30 – October 1, 2010 Location Bay Area, host Mozilla

November 17 – 18 Location TBD

8 Closure

The chairman reviewed open issues for the next meeting, indicated that a draft agenda for the next meeting would be circulated in the next couple of days and hope to produce agenda quickly. He requested TC39 members to submit within the next 2 weeks recommendations for the agenda items of the September TC39 meeting.

W3C got a request from Google for an ECMAScript internationalization library. We should invite these folks to the next meeting.

The next meeting will be in the Bay Area, CA at the kind invitation of Mozilla.

Mr. Neumann closed the meeting at approximately 16:00. He thanked the delegates for their hard work, Microsoft for their kind hospitality and Ecma International for the Social Event.

Technical Notes from Waldemar Horwat:

From: waldemar@google.com

To: es-discuss@mozilla.org

Subj: Day 1 and Day 2 meeting notes

ECMA needs permanent documentation of proposals to satisfy WTO requirements. Proposal is to have ECMA staff take snapshots of the wiki once the relevant wiki proposal urls are posted.

Microsoft executed an ECMA software license for the test project.

Google is expected to contribute Sputnik.

Two active projects for contributions: Test262, Harmony. Use separate software contribution pro.

Brendan will look into relicensing the ES3/4 grammar and semantic engine and validator as BSD so it can be part of the Harmony contribution.

John: Easiest for TC39 members to just license contributions with BSD licenses from the start. Cuts out a lot of the legalese for collaborative development, needing only one contributor agreement form from each company for each project.

No response from Philippe to John.

No enthusiasm within the group for travelling to France for a TC39 meeting later this year.

Currently WebIDL has no editor. There's a vacuum there.

Four different organizations are standardizing ECMAScript binary arrays:

* TC39

* WebGL typed arrays are happening now. It's too late to affect them.

* HTML5 binary arrays are in an earlier stage.

* There's also NodeJS.

It's awkward to coordinate all of this.

Rex Jaeschke's SC22 liaison document is somewhat out of date – it claims that the 3rd edition of ECMA-262 is the current one.

TC39's ES5 comments shall be submitted to JTC1 secretariat. This includes some Technical Corrections. It would be nice if there were no need for a Ballot Resolution Meeting. All this falls under the old ISO procedure because we filed the paperwork before July 1.

If/when ISO approves ES5, we'll backport the changes to the ECMA ES5 document and submit it to the GA as Edition 5.1. Don't want to name it "Edition 6" for small errata like these.

Allen: Should we include the new ECMA BSD software license in ES5.1?

Mark, Waldemar: Yes. Istvan: Wants to harmonize with ISO first.

Changes to module proposal from last meeting:

Nested modules have access to the complete lexical environment (not just the module bindings) of their parent modules.

The external modules slideshow is new, not yet reflected on the wiki.

```
module MyLib {
  function log(x) {...}
```

```
  module Util {
    module Even = load "even.js";
    module Odd = load "odd.js";
  }
}
```

This is like compile-time textual inclusion except that even.js and odd.js don't have access to the bindings MyLib, log, and Util.

However, odd.js has access to the binding of Even and even.js has access to the binding of Odd.

```
    module Even = load "even.js";
    module Odd = load "odd.js";
    module Imaginary = load "even.js";
```

This creates two entirely separate copies of even.js.

Load is like textual inclusion, not like an import:

```
even.js:
module Odd = load "odd.js"

odd.js:
module Even = load "even.js";
```

This blows up due to infinite inclusion.

Free variables inside odd.js and even.js are assumed to be module

references. This stirred up controversy.

```
even.js:
module A = load "foo.js"

foo.js:
import Odd.*  // Doesn't resolve Odd

To fix it:
even.js:
module Odd = Odd;
module A = load "foo.js"

foo.js:
import Odd.*  // Doesn't resolve Odd
```

except that "module Odd = Odd" doesn't work because modules are

letrec-bound. Proposed fix: introduce a special syntax for

re-exporting a module:

module Odd;

Now, what if instead of "module Odd = Odd", one wanted to express

something with the effect of "module Odd = Odd.SubOdd"?

Evaluation time expository example:

```
module A {...};
module B = C.foo;
module C = load "c.js";
```

works because the module B = ... statement doesn't do anything at run time. However,

```
module A {...};
module B = C.foo;
B.doSomething();
module C = load "c.js";
```

doesn't work because B is not defined by the time doSomething is invoked on it.

This also means that even.js cannot use "Odd" at the top level in the original example.

Question about what the following means in simple_modules on wiki: "A program is a sequence of scripts, evaluated from top to bottom. Each script is evaluated in a nested declarative environment record."

Possible answer: Each subsequent script on a web page is nested inside the prior one.

Objections: This is weird. A let statement can shadow a let statement using the same name. This is counterintuitive for sibling scripts on a page. Having a flat scope for the collection of scripts on a web page would be better.

The above means that the top-level scope can be extended dynamically.

This might clash with any predefined meaning of free variables such as considering them to be references to modules; it was a major hassle in ES4.

Why do we need overridable enumerators when we have overridable iterators?

Point: because one returns only strings that are property names while the other returns arbitrary data.

Counterpoint: If that's the criterion, then we shouldn't be using the for-in syntax for both. Leads to exactly the same problems when you think you're doing a for-in over properties when you're in fact getting values.

Error prone: for (x in Iterator.values([1,2,3,4,5])) vs. likely programming error for (x in [1,2,3,4,5])

Waldemar: If they are separate concepts, using the same for-in syntax for iteration and enumeration is too error-prone. If you intend to iterate on object Foo but accidentally pass in a Foo that's not an iterator, you'll get its property names instead of a visible error.

It's particularly insidious with arrays.

Mozilla's recent trials: Changing the behaviour of enumeration to a pure snapshot order under concurrent modification of the iterated object turned out to be a breaking change. People use object properties as work lists and sometimes delete future work list items.

Mozilla's current trial is an "ugly" and hard-to-describe deletion-filtering order that handles deletion on the direct object but not across the prototype chain. Too implementation-specific to make it suitable for a standard.

Conclusion: Futile to exactly specify enumeration order in obscure cases?

Proposed enumeration order:

1. Numeric order across all prototypes (entries that are considered to be array indices)
2. Prototype chain order
3. Creation order

Debate and some incredulity over scanning the prototype chain twice.

Prototype chain order should be the primary order, not secondary.

Behaviour of what happens when the iterated object is modified is still unclear.

Why is there a need for both next() and send() in the iteration protocol? Seems like unnecessary historical complexity.

Generators discussion.

Proxies: Discussing open issues on http://wiki.ecmascript.org/doku.php?id=harmony:proxies

How much validation to do? Do we do as much as we can stand?

(Brendan, Allen, Mark: Depends on who it is that is doing the standing:)

Allen: Some validation is expensive (looking for duplicate names in getOwnPropertyNames).

Copying and normalizing results (property descriptors etc.) also has cost.

Allen: A script could override Object.getOwnPropertyDescriptor and cause trouble.

MarkM: That's why for secure frames you'd have an initializer script that freezes things like Object.getOwnPropertyDescriptor.

Allen: If you do that, the initializer script could substitute Object.getOwnPropertyDescriptor with its own, more bulletproof version. This puts the burden of validation on those who want it.

Debate about whether having tight restrictions now and relaxing them later could be called "upwards compatible".

Waldemar: It's a stretch to put that label on it, just like I wouldn't call a change of functionality of a safe from having only the right combination open it to having anyone be able to open it an "upwards compatible" change.

Debate about the expense of copying/normalizing for validation.

Comprehensions: To be useful, need to have a bunch of basic helpers such as range(), properties(), values(), etc. with short names that can be used to construct comprehensions. These will be added to the proposal.

Would like to be able to interleave for and if clauses:

[x*y for (x in range(...)) if (isprime(x)) for (y in range(x+1...)) if (isprime(y))]

This is allowed by the proposal but not by the wiki.

The proposal has the grammar (for let? if?)+. This doesn't currently allow let-if-let-if combinations, where the second let is valid only if the first if returns true. Would be nice to allow orders like these too.

Debate about array comprehensions without a for clause:

```
for (x ...) {
  ... [x if (isprime(x))] ...
}
```

This would generate either a 0- or a 1-element array depending on the if condition. Duly noted.

Generator comprehensions: (foo for (key in obj))

Desugars to functions, not lambdas!

```
(function() {
  for (let key in obj) {
    yield foo;
  }
})()
```

Controversy for what happens if the generator expressions contain:

- arguments (make it work?)
- this (make it work?)
- yield (outlaw it?)
- var, break, return inside a let expression (don't bring back let expressions)

Require parentheses around a generator comprehension even if it's used as a function argument?

Python does.

Here's one thing that could happen if we don't require parentheses in some contexts:

Refactor

```
foo(bigexpression
    for (y in z))
```

into:

```
let i = bigexpression
      for (y in z)
foo(i)
```

which unfortunately would be transformed by semicolon insertion into something quite different:

```
let i = bigexpression;
for (y in z)
  foo(i)
```

const functions:

Waldemar: Grammatical and readability concerns about confusion between const x and const x(). "const" alone restricts syntactic options for destructuring. Other options: "const function", "def", ...

Erik: Common case is not to freeze everything. Freezing won't work with jquery listen functions etc.

Brendan: Makes it harder to rely on expandos.

Mark: Weak maps will solve the problem that expandos are used for now.

Waldemar: That would just bifurcate the problem, since browsers without weak maps will be around for a while.

Brendan: Weak maps won't make Firefox 4.

Brendan: Education would be good but will take a while.

Allen: Not willing to commit to this as a proposal yet because of concerns about its weight in making the language large.

Waldemar: OK with this feature, not with using just "const" as the syntax.

Doug: OK with this feature.

Internationalization API:

Prototype mistakes in the spec. Some aLocale.prototype.foo should be

Locale.foo; some should be Locale.prototype.foo.

MarkM: removeExtensions sounds too imperative (it actually returns a

new copy). Rename it.

Waldemar: Don't understand the spec. What is the pattern in parseDate("29. jul 1974.", {"pattern": "yy-MM-dd"}, locale);

doing? The supplied order is the opposite of the pattern and there are no dashes in the supplied string. Other pattern examples are

"yy/MM" and "yMMM". Since the separators don't necessarily appear in the matched string, what does the choice of "-" vs. "/" mean in the pattern?

Should this be developed by TC39 as a separate library standard?

MarkmM, Allen, Waldemar: Confusing which locale is which in aLocale.prototype.getDisplayLanguage(locale). The wiki documentation contradicts the verbal explanation. Also, the name should be something something different like displayLanguageOf.

Explanation of difference between patterns and skeletons. They use different syntaxes and purposes. Skeletons don't have delimiters or punctuation. A skeleton turns into a locale-appropriate pattern; if the skeleton order would seem weird in the locale, the locale can overrule the skeleton and produce a more appropriate pattern.

No current support for alternate calendars (i.e. Japanese calendar).

Setting collator strength should give a new collator instead of modifying in place. An even better solution would be a settings object instead of a single strength parameter.

someStringArray.sort(col.compare) doesn't work because compare is defined on Collator.prototype. You'd get the generic method not bound to col.

Collators will canonicalize unicode characters (i.e. precomposed ö == decomposed ö).

Any single collator will always induce a valid total order. The size of the equivalence classes will vary. For example, depending on strength, it may consider u == ü.

Discussed merits of non-BMP unicode escapes using the syntax \u{xxxxxx}. Generally liked, but we can't use it in character ranges in regular expressions without doing major surgery on regular expressions. Noncontroversial in other contexts.

String.direction questions: There are more than just the two possible directions. Vertical? A string containing parts with different directions? Indeterminate (example: empty string)?

Extended object API:

getPropertyDescriptor is useful for encapsulating property lookup on prototypes.

Allen, Waldemar: Concern about getPropertyDescriptors, getOwnPropertyDescriptors, getPropertyNames being too eager and thus asymptotically slow on objects representing large collections. Would want to be able to work with huge or infinite objects.

What would call the getPropertyNames trap? Only another call to getPropertyNames.

Agreed to retain only getPropertyDescriptor and getPropertyNames.

Next meeting (at Mozilla) moved to Sep 30/Oct 1.