Minutes for the: *24th meeting of Ecma TC39*

held in: *San Francisco, CA, USA*

on: *26-27 September 2011*

1 Opening, welcome and roll call

1.1 Opening of the meeting (Mr. Neumann)

The TC39 meeting (hosted by Mozilla in San Francisco, CA) was opened by **Mr. Neumann**, Chair of TC39 at approximately 10:15 AM on 26th September 2011 (TC39/2011/038 - Venue for the 24th meeting of TC39, San Francisco, CA, September 2011).

It was noted that before the TC39 meeting, on the 17th of August 2011 the TC39 ad-hoc group on internationalization (i18n Ad Hoc group) has also met. The report of that meeting is given under 4.2 below.

1.2 Introduction of attendees

John Neumann – Ecma International

Istvan Sebestyen - Ecma-International

Alex Russell - Google

Waldemar Horwat - Google

Allen Wirfs-Brock - Mozilla

Sam Tobin-Hochstadt - Northeastern University

Douglas Crockford - Yahoo!

Brendan Eich - Mozilla

Mark Miller - Google

Luke Hoban - Microsoft

David Fugate – Microsoft

Dave Herman - Mozilla

Eric Arvidsson - Google

Oliver Hunt - Apple

Norbert Lindenberg - guest – no affiliation

1.3 Host facilities, local logistics

Dave Herman welcomed on behalf of Mozilla the delegates and provided logistical information. It was announced that Google, Mozilla and Ecma international would host a social event on September 26th evening.

2 Adoption of the agenda (2011/039 Rev 1)

Ecma/TC39/2011/039 Rev 1 contained the Agenda for the 24th meeting of TC39, San Francisco, September 2011. This was agreed.

The relevant Ecma TC39 contributions for the meeting are the following:

- Ecma/TC39/2011/037 Draft minutes of the 23rd meeting of TC39, Redmond, July 2011
- Ecma/TC39/2011/038 Venue for the 24th meeting of TC39, San Francisco, September 2011
- Ecma/TC39/2011/039 Agenda for the 24th meeting of TC39, San Francisco, September 2011 (Rev. 1)
- Ecma/TC39/2011/040 Second draft Standard ECMA-262 6th edition, September 2011
- Ecma/TC39/2011/041 1st draft Technical Report on Test262 prepared by David Fugate (Rev. 4)
- Ecma/TC39/2011/042 Notes of the meeting of TC39 ad hoc group on Internationalization, 17 August 2011
- Ecma/TC39/2011/043 Status Report on test262 by David Fugate, September 2011
- Ecma/TC39/2011/044 TC39 chairman's report to the CC, October 2011
- Ecma/TC39/2011/045 Final draft Technical Report on Test262

Other documents are mentioned via their URL to the ES Wiki.

The more detailed technical notes by **Mr. Horwat** are attached to this report.

3 Approval of minutes from July 2011 (2011/037)

The minutes of the 23rd TC39 meeting in Redmond in July 2011 have been unanimously approved with no changes.

4 Status Reports

4.1 Report from Geneva

Mr. Sebestyen gave a short report. Basically since the End of July Meeting in Redmond, WA not much has happened in Geneva, as it was holiday time.

On the ECMAScript Trademark matters he said that it was on its way in Switzerland, the EU and USA. In Switzerland we have the mark, in the EU, US, Korea and Japan it is being processed. As soon as there is new information we will let TC39 know.

Mr. Sebestyen points out that the internal Ecma TC39 documentation and archival is still not up to the requirements as they should be. The major technical contributions are being posted outside of Ecma (e.g. ES-Wiki), and its mirroring to the TC39 Ecma internal archive is not secured. For the past two TC39 meetings the Secretariat did not prepare the pdf document reflecting the momentarily situation of the external sites. We had some emails on requirements with **Mr. Wirfs-Broch**, but that was all.

It was decided to do the following:

- The Secretariat will get in static HTML format copies (snapshots) of the ES Wiki on a regular basis. **David Fugave** has volunteered to do it (maybe with some help within Mozilla). The requirement is that no special software should be needed to present the content.

- Also the “ES Discuss” list should be archived. **Patrick Ch.** should subscribe to it. “ES5 Discuss” also exists (maintenance), and also a “Test262 discuss”. Then we will see how those “Discuss” lists can be archived.
- Archival is needed from the practical point of view at least until possible patent issues may come up. This is about 20 years.

It was also agreed that the TC39 RANDZ standardization goal should be put onto the TC39 website, when the update is being done.

*Secretariat’s Note: In the October 2011 CC meeting **Ms Valet-Harper** brought up that in the recent draft minutes of the TC39 meeting it said in Ecma/TC39/2011/046 “Draft minutes of the 24th meeting of TC39, San Francisco, September 2011 (Rev. 1)” that “it was also agreed that the TC39 RANDZ practice should be put onto the TC39 website, when the update is being done”. There was a discussion how this point should be formulated. The CC basically agreed that it was ok to state that the “TC has the intension (or goal) to develop a royalty free standard, such as for important web standards”, but of course no guarantee by Ecma can be given that this has been actually achieved. In other words the word “practice” should be avoided. The CC then requested that the new text should be first shown to the CC, before putting it up on the TC39 web page. As result of this “practice” in TC39/2011/046 rev2 was replaced by “standardization goal”.*

4.2 Report of the status for a Technical Report on interoperability/conformance tests

4.2.1 Prototype Website (http://test262.ecmascript.org and http://test.w3.org/html/tests/reporting/report.htm

Ecma/TC39/2011/043 Status Report on test262 by **David Fugate**, September 2011 was given. Good progress, some bug removed, a bunch of new tests have been added. Microsoft and Google had a good co-operation on Test262. But there are some items still open, that are not covered in tests yet.

The 1 page TR will be prepared for the December GA did not have the Ecma TR format. Correspondence took place between the Ecma Secretariat (**Patrick**) on the TR template and the Editor prepared the next draft.

The goal is to finish the formatting and to put the draft on the Ecma private web site for approval in the December 2011 GA by October 8, 2011.

But with the approval of the TR the work will continue and also ES6 Tests will be gradually added.

4.3 Report from the ad hoc on Internationalization standard.

4.3.1 Review of proposed draft standard

4.3.2 Multi-system prototype testing

4.3.3 Next steps

On August 17, 2011 there was a meeting on internationalization. Norbert Lindenberg reported on it. The results of that meeting had been sent by **Nebojša Ćirić** – who could not be in this meeting – by email.

The meeting notes of the TC39 internationalization ad-hoc meeting on August 17, 2011, are available here:

The notes were primarily taken by Jungshik Shin, Google.

Unfortunately the spec is not ready for a GA approval in December 2011. The group are still working on a draft document. The latest revision is available at:

https://docs.google.com/document/pub?id=1rsUxJQ03Ql6o3bh6RN7J81dtYZXE7OVsdQBw_h5ASnM

Mark Miller and Allan Wirfs-Broch are already added as editors, but if anybody else wants to take a look or leave a comment I'll add them to the editor list (I'll need their email addresses).

It was mentioned that additional testing was needed. The minutes of the August 17 meeting will be added to this minutes.

Next steps:

In the TC39 meeting the following has been decided:

Go for an Ecma General Assembly approval of the standard in June 2012. This delayed the original plan by about 6 months.

An immediate fast-track submission to JTC 1/SC 22 is planned.

4.4 Update of TC39 Web Page at Ecma home page

It was decided to update the TC39 webpage. **Mr. Neumann** will send the update instructions to the Ecma Secretariat. But first he wants to know what has happened to his earlier update instructions (The Secretariat has checked this, and the earlier update instructions were apparently not received).

4.5 Demo the ParallelArray (RiverTrail project name)

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

5.1 Status of working draft

5.2 Status of Classes

5.3 Status update on Binary Data

5.4 Synta Discussion

5.4.1 http://wiki.ecmascript.org/doku.php?id=strawman:block_vs_object_literal

5.4.2 http://wiki.ecmascript.org/doku.php?id=strawman:block_lambda_revival

5.4.3 http://wiki.ecmascript.org/doku.php?id=strawman:arrow_function_syntax

5.4.4 http://wiki.ecmascript.org/doku.php?id=strawman:paren_free

5.5 Update on Block Scoping Decisions and progress on specification

5.6 The .{ operator and object literal based class definition patterns

https://mail.mozilla.org/pipermail/es-discuss/2011-August/016187.html

https://github.com/allenwb/ESnext-experiments/blob/master/ST80collections-exp1.js

https://github.com/allenwb/narcissus/blob/master/harmony-extensions.md

5.7 Leak Hazard in Private names

6 Date and place of the next meeting(s)

November 16-17, 2011. Location: Silicon Valley, CA, hosted by Apple.

January 25-26, 2012. Location: Silicon Valley, CA, hosted by Yahoo.

7 Closure

The TC39 Meeting ended at 4:30 PM on 27 September 2011. **Mr. Neumann** has thanked the meeting participants for their good contributions, constructive discussions and the co-operative spirit of the group.

The group expressed appreciation to Mozilla and to **Mr. Dave Herman** and **Mr. Brendan Eich** for hosting the meeting and Google, Mozilla and Ecma international for hosting the Mexican dinner on the 26th September in Downtown San Francisco, CA.

Item 5 Attachment

Waldemar Horwat's Meeting Notes from September 26:

Test262.ecmascript.org presentation
Allen: There will be substantial changes to section numbering in ES6. Concerned about too much dependence on section numbers in Test262. However, there exist tools to rename Test262 files with new section numbers.

Debate sparked by test262's requirement that defineProperty throw if it can't create the property -- that's an invariant that even host objects are not allowed to violate. We can't test all possible property names and objects, so the authors of the test have chosen ones which some implementations are known to violate. Is this kosher?
Yes -- this is a normative invariant in the spec, even for implementation extensions.
Discussion over refactoring of such tests to parametrize them over the property names known to violate the spec. This way the tests can include different implementations' suspicious properties.
Debate over whether errors the test finds should be labeled as DOM errors or ECMAScript errors.

Test262 Technical Report will be submitted to the December GA. It must be approved and sent to ECMA members by October 8th to make it to the December GA.

Discussion about updating the ECMA TC39 web page.

ParallelArray demo.
Currently the implementation is very hacky (i.e. works mainly just for this demo), but shows great promise.
Discussion: How do we express arithmetic on number types other than IEEE double? This is a different problem from typed arrays in that we want to actually compute using other types (int8, float32, etc.) rather than just store using those types and convert to doubles for computation.

Allen's presentation on current state of the spec.
Question: We have an opportunity to make function declarations create immutable bindings if they occur inside blocks.
Answer: Consensus on making them consistent with function declarations outside blocks. Either we'd make all function bindings mutable or all immutable.

Discussion about object literal attribute modifiers. Objections to using := for immutable properties (intuition is opposite from that of Pascal). Disagreement over whether immutable properties should be nonenumerable -- some are the same in every instance, some are immutable but have different values in different instances.
No consensus on any concise way of marking properties nonenumerable.
DaveH: Prefer to play it conservative and do nothing rather than introduce a funky new syntax.

Evaluated property names:

```
{
  [foo]:1,
  get[1+2](){},
  ['prefix'+i++](){}
}
```

Waldemar: Concerned that these look too much like array literals or an array index expression on an array named "get", which is not a reserved word.
These can produce property name collisions at run time. What should happen?
A. Rely on DefineProperty semantics, which sometimes allow rebinding of properties
B. Throw

In choice A we'd then have to allow collisions in literal property names as well. Leaning towards choice B.

Method form is not a constructor. Why? To be consistent with built-ins. Hotly debated whether we should be consistent here (and consistent with what -- with built-ins, with bind, or with standalone functions?) and whether the extra constructors are useful.

super wiring: If 'super' is nested inside one or more functions inside an object literal, what does 'super' bind to?

```
{
  f:(0, function(){... super ...}),
  g:function(){... super ...},
  h(){... super ...},
  k:... super ...
}
```

Another hot debate with no convergence.
If it doesn't bind to anything, should super:
A. Be a compile-time syntax error?
B. Return null?
C. Return an empty object?
D. Throw an error when containing function is entered?
E. Throw an error when evaluated?
Not resolved. Allen's proposal currently has choice C if super is used in a property lookup and the value of 'this' if super is not used in a property lookup.
Debate over the meaning of:

```
function returnUndefined() {
  return super.foo;
}
```

What about

```
function returnUndefined() {
  super.foo = 42;
}
```

Error because there is no object.

What about assignments to super.foo in general, when there is a super?
Allen: These behave the same as assignments to this.foo
Waldemar: These should either do the 'super' thing or be an error. Silently changing the meaning to not do 'super' lookup is gratuitously unsymmetric.

Commas optional after method/getter/setter definitions. These currently always end with }. Note that an equivalent member definition that initializes the member to a function does not make the comma optional (and can't because of things like:

```
{
  g:function(){...}
  [expr] ...
}
```

where the [expr] is an array lookup on the function literal.

Debate over whether method properties should be writable and configurable or not.
Brendan: Use # to mark nonwritable/nonconfigurable properties.
Waldemar: Want to have easy ways to specify both writable and non-writable nonconfigurable properties.
MarkM: Nonconfigurable writable properties don't make sense because an adversary can freeze them.
Waldemar: That's a design bug we have to live with, but they're still useful in the common case.
Brendan: Accessors might solve this.

Waldemar: Accessors are too wordy and would also rely on private properties here.
Alex: Common accessor pattern of intercepting writes but passing reads through is too wordy and could use better support.

Waldemar: For methods, use # to mean nonconfigurable/nonwritable.
For value properties, use # to mean nonconfigurable/writable.
For value properties, use const to mean nonconfigurable/nonwritable.
None of these prefixes have any effect on enumerability.

Alternate proposal: use # in front of entire object to freeze (MarkM) or seal (Waldemar) object.

Reached consensus on the following # proposal:
before an object literal seals the object
before a propoerty makes that property nonconfigurable and nonwritable.
All method properties are nonenumerable.
All value properties are enumerable.

{x} desugars into {x:x} in the proposal.
DaveH: Would prefer to turn that into {x:void 0};
DaveH withdrew proposal, but now Waldemar wants to do that, in order to be able to define objects that contain uninitialized instance variables x, y, z, length:
#{x, y, z, length, ...}
without capturing the values of x, y, z, length from the outer environment.

Allen's "class" definition pattern:
DaveH: Too imperative for what should be a declarative construct
Brendan: Doesn't substitute for classes
MarkM: How would you freeze everything?
Getting the .prototype. and .constructor. sections out of order or omitting one would lead to weird errors.

DaveH:
Dislike the use of .{ for object extension. Like the idea in principle. It's not enough to say "this is the semantics we want, pending finding the right syntax".
Some like the syntax. Debate.
About as many syntax variants offered as there are people in the room.

Meeting notes from September 27:

Classes:
What do you get when you access an instance property before the directive defining it has been executed?
A. Throw (just as in temporal dead zone for let/const variables)
B. Get undefined
C. Property does not exist yet on the object -- prototype shows through

Waldemar objects to B because it would allow observers to see const properties get mutated and is future-hostile for guards (a guard expression would not have even been evaluated at the time of the property access).

Allen's idea: Separate property definitions from any code that refers to "this" using a barrier statement that deems that the object has been initialized. Works for single-level classes but not for inheritance, as the superclass could have leaked "this".
A suggestion was made to run all property definitions in all classes in the hierarchy before running imperative initializations in reverse hierarchy order. Waldemar doesn't know of any major language that does initializaiton backwards like this; the problem is that derived classes want to refer to the already initialized base class state so that they can initialize themselves.

Explored a different order: doing definitions in reverse hierarchy order and imperative initialization in hierarchy order.

At what point does a class instance acquire trademarks? When the constructor returns.
But then what about constructors that create a trademarked object and then put them into a registry of all objects with that trademark? The constructor can't register the object because it's not trademarked yet before the constructor returns. Yet another kind of two-phase initialization?

Discussion of __proto__. All implementations currently disallow using it to mutate the prototype chain of nonextensible objects.

Should we standardize __proto__ in Annex B?
MarkM + a few others: Yes
Waldemar, Doug: No

MarkM: Emphasizes that we can't allow one to add new private properties to nonextensible objects. Allowing that would create a hidden communication channel between two frozen entities given access to the same frozen object.

Back to throw/undefined/no-property debate.

Brendan: Fixing the const case (preventing observation of mutation) will also fix the guard case.

Exported bindings from modules are also reflectible as properties. If these are accessed before they execute (i.e. in the dynamic dead zone), they throw.

DaveH: Modules and classes are different things.

MarkM: Only have const classes?
Others: Only have non-const classes?

DaveH: Currently have nothing exactly like a const field on an object.
MarkM: Could specify it in terms of proxies.

DaveH: Proposed read barrier on const fields; no read barrier on let fields.
Luke: Still worried about performance of dynamic field access.
MarkM: Dubious about performance issue because the same case already arises for accessors.

Allen: We'd need to define a new instance member state: not yet initialized. We'd use this for reflecting on module instances as well.

DaveH: Not dogmatic about classes as sugar. OK to extend semantics in cases where the current ones are inadequate.

Brendan's summary:

```
class Widget {
  constructor(a, b) {
    public const k = a*b
    const c = a+b
    ...
  }
}
```

Similarities between const properties and lexical variables:
- Initialize-only
- Temporal dead zone

Differences:
- property vs. lexical binding (ignoring reflection on modules)
- pedagogy

Went off on a tangent to discuss a class idea:

```
class Point(x, y) {
  getX() {return x}  // Instance method because it refers to x
  getY() {return y}  // Instance method because it refers to y
  getInstanceX() {return this.x}  // prototype method
  getInstanceY() {return this.y}  // prototype method
  foo() {return getX();}  // Doesn't work -- need to use this.getX()
  public x = x, y = y;  // Instance variables
  ... constructor code here ...
  alert(this.x);
  public m = function() {return x+y}
}
```

No way to factor a function into two functions without indirecting via "this" as in the foo line above.
Adding a reference to x to a function changes where it lives.

Wondering why we're discussing this (which we've previously discussed) instead of discussing const and the dead zone.

DaveH: Don't go around throwing vetoes. We should not be finding the least objectionable thing.

Trying to understand Oliver's objections to the current class proposal.

Brendan's update on post-es.next paren-free.
Luke: concerned about gratuitously having two different ways of doing the same thing.
Brendan: Safe; omitting braces and parentheses will just yield a syntax error.
Waldemar: Opportunities for mischief when combined with semicolon insertion:
Start with ES5 code:

```
if (a + b)
  (x.y)()
z++
```

Now (erroneously) convert it to paren-free:

```
if a + b
  (x.y)()
z++
```

This doesn't give a syntax error; it silently changes the meaning of the program.

Block-lambdas:
Now the |'s are required (syntax doesn't work if they're not).