Minutes for the: ***26th meeting of Ecma TC39***
held in: ***Santa Clara, CA, USA***
on: ***18-19 January 2012***

1 Opening, welcome and roll call

1.1 Opening of the meeting (Mr. Neumann)

The TC39 meeting (hosted by Yahoo! in Santa Clara, CA) was opened by **Mr. Neumann**, Chair of TC39 at approximately 10:30 AM on 18th January 2012 (TC39/2011/056 Rev1 - Venue for the 26th meeting of TC39, Santa Clara, CA, January 2012).

It was noted that before the TC39 meeting, on the 17th of January 2012 the TC39 ad-hoc group on internationalization (i18n Ad Hoc group) has also met. The report of that meeting is given under 4.3 below.

1.2 Introduction of attendees

John Neumann – Ecma International

Istvan Sebestyen – Ecma International

Isabelle Valet-Harper – Microsoft (CC Chair of Ecma International) – part-time

David Fugate – Microsoft

Alex Russell - Google

Rafael Weinstein - Google

Waldemar Horwat - Google

Allen Wirfs-Brock - Mozilla

Sam Tobin-Hochstadt – North Eastern University

Douglas Crockford - Yahoo!

Brendan Eich - Mozilla

Mark Miller - Google

Luke Hoban - Microsoft

Dave Herman - Mozilla

Eric Arvidsson - Google

Gavin Barradough – Apple

Nebojsa Ciric – Google – part time

Norbert Lindenberg – invited guest – no affiliation – part time

1.3 Host facilities, local logistics

Doug Crockford welcomed on behalf of Yahoo! the delegates and provided logistical information. It was announced that Ecma international would host the traditional social event on January 18th evening.

2 Adoption of the agenda (2012/001)

Ecma/TC39/2012/001 contained the Agenda for the 26th meeting of TC39, Santa Clara, January 2011. This was agreed with minor changes.

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

- Ecma/TC39/2011/055 Minutes of the 25th meeting of TC39, Cupertino, November 2011
- Ecma/GA/2011/144 Minutes of the 102nd meeting of the Ecma General Assembly held in San Francisco, USA (6 December 2011)
- Ecma/TC39/2011/056 Venue for the 26th meeting of TC39, Santa Clara, January 2012 (Rev. 1)
- Ecma/TC39/2011/057 Draft "ECMAScript Globalization API Specification", 1 December 2011
- Ecma/TC39/2012/001 Agenda for the 26th meeting of TC39, Santa Clara, January 2012
- Ecma/TC39/2012/002 Notes of the ad hoc meeting on Internationalization, 17 January 2012
- Ecma/TC39/2012/003 Test262 status report, January 2012
- Ecma/TC39/2012/004 Binary data updates, January 2012
- Ecma/TC39/2012/005 One JavaScript presentation by Dave Herman

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 November 2011 (2011/055)

The minutes of the 25th TC39 meeting in Cupertino in November 2011 have been unanimously approved.

4 Status Reports

4.1 Report from Geneva

Mr. Sebestyen gave a short report, basically concentrating on the outcome of the GA meeting in December and what the meeting had to say about TC39 activities.

In the GA Mr. Breidthardt (CC Chairman) has reported about the activities of TC39: The “Internationalization” project made good progress. The work will be tested by multiple implementations. Submission for approval by the GA is expected in June 2012. *(TC39 has requested **Mr. Sebestyen** that the new standard should get the Ecma number ECMA-402. This has been reserved by the Secretariat.)*

TC39 has developed a set of tests (to be approved as TR at this GA) against the ES 5 specification. Tests (referred to as “Test262”) are contributed by members under the Software Licensing Policy approved by the Ecma GA. TC39 members are applying these tests to their browser implementations. The test site is accessible at http://test262.ecmascript.org.

The CC recommended to approve the TR on ECMA-262 Test Suite (GA/11/097).

Good progress on the next edition of ES (“Harmony”). Basically the list of features for the next edition has been frozen. Approval of the next major Edition is planned for 2013.

TC39 also raised the question 'how to deal with test contributions from a non-Ecma member' and possibly also Open Source SW contributions?
NOTE: The current version of the Test Suite does not contain any such contributions, so this is a question for future versions of the Test suite as well as for ES “Harmony”.

The GA had no questions on the report.

Then the GA voted on the new TR on ECMA-262 Test Suite (GA/11/097 – TC39/11/045)

In favour: 17 - unanimous

Against: none

Abstain: none

This Standard will be published as ECMA TR/104 1st edition.

Then a long discussion started about the implications of the Ecma Patent policy on TC39 and about some of the events that took place around the December GA, and since then:

Mr. Sebestyen reported that two set of patent statements have been submitted by Intel, and published as GA/11/126 and GA/11/127. They contain “blanket” RAND patent declarations from Intel related to all Ecma standards and Technical Reports submitted for approval to the June and to the December 2011 GA. There was in the GA a discussion on the Intel submissions. Some Ecma members did not understand why Intel submitted patent declarations to all Ecma Standards and Technical reports, and not to selected ones. **Mr. Wennblom** responded that according to the understanding of the Intel legal team of the Ecma patent policy this had to be done exactly so. Google and Yahoo requested for more details, but in the absence of those they disapproved most final drafts submitted for approval to the December 2011 GA. The above ECMA TR/104 was an exception to that.

Mr. Sebestyen said at the GA that he would meet with Intel on December 9 and hopes to get additional information that would allow better to understand the concerns. The goal is that those concerns should be brought to the attention of the CC, in order that they can be discussed there. At the time of the TC39 meeting this discussion with Intel was still going on.

TC39 was rather disappointed with this development. They confirmed again their intention to target related to ECMAScript for RANDZ-standards and -Technical Reports. They have felt that for base Web standards they have to follow that by the Industry well accepted practice (also followed by W3C and IETF) otherwise the standards would not be taken by the market.

Mr. Sebestyen has also reported about other Ecma IPR Policy matters discussed in the GA: On Text copyright matters (GA/11/120-Rev1) the GA approved a new copyright statement that is also suitable for translated and derived work.

The possible extension of the TC39 experimental software copyright policy was discussed: What about 3rd party (non-Ecma) contributions and what about use of free, open source software. **Mr. Sebestyen** confirmed that the GA has acknowledged the CC proposal:

- Ask TC39 for a (prioritized) list of issues including a detailed 'problem description' (work in TC39 will start in December / January*);(Not done yet…)*
- Re-establish the 'Software Copyright Ad-Hoc' group;
- Members of this Ad-Hoc group can be CC members, Ecma Management, plus invited IPR experts from any Ecma Member;
- The Ad-Hoc should review the input from TC39, finalize the problem statement and propose suitable solutions.

In the GA discussions it was brought up and agreed that the obligations of 3rd party members should not be different from the obligations of Ecma members both for contributions and for approved standards / technical reports. This should be true both for patent policy matters and software copyright matters. This should be brought to the attention to the IPR group.

In the GA **Ms Auber** observed that the current external contributors form on text standards is done in Ecma on a case by case basis by the different TCs. For software based standard such external contribution form does not exist.

In the TC39 discussion it was also reported about the CC discussion on the proposal to a slight change on the Ecma patent policy, namely disclosures of patents after the publication of a standard should identify the patents. Some TC39 member felt that even for non-late

disclosures the patents should be identified. **Mr. Sebestyen** reported that for the time being the proposal failed, there was no agreement on it. Also it was felt that the changes in the patent policy should be larger before a new revision is published.

Mr. Sebestyen then noted of the slight change of the Ecma By-laws and Rules (GA/11/119) by the GA:

- For the election process for Ecma Management and CC
- For proposals for new Work

Any new work item proposal in a TC or TG shall be supported by at least three members of whom there is at most one NFP.

Mr. Sebestyen points out about the internal Ecma TC39 documentation and archival is still not up to the requirements as they should, but since the last meeting some progress has been made.

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

Mr. Sebestyen reported that the TR/104 has been approved at the December GA. For the actual URL link a note points out that this has to be done in the final editing and publication process. That will be done after the approval of the TR. This was confirmed at the meeting but so far it could not be done. It was, however, confirmed that this takes a relative low amount of work, and a Microsoft “legal” go ahead to make the changes is expected in the near future.

David Fugate gave a report (see: Ecma/TC39/2012/003 Test262 status report, January 2012). He said that good progress had been made, 21 bugs have been closed, text is available for 100 more.

More details on:

http://wiki.ecmascript.org/doku.php?id=test262:test262

4.3 Report from the ad hoc on Internationalization standard

4.3.1 Review of proposed draft standard

On January 17, 2011 there was a separate ad-hoc group meeting on internationalization (i18n Ad Hoc group). **Norbert Lindenberg** and **Nebojša Ćirić** reported on it. The results of that meeting had been published as Ecma/TC39/2012/002 Notes of the ad hoc meeting on Internationalization, 17 January 2012.

The latest version of the draft Standard is published as Ecma/TC39/2011/057 Draft "ECMAScript Globalization API Specification", 1 December 2011.

As reported in the September meeting the spec has to ready for a GA approval in June 2012.

So the group is finalizing the draft document. The latest revision is available at:

http://wiki.ecmascript.org/doku.php?id=globalization:specification_drafts

4.3.2 Multi-system prototype testing

Still going on.

4.3.3 Next steps

On the request of TC39 the Secretariat has promised to reserve the ECMA-402 number for this standard. *This has been done after the meeting, the number can be confirmed.*

In order to approve the standards on June 27, 2012 by the GA the Ecma Secretariat has to publish it by April 27, 2012. This means that TC39 has to approve the standard at its March 28-29 meeting.

This means that the final draft of ECMA-402 should be distributed to TC39 by March 21, 2012.

After Ecma approval an immediate fast-track submission to JTC 1/SC 22 is planned.

4.4 IPR

4.4.1 Availability of TR and related software on Ecma Website

4.4.2 IPR ad hoc of the CC and Experimental Software IPR issues

See discussion under 4.1 above.

4.5 Update of TC39 Web Page at Ecma home page

On the TC39 website it was mentioned that an updated text will come from **Mr. Neumann**, based on that the TC39 web pages will be updated.

This was done and the TC39 web pages have been updated.

4.6 Status of ES 6 Draft Specification

Draft Specification for ECMA-262 Edition 6

Current Working Draft: **January 16, 2012**

Changes marked as “Rev 5”

This is an intermediate draft containing incomplete changes.

Changes include:

- Various proses descriptions and introductions have been converted into non-normative notes.
- Some preliminary work was done in support of super references and super calls (incomplete)
- Various editorial minor corrections
- Updates to function declaration instantiation in support of extend mode formal parameter scoping decisions made at Nov 2011 TC39 meeting.
- Changes to spread in destructing assignment in ArrayLiteral to conform to decisions made at Nov. 2011 TC39 meeting.
- Added static and runtime semantics for concise method properties in object literals.
- Converted ES5 algorithmic strict mode errors into declarative early errors for Unary and Postfix expressions
- Added is and isnt operators
- Eliminated “augments” as a possible formal parameter name for extended code

Mr. Wirfs-Brock has presented the latest draft from the Wiki. It has the date January, 2012. The draft is marked with “Rev. 5”. He explains all the changes he has made to the previous version and requests TC39 members for feedback.

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

See for details **Waldemar Horvat**`s technical notes.

5.1 Versioning

See the presentation Ecma/TC39/2012/005 “One JavaScript” by **Dave Herman**. This was discussed at length in the meeting. See for further details **Waldemar Horvat**´s notes below.

5.2 Modules and Binary Data: (TH)

5.3 Object exemplars (Allen)

5.4 Decoupling [ ] and property access (Allen)

5.5 Magic Proto Property

http://wiki.ecmascript.org/doku.php?id=strawman:magic_proto _property

5.6 Override Mistake

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

5.7 Spec Drafting priority

5.8 Block Lambda

6 Date and place of the next meeting(s)

March 28-29, 2012 Google (Silicon Valley)

May 23-24, 2012 North-Eastern University (Boston)

July 25-26, 2012 Microsoft (Redmond)

September 24-25, 2012 Mozilla (Silicon Valley)

November 28-29, 2012 Apple (Silicon Valley)

7 Closure

The TC39 Meeting ended at 4:30 PM on 19 January 2012. **Mr. Neumann** thanked the meeting participants for their good contributions, constructive discussions and the co-operative spirit of the group.

The group expressed appreciation to Yahoo! and to **Doug Crockford** for hosting the meeting and Ecma international for hosting the TC39 dinner on the 18th January in Santa Clara, CA.

Item 5 Attachment

Technical meeting notes from Waldemar Horwat:

From: waldemar@google.com
To: es-discuss@mozilla.org
Sent: 1/18/2012 8:27:38 P.M. Eastern Standard Time
Subj: Jan 18 meeting notes

My rough notes from today's meeting.

Waldemar

DaveH: One JavaScript (Versioning debate)
It's inevitable (and necessary) that ES6 will have some breaking changes around the edges. How to opt-in?
DaveH's versioning proposal: A module block or file include is the only in-language ES6 opt-in. Modules can appear only at the top level or inside another module. This avoids the problem of a "use strict" nested in a function inside a with.

Brendan:
```
var obj = get_random_obj();
var x, prop = 42
with (obj) {
  x = function() {
    "use strict";
    return prop;
  }();
}
```

Differences between the de facto (traditional) semantics and ES6 (i.e. semantic changes instead of mere syntax additions):
- ES5 strict changes
- static scoping (really means static checking of variable existence; see below)
- block-local functions
- block-local const declarations
- tail calls (yikes - it's a breaking change due to Function.caller)
- typeof null
- completion reform
- let

DaveH: Thinks we may be able to get away with enabling completion reform and let for all code.

Allen: Would a class be allowed outside a module?
DaveH: Yes, but it would not support static scoping, block-local functions, etc.
MarkM: Classes should not be allowed in traditional semantics. If you want a class, you need a "use strict" or be inside a module.

Waldemar: Given that you can't split a module into multiple script blocks, making modules be the only in-language opt-in is untenable. Programmers shouldn't have to be forced to use the broken scope/local function/etc. semantics to split a script into multiple script blocks.
DaveH: Use out-of-language opt-ins.

MarkM: Wants a two-way fork (non-strict vs. strict) instead of a three-way fork (non-strict vs. strict vs. ES6-in-module).

MarkM: Does a failed assignment inside a non-strict module throw?

DaveH: Most of the differences between strict and non-strict are code bugs.
Luke, MarkM: No. Their developer colleague experience shows that there are plenty of changes to non-buggy code that need to be made to make it work under strict mode.

Allen, Waldemar: It's important to support the use case of someone writing global code using the clean new semantics and not having to learn about the obsolete traditional compatibility semantics.

Can "use strict" be the ES6 opt-in?

What DaveH meant by static scoping (i.e. static checking):
What happens when there's a free variable in a function?

Nonstrict ES5 code:
- Reference error if variable doesn't exist at the time it is read; creates a global if doesn't exist at the time it is written.

Strict ES5 code:
- Reference error if variable doesn't exist at the time it is read or written.

Static checking goal for ES6 modules:
- Compilation error if variable doesn't exist at the time module is compiled.
- Reference error if variable doesn't exist at the time it is read or written.
(It's possible to get the latter error and yet have the module compile successfully if someone deletes a global variable outside the module between when the module is compiled and when the variable is read or written at run time.)

Discussion of whether it is important to support non-statically-checked binding in modules.

MarkM: typeof is used to test for the existence of globals. If the test succeeds, they then proceed to use the global directly. This would then be rejected by static checks.

DaveH: Doesn't see a way to do static checking with strict code (due to, for example, the "with" case illustrated by Brendan earlier).

MarkM: The cost of having three modes is higher than the cost of not supporting static checking early errors.

DaveH's new proposal: Other than static checking, attach the incompatible ES6 semantics to the strict mode opt-in. These semantics are upwards-compatible with ES5 strict mode (but not ES5 non-strict mode). The semantics inside a module would be the strict semantics plus static checking.

Do we want other new ES6 syntax normatively backported to work in non-strict mode?
Waldemar, MarkM: Not really. This requires everyone to be a language lawyer because it's introducing a very subtle new mode: ES6 with nonstrict scoping/const/local semantics. If an implementation wants to backport, the existing Chapter 16 exemption already allows it.
DaveH, Brendan: Yes. People won't write "use strict". Don't want to punish people for not opting in.
Alex: Split the middle. Backport new ES6 features to non-strict features where it makes sense.

Waldemar, DaveH: Want to make it as easiy as possible to make a strict opt-in for an entire page instead of littering opt-ins inside each script.

Allen: Backporting increases spec complexity and users' mental tax. The main costs are in making lots of divergent scoping legacy issues possible.

Doug: Modules are sufficient as an opt-in, without the need for a "use strict" opt-in.
Waldemar: No. Having multiple scripts on a page would require each one to create its own module, and

then issues arise when they want to talk to each other -- you'd need to explicitly export const bindings, etc.
MarkM: No. The typeof test won't work. Also, this would make it impossible to write code that's backwards compatible with older browsers that don't implement modules.

Which ES6 features can be backported into non-strict mode?
(blank: no significant issues;
?: possible issues;
x: semantics conflict with de facto web)

? let (syntax issues)
x const (divergent semantics)
x function in block (divergent semantics)
? destructuring (syntactic conflict with array lookup)
parameter default values
rest parameters
spread
x tail calls (because of Function.caller)
direct proxies
simple maps and sets
weak maps
is / isnt (egal)
iterators
? generators (interaction with scoping issues and yield keyword)
generator expressions
comprehensions
private names
quasi-literals
pragmas (controversial)
? completion reform (Brendan: might be able to get away with it without breaking the web, but we don't know yet)
x typeof null (Erik: It breaks the web)
class
super
n/a modules
methods in object literals
<|
({[computed_name]: value}) (MarkM: what should happen with duplicate names in nonstrict mode?)

Brendan: Kill typeof null. Replace it with Ojbect.isObject?

How are the names introduced by generators and classes scoped in nonstrict mode?

MarkM: Example of code that might accidentally work one way in all current browsers but not in ES6:
(another foo is also predefined in an outer scope.)

```
if (...) {
  function foo() {...}
} else {
  function foo() {...}
}
foo();
```

// foo will call one of the two foo's defined above in ES5 nonstrict, although it's implementation-dependent which. On some browsers the first definition wins; on some the last definition wins; on some the one which corresponds to the true branch of the if wins. In ES6 strict it will call the outer scope foo.

Discussion about whether we can move the web to the new local function and const semantics even in nonstrict ES5 mode. Also discussed an alternative of whether we can require nonstrict mode to support limited usage scenarios such as having the following work:

```
if (...) {
  function foo() {...}
  ...
  foo();
}
```
Waldemar: This doesn't work in current ES5 non-strict if the if is inside a with statement because an existing implementation might hoist foo across the with and then foo() could refer to a field of the with'd object. This also might not work in the presence of other definitions of foo inside the same function.

Not clear if specifying such limited cases in the normative spec is useful.

Current tentative decision is to support let, const, and local functions in nonstrict ES5 in the same way as in strict ES6. Fallback to either specifying limited cases or doing the ES5 nonstrict status quo (i.e. syntax error + Clause 16) if experiments show this to not be viable. We won't resolve this discussion without running some experiments.

Tail calls:
Luke: Remove them altogether.
Waldemar: If we support them only in strict mode, the failure mode is someone copying-and-pasting code from a module to the global level and silently losing tail recursion, leading to very confusing behavior.
Debated.
Waldemar: We can require tail calls in non-strict mode by taking advantage of the fact that Function.caller only remembers the last invocation of each function. Thus we can do an amortized algorithm analysis that allocates the cost of storage of one stack frame link at the time we create the function. This makes it possible to implement tail calls in non-strict mode while supporting Function.caller.
Tentative decision is to support tail calls in strict mode only.

Resolved: Named generators behave in non-strict mode the same as in strict mode. "yield" is a contextual keyword in non-strict generators.

How to resolve let[x] = 5 in nonstrict mode? Will need to do experiments.
Also, do we require no-line-terminator between "let" and the identifier? Probably not, because if we do then we'd get this annoying hazard in non-strict mode:

```
{
  let x = 7;
  if (...) {
    let
      x = 5;
  }
  // Now x is 5 because the second "let" is just a useless identifier expression with an inserted semicolon, followed by an assignment to the existing x!
}
```

Gavin: Module syntax hazard if we have no-line-terminator between "module" and the identifier:

```
module
{
  ...
}
```
gets interpreted as a useless expression (the identifier "module") followed by a block.

Completion value reform: Let's experiment.

Octal constants:
Useful as arguments to chmod.
Proposal for 0o123 (as well as 0b01110).
MarkM: Concerned about 0O123.

Waldemar: Nothing unusul here. We've lived with 36l (meaning 36 long instead of 361) in Java and C++ for a long time.
Alternative for octal: 8r123 (but then we'd also want 16r123, 2r0101, and maybe more).
Decided to allow 0o and 0b. Unresolved whether to allow 0O and 0B. Persistent weak feelings on both sides on the upper case forms.

Use __proto__ in object literals to do a put (assuming that a __proto__ getter/setter was created in Object.prototype) instead of a defineProperty? All modes or only nonstrict mode?
Allen: Make such use of __proto__ to be a synonym for <|. If a <| is already present, it's an error.
DaveH: __proto__ is ugly. Don't want it in the language forever.
Waldemar: What about indirect [] expressions that evaluate to "__proto__"? In Firefox they evaluate to accesses that climb the prototype chain and usually reach a magic getter/setter-that-isn't-a-getter-setter named __proto__ that sits on Object.prototype.
MarkM: Likes the ability to delete __proto__ setter and thereby prevent anything in the frame from mutating prototypes.
Waldemar: How do you guard against cross-frame prototype mutations?
DaveH: __proto__ is in the "omg, what were we thinking" category.
Waldemar: Opposed to making __proto__ mutate prototypes other than at object construction. This is getting insanely complex.
Unresolved.

Second day meeting notes:

Revisited octal/binary constants.
Waldemar: Note that we currenty allow both upper and lower cases for a, b, c, d, e, f in hex literals as well as e in exponents and x in hex literals. Breaking symmetry by forbidding upper case for b and o to avoid lookalikes would be "nanny language design". We can't forbid lookalikes anyway:

```
{
  let y = 3;
  {
    let Õ = 7;
    return y;
  }
}
```

This is a valid ES6 strict function body that returns 3, of course. The second let defines a local variable named with a Cyrillic letter.

Decision: Allow upper case B and O to start binary or octal literals.

Complex discussion about what's feasible in static checking of references from modules. Discussed issues with integrating script code eval'd from cross-origin XHR requests and the (anti-)merits of using var to declare globals.

MarkM: Abandon static checking.
Waldemar: MarkM's example from yesterday is an important use case: using a typeof check to see if a variable is defined and later in the same module conditionally dynamically defining it if it wasn't and, even further down in the module, using it. A static check would flag the uses and prevent the module from loading.
Allen: This is a good use case for pragmas.

MarkM, Waldemar: Example of a successful strict transition: Nowadays the two of us don't even know non-strict Perl because it's too weird. We learned the cleaner strict version, always set the pragma, and stay within the strict version. Anything non-strict is considered to be broken until it can be upgraded to strict.

Philosophical discussion about modality, whether a "mode" should be called a "dialect", etc.

Summary of static checking pain points that need resolutions:
- Out-of-order asynchronously loaded scripts
- Concatenation of scripts
- Dynamic testing for variable existence and conditional use if it exists
Resolution: Will do more work and come back to this area.

Allen: Do top-level let and const create properties of the global object?
DaveH (channeling Andreas): No, but functions do.
Sam: If they don't then the T-shirt Erik is currently wearing wouldn't work.
Many agreed that we can't break the T-shirt.
DaveH: What if there are duplicate top-level bindings?
MarkM: Top-level let should behave as much as possible the same as concatenated scripts.
Waldemar: What about read-eval-print loops?
MarkM: Each eval inserts one more nested scope.
Waldemar: What if someone does this:
> let x = 3;
> function foo() {return x;}
> let x = 7;
This would leave foo returning 3.
MarkM: That's the proper behavior. Anything else has problems.
Waldemar: [incredulous]
Allen: We shouldn't spec read-eval-print loops.

DaveH's summary of issues:
- Can we get to a world where var is completely unnecessary?
- Can we do the above for all important top-level patterns?
- Can we do the above without ever needing nonstrict mode?
- Should let have consistent semantics in all contexts?
Consensus on:
- Should multiple scripts or scripts and attributes share lexical scopes other than via global objects? No.
- Consensus that concatenating multiple scripts via scope nesting semantics is undesirable.

Sam: What about global scope polluters?
let Object = 7;
should be an error. But implementations have other sources of polluters (such as predefined implementation features or other HTML elements on the page) that can cause global clashes.
Allen: Top-level declaration overriding is allowed if the existing property has the attributes writable:true and enumerable:true (and the new declaration stays enumerable). Configurable doesn't (sic) enter the decision.
Allen: See https://bugs.ecmascript.org/show_bug.cgi?id=78
DOMs now declare properties on the prototype of the Window object.

Internationalization report
Discussion about a name (and domain name) for the internationalization test suite. 402 is the current next unused ECMA standard number. We probably shouldn't wait for two more ECMA standards to be published.... Mild consensus on putting the test suite in a subdirectory of test262.
Discussion about whether to keep internationalization as a separate document or to merge it into ECMA-262. Current decision is to keep it separate for faster evolution of both standards and to avoid bureaucracy with ISO.
Discussion about the choice of name for the global variable via which the internationalization API is accessed. Possibilities are Globalization and Intl. Need to look for web breakage these would cause.
Everybody needs to review internationalization spec before the vote in March.

Debate over whether to use TypeError or RangeError for domain errors.
DaveH: Use TypeError
Allen, Waldemar: Use RangeError. Note that RangeError is used for more than just numeric intervals -- passing 3.5 to a function that expects an integer in [0, 2^32) throws a RangeError in ES5.1.

Test262 status report: No invalid test cases left.
MarkM: Want some means to avoid having test262 penalize implementations that anticipate incompatible breaks that have been approved by the committee (possible example: completion reform in ES6).
Allen: Distinguish between spec bugs and breaking spec revisions.
DaveH, Sam: In reality the test suite provides political pressure. Want to avoid applying that pressure for features that we decided to break in the future spec.
Alex: Create a category of deprecated tests that, just like the nonnormative ones, run but don't factor into the score.
DavidF (in response to discussion assertions about test262 always tracking only the latest ECMAScript version): Microsoft wants to retain an ES5.1 variant of test262 even after ES6 is released.
Consensus: Create a third category of tests for features that are "no longer endorsed".

Mozilla hasn't signed a contributor agreement yet. Waiting for having something to contribute.

Istvan's report about the chaos that Intel caused at the last GA meeting and the developments since then.
Allen: We should formalize the RANDZ for TC39.
Intel will be here for the next meeting of TC39.
TC45 also had an understanding that their contributions would be RANDZ.
MarkM: If we can't formalize RANDZ in TC39 somehow, several members will desire to take the standards activity to an organization that can.
Varied and long discussion about IPR policy developments.

ECMAScript® got registered as a trademark in Switzerland.

At next meeting MarkM will present a tiny API proposal for maps and sets.
Allen: Reset agenda for next meeting. Don't include zombie carryovers.
Brendan: Gavin is working on classes.

ArrayTypes:
```
var Int8Array = new ArrayType(Int8);
var arr1 = new Int8Array(100);
var arr2 = new Int8Array(256);

var MyStruct = new StructType({extra: Int8Array(100)});
```
Note that Int8Array does different things when called with or without new. Debated how to do this -- specialize on new (can be done via proxies) or on whether the "this" value is undefined or not.

Allen, DaveH: Classes should support the ability to express APIs such as this one.

Waldemar: How would this work?
```
let t = Int8Array(100);
var arr3 = new t;
```
Luke: It wouldn't.
Some debate ensues.
DaveH: There should be two kinds of array type objects: sized and unsized. Calling an unsized one as a function with a length argument returns a sized array type object. Can call new on the resulting sized array type to obtain an array instance. Luke now agrees.

Can construct an array value as a view over part of an ArrayBuffer.
Waldemar: This brings up endian issues.
Luke: Array buffer is exposed only if you allocate an array using an explicit ArrayBuffer. The forms that create arrays without explicit ArrayBuffers don't expose endianness.
DaveH: The endianness ship has sailed with TypedArrays. It's implementation defined.
DaveH: People use this to aid with file I/O.
Waldemar: The two statements above contradict each other. All implementations will want to make the

same choice to avoid nondeterministic behavior, particularly if people use it for file I/O. We should pick the main one and mandate it.
Brendan: It's already specified as little endian for arrays and big endian for data view.
Sam: Looked at the Kronos spec and it doesn't say anything about non-data-view endianness. It's not specified.
DaveH: We should try to spec endianness if possible.

Debate over whether the following should be an error in strict mode (or "extended mode" if there is such a thing). It isn't an error in ES5.1 strict.
function (e) {var e;}
Consensus: No, we should not change strict mode incompatibly like this. This is valid in all modes.
Brendan: It should be an early error to have a var hoist across a catch-block with the same variable name. Also, "function (e) {let e;}" should always be an error.

Allen, Sam: Module mode will be the same as strict mode, except possibly for additional free variable static checks.

Discussed Allen's list of current ES6 extended mode breaking changes from the "ES6 opt-in, reloaded" thread.
Consensus: For each of these, either we introduce it in all code (maybe all strict code) or we don't do it.

Discussion about scope of for-bindings.
for (var x = ...;;) {...} will, of course, retain ES1 semantics.
for (let x = ...;;) {...}
Allen: This will behave as in C++: x is bound once in a new scope immediately surrounding just the for statement.
DaveH: Strangely enough, this creates a new x binding in Dart at each iteration.
There's an alternative semantics that creates an iteration-local second x inside the loop and copies it back and forth. Debate about whether to go to such complexity. Many of us are on the fence.
Waldemar: What happens in the forwarding semantics if you capture the x inside a lambda in any of the three expressions in the head?
If this happens in the initializer:
DaveH's option: The lambda would capture an outer x.
Alternative: The lambda captures a hidden second x.
Waldemar's option: The lambda would capture the x from the first iteration. The let variable x is bound once through each iteration, just before the test, if
```
  for (let x = expr1; expr2;) {...}
```
were:
```
  while (true) {
    let x = first_iteration ? expr1 : value_of_x_from_previous_iteration;
    if (!expr2)
      break;
    ...
  }
```
MarkM: Just discovered that his desugaring has the same semantics as Waldemar's option.

for (const x = ...;;) {...} behaves analogously to let, whatever we decide let will do.

Those opposed earlier to new-binding-per-iteration hop back onto the fence. Waldemar, Brendan, and DaveH hop off the fence to now support it, as it will cure an annoying gotcha in practice. Looks like we have support for it now.

End of meeting.
