Minutes of the: ***20th meeting of Ecma TC39***

held in: ***Sunnyvale, CA, USA***

on: ***19 – 20 January 2011***

1 Opening, welcome and roll call

1.1 Opening of the meeting (Mr. Neumann)

The meeting (hosted by Yahoo! Classroom 3 in Building C (second floor), 700 First Avenue Sunnyvale, CA, USA) was opened by **Mr. Neumann**, Chair of TC39 at approximately 10:00 AM on 19th January 2011 (2010/064-Rev.1): Venue for the 20th meeting of TC39, Sunnyvale, January 2011).

1.2 Introduction of attendees

Douglas Crockford – Yahoo

John Neumann – (Microsoft, Mozilla, Yahoo)

Rex Jaeschke – SC 22 Chair

Nebojsa Ciric – Google

Mark Miller – Google

Cormac Flanagan – UCSC

Sam Tobin-Hochstedt – Northeastern University

Dave Herman – Mozilla

Waldemar Horwat – Google

Allen Wirfs-Brock – Mozilla

David Fugate – Microsoft

Luke Hoban – Microsoft

Alex Russell – Google

Erik Arvidsson – Google

Martin Maly – Google

Peter Hallam - Google

Istvan Sebestyen – Ecma-International (by phone)

Peter Constable – Microsoft (by phone)

Tom Van Cutsen – Google (by phone)

1.3 Host facilities, local logistics

Mr. Crockford welcomed on behalf of Yahoo! the delegates and provided logistical information. It was announced that Ecma international would host a social event on January 19th evening.

2 Adoption of the agenda (2011/001-Rev.1)

Adopted as presented, noting meeting start time on the 20th is also 10:00 AM.

3 Approval of Minutes from November (2010/063)

Minutes approved as presented.

4 Report from the Secretariat

Mr. Sebestyen has reported about the December 2010 GA meeting based on GA/2010/175:

4.1 Review GA results of importance to TC39

"7.7 TC39 - ECMAScript
Activity report: TC39/10/056.

Ms Valet-Harper has reported about the activities of TC39. Lot of activities, it is a large, productive and active group.

Dr. Sebestyen reported that the JTC 1 fast-track on ECMAScript Edition 5 (ECMA-262, or ES5) was submitted to JTC 1 a year ago. It is the last Ecma standard that follows the old JTC 1 fast-track approval process (the "magic time" for the change from the "old" to new" process was July 1, 2010 - where ongoing fast-tracks followed the old rules, and new submissions would follow the new rules). In the JTC 1 ballot (that ended early August 2010) the standard got several comments (from Japan, Russia and Ecma International) but only one negative vote (from Japan). He said that there was a lot of interaction between the Ecma Secretariat, the JTC 1 Secretariat and the SC 22 Secretariat on the procedural issue how to deal with this fast-track. Since in the proposed disposition of comments (deliver to SC 22 mid November 2010) all of the Japanese comments were taken positively into account he thinks that a BRM might be avoided if the proposed disposition of comments report would be approved by Japanese NB. The Japanese NB has promised to give a final reply on that by the end of 2010. If no BRM would be needed JTC 1 would go to final publication. If there was a BRM, it would take place on January 18, 2010 in the San Jose, CA area. If the standard got technically changed by the BRM a new 2 months ballot would be needed - was explained to the Ecma Secretariat.

The JTC 1 edition of the ECMAScript5 specification incorporates a number of editorial and technical corrections including those listed in an ES5 errata.

In order to keep the JTC 1 and Ecma standards in strict alignment TC39, has prepared a revision to the ES5 spec. whose content is identical to the JTC 1 version. It also includes a new Annex F that lists the technically significant changes incorporated into the revision. This revision will be known as ECMA-262, Edition 5.1 (or ES5.1).

The final draft of the ES5.1 spec. is available on the TC39 web pages. TC39 is continuing its longer term work on "ECMAScript Harmony" which is intended to be the next version to include any new features (planned for 2013).
It was reported that a new TC39 project about "Internationalization" has started.

Ms Valet-Harper reported that the browser implementation testing activity of ECMAScript Edition 5, which is a public process (based on W3C example), is well received and will continue.

The GA congratulated TC39 and its Chair for their fine work."

TC39 took note that due to the hard work of all involved, the BRM on ES5 was not needed, as all issues had been resolved in a satisfying way. Both **Mr. Neumann** and **Mr. Sebestyen** expressed their gratitude to TC39 and especially to the Editor **Mr. Wirfs-Brock** for their hard and excellent work.

Mr. Sebestyen also mentioned that in terms of standards download ECMA-262 is still the number 1 download from the Ecma website, where only this standard accounted for some 28% of all downloads. The downloads of all Ecma standards in 2010 were about 103,000.

Regarding the question on the trademark status of ECMAScript **Mr. Sebestyen** will contact the Ecma Trademark lawyer.

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

The meeting reviewed the status of co-operation with Khronos: No new activity, however, it has been noted by Secretary General that regarding IPR policies there is no apparent impediment to working with them in the future. No additional agreements required.

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

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

Report follows at the end of minutes as Attachment 1.

4.4 Report from the ad hoc on Internationalization standard.

(The meeting of the Ad hoc Group in Internationalization has taken place, one day before the TC39 meeting, on January 18, 2011 also at the Yahoo! premises)

- AHG: The work should be a separate standard, and we should approach ISO only when substantial part of the standard is implemented (missing parts).
- Some of the work might better be done as a part of the core language
- TC 39: Proceed on separate standards track under tc39, i.e., to result in a separate Ecma standards document with its own number. There are various points of contact between this future internationalization standard and the ongoing work on EcmaScript 262. The points of contact (such as the *locale* methods) are manageable, and tc39 as a whole deciding which issues to solve within which of these standards.
- TC 39: No intention to fast-track at this point (ISO)
- We would like to try to drive for an early timetable for this first version: agreeing on a spec that we can start implementing against by June of this year.
- Pruned down current proposal to a functionality that can be implemented in short time on all platforms.
    - LocaleInfo
    - Collation
    - Date and time formatting but no parsing
    - Number formatting but no parsing. Needs additional discussion about currency and precision
    - No parsing, timezones, complex patterns/skeletons, normalization, case folding, regexes (UCS2 vs UTF16)
- Focus of the meeting was on feasible functionality and making sure that API design mashes well with JavaScript language (Mark, Allen and Doug helped out with that)
- Made a top level, aggregator object, LocaleInfo and put all other parts of API under it
    - Avoids pollution of global namespace
    - Easier to pass it around
- LocaleInfo object can serve inferred values if developer constructs it so (currency, region from locale) which is somewhat dangerous, but useful. We would let developer know if a value is inferred or explicitly set.
- All objects in the API are immutable, but can be cloned/derived to create variations using derive function
- Throw errors, don't return undefined - one can better diagnose the problem
- Long discussion about default locale and what it means for various deployments
    - Server side doesn't mean much, one has to set the locale for proper response
    - Client side would benefit from having system locale as default locale

- Browsers/webapps would benefit from having default locale set from external source (lang param from URL, cookie) - system/browser locale doesn't make make much sense
- Mozilla team presented localization proposal for message formatting, that reminds of ICU message formatting and is going to be used in Mozilla localization. Their work may be included in one of the iterations of the standard produced by AdHoc group.

4.5 W3C Joint work items

No related work activities at this time.

5 Progression of ES 5

5.1 Editor's Proposed Disposition of Comments Report

Historical review was given by the Editor, Allen Wirfs-Brock.

5.2 Report from the BRM (if held)

As mentioned above, no BRM was held.

5.3 Next steps

There is unanimous approval to forward ES 5.1 to the CC for recommendation to the GA in June 2011 for approval. This document aligns with the final approved ISO 16262 Edition 3 standard.

TC39 also expressed its strong appreciation to Mr. Wirfs-Brock for the outstanding job he did as editor of ES 5 and ES 5.1.

6 Discussion of ES harmony (technical contributions are available and can be found on the ES wiki – they have been captured with the date of January 24, 2011 as TC39/2011/009)

Informal notes of technical discussion attached as Attachment 2.

6.1 A new strawman Private Name proposal

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

Delayed until next meeting.

6.2 "Soft Fields" and "Names vs Soft Fields"

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

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

Delayed until next meeting

6.3 guards

<http://wiki.ecmascript.org/doku.php?id=strawman:guards>

6.4 "Private Names"

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

Duplicate of 6.1

6.5 The revised API is documented at:

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

6.6 parameters property of functions

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

a) Proper tail calls
http://wiki.ecmascript.org/doku.php?id=strawman:proper_tail_calls

b) Regexp x Flag
http://wiki.ecmascript.org/doku.php?id=strawman:regexp_x_flag

c) RegExp y Flag
http://wiki.ecmascript.org/doku.php?id=strawman:regexp_y_flag

d) JSON.path
http://wiki.ecmascript.org/doku.php?id=strawman:json_path

e) Object.isObject
http://wiki.ecmascript.org/doku.php?id=strawman:object_isobject

f) Array.create
http://wiki.ecmascript.org/doku.php?id=strawman:array_create **and**
http://wiki.ecmascript.org/doku.php?id=strawman:array_subtypes

g) Number.isFinite
http://wiki.ecmascript.org/doku.php?id=strawman:number_isfinite

h) Number.isNaN
http://wiki.ecmascript.org/doku.php?id=strawman:number_isnan

i) String.prototype.dup
http://wiki.ecmascript.org/doku.php?id=strawman:string_dup

j) String.prototype.format
http://wiki.ecmascript.org/doku.php?id=strawman:string_format

7 Date and place of the next meeting(s)

The meeting agreed on the following schedule:

March 22 – 24 Location San Francisco (Google)

May 24 - 26, at UCSC Santa Cruz

July 26 - 28 or 27 - 28 at Microsoft Redmond

September 28 - 29 at Mozilla

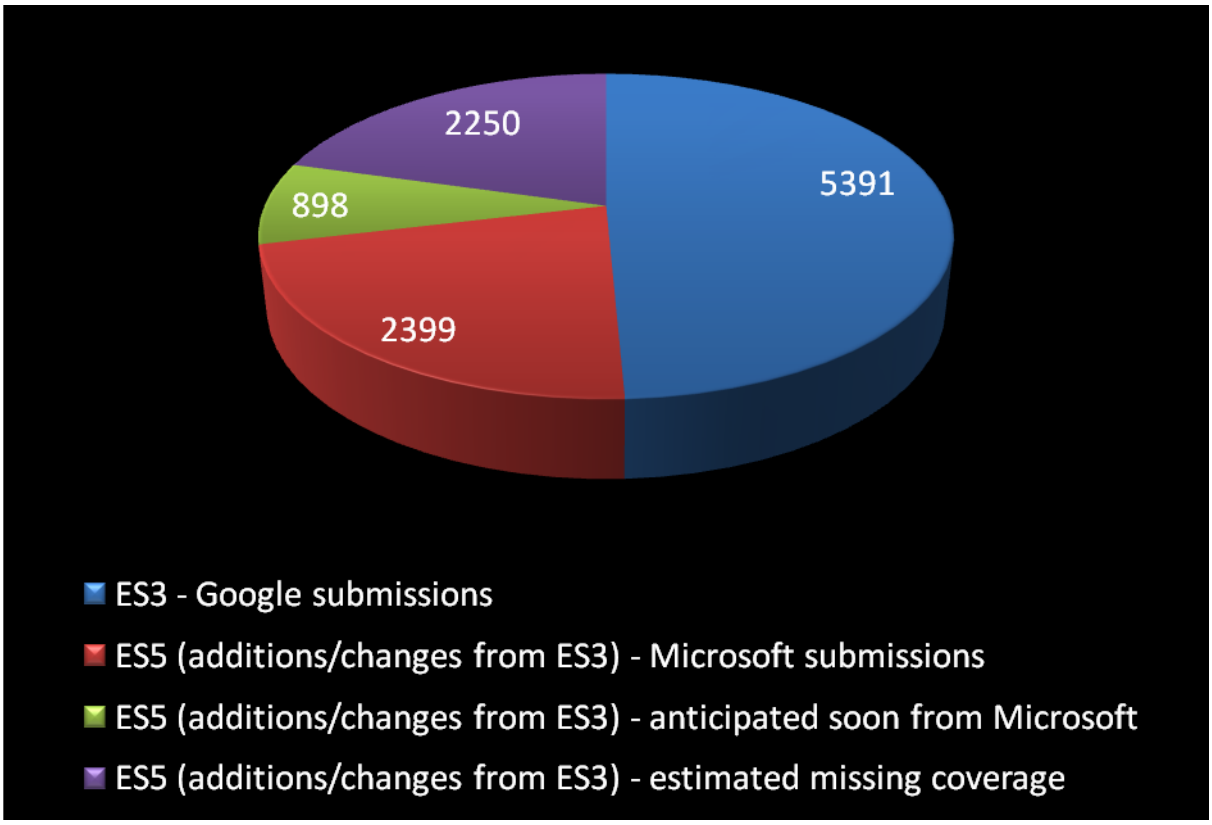
November 16 - 17 at Apple

8 Closure

Mr. Neumann expressed his appreciation to Yahoo! for the meeting facilities and to Ecma-International for the gracious dinner on Wednesday. The meeting closed of Thursday around 4:15.

Attachment 1: 4.3.1 Status Summary

Test262 will soon contain 8,688 tests (7,790 tests as of today) comprised from Google and Microsoft submissions:



With the following breakdown by chapter:

ES5 Chapter	Google Submissions	Microsoft Submissions	Total Submissions
Chapter 7	471	0	471
Chapter 8	114	0	114
Chapter 9	130	0	130
Chapter 10	94	16	110

Chapter 11	**1088**	**53**	**1141**
Chapter 12	**417**	**27**	**444**
Chapter 13	**112**	**0**	**112**
Chapter 14	**5**	**0**	**5**
Chapter 15	**2960**	**3224**	**6184**
Annex C	**0**	**0**	**0**
Total	**5391**	**3320**	**8711**

*Disclaimer: Rating of red, yellow, green assumes Google submissions are complete with respect to ES3

With more contributions, we believe we can be much closer to complete “Beta quality” coverage of the ES5 specification within a few months.

Sections Missing *ES5* Test Coverage Outright

7.6.1	**12.6.4**	**15.1.1.1**	**15.2.4.2**	**15.4.4.12**	**Annex C**
7.8.5	**12.7**	**15.1.1.2**	**15.3.4.5.1**	**15.9.1.15**	
11.8.2	**12.8**	**15.1.1.3**	**15.3.4.5.2**	**15.10.2**	
11.8.3	**12.9**	**15.1.2.2**	**15.4.4.4**	**15.10.4**	
12.6.3	**13.2**	**15.1.2.3**	**15.4.14.10**	**15.11**	

Sections with Insufficient *ES5* Test Coverage

15.2.3.2	**15.2.3.10**	**15.4.3.2**	**15.4.4.19**	**15.5.4.20**
15.2.3.5	**15.2.3.12**	**15.4.4.15**	**15.4.4.20**	**15.9.4.4**
15.2.3.7	**15.2.3.13**	**15.4.4.16**	**15.4.4.21**	**15.9.5.43**
15.2.3.8	**15.2.3.14**	**15.4.4.17**	**15.4.4.21**	**15.12.2**
15.2.3.9	**15.3.4.5**	**15.4.4.18**	**15.4.4.22**	

Call to Open Test Submissions to Public

The submissions thus far have been critical to getting this project bootstrapped and I want to thank Google and Microsoft for them. We still have a long way to go. As you can see from the table on the previous page of this report, there are still areas of the specification with no tests as well as areas that don't have as much coverage as we want. I'd like to encourage others to contribute tests to the suite, particularly in the areas where there is no coverage today, such as strict mode. Microsoft will continue to contribute tests to the suite over the next few months to help round out other areas.

With that said, the suite is now in a position where it is more than ready for members of this technical committee to work with it, send feedback to the alias, and open bugs.

Please note that there's an open issue around bugs.ecmascript.org that Dave Herman from Mozilla is working to resolve.

Based on the projected rate of contributions I believe we will be on track to have good (shallow but broad) coverage of the entire ES5 specification by the next committee meeting for us to review. I recommend that we drive to opening the suite to the community more broadly with an open beta starting after the next committee meeting to help get more people looking at the progress and finding issues with the suite and harness.

Discussion of test suite host site: Test262.ECMAScript.org

Test262.ECMAScript.org page is complete. As discussed at the last committee meeting, the results page has been removed. Moving toward Beta, we recommend that the Test262. ECMAScript.org site should be made public and a link added to that site from the development page of ECMAScript.org.

Attachment 2: Meeting informal technical notes from Waldemar Horwat:

Internationalization standard (4.4): Part of E262 or separate track? Pros and cons to each one, and either would be workable. There is a substantial area of interaction (ES5 locale methods, normalization, and such) between them that will need to be addressed regardless of which approach we take.

Lunch discussion over whether we want all locale-specific behavior (examples: date formatting) to be implementation-defined or whether we'd want to specify it for at least the major locales. Having everything implementation-defined makes testing a hassle and will result in different behavior on different platforms/browsers (as is happening today). On the other hand, specifying the results for a lot of locales is a lot of work.

Waldemar: Either would work. Personal preference is to make it part of E262 if it's small (in terms of number of pages of standard) or make it separate if it's large.

Agreed not to fast-track internationalization library (if it's a separate standard) for now. It's going to be evolving too quickly.

(5.3) Allen: Update on ISO fast track and ES5.1. Applause.

TC39 requested a vote on ES5.1 at the upcoming summer GA.

Allen: There's a significant community of users who are used to the classical patterns of encapsulation (rather than using closures for all encapsulation).

Waldemar and Dave: Important to be able to late-bind design decisions and change them easily. This means that a programmer should be able to relatively easily take code that uses a public object property and refactor the code to make it private (or vice versa) without having to restructure the code.

From the mailing list discussions, we have, roughly speaking, two main proposals:
A. Treat private properties like internal properties of objects. Don't allow sticking private properties after the fact on unaware objects such as frozen ones.

B. Treat private properties as syntactic sugar for weak maps. Allow private properties on any object that works as a key for such a map.

Waldemar: B inherently supports strong encapsulation. A doesn't as currently proposed due to its reflection API, but there is no inherent reason why the reflection API needs to pierce privacy.

Discussion on role of reflection API, and what privileges you'd need to use it. Java private fields may or may not be readable via the reflection API depending on the security settings. Most of those seem to fall under the category of being debuggers/profilers/white-box-tests that can have special privileges.

MarkM, Waldemar: Prefer separate (unavailable without special privileges) debugging API for reflecting on private properties.

(6.1 and 6.2 deferred to March or May)

Guards (6.3)

Dave: This is abstract. Also, would rather do a general user-extensible syntax for everything, which is perfectly doable as part of the modules proposal.
Waldemar: Defining a general user-extensible syntax is far more abstract and seems like a pie in the sky. Would want to see a solid proposal before even considering such a thing.

In function type guards, should later guards be able to depend on the values of earlier parameters? MarkM: No.

:: syntax seems fine.

Waldemar and MarkM will revive one of the past proposals.

(6.4 is duplicate of 6.1)

(6.5 deferred to Morning of 20th.)

Parameters property of functions (6.6)

Doesn't interact well with code minimizers. Currently can extract names of local parameters from a function's toString, but a number of us like that this is difficult, as it discourages features that make alpha-renaming refactoring difficult.

Allen prefers and wrote a mirror interface proposal on the same wiki page.

Briefly discussed relation to passing parameters by name. Brendan: passing object literals is now simple enough that an extra feature for passing parameters by name has little demand. Example:

```
function f({foo, bar}) {...}

f({foo, bar});
// Same as f({foo: foo, bar: bar})
```

Tails calls: (6.6a)

Waldemar concerned about interaction with guards, which would make tails call no longer be tail calls.

What are the use cases? Not many, but they show up occasionally: state machines, event handlers, etc.

Dave: Tail calls won't help with this kind of event soup:

```
XHR.get(x, #(y1) {
  XHR.get(y, #(y2) {
    XHR.get(z, #(y3) {
```

```
      ...
    })
  })
})
```

Moved tail calls into proposals. We'll need to check with implementors to see how feasible this would be to implement. It will be an issue for Rhino.

/x modifier in regexp: (6.6b)

Waldemar: Comments would make the regexp syntax ambiguous. Without comments, worried about having a mismatched / or mismatched \ escape start interpreting a whole bunch of source code as the body of a regexp. Currently that's stopped by the end of the line, just like a mismatched quote can only cause mayhem until the end of the line. Alternatives are to do line continuations by using \<newline> or have a different character sequence to introduce a multiline regexp.

Brendan:

```
#/ ...
  // this is a comment
  /* an so is this
  */
/#g
```

Discussed whether
#/ ... //comment /#
should be ok. Waldemar: yes, it should, because it falls out of the two-phase process that we use to first find the end of the token and then send the body to:
new RegExp(" ... //comment ", "#")

White space in a character class is not elided.

/y modifier in regexp: (6.6c) OK

JSON path: (6.6d)

Waldemar: Is there a way to follow a path using regular ES operators rather than just path strings? This has powerful operators such as tree and range searching, so it will tempt people to do the path equivalent of eval("x." + field) where they want to do x[field].
What is the grammar for expressions in path strings? Is it the full ECMAScript eval grammar plus things like the @ and $ operators?

MarkM: Quasiquote syntax example
Waldemar: Quasiquote syntax doesn't work well here because JSON path expressions can have various free variables ($, @, etc.) that are instantiated by the search.

XPath structures are trees, but here the ECMAScript objects being queried can contain arbitrary graphs.

Object.isObject: (6.6e)

MarkM, Waldemar, Erik: Pleasantly surprised by Brendan's bold

alternative to fix typeof null to be "null". Very much in favor of it.

How to write transitory code that works in both Harmony and ES5? Not hard. Just need to consider both cases.

Transition to chatting about Harmony and code harmonizers:

A tangent: Allen: We have a discrepancy between function identifier binding in function expressions vs. function statements. In function expressions the body can reliably use it to recurse, while in a function statement it could have been rebound by the time the function runs.
Brendan: Never heard anybody complain about this.

MarkM: If we're making a harmonizer, let's get rid of semicolon insertion as well.

**** End of first day informal notes ****

Once again, here are my raw meeting notes.

Waldemar

Discussion of isNaN and isFinite.

Can/should we fix these in place rather than creating more functions?

Allen: Existing usage is consistent with normal numeric coercions done by other operators such as -.

Doug: Would it be useful to have something that doesn't do the coercions?
Allen: It would be a convenience.
Erik: Not sure if it's worth the extra API.

Decided to accept the proposal.

String duplication: Accepted proposal, but rejected proposed arbitrary limits of 255 or 4294967296 repetitions. Per Allen's comment on wiki, behave as though using ToInteger.

Proxy default handler:
Some trivial bugs in the code: Calling getOwnPropertyDescriptor etc. with only one argument. desc in "desc.configurable = true" can be undefined.

set/put/canPut problem discussion.
Allen: Clean up the list of primitive methods and handlers.
MarkM: All existing uses of put can be written in terms of set.
Waldemar: Would want a more generic way of invoking [[set]] rather than having to instantiate a new default proxy.
Brendan: Issue remains with prototype chain.

Agreed to move this to proposal stage, with some open issues.

Discussion about typeof result strings:
Dave: We should set expectation that typeof can return new things in the future.
Doug: IE can already return a typeof of "unknown".
Waldemar: If we don't add new return strings, in practice it doesn't matter what we say in the committee. If we do add new ones on occasion, then folks will pay more attention.

Brendan, MarkM: should have a "second hand of fingers as backups to break".

MarkM: Trouble with === generalization recurring into records that contain zeroes or NaNs.

Waldemar: To clarify, I'd very much would want === to be an equivalence relation but don't think that we can. A couple examples:
- switch (-0) {... case 0: ...}
- x !== x as an idiom for NaN testing

Discussion of MarkM's semicolon insertion proposal in response to yesterday's meeting notes.
MarkM: Parse with and without virtual semicolon and if both succeed then fail.
Waldemar: How far do you backtrack second parse to see if it fails?
One token? Rest of program?
One token would be tenable; rest of program, not so much.

Example pro:
```
a()
(function() {...})();
```

Examples con:
```
x = a + b
      + c
      + d;

x = a[longindexexpression]
    [anotherlongindexexpression];
```

Brendan: Perhaps it's parentheses that are unique?
Brendan: Make a pragma to turn off semicolon insertion as a strawman.

Dave: Recognize real pragmas, not just string literals.

Do we automatically insert a semicolon after the "no ASI" pragma?

Array create:

Allen would like to eliminate the [[class]] property -- it has too much undesirable conflation of separate concepts:
- magic length property?
- exploded by Array.concat?
- JSON serialized using [] or {}?
- recognized by Array.isArray?
- postMessage special treatment?
- transmitted via proxies?

Brendan: Make a quasiquote special interest group for brainstorming?

Allen: Semi-annual language innovation workshops, independent of day-to-day standards process?

Proposals on people's radar for Harmony:

Allen:

private names
enhanced object literals
[[Class]]
Object.hash
math enhancements
array protocols
spec MOP vs. proxy MOP
numerics, operator overloading

MarkM:

classes and traits
soft fields
generative module expressibility (deferred compile time of modules parametrized by modules)
quasis
better random
concurrency
Function.prototype.toString
simple maps and sets

Dave and Brendan:

records and tuples
expression forms
catch guards
generators
array comprehensions
generator expressions
yield*
modules (also includes global object reform)
module loaders
pattern matching
conditional expressions
enumeration
binary data
pragmas
paren-free
versioning

Waldemar:

guards/types
zero-inheritance classes

Erik:

#-functions

Doug:

modulo operator

Luke:

binary data
Function.create

Tom:

extended Object methods

Alex:

promises/deferred/futures
events

Discussion of Brendan's blog post:

case A:
```
function f(z) {
  return #{x: 42, m: #(a) {a*a}};
}
```

case B:
```
function f(z) {
  return #{x: 42, m: #(a) {a*a}, m2: #() {z}};
}
```

```
r1 = f(1)
r2 = f(2)
```

r1 === r2 in case A but maybe not in case B
r1 egal r2?

MarkM: Function comparison should not reveal captured values.

NaN/zero equivalence debate again.

Tuples are immutable and don't contain holes.
Waldemar: What's a convenient way of replacing the nth element of a tuple (to make a new tuple)?
Brendan:
```
t = #[0, 1, 2]
u = #[t[0], 5, t[2]] === #[0, 5, 2]
v = #[...t[0:2], 6] === #[0, 1, 6]
```

How shallow is #? Do we need the #'s for the getter and setter below?
```
#{get #p() {...},
  set #p(q) {...},
  oldSchool: function() ...,
  #newSchool() {"Yay!"}
}
```

Declaration syntax:
```
const #foo(x) {x}            // Redundant const?
let #foo(x) {x}
```

var #bar(y) {y.z = #() {bar}} // Illegal?
#() {} // expression
#f() {} // expression or declaration
MarkM and Waldemar would prefer #f() to default to const-bind rather than let-bind. General feeling in that direction.
Implicit block-hoisting for mutual recursion?

Lexical «this»: If #-functions are «this»-transparent, then can't use them as methods. Any way to override?
Dave: #(this, x, y) {...}
Allen: #(this: x, y) {...}
Consensus semantics, regardless of syntax choice for #-functions' «this» parameter:
- If you don't mention «this» as a parameter then you always get «this» from the outer lexical scope.
- If you do mention it then you always get «this» from the call site, or undefined if it's not called using the . notation or an equivalent.

Where do #-functions inherit from? Do they have call and apply methods?

Discussion of how to refer to obsolete features in the new spec. We don't want to have to point to ES5.1 to describe things like «with». Also there can be shared heaps where you can call a legacy function that uses «with» and is stored in a variable in a module or other new feature.

Can we put «with» into Annex B?

MarkM: There is a cost to maintaining spec compatibility with «with» and global objects: Global environment records.
Brendan: We have to do this anyway for DOM event handlers.

Upcoming TC39 meetings:
Mar 22-24 (3 days for Harmony), at Google San Francisco
May 24-26, at UCSC
July 26-28 or 27-28 at Microsoft
September 28-29 at Mozilla
November at Apple

Thank you Doug and Yahoo for hosting!

_____