Minutes of the: *27th meeting of Ecma TC39*

held in: *San Francisco, CA, USA*

on: *28-29 March 2012*

1 Opening, welcome and roll call

1.1 Opening of the meeting (Mr. Neumann)

The TC39 meeting (hosted by Google in San Francisco, CA) was opened by **Mr. Neumann**, Chair of TC39 at approximately 10:15 AM on 28th March 2012 (TC39/2012/006 - Venue for the 27th meeting of TC39, San Francisco, CA, March 2012).

1.2 Introduction of attendees

John Neumann – Ecma International

Istvan Sebestyen – Ecma International

David Fugate – Microsoft

Alex Russell - Google

Rafael Weinstein - Google

Waldemar Horwat - Google

Allen Wirfs-Brock - Mozilla

Roozbeh Pournader - Google

Douglas Crockford - Yahoo!

Brendan Eich - Mozilla

Mark Miller - Google

Luke Hoban – Microsoft

Bill Ticehurst - Microsoft

Dave Herman - Mozilla

Rick Hudson – Intel

Stephan Herkut - Intel

Andreas Rossberg - Google

Ross Dreyer – Google (part time)

Oliver Hunt – Apple

Nebojsa Ciric – Google – part time

Norbert Lindenberg – invited guest – no affiliation

1.3 Host facilities, local logistics

Mr. Horwat welcomed on behalf of Google the delegates and provided logistical information. It was announced that Ecma international would host the traditional social event on March 28th evening.

2 Adoption of the agenda (2012/011 Rev. 3)

Ecma/TC39/2012/011 Rev. 3 contained the Agenda for the 27th meeting of TC39, San Francisco, March 2012. This was agreed with minor changes.

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

- Ecma/TC39/2012/008 Minutes of the 26th meeting of TC39, Santa Clara, January 2012
- Ecma/TC39/2012/009 Software Copyright Submission Form from Google to the draft new ECMA-402 (ECMAScript internationalization) standard
- Ecma/TC39/2012/010 Draft "ECMAScript Internationalization API Specification", 23 February 2012
- Ecma/TC39/2012/011 Agenda for the 27th meeting of TC39, San Francisco, March 2012 (Rev. 3)
- Ecma/TC39/2012/012 Revised Intel patent statement and licensing declaration form regarding ECMA-262
- Ecma/TC39/2012/013 Third draft Standard ECMA-262 6th edition, February 2012
- Ecma/TC39/2012/014 Intel proposal to enable data-parallelism in web applications
- Ecma/TC39/2012/015 Presentation "Full Unicode in ECMAScript" by Norbert Lindenberg
- Ecma/TC39/2012/016 Presentation "River Trail" by Intel
- Ecma/TC39/2012/017 Draft specification "River Trail API" by Intel, March 2012
- Ecma/TC39/2012/018 Test262 Status Report, March 2012

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 January 2012 (2012/008)

The minutes of the 26th TC39 meeting in Santa Clara in January 2012 have been unanimously approved.

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

4.1 A new ES6 draft is available at

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

Mr. Wirfs-Brock has presented the latest draft from the Wiki. It has the date February, 2012. The draft is marked with “Rev. 6”. He explains all the changes he has made to the previous version and requests TC39 members for feedback.

4.2 Syntax Issues

http://wiki.ecmascript.org/doku.php?id=harmony:iterators#syntax_issues

These are two small issues, one about the for-in loop, the second about for-in and its new sibling, the for-of loop/comprehension-head/generator-expression-head form. Fine to group them together

4.3 Proto A newer versions with several corrections:

http://wiki.ecmascript.org/lib/exe/fetch.php?id=strawman%3Amagic_proto_property&cache=cache&media=harmony:draft_proto_spec_rev2.pdf

4.4 Intel proposal

Mr. Hudson from Intel gave a detailed presentation on “River Trail” (TC39/2012/016). He was supported by **Mr. Herkut**. The intention of the presentation is to present the idea to TC39, to be incorporated into future work of TC39. The basic idea is to extend Javascript for being suitable for parallel processing. It exists a prototype for showing that it runs. It was announced that next week the specification will be published on the ES-Wiki and on the Ecma website (TC39/2012/017 and TC39/2012/014).

The reaction of TC39 to the proposal was positive.

The goal is to get Javascript paralelism into browsers.

There was a debate about whether to do this for ES6 or ES7: “Let's just do the work. The browsers can ship it when it's ready, regardless of when the standard becomes official.”

It was decided that structurally this will part of the mainline ECMAScript work, (es-discuss + meetings), and not in separate meetings as was done with internationalization.

4.5 Modules and Binary Data

4.6 Override Mistake

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

4.7 Support for full Unicode character set

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

http://norbertlindenberg.com/2012/03/ecmascript-supplementary-characters/index.html

4.8 Revisit for(let;;) binding alternatives

https://bugs.ecmascript.org/show_bug.cgi?id=311

4.9 Resolve scoping rules for global lexical declarations

https://bugs.ecmascript.org/show_bug.cgi?id=312

4.10 Review proposal for __proto__ in current ES6 draft, and other approaches discussed at

https://bugs.ecmascript.org/show_bug.cgi?id=313

4.11 Consider promotion from strawman to proposal status for "do expressions":

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

4.12 Consider promotion from strawman to proposal status for "block lambda revival"

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

4.13 Consider promotion from strawman to proposal of “arrow function syntax”:

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

(but possibly using fat arrow syntax)

4.14 Community to request to add Math.clz (count leading zeros) function

https://mail.mozilla.org/pipermail/es-discuss/2012-February/020779.html

4.15 Review updates to Math libraries additions based on community feedback:

http://wiki.ecmascript.org/doku.php?id=harmony:more_math_functions (to be updated)

4.16 Review candidate spec approach for “regexp_match_web_reality”:

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

4.17 Review updates to ‘observe’ strawman based on feedback from Rafael and Arv:

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

4.18 Proposal to change behaviour of DaylightSavingsTA:

https://mail.mozilla.org/pipermail/es-discuss/2012-March/020832.html

5 Edition 5.1 Issues

5.1 Global declarations shadowing inherited global propertiesResolving

https://bugs.ecmascript.org/show_bug.cgi?id=78

5.2 Problems with restrictions on non-strict Function caller property

https://bugs.ecmascript.org/show_bug.cgi?id=310

6 Report from the ad hoc on Internationalization standard (Ecma/TC39/2012/010)

6.1 TC39 review and last feedback prior to preparation of the final draft

Norbert Lindenberg and **Nebojša Ćirić** reported on the latest status of the work.

The latest version of the draft Standard is published as Ecma/TC39/2012/010 Draft "ECMAScript Globalization API Specification", 23 February 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

6.2 Multi-system prototype testing

It is going on.

6.3 Next steps: Time Line to success.

There will be a delay in approval. It is now expected to be ready for a GA approval in December 2012.

On the request of TC39 the Secretariat has promised to reserve the ECMA-402 number for this standard. This has been done after the January 2012 meeting. However, with the delay it cannot be guaranteed that also for December 2012 this number can be reserved.

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

7 Status Reports

7.1 Report from Geneva

Mr. Sebestyen reported briefly about the latest developments in the Ecma office.

7.1.1 Updated Intel Patent Declaration on ECMA-262

Mr. Neumann has announced that after the lively discussions on this topic in the January TC39 meeting new discussions with Intel have resulted in a new patent declaration from Intel on ECMA-262 (TC39/2012/012). Intel had no intention to cause a "tsunami" in TC39. The new statement declares now free royalties for patents of Intel that are needed to implement ECMA-262. That solves the Intel problem. **Mr. Neumann** pointed out that no other TC39 member has filed any patent statement on ECMA-262, and he would like to bring up this point in the meeting. He said that the IPR group of Ecma may discuss the matter, if such declarations should be requested and collected from other members as well. **Mr. Wirfs-Brock** added that a solution might be if Ecma work of this technical committee working on "essential internet standards" might automatically require royalty free statements (because that is what "essential internet standards" require today). **Mr. Neumann** agreed that such a solution might also work but this new concept has to be included into the Ecma patent policy. **Mr. Horwat** noted that this was included into earlier meeting notes but **Mr. Sebestyen** noted that for several GA representatives the TC39 meeting minutes are unfortunately not visible enough and that caused some confusion.

7.1.2 Possible TC39 input to IPR ad hoc group

Mr. Sebestyen said that from TC39 there is only one contribution to the IPR Adhoc Group, namely the response to the CC and GA request on the possible extension of the TC39 experimental software copyright policy. He, however, also informed TC39 that there is a proposal by an Ecma Member for a solution how to treat in the Ecma patent policy "essential internet standards", such as ECMAScript. On request the proposal was distributed to TC39 members on the TC39 reflector. Several TC39 members have expressed interest in such a topic. Of course this is not being discussed in TC39 but in the IPR Adhoc group, where Ecma members are invited to participate.

7.1.3 Discussion on the possible extension of the TC39 experimental software copyright policy

Mr. Sebestyen said that finally the TC39 contribution to the IPR Adhoc Group had been prepared and submitted to the IPR Adhoc. The content of the contribution has been discussed over the TC39 reflector. It contains basically two sets of issues, TC39 requirements for extending the software copyright policy for non-ecma members and the use of open source content (basically the input from **Mr. Wirfs-Brock**), and the other set is basically feedback from the Ecma Secretariat on the implementation experiences of the software copyright policy (Basically the input from **Mr. Sebestyen**). He said that the IPR Adhoc is going to meet at the beginning of April first, and TC39 will be kept informed about the progress made. TC39 experts (including legal experts of members) are invited to participate in the work of the Adhoc Group.

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

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

Mr. Fugate presented TC39/2012/018 containing the "Test262 Status Report":

To the more than 11 thousand test cases almost 400 has been added just the last month.

- David Fugate will move to another role within Microsoft as of April 1
- Bill Ticehurst will be backfill for David for test262
- New Ecma contributor's agreement form on the way from Microsoft covering all future versions of ECMAScript
- Focus will primarily be on new tests for ES6
- Others encouraged submitting tests for ES5.1 and ES6

It has been announced that **Mr. Fugate** will in the future not work any longer on the test262 project and it will be replaced by **Bill Ticehurst**. The TC39 meeting has expressed its gratitude to **Mr. Fugate** for his excellent work and wished him lot of success in his future tasks.

8 Date and place of the next meeting(s)

May 21 - 23, 2012 at Mozilla (**Mountain View or San Francisco)**

July 25 - 26, 2012 at Microsoft **(Redmond)**

September 25 - 26, 2012 at Northeastern University **(Boston)**

November 28 - 29, 2012 at Apple **(Cupertino)**

9 Closure

The TC39 Meeting ended at 5:15 PM on Thursday, 29 March 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 Google and to **Mr. Horwat** for hosting the meeting and Ecma International for hosting the TC39 dinner on the 28th March in San Francisco, CA.

Attachment

Technical meeting notes from Waldemar Horwat:

March 28:

From: waldemar@google.com
To: es-discuss@mozilla.org
Sent: 3/28/2012 8:43:00 P.M. Eastern Daylight Time
Subj: March 28 meeting notes

Here are my rough notes from today's meeting.

Waldemar

IPR discussion
Intel changed their ECMAScript patent declaration to RANDZ.
Now they wonder why no one else made a RANDZ declaration.
Istvan described the history.
Mozilla is also unhappy with the current state of affairs. Even
though this instance turned out well, it shows the potential for
problems.
Lots more IPR discussion

Rick Hudson, Stephan Herhut: River Trail proposal
Proposal will appear on wiki shortly.
Deterministic, except for things such as array reduction order if the
reduction operation is nonassociative.
Parallel arrays are immutable.
Various parallel methods take kernel functions to operate on subcomponents.
MarkM: Are the requirements on the kernel functions to allow them to
be optimized well-defined?
Rick: Yes

```
var a = new ParallelArray(...)
var b = a.map(function(val) {return val+1;});
```
Allen: This can be done today sequentially by replacing ParallelArray
with Array.

```
var b = a.combine(function(i) {return this.get(i)+1;});
var sum = a.reduce(function(a, b) {return a+b;});
```

Competitors:
OpenCL: GPUs do poor job of context-switching.
WebCL: Too many things undefined.
Browser-provided webworkers (task parallelism).

Waldemar: Can the kernels do nested parallel operations, such as
what's needed for a matrix multiply?
Rick: Yes
Waldemar: What data types can you use?
Rick: This builds on the typed array work.

Some desire for not having too many array times. Can we unify ParallelArray with Array?
DaveH: No. Holes alone cause too much of a semantic dissonance.
Waldemar: Would like to unify ParallelArray with typed arrays.
DaveH: No, because ParallelArrays can hold objects and typed arrays can't.
Waldemar: Why not?

Discussion back to knowing which kernels can be optimized.
DaveH, MarkM: Nondeterministic performance degradation is preferable to nondenterministic refusal to run the code. This leaves implementations space to grow.
What about throwing exceptions only for functions that can never be optimized because they mutate state?

Waldemar: Is this optimized? (Note that there are several different issues here.)
let y = ...;
function f(x) {return x+y;}
a.map(f)

Note that merely reading mutable state is not necessarily a cause for deoptimization because parallel functions don't run in parallel with other code, so that state stays fixed for the duration of the parallel operation.

Allen: Concerned about every closure carrying along sufficient information to do the kind of abstract interpretation needed to optimize it as a ParallelArray kernel.
Allen's issue summary:
- Do we want to do this?
- If so, how do we structure the activity (separate standard or part of ESnext or ESnextnext)?
- Data parallelism or parallelism in general?
Rick: Our goal is to get it into browsers.
Debate about whether to do this for ES6 or ES7.
Brendan, DaveH: Let's just do the work. The browsers can ship it when it's ready, regardless of when the standard becomes official. Need to get new features user-tested anyway.
Structurally this will part of the mainline ECMAScript work (es-discuss + meetings), not in separate meetings as was done with internationalization.

Allen's spec status.

Olivier: Concerns about latency issues related to module fetches blocking. Multiple script tags can fetch their scripts concurrently; modules have latency problems such as:
<script src=A.js>
<script src=B.js>
<script src=C.js>
vs.
<script src=C.js>
where C.js is:
module A at "A.js"
module B at "B.js"
// use A and B

Alex: Have modules block document.write.

Long debate about asynchronous loading approaches.

Olivier: To get better latency, you can make your first module load simple, but after that you'll need to use the AMD syntax again.
DaveH: Change the HTML semantics.
Alex: Evaluate the outer module asynchronously in script blocks such as the one below, because there is no non-module code after the module:

```
<script>
module M {
  module A at "A.js"
  ...
}
</script>
```

Olivier: This will error out if followed out by the otherwise correct:

```
<script>
M.foo();
</script>
```

DaveH proposal: Bifurcate grammar so that scripts (whether in <script> tags or in files included from those) cannot contain static module load statements, not even in syntactically nested modules. Modules can include such statements but can only be parsed via dynamic loads.

```
<script>
System.load("interp.js", function(m) {...});
</script>
```

interp.js contains:

```
module stack = "stack.js";
export function evaluate(...) {...}
```

stack.js contains:

```
export class Stack {...}
```

The body of an eval would be treated like a script body, not a module body. This avoids the tarpit of dealing with synchronous i/o in eval.

For-of design variants:
Variant 1:

```
import keys from "@iter"
for let k of keys(o)
```

Variant 2:

```
for own (let k in o)
```

Current way:

```
Object.keys(o).forEach(function(...){...});
```

Picked variant 1.

DaveH: The people want Array.of and Array.from and don't want to wait for the ... syntax.
Array.of(a, b, c) ≡ [a, b, c]

Array.from(a) ≡ [... a]

Brendan: Disallow initializer in:
for (var i = x in o)
MarkM: OK, as long as Brendan is the first penguin through the ice hole.
Mozilla will take it out.

Brendan's arrow proposal:
ArrowFormalParams_opt -> Block_opt
ArrowFormalParams_opt => AssignmentExpr

ArrowFormalParams:
(FPL)
(this Initialiser_opt, FPL) // Makes this dynamic
(this Initialiser_opt) // Makes this dynamic
Identifier

Brendan: -> and => both use static this.
DaveH: -> defaults to dynamic this, => defaults to static this
Debate about Tennent Correspondence Principle and to what extent this complies.

ArrowFormalParams is actually parsed as the cover grammar:
(CommaExpression)
with later semantic restrictions on what the CommaExpression can
actually contain.

Waldemar: A lot of problems that are individually nonfatal but too
much in combination.
1. Two different arrows is confusing. Will keep using the wrong one
and get weird results. Examples:
  -> {a:b} // intended to return an object literal, instead has a
labeled expression
  -> {} // intended to return an empty object, instead returns undefined
  => {} // intended to return undefined, instead returns an empty object
2. Having both dynamic and static this options is overkill.
3. TCP hazards
4. Cover grammar introduces a lot of spec complexity.
5. Different 'this' defaults in DaveH's variant of the proposal

Wild debate.
Poll of who objects to which problem in the proposal:
1. AWB, MM, AR, AR, LH, DC, WH, DH
2. AWB, MM, AR, AR, LH, BE, DC, WH, DH
3. AR, AR, LH, DC, BE
5. AWB, MM, AR, AR, LH, BE, DC, WH, DH

If we choose only one arrow, which is it? Flow of debate assumed it's
=>, with a do expression used to place statements there if needed.
Luke, Waldemar: Use only one arrow, but allow either a block or an
expression there. If it starts with a {, it's always a block.
MarkM: What about => function foo(){}?
No issue there; it's a function expression. Only => { starts a statement.

MarkM: => assignment cannot be TCP due to yield.
Debate over meaning of TCP.

Some people have changed camps. The ones above are from the original poll.

Confusion about the meaning of poll polarity on item 3.
New results:
WH, AWB, MM: object to 1, 2, 5; want TCP
BE, DH, AR, AR, LH, OH: object to 1, 2, 5; don't want TCP
DC: object to 2, 5; don't want TCP

DC switched to middle camp.
AWB, WH, MM prefer to keep TCP but are ok with switching to middle camp.

Consensus on:
- Have only one arrow, namely =>
- this is always static. No provision for dynamic this.
- Other than the treatment of this, the function behaves like a normal function. return, break, etc. behave as though the function body were in a function(...){...}.

To be discussed later:
The thing that comes after the arrow:
1. Always an expression (possibly a do expression)?
2. A block if it starts with {, otherwise an expression.

_____
es-discuss mailing list
es-discuss@mozilla.org
https://mail.mozilla.org/listinfo/es-discuss

Meeting Notes from March 29:

From: waldemar@google.com
To: es-discuss@mozilla.org
Sent: 3/29/2012 8:02:29 P.M. Eastern Daylight Time
Subj: March 29 meeting notes

> Rough notes from today's meeting.
>
> Waldemar
>
> ---
> ES 5.1 quirks in defining globals when the global object's prototype already contains a property with the same name:
> https://bugs.ecmascript.org/show_bug.cgi?id=78
> Waldemar: What if the prototype contains a getter or setter?
> Doesn't matter for the purpose of defining globals.
> Agreed on the solution proposed in the bug.
>
> John: Do we have a process for tracking ES5.1 errata?
> Allen: Yes.
> John: WIll try to get an official errata on the ECMA site.
>
> Strict function caller leak: https://bugs.ecmascript.org/show_bug.cgi?id=310
> We should correct this in the errata. MarkM might have more comments but he's not here.
>
> Luke: 15.10.4.1 Regexp.source on empty regexps on all browsers returns the empty string instead of (?:), which is incompatible with the spec text. That was a change to the spec in ES5 but browsers didn't obey it.

Allen, Waldemar: This isn't a spec bug. It was an intentional change.
Luke: It breaks prototype.js.
DaveH: If this is the only breakage, change the spec to insert the regexp source into /(?: and)/ instead of / and /.
Waldemar: It's not the only problem. Escaping of slashes is also a problem here.
Luke: Browsers diverge here. Firefox is the only browser that does escaping.
Debated.
No consensus.

David Fugate: Test262 update slideshow
Microsoft sources now include ECMA copyright boilerplate instead of Microsoft's.
MarkM will do likewise for Google sources.
19 new bugs
26 bugs resolved
Prepopulated bug reports
International402 mini-suite framework now (quietly) live, with one fake test
Bill Ticehurst will replace David Fugate's role on Test262.
Development focus will shift to ES6 tests.
Allen: Section numbers will change in ES6, which will impact test suite organization.

Norbert: Internationalization
Final draft won't make it for June. Will try for December. Allen reviewed half of the spec and had lots of substantial comments. There are quite a number of fixes that need to be made to the spec.
Discussion about implementations and overspecification vs. underspecification.

MarkM: Quick follow-up on TCP correspondence from yesterday. What should var do?
MarkM proposed that var be statically rejected if it would cross a closure boundary.
Waldemar: Since we gave up on TCP, => should work just like a function except for 'this'.
Discussion cut off by Luke, since it looks like it won't be a quick one.

Discussion of hypot, hypot2.
hypot is the square root of the sum of squares and takes either two or three arguments.
hypot2 is the sum of squares and takes either two or three arguments.
Waldemar: How is hypot2 better than just doing x*x + y*y?
Luke: It's just ergonomics.
General reluctance about the hypot2 name because it looks like the 2 means two arguments (as in atan2). Some debate about other function names (hypotSq? sumOfSquares?).
MarkM: How is hypot better than sqrt(x*x + y*y)?
It's potentially more efficient and more accurate. It is widespread in numeric libraries.
Consensus: hypot will support just two or three arguments. hypot2 dropped.
Waldemar, MarkM: Why not one or zero arguments? It would be 0 for zero arguments and abs for one argument.
Allen, DaveH: If you pass one argument to hypot, you'll get NaN.
Luke: It's not variadic.
Waldemar: Why isn't it variadic?
Luke: 2 or 3 is the 99% use case.
Waldemar: 2 or 3 arguments is the 99% use case for max.
Waldemar: If it's not variadic and takes only 2 or 3 arguments,

you'll get silent mistakes. If you pass in four arguments, you'll get the hypot of the first three, and the last one will be silently ignored. That's bad.
Luke: Will go back to the experts to explore implementing variadic hypot.

Add parseInt and parseFloat to Number, matching isNaN and isFinite?
Use the name Number.parse instead of Number.parseFloat?
Should these functions be specified as independent functions from the existing global parseInt and parseFloat, or should they be the same function objects?
Consensus: Not duplicting parseInt and parseFloat into Number. May consider doing a Number.parse in the future.

Math.cbrt cube root: Handles negatives, more accurate than pow.
Approved.

Naming: Math.sign vs. Math.signum?
Sticking with Math.sign.

Should the spec try to enforce accuracy?
No. None of the existing math libraries spec accuracy.

MIN_VALUE is 0 on ARM implementations because they don't represent denorms. This is a severe spec violation. However, apparently turning off denorms is a major power savings on ARM.
Allen, Waldemar: MIN_VALUE must be a nonzero value. If a platform doesn't represent denorms, it should make MIN_VALUE be the smallest positive normalized number on that platform.

Number.toInteger: Just does the internal ToInteger. Not the same as floor.
Renamed it to toInt.

Count leading zeros: Number.clz.
Consensus that we want it.
Should we include "32" in the name?
Waldemar: "32" not needed in name. None of the other bit operators include "32" in the name.
MarkM: Prefers clz32 but doesn't feel strongly.

Hex floating point literals:
Waldemar: Other languages include these things. They're rarely used but when you want one, you really want one. Use cases are similar to that of hex literals.
Will explore adding them.
MarkM: 0x3.p1 currently evaluates to undefined. This would be a breaking change.
Waldemar: Not clear anyone would notice. How did other languages deal with this?

Relocated and rescheduled meetings:
May 21-23 Mozilla, Mountain View (May 21 is internationalization)
Sep 25-26 Northeastern, Boston
The July and November meetings are the same are previously scheduled.

DaveH's presentation about module loaders
Discussion about cross-origin problems and modeling iframes
About loader providing a function to define the entire set of built-ins:

Allen: A lot of intrinsics refer to the built-ins. The above function defines the ones that ECMAScript knows about. What about other, implementation-specific ones? Have the environment provide a way for making environments that include such things?
Luke: Objections related to WebIDL [I didn't follow the logic]
Discussion about the role ECMAScript should play in the hierarchy of web infrastructure.

Waldemar: Is there any notion of a parent environment, other than whatever environment intrinsics get the built-ins from? (quietly hoping the answer is no)
DaveH: No.
Waldemar: What about the dynamic craziness related to inner and outer window objects?
DaveH: Will need extra hooks for that.

Waldemar: What creates new intrinsic worlds?
DaveH: Creating a Loader does, as in:
l = new Loader(System, {intrinsics: null}); // Can also specify existing intrinsics to share a world
a = l.eval("[]");
Object.getPrototypeOf(a) != Array.prototype
b = l.eval("[]");
Object.getPrototypeOf(b) != Array.prototype
Object.getPrototypeOf(b) === Object.getPrototypeOf(a)

Allen: Can DefineBuiltIns be called any time?
DaveH: It can be called on any object at any time. Don't want to get now into esoterica of what happens if there are existing properties, setters, etc.

DaveH: Imperative module replacement example:
@websockets exists as version 1. An implementation wants to improve it to v2:
import "@websockets" as ws;
System.set("@websockets", polyfills(ws));

Waldemar: What can you provide to the intrinsics parameter of the Loader constructor, other than the previously covered examples of null and another Loader?
DaveH: Nothing else is allowed there.
Waldemar, Luke: Then we'll need a notion of a loader type just like we have a notion of an array type. At some point we'll need to say what happens when you pass an object that inherits from a Loader, a proxied Loader, etc.

Module syntax alternatives:
module x at "foo.js"
module x = "foo.js"
module "foo.js" as x
module x at "foo.js"
Don't have time to discuss these now.

Unicode presentation
In some places ES5.1 treats unicode characters as arbitrary 16-bit chunks. In other places it has special provisions for surrogate pairs.
For much text processing it doesn't matter; either works. The trouble

points are:
- Supplementary characters within source code identifiers
- Regular expressions
- String comparison
- Case conversion

Norbert's alternatives:
1. UTF-32 strings
2. UTF-32 or UTF-16 strings
Waldemar: Either of these would be nightmares because they'd provide two different ways of encoding the same supplementary character, and by Murphy's Law you'd get the wrong one at the most inconvenient time. Wasted a lot of time in Perl which has this problem. A far better solution would be to keep representation as always UTF-16 and just fix the functions to understand UTF-16 better.
Olivier: For ECMAScript to adopt the UTF-32 model, you'd need an amazingly compelling reason that would blow all other ones out of the water.
Consensus: No one wants UTF-32 strings.

Norbert's favored third option: Stick with UTF-16, change functions to understand UTF-16.
MarkM: Careful about breaking compatibility! Change functions or create new ones?

Regular expressions:
/u mode that matches via UTF-16 code points instead of code units.
/./ would match a code point; supplementaries can be in ranges.
Clear that this must be done via a mode; this would break too much stuff without a mode such as /u.

Waldemar: Why not graphemes?
Nebojša: The Unicode folks tried it; it became too difficult.

Case conversion UTF-16 fixes are uncontroversial?
Waldemar: Not so. When designing ES3 we intentionally disabled some Unicode case conversions to avoid nasty surprises. For example, we wanted /[a-z]/i to match only the ASCII letters. Had we allowed the true Unicode conversions, this would also match the non-ASCII Turkish dotless lower-case i (ı) or upper-case dotted I(İ). There were similar issues around ß expanding into SS. How will we deal with situations like this with /u?

Make /u be the little red switch for:
1. Unicode code point semantics
2. Unicode based \d\D\w\W\b\B
3. Unicode case folding
4. Remove some/all identity escapes to allow \p, \X, \N
5. Don't match web reality?

Can we do only 1+3+4+5, without 2?
Waldemar: That makes no sense. Given 3, you shouldn't expect \w to match the same characters as in non-/u mode because, at the least, /\w/i will match ı or İ.
Consensus on doing 1+3+4+5. 2 needs discussion.

< and > will work as they do now on strings -- compare code units as

unsigned 16-bit integers.
trim: There are currently no supplementary white space characters, so we can redefine it to support supplementary characters without breaking anything.
toLowerCase and toUpperCase: Safari already converts supplementary characters with apparently no ill effects. Consensus on redefining these to work properly on supplementary characters.

Base the spec on Unicode 5.1. Implementations will be permitted to support later versions if they choose.

String.fromCodePoint(cp0, cp1, ...)
This accepts integers between 0 and 0x10FFFF. If the integer is in the surrogate range, it will produce an unpaired surrogate.

String.prototype.codePointAt(pos)
Here pos is the code *unit* position. If it's the second surrogate of a pair or an unpaired starting surrogate, it will return the code unit of the surrogate.

String.prototype.[iterator]
Should return code point strings (of length 1 or 2), not numbers.

Waldemar: Would also want an iterator for graphemes.
DaveH: The default iterator should return code point substrings.

Code point escape:
"\u{20BB7}" === "\uD842\uDFB7"
V8 doesn't currently throw a syntax error for \u{, but that will get fixed.
Exclude 0xD800-0xDFFF? No.

Maximally minimal classes:

Alex Russell and **Allen Wirfs-Brock** initiated the discussion as a status up-date to TC39. We pointed out that this proposal had recently been extensively discussed on es-discuss and that it appear to have considerable support from most of the participants in that discussion.

Luke Hoban: These aren't good enough to be a net win.

I'm not sure whether this is an exact quote. **Luke Hoban** certainly did raise the issue of whether classes, as defined by this proposal, added enough functionality to ES to justify the inherent complexity of a new feature.

Allen Wirfs-Brock and **Alex Russell** reiterated that this proposal is only trying to address the most common class definition use cases but in a way that allows for future extensions to address a broader range of use cases. There is significant value in what the proposal provides even if it doesn't do everything any might want.

Dave Herman stated he has some minor design issues he wants to further discuss, but that overall the level of functionality in this proposal was useful and would be a positive addition. He supports it.

Waldemar Horwat: These don't address the hard problems we need to solve.

Concerned about both future-hostility (making it cumbersome for future

classes to declare, say, object layout without cumbersome syntax by

taking over, say, const syntax) and putting developers into a quandry

We discussed this concern quite a bit and did not identify any specify ways in which the current proposal would block future extensions. **Waldemar Horwat** was asked to provide

specific examples if he comes up with any. **Allen Wirfs-Brock** pointed out that future syntactic additions can also enforce new semantics. For example addition of a per instance state declarations and a "const" keyword to the constructor declaration could cause ad hoc this.property assignments to be statically rejected, if that was a desired semantics.

-- if they want to do anything more sophisticated, they'll need to

refactor their code base away from these classes. Unless one choice

is clearly superior, having two choices (traditional and extended

object literals) is better than having three (traditional, extended

object literals, and minimal classes). Minimal classes currently

don't carry their weight over extended object literals. Perhaps they

can evolve into something that carries its weight, but it would be

more than just bikeshedding.

The above is a statement of **Waldemar Horwat's** opinion. Other opinions expressed in the discussion aren't record in the original notes.

Alex Russell: We need to do something.

Allen Wirfs-Brock and **Alex Russell** also expressed that it is unlikely that any class proposal that significantly goes beyond will be accepted for ES6.

Debated without resolution.

In summary:

Waldemar Horwat should identify any specific ways that the syntax or semantics of this proposal would be future hostile.

Waldemar Horwat, **Luke Hoban**, and **Mark Miller** expressed varying levels of concern as to whether the user benefit of the proposal was sufficient to justify its inclusion. In order to resolve this question, both sides of the issue really need to provide better supporting evidence for the next meeting.
