Minutes for the: *34th meeting of Ecma TC39*

in: *London, UK*

on: *21-23 May 2013*

1 Opening, welcome and roll call

1.1 Opening of the meeting (Mr. Neumann)

Mr. Neumann has welcomed the delegates at Google, at the Belgrave House, in London.

Companies / organizations in attendance:

Mozilla, Google, Microsoft, eBay, jQuery, Yahoo, Northeastern University, Adobe, VUB

1.2 Introduction of attendees

John Neumann – Ecma International

Istvan Sebestyen – Ecma International

Tom van Cutsem – VUB

Mark Miller – Google

Jasvir Nagra - Google

Douglas Crockford - eBay

Luke Hoban - Microsoft

Allen Wirfs-Brock - Mozilla

Erik Arvidsson - Google

Bernd Mathiske - Adobe

Alex Russell - Google

Eric Ferraiuolo - Yahoo

Yehuda Katz - jQuery

Sam Tobin-Hochstadt - Northeastern Univ

Dave Herman – Mozilla (via conferencing)

Andreas Rossberg - Google

Brendan Eich - Mozilla

1.3 Host facilities, local logistics

On behalf of Google **Alex Russell** welcomed the delegates and explained the logistics.

1.4 List of Ecma documents considered

Ecma/TC39/2013/015	Ninth draft Standard ECMA-262 6th edition, March 2013
Ecma/TC39/2013/016	Draft minutes of the conference call of the IPR adhoc on 25 March 2013
Ecma/TC39/2013/017	Venue for the 34th meeting of TC39, London, May 2013
Ecma/TC39/2013/018	Minutes of the 33rd meeting of TC39, Sunnyvale, March 2013
Ecma/TC39/2013/019	Experimental TC39 RF TG Policy package
Ecma/TC39/2013/020	TC39 chairman's report to the CC, April 2013

Ecma/TC39/2013/021	Mozilla patent statement regarding TC39 specifications, March 2013
Ecma/TC39/2013/022	Notes of the ad hoc meeting on Internationalization, 19 April 2013
Ecma/TC39/2013/023	Agenda for the 34th meeting of TC39, London, May 2013 (Rev. 2)
Ecma/TC39/2013/024	Non-Ecma-member contributions to the planned TC39 RF TG Work
Ecma/TC39/2013/025	Tenth draft Standard ECMA-262 6th edition, May 2013
Ecma/TC39/2013/026	"Notification Proxies: update TC39 May 2013" by Tom Van Cutsem
Ecma/TC39/2013/027	"ES6, ES7, ES8...Roadmap" by Mark S. Miller, May 2013

2 Adoption of the agenda (2013/023-Rev2)

The agenda was approved.

3 Approval of minutes from March 2013 (2013/018)

The minutes of the March 2013 TC39 meeting have been approved as presented. Individuals that took technical notes were recognized and appreciation extended.

Erik Arvidsson volunteered to take technical notes. The technical notes are included in Annex 1 of the minutes. They reflect the discussions correctly and in great details. The technical notes- have been already shared with TC39 and feedback was taken into account.

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

4.1 Object.freeze: Do nothing instead of throwing a TypeError. (Doug)

4.2 WeakSets, design follow up from September (Rick)

https://github.com/rwldrn/tc39-notes/blob/master/es6/2012-09/sept-19.md#weakset

4.3 WeakRefs, review Mark Miller's security solution: (Rick)

https://mail.mozilla.org/pipermail/es-discuss/2013-January/028542.html

Defer ES 7

4.4 Notification Proxies (carry-over of point 4.3.1 from previous agenda) (Tom)

4.5 Roadmap (Mark)

4.6 Communicating Event Loop Concurrency (Mark)

4.7 Promises (Mark)

4.8 Relationships (Mark)

4.9 StringTemplates - the meaning of omitting the name of the handler (Mark)

4.10 Module Naming

https://mail.mozilla.org/pipermail/es-discuss/2013-April/030165.html

4.11 Unique Symbols vs Private Symbols vs Weak Maps vs Relations (Andreas)

4.12 Revisit the @-names discussion/resolution from the November meeting

A significant body of work has emerged that I feel makes a strong case towards making an exception to the cut-off deadline. I'd like to urge everyone to take a moment to review this: https://github.com/Benvie/continuum/tree/gh-pages/engine/builtins

4.13 Endianness of Typed arrays

https://mail.mozilla.org/pipermail/es-discuss/2013-March/029505.html

4.14 Module update/Module Naming

4.15 Proxy issues

4.16 Spec update

4.17 Object Observe implementation report (Rafael)

4.18 __Proto__

4.19 Generator Update

5 Edition 5.1 Issues

6 General implementation experience

7 Second edition of ECMA-402

7.1 Status report

8 Test 262 Progression

8.1 Status report

9 Status Reports

9.1 Report from Geneva

9.1.1 Brief report from the IPR meeting

Mr. Sebestyen has reported that a series of meetings on TC39 Policy issues were held in the IPR Group. Slow progress on the software contributions by non-Ecma members, like 3rd party and Liaison organizations. He felt that the current contribution license text is already so broad that it could also be applied by 3rd parties and liaison organizations. This, however, has to be verified by the IPR Group.

9.1.2 Approval of the TC39 RF TG operating procedures (2013/019)

Significant progress on this topic by the IPR Group and the CC. Basically the policy and the respective operational documents (such as forms) have been completed, distributed, and the Ecma GA will vote on June 11, on 2 docs about the TC39 RF TG (royalty free task group) how it should function. Also there will be a vote on the scope of such TC39 RFTG. If approved, then the royalty free group within tc39 will be created. This will be a transition from the current RAND way of working. Therefore all TC39 members are required to sign up using the new form to the TC39 RFTG. It is foreseen that all further development of the ECMAScript standard would be done by this RFTG. The plan is to collect the applications to the new RFTG by about mid September. The hope is to make switch in 2 or 3 months. The transition should happen without interruption. It means that until sufficient members of TC39 members have not signed up to the TC39 RF TG only the existing RAND Ecma policy is applied. The new TC39 RF policy will apply only to the TC39 RFTG. It is experimental, like the TC39 software copyright policy.

On the question of status of patent statements on ECMAScript **Mr. Sebestyen** replied that Intel submitted royalty free statement on ECMA-262. On the encouragement to file other RF statements Mozilla followed. Ecma has no other statements on record.

9.1.3 Approval of the Non-Ecma-member contributions to the planned TC39 RF TG Work (2013/024)

Mr. Wirfs-Brock explained that this is a new "Public RF channel" that was requested by some members of the IPR group. We in TC39 never talked about that here, but it is a good idea. This allows someone to write up a proposal and we are allowed to read it and maybe even incorporate it.

Mr. Sebestyen added, it will be an additional channel from non members via a form on the TC39 web site (Click through process, contributor has to agree to the RF TF policy. Then submission of contribution through the website).

Replying to question **Mr. Sebestyen** said that the IPR ad hoc group is still working out a solution for a software contribution from non members, but the IPR ad hoc group has not finished their work. So the external contribution is not yet for software. This will be the next step.

9.1.4 Ecma recognition program

Mr. Sebestyen explained that now there is an Ecma recognition program for doing significant work in Ecma standardization in place. TC39 should prepare a list of those who should receive such recognition (certainly the Editors of ECMAScript standards). **Mr. Neumann** suggested including also long term TC39 contributors in the list (who have participated in the TC39 work for longer than a year in the past 5 years).

9.2 Json

Mr. Neumann, Mr. Crockford, Mr. Wirfs-Brock and **Mr. Sebestyen** have reported that the IETF wants to put the informal (and expired) JSON RFC 4627 onto the Standards Track. Therefore they are recharting the JSON roup in IETF.

There are differences between RFC 4627 and the ECMAScript specification in the rules for parsing JSON. How the deal with the differences have to be solved. The resulting standard will be jointly published as an RFC and by Ecma. Ecma participants will be participating in the working group editing through the normal process of working group participation. The responsible IETF Area Director will coordinate the approval process with Ecma so that the versions of the document that are approved by each body are the same.

The project has been approved by TC39, and if the new RFTG is approved by the Ecma GA it would be done under that regime.

Currently IETF lists January 2014 as the date of publication of the standard.

10 Date and place of the next meeting(s)

Schedule 2013 meetings:

- July 23 – 25, 2013 (Microsoft - Redmond)
- September 17 – 19, 2013 (Bocoup - Boston)
- November 19 – 21, 2013 (PayPal - San Jose)

11 Closure

Mr. Neumann thanked **Google** for hosting the meeting in London, the TC39 participants their hard work, and **Ecma International** for holding the social event dinner Wednesday evening. Special thanks goes to **Erik Arvidsson** for taking the technical notes of the meeting.

Annex 1

Technical Notes (by Erik Arvidsson):

Tuesday May 21

John Neumann (JN), Allen Wirfs-Brock (AWB), Eric Ferraiuolo (EF), Erik Arvidsson (EA), Luke Hoban (LH), Doug Crockford (DC), Yehuda Katz (YK), Brendan Eich (BE), Sam Tobin-Hochstadt (STH), Alex Russell (AR), Dave Herman (DH) (calling in), Bernd Mathiske (BM), Andreas Rossberg (ARB), Mark Miller (MM), Tom Van Cutsem (TVC), Jasvir Naga (JNA), Istvan Sebestyen (IS)

Dinner is Wednesday ~14 people

JN: Going through the agenda

Adding __proto__

Unifying iterator/generator APIs

Talking about getting user stats for test-262...

YK: Prioritize ES6 items. So that we don't get do ES7+ items before

Minutes approved unanimously

4.1 Object.freeze

DC: Today Object.freeze throws when primitives are passed in. Suggesting not throwing when a value type is passed in.

MM: Object.isExtensible would return false for primitives

EA: This would give an inconstint view for primitives.

AWB/YK: (In strict mode) numbers and strings lazily box so the assignment never fails.

MM: Proxies are allowed to be non extensible and throw away.

ARB: Is the suggestion to lazily wrap primitives?

MM: No, then isExtensible(7) would return true because the wrapper is extensible.

AWB: In most of the new changes we are not doing unnecessary coercion.

YK: The Chrome dev tools, console.dir(7), says "no properties" which supports treating these as empty objects.

MM: The only observable wrapper is the `this` wrapper in non strict mode.

AWB: In the new spec, Object.setPrototypeOf(7) throws.

MM: Agrees violently!

Conclusion: DC+AWB to work out the details

4.2 WeakSet

Do we need them?

MM: Trivial shim around WeakMap.

YK: Often wanted it

AWB: Adds no new capabilities.

AR: We should not limit ourselves to what is a new primitive capabilities

AI(AWB): add to spec

Consensus to add WeakSet.

4.4 Proxies

TVC's presentation on Notification Proxies:
https://docs.google.com/file/d/0B9iYRsLxmdqUd1RsdHZtazliWmc/edit?usp=sharing

Arguments against:

- shifts the burden from spec writers/implementors to users (need to use shadow target even for non-frozen objects)

- implementors will deal with spec bugs related to invariant violations as they come up

Conclusion: Notification proxies are not approved. MM & TVC are still happy with direct proxies.

Proxy Invoke Trap and wrong |this|-binding on built-in methods

AWB: with current default behavior of "get", "Caretaker" will break on built-ins such as Date, because the |this| binding is by default set to the proxy, so the Date built-in method will not find the correct private state.

ARB: Same issue with binary methods

...

STH: We should add invoke trap but not change the object model

MM: Pleasant to have. Separate from private state.

AWB: used to think this was an issue with proxies, but convinced that it's an API issue: we need to provide default handlers that do the right thing, and which users can subclass. In particular, want a handler that, on forwarding, rebinds |this| to the target.

STH: If you want to proxy a Date method the underlying `this` needs to be a non wrapped Date object.

TVC: previously proposed a Handler API that defines derived traps and fundamental traps, allows you to subclass and inherit correct behavior for derived traps. Can be used as the basis.

AWB/TVC: invoke trap would make it easier to control |this|-binding

DH: Never liked breaking the semantics of [[Get]] + [[Call]]

TVC: there already exist invoke-only properties on platforms with __noSuchMethod__

AWB: For a [[Call]] it might be important to control `this` but by the time the [[Call]] is happening you do not know what `this` to use.

DH: ActionScript has a proxy and they do have an invoke trap.

BM: The most common action is to invoke a method.

? : we already gave up on the |this| invariant for accessors: in ES5, if obj.x is a getter, |this| will always be bound to obj in the getter. With proxies this is no longer true.

AI(AWB, TVC): Add spec for invoke. Tom and Allen to work out details of a Handler API that accommodates both "caretaker" (aka forwarding) and "virtual object" use cases.

Consensus: Add invoke trap.

4.11

MM: Everybody in this room wants classes and want to post pone private state to after ES6

ARB: Disagrees.

ARB: Based on feedback, people do not want unique symbols, only private symbols.

MM: Private symbols do not work with proxies.

TVC: can still use WeakMap for private state.

DH: The most common cases where true information hiding is self hosting. The stakes are too high for the browser engines.

YK: If "iterator" would be a private symbol, you cannot create a proxy that will work with for-of loops.

ARB: Symbols (unique and private) and relations overlap.

BE: If we add symbols now we are stuck with them.

LH: Future users will be confused. They will not know what to use

BE: Unique symbol is very different from class private syntax.

AWB/MM: If we first did relationships we might not need symbols.

MM: Relationship published but not reflective.

MM: Difference between relationships and symbols: where is the mutability? This forces us to have both relationships and unique symbols.

Conclusion: ?

Report from Geneva

IS: IPR, vote on june 11, 2 docs about policy. royalty free task group should function. When approved, create royalty free group within tc39. must transition to the royalty free task group. Collect before November. Hope to make switch in 2 or 3 months.

IS: Intel submitted royalty free statement on ecma 262.

BE: So did Mozilla

IS: IPR ad hoc group, work out a solution for a software contribution from non members. IPR ad hoc group has not finished their work.

AWB: Public RF solution? We never talked about that here.

IS: Additional channel from non members. Compromise; Fill in form on tc39 web site. Click through process. Agree RF TF. Can submit contribution through the web site.

AWB: But not the software.

IS: It is still missing. It will be the next step.

AWB: This allows someone to write up a proposal and we are allowed to read it and maybe even incorporate it.

IS: There is an ECMA recognition program. List contributors. Requests a short list of nominees.

JN: Nominees before Thursday morning.

STH: And maybe a trophy?

Item 9.1.2

JN: RFTG mode. Keep things transparent. Unanimously approved

JN: Doc 24. Any objections? Unanimously approved

4.13 Endianness of Typed array

ARB: Remember it as if we should specify this.

BE: Endianness in Typed Arrays is unspecified.

DH: Keep it open for now... Same system to same system. Using data view, which is explicit, there is no problem.

STH: We don't know what WiiU will do?

AWB: Or they decide not to comply to the spec

DH: WebGL is endian agnostic.

Conclusion: Leaving it unspecified in ES6.

4.18 __proto__

STH: Recollection, first as data property, then as an accessor. Then discussed the power of that setter. Set the [[Prototype]] in the [[Realm]]. Then Allen wrote the spec. Realized that there were some problems with that design. Roughly the same power as Object.setPrototypeOf.

MM: Existence of a setter... as long as we have the extensibility restriction, that is sufficient.

AWB: Why restrict __proto__ and not other

DH: Objects belonging to a realm is a bad idea.

MM: No more reason to restrict the setter.

STH: Bind __proto__ setter to the object upon extraction

MM: In SES objects that are non extensible. Not going to remove __proto__ going forward.

ARB: If Object.prototype.__proto__ is a data property, making it non writable prevents other objects to use assign to set __proto__.

AWB: If Object.prototype.__proto__ is an accessor that just calls Object.{set,get}PrototypeOf.

AR: Best practice on the web is important even in the future.

TVC: If we have O.p.__proto__ do we want Object.setPrototypeOf or just Reflect.setPrototypeOf?

AWB: Makes sense to have Object.setPrototypeOf for consistency.

EA: Where do we draw the line (Object.x or Reflect.x)?

DH: People will need to be able to get this before we have a reflect module.

TVC: We need both because they have different return value (reflect setPrototypeOf returns boolean success value).

Conclusion: __proto__ is an accessor on Object.prototype. The setter mutates [[Prototype]]. There is no "poison pill". We will provide both Object.setPrototypeOf and std:reflect setPrototypeOf.

Naming of @@iterator

AWB: Suffix with $

STH: Opposed to special naming. People don't do this kind of naming convention. Why do we want to introduce this concept?

```
class Foo {
  *[iterator]() {
    yield ...
  }
}
```

Conclusion: No special naming

Generators and iterators

AWB: send is gone in favor of next(arg) (only first arg is passed through in yield*)

YK: Whether generators return a frozen object or not?

BE: close is gone

Wednesday May 22 2013

John Neumann (JN), Allen Wirfs-Brock (AWB), Eric Ferraiuolo (EF), Erik Arvidsson (EA), Luke Hoban (LH), Doug Crockford (DC), Yehuda Katz (YK), Brendan Eich (BE), Sam Tobin-Hochstadt (STH), Alex Russell (AR), Dave Herman (DH) (calling in), Bernd Mathiske (BM), Andreas Rossberg (ARB), Mark Miller (MM), Tom Van Cutsem (TVC), Istvan Sebestyen (IS), Jasvir Naga (JNA)

4.16 Spec update

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

YK: ToPositiveInteger is needed by JSIDL

AI(YK+AWB): Put an algorithm in the spec that DOM can use so that we get the same behavior in JS and DOM.

6 General implementation experiences

ARB: We started implementing generators but things are pretty smooth.

BE: Doing modules at the moment.

AWB: Bunch of bug fixes in the spec related to classes.

4.9 String templates

MM: Suggests status quo.

AR: Objects.

MM: Controversy related to tag-less templates. Alternatives include making tag-less templates an error, delayed evaluation (contextually provided)

AR: Econics: naked interpolation is too attractive. Should always have a tag to encourage users to think about which behavior is correct.

YK: I cannot support Alex's proposal.

STH: What would the name of this tag be?

AR: Something that is imported from a module.

STH: Concerned about short names and conflicts.

YK: People will just use 's' without thinking.

YK: People should use HTML templating engines.

DC: Alex's testimony about application developer feedback is relevant.

LH: it sounded like Google engineers were using a template system

EA: Correct.

MM: Does anyone prefer taking out TS if they don't get tag-less TS?

Everyone: Agrees that it is better to require tag to remove TS from ES6.

AR: Strings are always used later in some context. Communicating the intent

AWB: String concat vs string interpolation have the same issue.

LH: Assumes that maybe only 20% of the uses ot TS are susceptible to XSS

MM: Removing tag-less does not reduce XSS because people will just use s`. TS helps people transition to a better world. Once they have have a TS it will be easy to add an html tag at the front as needed.

ST: It will be painful to import String raw and alias that to s.

MM: Maybe put tag-less in appendix? Withdrawn idea because no one likes it.

YK: You should not have use string based APIs.

AR: Willing to abstain but “Y’all are making a big mess”

BM: Half convinced by Alex.

LH: Different code bases will use different tags for normal string interpolation so moving between code bases will be hard to.

AR: That is a good thing. Forces people to think.

MM: Template strings in E.

STH: Lots of contexts where XSS is not an issue.

BM: More ways to XSS is a bad thing.

BE: if people have to import s then the economics change and people will stick to +

Conclusion: AR and BM sustains. We continue with the status quo (tag-less ts is supported)

JSON

DC: IETF wants to change JSON

MM: The 2 documents should have exactly the same text except for boilerplate.

IS: Should it be done in TC39?

DC: Most of the work will be on the mailing lists

AWB: Who will be the editor?

DC: Hopes they (IETF) will provide an editor.

JN: Should this be fast tracked to ISO?

DC: That makes sense.

JN: How long do you expect this to take?

DC: Has taken a long time to coordinate and get started.

DC: 5.1 specs the 2 functions that uses the JSON format.

4.10 Modules

STH: Progress since last meeting. Discuss “module naming”, “naming standard modules”.

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

STH: Wiki is up to date with the current proposal. Spec is “wiki complete”. Jason Orendorff of Mozilla has worked on flushing out semantic issues. Moz is implementinb parsing of modules.

STH: Syntax: Made a couple of changes.

A. To support anonymous exports

```
export default expr;

import $ from ‘jquery’;  // imports default anonymous export
```

If there is no default then the above is an error

```
import {ajax} from ‘jquery’;

import {ajax as A} from ‘query’;
```

to reduce confusion and to make it clear that this is not destructuring.

```
module fs from ‘js/fs’
```

fs is a module instance object

The following is not valid:

```
import {...} from fs;  // SyntaxError
```

Renaming on export:

```
let foo = 13;
export {foo as bar};
export {foo};
```

The following is not valid:

```
export foo;
```

STH: The only evaluation here is “13”. The rest are just bindings that are shared with the outside/module importer.

MM: Bad idea to allow external modules to assign to imports.

DH: Imported bindings are read only to the importer.
AWB: This is new semantics to the language. Is there a list of these new semantics modules introduce?
AWB: Is there a way to get the default export from the instance module obejct.
STH: There will be a well known symbol name to get to it.
AWB: Does module instance objects inherit from Object.prototype.
DH: No. Because we do not want any pollution.
JNA: Is it an error to assign to an imported binding?

```
import {ajax} from 'jquery';
ajax = 14;  // Error
```

AR: What is the reason for not extending Object.prototype or some other object?
YK: To prevent people from expecting toString to be there (???)
DH: fs.readFile We don't want to statically check this deeply inside an expression.

```
fs.toString
```

THS: The plan is to allow the above to be a static error in the future.
DH: To keep things clean.
AWB: Concerned about the dot operator
ARB: Don't want less checking if you do not use import.
DH: Do not want refactoring hazards.
ARB: This only affect the static semantics.
AWB: Can you use square bracket?
STH: Square bracket is dynamic.
AR: This is only a static check that is lost. At runtime there will still be errors.
LH: Concerned about default export. Now people will have to decide which approach to use.
STH: This is already the case in Node.js today.
LH: Today you might get any object, it might be callable with properties.

```
var fs = require('fs');  // module instance
var glob = require('glob');  // function with properties
var parse = require('parse');  // function

module fs from 'fs';
import glob from 'glob';
import {sync} from 'glob';
import parse from 'parse';
```

Lots of discussion...

```
import {sync} from 'glob';
```

alt

```
import glob from 'glob';
var {sync} = glob;
import {ajax} from 'jquery';
```

LH: Prefers "export =" and lose static checking when people opt in to single anonymous export.

STH/YK: We already agreed that we want static checking.

LH: Even for new things being built, this is causing a confusion.

AWB: It is unclear when and what you want to export as the default export.

BM: Wants

```
import default $ from 'jquery'
```

to ensure that people have to be explicit about what they import.

DH: This is just syntax and we are wasting time "bikeshedding"

AWB: What is the best practice? Is there a single module containing Map, Set & WeakMap or...

YK: WeakMap should be its own import:

```
import WeakMap from 'collections/WeakMap';
```

BE: We have to pay attention to what Node/AMD do today.

YK: AMD tries to make modules small to reduced byte size of the dependencies.

STH: And now to semantics

https://github.com/jorendorff/js-loaders/blob/master/browser-loader.js

STH: Major things that changed. Use options object more consistently.

STH: The wiki page is up to date.

STH: Need to decide whether the browser loader is in the appendix or if it is in some w3c spec. Want core language semantics to treat the names as strings, not the semantics of these strings.

STH: Bulk loading. One HTTP request to load multiple modules. Possible to implement. Create fecth hook. Stores module notations in a side table. In the xhr response, split the result and call the different fulfill hooks.

EF: Sounds like what we do today in YUI loaders.

EF: How would you write the HTML?

DH: Initial script tag with configuration. Second script tag as usual. Alt 2 is to have configuration and dynamic module load in the same script block.

```
<script>
ondemand
</script>
<script src="main.js" async></script>
```

alt 2

```
<script>
ondemand
System.require("main.js", function() { …. });
</script>
```

DH: script[async] today have to use an external src.

STH: Naming and declarations of modules.

ARB: Presenting slides...

AWB: The rate that internal vs external names changes is very different.

STH:

```
module 'm' { … }
module 'n' {
  import x from 'm';
  …  // this part is not executed.
}
import x from 'm';
```

STH: Configuration step is mostly about other people's code.

….

```
<script>
module 'm' { … }
module 'n' {
  import m from 'm';
  function f() {
    Loader.eval("import m from 'm'");
  }
```

```
}
</script>
```

m is fixed at compile time

ARB: Not opposed to logical modules. Wants both lexical and logical
DH: Not opposed to lexical modules.
YK: Too late to work out lexical modules for ES6.
ARB: If we wait we will have redundancy.
YK: Want declarative form to be able to prefetch etc.
BE: I want lexical modules (in the future) but logical modules are easier to use.
ARB: Since I don't seem to be able to convince anyone I'm going to drop this
ARB: For the record. Major concern about the global registry becoming the new global object.

Conclusion: Move along with Dave and Sam's proposal. Work on lexical modules for ES7

Promises vs Monads

MM: Presenting...

Thursday May 23 2013

John Neumann (JN), Allen Wirfs-Brock (AWB), Eric Ferraiuolo (EF), Erik Arvidsson (EA), Luke Hoban (LH), Doug Crockford (DC), Yehuda Katz (YK), Sam Tobin-Hochstadt (STH), Alex Russell (AR), Dave Herman (DH) (calling in), Bernd Mathiske (BM), Andreas Rossberg (ARB), Mark Miller (MM), Tom Van Cutsem (TVC), Istvan Sebestyen (IS)

Promises vs Monads

MM: Continuing from yesterday

AR: https://github.com/slightlyoff/Futures/blob/master/Promise.idl

STH: Don't like resolve but not willing to die on this hill.

AR: DOM has a bunch of ad hocs APIs to do promise like things.

YK: Mozilla is also actively working on APIs using promises.

AR: A lot of methods today return void so we can change these to return a promise. This is forward compatible.

AR: then does recursive unwrapping

...

Next Meetings

July 23 - 25 @ Microsoft, Redmond

Sept 17 - 19 @ Bocoup, Boston

Nov 19 - 21 @ PayPal, San Jose

ES6, ES7, ES8... Mark's Strawman Roadmap

LH: The important part is not the features but the process.

AWB: Can things be decoupled?

LH: These kind of structural questions are the important part

MM: Suggests "currency" to be the main theme.

AWB: Thought about the event loop. All we need is a processing queue... put things in the front and the back.

DH: Only need to add to the back.

AWB: OK.

STH: The callback is called at some later point.

AR: Don't think we need to specify the order.

STH: If we are going to specify promises etc we need to be able to specify things in detail. We can be loose in ES6 and then come back in ES7 and provide a more tight spec.

DH: We could specify the pending events as a set or something.

DH: Not sure if there is a consensus that we want a fast small ES7. Not opposed to a modularized approach.

AR: Are there any browsers that are not shipping stable ES6 features today.

YK: Yes. V8.

AWB: Where we have problem today is that there is a lot of interdependency.

MM: These ("concurrency") are coupled together to the event loop

AWB: We can do it as a separate non 262 spec

DH: Opposed to a separate spec. Introduces versioning confusion.

AWB: Roll up

DH: Think of all the extra overhead.

STH: Big difference with 402 since it was run by different people.

LH: Lack of confidence in new features has been an issue for implementers. Good exceptions were Object.observe and Proxies where the wiki contained a mostly complete spec.

AWB: We need to have wiki proposals be deltas to the spec.

TVC: We could have "stable" wiki pages. These would have complete spec deltas.

DH: Very concerned about over modularizing.

AWB: We need to find a way to work faster and be less monolithic.

DH: Agree. ES6 process has blocked implementation work.

LH: We are not committed to our designs.

STH: We are not resolving issues until we start to spec. We are not getting feedback until engines starts to implement.

EA: The problem is that we didn't start to spec things until very late. We had agreements on features long before there was any spec drafts for them.

YK: More from our champions before we get to concensus.

ARB: Lots of the proposals were very vague.

AWB: The more complete spec you bring to tc39 the better chance you have to reach consensus.

ARB: Lack of early spec leads to lack of early implementations...

AWB: ...which leads to lack of feedback.

LH: Not more work, just doing the work earlier before things pile up too much.

DH: Need to look at the dependency graph. Hold of the work of later feature.

ARB: We need to higher bar before we accept proposals.

MM: What we agreed to 2 years ago was that the features are the one we want to spend work on speccing.

LH: Less features to bite of.

DH: A lot of us have a hard time not getting too engage in too many features.

YK: if we focused more effort on managing the overall complexity instead of getting stuck on a lot of technical discussions (and nit picking).

DH: Object.observe and Proxy moved fast but are fairly isolated features

TVC: Didn't involve syntax.

AWB: With ES6 we had a long backlog.

DH: A language will have smaller and smaller complexity budgets as it grows.

AR: ES future needs events

DH: Since this is Mark's wishlist people will throw in their pet features.

MM: This is the direction I am going to work.

LH: There is a page on the wiki outlining the goals.

LH: Looking for 2 things: Something that would allow earlier implementations. Have not brought proposals (over the last 2 years) because we have been blocked by ES6.

LH: When is the appropriate time to bring new proposals to TC39?

AWB: We are free to do what we want. We can issue 6.1, 6.2 etc or technical reports which would serve as a recommendation.

DH: We cannot exclusively work on ES6.

YK: Time at f2f is the most important. Champions can go off and do what they want.

DH: Suggests adding non ES6 items to the agenda. We will prioritize the non es6 stuff we can get to given our limited time.

YK: We should reinstate the rule that agenda items needs links to wiki pages.

YK: Spec language is good but examples at the top are a must.

ARB: Add step after proposal. For example "stable" or "spec" which a proposal gets promoted to once there is a spec draft, good enough to start implementing.

DH: Strawman: Anything goes.

YK: Proposals used to mean approved.

DH: 3 sections: strawman, proposal, spec/candidate. Keep strawman. Work on improving as a proposal, and when mature enough promoted to next level.