Ecma/TC39/2015/017
Ecma/GA/2015/009

Minutes of the:	*44th meeting of Ecma TC39*
in:	*San Francisco, CA, USA*
on:	*27-29 January 2015*

1 Opening, welcome and roll call

1.1 Opening of the meeting (Mr. Neumann)

Mr. Neumann has welcomed the delegates at the Mozilla office in San Francisco, CA, USA.

Companies / organizations in attendance:

Mozilla, Google, Microsoft, Intel, eBay, jQuery, Yahoo!, Facebook, Meteor, Twitter, Netflix, Shape Security

1.2 Introduction of attendees

Brian Terlson - Microsoft

Dmitry Lomov - Google

Andreas Rossberg - Google

Adam Klein - Google

Jordan Harband - Twitter

Jafar Husain - Netflix

David Herman - Mozilla

Ben Newman - Meteor Development Group

Chip Morningstar - Paypal

Erik Arvidsson - Google

Peter Jensen - Intel

Caridy Patino - Yahoo

Eric Ferraiuolo - Yahoo

Rick Waldron - jQuery (on phone)

Waldemar Horwat - Google

Michael Ficerra - Shape Security

Allen Wirfs-Brock - Mozilla

Sebastian Markbage - Facebook

John Neumann - Chair

Jonathan Turner - Microsoft

Jef Morrison - Facebook

Brendan Eich - Self (on phone)

Mark Miller - Google

Domenic Denicola - Google (on phone)

Yehuda Katz - jQuery

Kevin Smith - jQuery

István Sebestyén - Ecma (on phone)

Sam Tobin-Hochstadt - Indiana University (on phone)

Misko Hevery - Google

Igor Minar - (Google)

1.3 Host facilities, local logistics

On behalf of Mozilla **Allen Wirfs-Brock** welcomed the delegates and explained the logistics.

1.4 List of Ecma documents considered

Ecma/TC39/2014/059	Minutes of the 43rd meeting of TC39, Santa Clara, November 2014
Ecma/TC39/2014/060	Draft Standard ECMA-262 6th edition, Rev. 29, December 6, 2014
Ecma/TC39/2014/061	Venue for the 44th meeting of TC39, San Francisco, January 2015
Ecma/TC39/2014/062	Submission of 28 IETF standards (MSP)
Ecma/TC39/2015/001	TC39 RF TG form signed by Twitter
Ecma/TC39/2015/002	Draft Standard ECMA-262 6th edition, Rev. 30, December 24, 2014
Ecma/TC39/2015/003	Agenda for the 44th meeting of TC39, San Francisco, January 2015 (Rev. 1)
Ecma/TC39/2015/004	Draft Standard ECMA-402 2nd edition, Rev. 6, January 12, 2015
Ecma/TC39/2015/005	Minutes of a TC39 ad hoc video call held on 7 January 2015
Ecma/TC39/2015/006	Draft Standard ECMA-262 6th edition, Rev. 31, January 15, 2015
Ecma/TC39/2015/007	Responses to the Ecma Contribution License Agreement (CLA), 26 January 2015
Ecma/TC39/2015/008	Allen Wirfs-Brock's slides, 27 January 2015
Ecma/TC39/2015/009	Allen Wirfs-Brock's slides on "Bug 3395", 28 January 2015
Ecma/TC39/2015/010	Allen Wirfs-Brock's slides on "ECMA-262 Introduction Text Needed", 29 January 2015
Ecma/TC39/2015/011	Dave Herman's slides on "Interfacing with the Loader Spec", 29 January 2015
Ecma/TC39/2015/012	Andreas Rossberg's slides on "Experimental New Directions for JavaScript", 29 January 2015
Ecma/TC39/2015/013	Venue for the 45th meeting of TC39, Paris, March 2015

2 Adoption of the agenda (2015/003-Rev1)

Agenda for the: 44th meeting of Ecma TC39

```
in: San Francisco - Mozilla
```

```
on: 27 - 29 Jan. 2015
TIME: 10:00 till 17:00 PST on 27th and 28th of Jan. 2015
      10:00 till 16:00 PST on 29th of Jan. 2015
LOCATION:
    Mozilla Foundation
    2 Harrison Street
    San Francisco, CA 94105
    USA
```

1. Opening, welcome and roll call
   - i. Opening of the meeting (Mr. Neumann)
   - ii. Introduction of attendees
   - iii. Host facilities, local logistics
2. Adoption of the agenda (2015/003)
3. Approval of the minutes from Nov 2014 (2014/059)
4. ECMA-262 6th Edition
   - i. ES6 End-game schedule review (Allen)
   - ii. Final RF Patent Opt-out Periods for ECMA-262-6 and ECMA-402-2(Allen)
   - iii. ES6 draft status report (Allen)
     - a. Recap of changes in recent drafts
     - b. Review significant unresolved bugs and issues
   - iv. Subclass instantiation reformation: status and open issues (Allen)
     - a. (see doc TC39 2015/005 on Ecma file server for miniutes of Jan 7 conference call)
     - b. https://github.com/tc39/ecma262/blob/master/workingdocs/ES6-super-construct%3Dproposal.md
     - c. `new.target`: in or out as user level feature for ES6?
   - v. `@@toStringTag` (rationales) (Jordan Harband) (es-discuss thread)
     - a. Should non-builtins be able to pretend to be builtins?
       - a. If yes?
         - a. Shims and faithfully built imitations can follow builtin code paths.
         - b. https://people.mozilla.org/~jorendorff/es6-draft.html#sec-object.prototype.tostring step 17.b. can be deleted.
       - b. If no?
         - a. Missing unspoofable builtin values (`Math`, `JSON`, `Object`, `Null`, `Undefined`): spec bug
         - b. Should `Object.prototype.toString` add a prefix to *all* non-built-in `@@toStringTag` values rather than a whitelist?

c. Should builtins be modifiable to pretend to be something else?

a. No?

a. Should built-in `@@toStringTag` values have `{ configurable: false }`?

b. Does that cover just `*.prototype[@@toStringTag]`, or also own nonwritable nonconfigurable properties on every builtin instance?

c. Should that apply to only ES5 builtins, or to every builtin?

vi. Proposed: class method syntax induces a non-enumerable method property, in contrast to method shorthand syntax in object literals and in concert with built-ins in the core language. (Brendan, et al.)

vii. Const in sloppy mode (Brian)

viii. Module loader spec integration (Dave)

ix. ES6 Introduction Text (Allen)

5. ECMA-402 2nd Edition

   i. Status Update (Rick)

6. ECMA-262 7th Edition and beyond

   i. Reflect.isCallable and isConstructor (Jason Orendorff)

   ii. Experimental new directions for JavaScript at Google (Andreas Rossberg)

   iii. Move Decorators to Stage 0 (possibly Stage 1) (Yehuda Katz)

   iv. Updates to Object.observe (Erik Arvidsson)

7. Test 262 Status

8. Report from the Ecma Secretariat

9. Date and place of the next meeting(s)

   i. March 24-26, 2015 (Paris - Inria)

   ii. May 27 - 29, 2015 (San Francisco - Yahoo)

   iii. July, 28 - 30, 2015 (Redmond, WA - Microsoft)

   iv. September 22 - 24, 2015 (Potland, OR - jQuery)

   v. November 17 - 19, 2015 (San Jose - Paypal)

10. Group Work Sessions

11. Closure

The agenda was approved.

3 Approval of the minutes from the November 2014 meeting (2014/059)

2014/059, the minutes of the 43rd meeting of TC39, San Jose, November 2014 were approved without modification.

4 General

The plan remains for GA approval of ES6 in June 2015.

TC39 has a close to "final draft" at Jan 2015 meeting but not the very final yet. TC39 will vote to accept and forward that to the GA in March 2015 meeting in Paris. At that time TC39 RFTG need to start a final “RF out-out period” of 60 days. It has to be finished before the GA approval on June 17, 2015.

In June 2015 a little changed ECMA-402 (Internationalization) will also come out. The same is true what was said for ECMA-262 6th Edition above.

The size and significance of ES6 for the GA as follows: ES5.1 approximately 250 pages. ES6 approximately 650 pages.

The plan is for "ES7" to have GA approval in June 2016 with yearly editions to follow.

Given the yearly release schedule TC39 would like to move away from informally talking about edition numbers (ES3, ES5, ES6, etc) and starting talking in terms like ES 2015, ES 2016, etc.). To encourage this, we would like to change the titling of ECMA-262 as follows:

Currently the document title is "ECMAScript Language Specification". TC39 would like ES6 to be titled: "ECMAScript 2015 Language Specification", ES7 would be "ECMAScript 2016 Language Specification", etc. We're just talking about the title text, the formal document numbering would remain the same, i.e. ECMA-262 6th Edition, ECMA-262 7th Edition, etc.

TC39 asked in November 2014 if that is possible? The GA answer was “yes”.

5 For details of the technical discussions see Annex 1

6 Status Reports

6.1 Report from Geneva

Mr. Sebestyen briefly reported about the 2014 GA meeting in Sapporo. He said that the June 2015 approval of ED 6 has been seen by the GA as very important and should not suffer further delay. Otherwise the GA was happy with the progress and expressed its thanks to TC39 for its hard work.

Ecma has elected new Management: **Mr. Yamashita** (Hitachi, Japan) became new president of Ecma. **Ms. Valet-Harper** (Microsoft) is Vice President and **Dr. Friedrich** (IBM) is the new Treasurer. We have two CC members, **Ms. Porath** (Intel) and **Mr. Cargill** (Adobe) but the CC Chair position is currently empty and it will be filled from meeting to meeting on ad-hoc basis.

Mr. Sebestyen said that Ecma have difficulties to ensure that all TCs are chaired by individuals coming from an Ordinary Member (Ecma Rules 6.2.7). Therefore the CC suggested that in exceptional cases the TC may approve exceptions to this rule and appoint individuals of non-Ordinary Ecma Members as Chair (Vice Chair). The December 2014 GA has endorsed this idea.

The GA agreed to the following change:

“6.

Technical Committees

6.2.7

At the first meeting of the TC which takes place after the end of the initial period, a Chairman and Vice-Chairman shall be elected from among the ordinary member representatives. However, in exceptional cases – when no ordinary member representatives are available for the chair - the TC may appoint individuals of non-Ordinary Ecma Members as Chair and / or Vice-Chair."

6.2 Json

Mr. Sebestyen has reported that in the European Commission the "identification" and "recognition" process has continued. It is rather heavy. The government of the Netherland has submitted a long list of IETF Specifications for identification and Recognition, among them JSON tailored for IETF applications (which is included in ECMAScript and as such it is also an ISO/IEC JTC 1 Standard). The two quoted IETF specification is the old expired IETF JSON RFC from 2006 and the other the not yet approved IETF JSON standard. Note: since the original transmission the new RFC 7159 is the document they want to have recognition, which makes informative (but not normative) reference to ECMA-262 and ECMA-404. Regarding ECMA-404 (JSON syntax and Interchange format) the fast-track has not been carried out yet waiting for confirmation by TC39. Some GA members expressed concern about the EC "identification" and "recognition" plans, that ECMA-404 and RFC 7159 may diverge in the future. TC39 conformed at the January 2015 meeting the fast-track of ECMA-404 to JTC 1. **Mr. Morningstar** volunteered to be the editor of the JTC 1 fast-track standard.

7 Date and place of the next meeting(s)

Schedule 2015 meetings:

- March 24-26, 2015 (Paris, France; INRIA)
- May 27-29, 2015 (San Francisco, Yahoo)
- July 28-30, 2015 (Redmond, Microsoft)
- Sept. 22-24, 2015 (Portland, jQuery)
- Nov. 17-19, 2015 (San Jose, Paypal)

8 Any other business

None.

9 Closure

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

Annex 1

https://esdiscuss.org/notes/2015-01-27

2015-01-27 Meeting Notes

Brian Terlson, Jonathan Turner, Jordan Harband, Allen Wirfs-Brock, John Neumann, Rick Waldron, Eric Ferraiuolo, Jeff Morrison, Sebastian Markbage, Erik Arvidsson, Peter Jensen, Yehuda Katz, Dave Herman, Waldemar Horwat, Alan Schmitt, Michael Ficarra, Lee Byron, Dmitry Lomov, Arnaud Le Hors, Chip Morningstar, Caridy Patino, Domenic Denicola, Mark Miller, Kevin Smith, Andreas Rossberg, István Sebestyén, Sam Tobin-Hochstadt, Brendan Eich, Jafar Hussain

Introductions

John Neumann: No changes to agenda? We can change the order as we go.

John Neumann: Any objections?

John Neumann: No objections

John Neumann: Previous meeting minutes?

Conclusion/Resolution

- Approval of agenda
- Approval of minutes

4.1 ES6 End-game schedule review

(Allen Wirfs-Brock)

(include link to slides)

Allen Wirfs-Brock: June, Ecma GA meeting:

- Approve standards
- Meet twice a year, next in Dec

Allen Wirfs-Brock: We want ES6 to be approved in June this year. What does it take to achieve that?

Allen Wirfs-Brock: Final editorial works needs to be done. Takes a couple of months. Needs to start April 1st.

Allen Wirfs-Brock: Before that TC39 needs to approve the ES6 spec. March 24-26 meeting...

Content is in slides...

István Sebestyén: Question about different dates and document versions. His problem was that the documents for opt out are not the final documents that go to the GA for approval.

Allen Wirfs-Brock: Will get to it on the next slide

Allen Wirfs-Brock: Everything I say here also applies to ECMA 402 (the i18n spec). Rick Waldron is responsible for that and it needs to follow the same milestones.

Rick Waldron: This will not be a problem for 402

István Sebestyén: The opt out period needs to be over before June 17 latest, because that is the day of the Ecma GA, when Edition 6 is up for approval. This means that theoretically we may start the opt-out on April 17 the latest. In practice of course that is too late, we should find an earlier opt-out starting date.

István Sebestyén: Anticipate large changes?

Allen Wirfs-Brock: No, I hope not.

Dave Herman: Had some module related issues but did not want to bother Allen since he was busy related to subclassing issues. If we don't resolve these issues then we will not be done.

Allen Wirfs-Brock: Does anyone expect to have issues that will cause the spec to slip?

Dave Herman: There are issues that need to be addressed, regardless of consequence.

Allen Wirfs-Brock: Need to resolve those issues at this meeting. These dates are really hard deadlines.

Dave Herman: Can try to assemble a list of these issues this week.

István Sebestyén: Is the opt out going to be on revision 31 or on the final spec?

Allen Wirfs-Brock: Can we do that with the final? Can we start the opt out on April 1st?

István Sebestyén: Why don't we start the opt out on Feb 20, and use whatever spec draft we have then?

Rick Waldron: Why can't the opt out start on Apr 1st.

István Sebestyén: Would also work. But earlier is better.

Rick Waldron: Gived more time to wrap up more issues.

Allen Wirfs-Brock: It does not really change much.

Allen Wirfs-Brock: 2 things going on: 1) TC39 approving the spec.

Allen Wirfs-Brock: Feb 20 gives us (TC39) 30 days before the f2f meeting to review the last spec. *Feb 20 is the day!*

Allen Wirfs-Brock: We don't need multiple opt outs. So the opt out period can start on Feb 20th (instead of Feb 1st as in the presentation).

4.3 ES6 draft status report

(Allen Wirfs-Brock)

(Same slides as previous item)

Allen Wirfs-Brock: Three drops since last meeting, changes in the slides as well as harmony:specification_drafts

Allen Wirfs-Brock: The HTML comment extensions in B.1.3 need to be reviewed against what browsers actually do.

Waldemar Horwat: I'm validating grammar from main body of the standard (and found some minor bugs), but not going to validate browser extensions.

Domenic Denicola: I will try to do that, maybe.

Erik Arvidsson: If we add the module tag (script[type=module]) do we need to keep this open?

Dave Herman: We can always remove the forbidding phrase if we see a need for it.

Mark Miller: Want to forbid in strict mode...

Allen Wirfs-Brock: Not possible.

Mark Miller: Ok.

Jordan Harband: Browsers (IE vs Safari vs Chrome/Firefox) do different things in regards to HTML comments. In fact, even a single browser does different things depending on where the source code is coming from (script block, eval, console, ...)

Mark Miller: if you don't require [HTML comments] or forbid it, it's optional, [which makes it hard to fix in modules later]

Waldemar Horwat: I'm objecting to have different behavior for browser ECMAScript vs non browser ECMAScript.

Dave Herman: The following program is valid when HTML comments are not supported

```
x

<!--y-->

z
```

Mark Miller: This is enough to show that there is an XSS issue. If the defender assumes HTML comments works or not.

Waldemar Horwat: You can't go down the combinatorial explosion of different browsers' or different contexts' parses. The only sane thing is to reject this of you are writing a validator.

Mark Miller: Waldemar do you agree with if we cannot prohibit it everywhere we have to allow it everywhere?

Waldemar Horwat: No.

Mark Miller: Would it be more predictable if it were allowed everywhere?

Waldemar Horwat: Not in practice because existing browsers differ in parsing the extensions. Regardless of what we enact for ES6, for security purposes you have to handle the browser diversity anyway.

Mark Miller: I give up.

Yehuda Katz: You can use CSP to prevent eval.

Mark Miller: I think the current spec is fine. It is no worse thatn the current situation.

Allen Wirfs-Brock: We need some eyes on this grammar.

Conclusion/resoltuoin

- Use the new annex spec for html comments as in the latest draft
- Need to review the grammar

Allen Wirfs-Brock:

```
let x = expr
```

gets to `x`, captures that identifier, then evaluates `expr`.

Andreas Rossberg: Not clear when this is observable?

Dave Herman:

```
obj = {x: 'here'};
with (obj) {
  x = (delete obj.x, 42);
  // {x} = (delete obj.x, {x: 42});
}
```

ecmascript#3463

Allen Wirfs-Brock: New document title

Yehuda Katz: I recalled that we talked about not changing the name until we announced the new train model.

(still during the break) discussion between Mark Miller and Yehuda Katz: about Window/WindowProxy issue, discussing concerns, esp as relates to jQuery.**Mark Miller:** preferred to keep invariants, add test262 tests, wait for browsers to fix **Yehuda Katz:** points out Object.prototype.toString.call(window) results differ (Chrome: 'global', Safari: 'window', eg) ... more discussion about configurability, differences between Window and a WindowProxy ...

4.4 Subclass instantiation reformation: status and open issues

(Allen Wirfs-Brock)

The presentation continues

Allen Wirfs-Brock: Walks through consensus from Jan 7 ad-hoc meeting (see slides)

Yehuda Katz: We should opt on the side of future proofing and fewer features.

Waldemar Horwat: Is there any cases where these uniitiallized objects can escape?

Allen Wirfs-Brock: Only if the constructor calls out before it is done with the initialization.

Allen Wirfs-Brock: Not having to worry about reentrancy or uninitialized objects leaking allowed Allen Wirfs-Brock to eliminate 20-25 pages of spec.

Waldemar Horwat: Where exactly does a constructor's [[Prototype]] point in the no-extends, extends null, and extends non-null cases?

Allen Wirfs-Brock:

- extends non-null: the superclass
- extends null: Function.prototype
- No extends: Function.prototype

Waldemar Horwat: If someone mutates prototypes, the code uses the new superclass in some cases and the old one in other cases. In particular, the extends-null case uses the old one, while replacing a non-null case with another non-null case uses the new one.

Allen Wirfs-Brock: Have ideas on fixing that.

Yehuda Katz: Can we make returning anything but undefined or an Object throw an exception?

Allen Wirfs-Brock: Then we would have different sementics between class constructors an FunctionDeclarations/FunctionExpression

Waldemar Horwat: Why are the exceptions in points 8 and 9 of Allen Wirfs-Brock's slides different? That seems weird.

1. When a function implicitly returns from a [[Construct]] invocation, if its this binding is still uninitialized a ReferenceError is thrown.
2. When a function explicitly returns a non-object from a [[Construct]] invocation and its this binding is still uninitialized, a TypeError is thrown.

Yehuda Katz/Mark Miller/Waldemar Horwat: Thought there was agreement on the list to make class constructors non-callable.

Andreas Rossberg: What about arrow functions?

Allen Wirfs-Brock: super (like this) needs to work inside arrow functions. And it needs to work as if the super call was done outside the arrow function. This more or less means that the lookup of NewTarget, HomeObject etc is looked up in the outer scope of the arrow function.

Waldemar Horwat: What if super is reentrant? Stashing it away via arrow functions allows code to reenter it.

Andreas Rossberg: Surprised and concerned about this as well. Implications unclear when super() is allowed to leak from constructor through an arrow function.

Allen Wirfs-Brock: You can call super twice but you will get an exception when you try to bind this after the second call.

Andreas Rossberg: Possible implementation concerns?

Dmitry Lomov: Should work fine.

Waldemar Horwat: Wondering about calling super twice and having it throw only after the second call completes and tries to assign to this. Not sure about all the implications yet, but ok with it — the alternatives are worse.

Mark Miller: TCP for `super.foo` is clear bt what about `super()` in a constructor?

Allen Wirfs-Brock: Both works in the latest draft.

Waldemar Horwat: What is the NewTarget in the case of `new super()`?

Allen Wirfs-Brock: The NewTarget is whatever was part of the `new func` so in this case `funct`. The NewTarget gets forwarded down.

Waldemar Horwat: OK, so if you have a chain of [[construct]] calls, it's the base of the chain. What happens when new super() gets called from a function invoked via [[call]] instead of [[construct]]?

Allen Wirfs-Brock: Error. Have to verify how it is specced.

Dmitry Lomov: We could also use the function itself.

Allen Wirfs-Brock: That is an option. [Checking spec.] -- throws a refernce error if there is no NewTarget which is the case when the function was [[Call]ed and not [[Constructed]].

Allen Wirfs-Brock: My suggestion is to keep `new super()`.

Yehuda Katz: Wants to dissallow all forms of super outside of class bodies

Allen Wirfs-Brock: The primary utility is that you can use it in cases where `super()` does not work.

Dmitry Lomov: You cannot use `new super()` in method.

Yehuda Katz: Wants to do a max-min approach here.

Dmitry Lomov: are there critical use cases for new super() in constructors? I guess no

Allen Wirfs-Brock: confirms

Brian Terlson: how concern are you about removing `new super`?

Allen Wirfs-Brock: Should not be too bad. Pretty self contained.

Brian Terlson: Anyone defending new Super()? *crickets*

Allen Wirfs-Brock: Next, do we need syntax to manifest NewTarget?

Yehuda Katz: Early strawman was `arguments.constructor`.

Mark Miller: `<keyword>`.prop gives us new

Allen Wirfs-Brock: Sees options in the future to add things like `function.callee`

Allen Wirfs-Brock: Maybe it could be `class.target` but `new.target` is more accurate. These are called MetaProperties in the spec.

Mark Miller: Thinks that we should include `new.target`. Understands the objections but it is a very useful thing to have and it reads very well.

Mark Miller: Anyone object to have `new.target` in ES6?

Yehuda Katz: Wants to talk about callable constructors before we make a call [no pun intended].

Yehuda Katz: OK with `new.target` if constructors are not callabele.

Allen Wirfs-Brock: Where is `super()` allowed? Currently only allowed in class constructor bodies.

Andreas Rossberg: Makes sense at this point. We have had multiple threads about this?

Domenic Denicola: Would `new.target` work in FunctionDeclaration/FunctionExpression?

Allen Wirfs-Brock: Shoudl work.

Yehuda Katz: Objects. We should opnly allow `new.target` in places where super is allowed.

Allen Wirfs-Brock: `new.target` can be used anywhere where [[Construct]] is used

Dmitry Lomov: `new.target` gets set/bound when [[Construct]] is done. This is valid for normal functions as well constructors

Andreas Rossberg: There is no design issues here. It can only mean one thing.

Erik Arvidsson: `new.target` is tied to [[Construct]], not super.

Yehuda Katz: OK.

Waldemar Horwat: What happens if you do ES3 style constructor/prototype?

Allen Wirfs-Brock: The NewTarget gets set as you do [[Construct]]

Erik Arvidsson: You can pass along NewTarget if you use `Reflect.construct`

Yehuda Katz: A possible semantics for new.target on [[Call]] is to throw.

Mark Miller: represents Yehuda Katz's point. If class constructors are not constructable and we disallow new.target in function declarations, then we don't have to answer the question of what happens with new.target on [[Call]]

Allen Wirfs-Brock: ???

Domenic Denicola: Do we want people to refactor functions to classes. Then supporting [[Call]] and [[Construct]] is important.

Allen Wirfs-Brock: Not sure if we are going to resolve these things quickly post ES6?

Yehuda Katz: No chance of getting concensus for ES6.

Domenic Denicola: If someone writes a FunctionDeclaration and wants to enforce that it is called with new, then if we don't have `new.target` people will go back to instanceof and other broken heuristics.

Dmitry Lomov: You could also use new.target usefully in a [[Call]] context - to assert that it is not a [[Construct]] context

Brendan Eich: We don't know what the future holds but we do know that functions in the past have been callable and constructable. What we do know is that`new.target` is useful.

Waldemar Horwat: If we're going to use new.target === undefined to distinguish [[Call]] from [[Construct]] contexts: Is is possible for new.target to be undefined inside a [[Construct]] context? Even if it's called via reflection?

Allen Wirfs-Brock: No, that's never possible.

Mark Miller: Reflect.construct(??, null) is a type error?

Allen Wirfs-Brock: Yes, it is a TypeError

Allen Wirfs-Brock: null or undefined

Mark Miller: undefined is least surprising.

Erik Arvidsson: Because missing is often signalled with undefined.

Brendan Eich: Strict functions uses undefined for missing this.

Brian Terlson: Is `new.target` undefined in global scope.

Allen Wirfs-Brock: It is SyntaxError in global scope, just like super and return.

Mark Miller: Is it a syntax error in concise methods? They do not have a construct.

Allen Wirfs-Brock: That would include more exceptions and reduce consistency.

Mark Miller: `new.target` is also tcp through arrow functions.

Conclusion/resolution

- Remove `new super()`
- `new.target` is available in all functions. It is `undefined` in a non [[Construct]] call.
- `new.target` in arrow functions gets the NewTarget of the outer context (not the arrow funciton context).

super() outside class constructor body

(Champion???)

Dave Herman: ????

Yehuda Katz: A constructor is both callable and constructable. If you use `super()` in call context you do not know if it is a call or construct.

Brendan Eich: super-dot is orhtogonal to super-call

Mark Miller: Do we have concensus on super() is only allowed in constructor? And constructors are only [[Construct]]able, i.e. their [[Call]] trap throws?

Mark Miller: Would like to postpone toMethod post ES6?

Yehuda Katz: Current design is insufficient.

Domenic Denicola: What about if someone wants to use super in functions'

Yehuda Katz: They should just use classes.

Brendan Eich: Deferring toMethod and making a call on a class throw seems like the most future proofing.

Waldemar Horwat: Can you explain what toMethod is?

Allen Wirfs-Brock: Explains the spec

Yehuda Katz: Concerned about having `toMethod(homeObject)` and then change the arity to `toMethod(homeObject, propertyName)`.

Allen Wirfs-Brock: Just reserve the second argument.

Dmitry Lomov: As a user, what do I pass?

Waldemar Horwat: Also wants to defer toMethod.

Allen Wirfs-Brock: Wants one more go and then I'll give up

Allen Wirfs-Brock: People that wants to do meta programming. Polyfills, compilers. All of these people are now shut out.

Domenic Denicola: Without toMethod you cannot add a method that wants to call super to an existing class. E.g. for a polyfill on a DOM class; especially custom element callbacks.

Mark Miller: When we started we set out with a design that could explain classes using ES5 semantics. We've lost this with TDZ.

Domenic Denicola: Feels strongly that classes should desugar to smaller primitives. Since we have already lost that we can/should come up with a more consistent design post es6.

Dave Herman: We have to go from both ends. We are not abandoning, just deferring.

Brendan Eich: We want kernel semantics. But we should not rush this. We will regret rushed decisions. We do not have solid design at this time so we must defer.

Dmitry Lomov: No `super()` outside class constructors since this is already initialized.

Allen Wirfs-Brock: My bottom line; `new.target` and `super.property` are the essential things.

Dmitry Lomov: extends null?

Allen Wirfs-Brock: We can drop the special dynamic case for extends null. Can have the constructor return an object it makes itself via Object.create.

Erik Arvidsson: If you have an extends clause, the default constructor calls super. If the extends expression evaluates to null then the super() call will throw. So you have to provide your own constructor

```
class Foo extends null {
  constructor() {
    return Object.create(new.target.prototype);
  }
}
```

Allen Wirfs-Brock: If not null then the extends expression must be IsConstructable.

Dmitry Lomov: What is the canonical way to create a class that has null as the [[Prototype]] of the instance

Mark Miller:

```
class C {
  constructor() {
    return Object.create(new.target.prototype);
  }
}
C.prototype.__proto__ = null;
```

Waldemar Horwat: We special case the null value in the extends clause. If null we use %FunctionPrototype% as the superParent. We use null as the protoParent. Get rid of the specialized treatment of super() for null extends; instead, use the same semantics for null and non-null extends, which will make any attempt to call super() with a null extends throw.

```
class C extends null {
  constructor() {
    return Object.create(new.target.prototype);
  }
}
```

Allen Wirfs-Brock: `super.property` in object literals works in concise methods and accessors.

Allen Wirfs-Brock: Do we have concensus to remove `super.property` from concise methods in object literals.

Yehuda Katz: I think that is unfair. We have made a lot of changes.

Dmitry Lomov: Concise methods are not constructable. Correct?

Allen Wirfs-Brock: Correct.

Domenic Denicola: We cannot reopen unrelated issues at this time of the process.

Yehuda Katz: I'm passionate about the syntactic sugar related to `super()` for `super.propertyName()`. This syntax got removed and we had to do some fairly significant changes. And this is related.

Waldemar Horwat: Was under the impression that `super.property` was only going to be available inside class methods.

Allen Wirfs-Brock: There is no distinction between class methods and object methods.

Mark Miller: When we talked about this before I was also under the assumption that this was about inside classes and outside classes.

Conclusion/resolution

- Dropping toMethod from ES6.
- No `super.property` in FunctionExpression/FunctionDeclaration (as well as generators).
- No `super()` outside class constructors.
- Class constructor [[Call]] always throws.
- `extends null` does not have special dynamic default constructor. The syntactic presence of extends clause causes [[ConstructorKind]] to be derived.
- No concensus on super inside concise methods of object literals. (to be revisited)

Methods in class bodies enumerability?

Allen Wirfs-Brock: Is anyone objecting to making methods in class literals non enumerable?

No objections

Conclusion/resolution

- Methods in class bodies are non enumerable.
- Concise methods in object literals are still enumerable.

https://esdiscuss.org/notes/2015-01-28

2015-01-28 Meeting Notes

Brian Terlson, Jonathan Turner, Allen Wirfs-Brock, John Neumann, Rick Waldron, Jeff Morrison, Erik Arvidsson, Peter Jensen, Yehuda Katz, Dave Herman, Waldemar Horwat, Dmitry Lomov, Domenic Denicola, Kevin Smith, Andreas Rossberg, István Sebestyén, Sam Tobin-Hochstadt, Michael Ficarra, Jordan Harband, Chip Morningstar, Mark Miller, Ben Newman, Brendan Eich, Adam Klein, Igor Minar, Misko Hevery

Report from the Ecma Secretariat

(István Sebestyén)

István Sebestyén: Strange JSON development: The government of the Netherlands wants to get an EU recognition on IETF's JSON future standard as part of the European Commission's Multi-Stakeholder Platform. Netherlands submitted 27 IETF standards to the Multi-Stakeholder Platform, and IETF's JSON happens to be among them. Normally only long-standing in the market already proved standards should be recognized. Also no "consortium specification" should be recognized where appropriate International Standard (like of ISO, IEC, ITU) exists. JSON in ECMA-262 is already part of ISO/IEC 16262:2011. In addition TC39 has standardized in 2013 ECMA-404 (JSON). We had originally a TC39 decision to fast-track it to JTC1, but after discussion the GA in December 2013 has approved the possibility of a Postal Balot on the JTC1 fast track upon a new request from TC39. The iisue has been reported to the last Ecma GA and was worried about the situation.

Waldemar Horwat and István Sebestyén: [Discussion about what the Multi-Stakeholder Platform is and its implication on governmental activities]

István Sebestyén: Recommend resuming the fast-track of ECMA-404 (JSON).

Allen Wirfs-Brock: We don't currently have an editor for the JSON spec. Who wants to do it?

Chip Morningstar: I volunteer!

Waldemar Horwat: Timeline? June GA?

István Sebestyén: Will discuss it with ECMA management. Might do it earlier via a letter ballot.

ES6: Generator issues

(Allen Wirfs-Brock)

Waldemar Horwat/Dave Herman: Ask for presentation.

Allen Wirfs-Brock:

Dave Herman: Earlier presentation was careful to allow try/finally to abort the return. The point was to allow infinite iterators that refuses to return.

Ben Newman: Filed this bug. You could change the return method to return `{done: true}`.

Dave Herman: You can have wrappers that are specific for for-of loops.

Allen Wirfs-Brock: THe original proposal required a falsy done value. If it wasn't falsy an error was thrown.

Dave Herman: You could change the for-of loop by wrapping the iterable to not continue the iteration of the underlying

```
var nats = getNats();
for(x of nobreak(nats)) {
  if(cond) break;
  ...

  for(x of nobreak(nats)) {
    if(cond) break;
  }
}
```

```
function nobreak(it) {
  return {
    next(x) { return it.next(x); },
    throw(x) { return it.throw(x); },
    return(x) { return {done: true}; }
  };
}
```

Yehuda Katz: If someone asks to cleanup, by calling return, then you should clean up.

Kevin Smith: There should be an equivalent of an iterator with a return and without a return. (?)

Dmitry Lomov: having a nobreak is an abstraction breakage ; you have to know whether your iterator is infinite or not, or rather, has something useful to do in return

Waldemar Horwat: Why is return(x) stubbed out but not throw(x) in Dave Herman's nobreak?

Waldemar Horwat: What happens when there is a throw inside the for-of loop body. Does it call "throw" or "return"?

Allen Wirfs-Brock: (looks up semantics)

Brendan Eich: If there is a throw in the for-of, the "return" method is called, not the "throw" method.

Dave Herman:

```
var nats = getNats();
nats.return = () => ({done: true});

for(x of nobreak(nats)) {
  if(cond) break;
  ...
  for(x of nobreak(nats)) {
    if(cond) break;
  }
}

function nobreak(it) {
  return {
    next(x) { return it.next(x); },
    throw(x) { return it.throw(x); },
    return(x) { return {done: true}; }
  };
}
```

Allen Wirfs-Brock: Why can't we make that an iterator that says that it is done, that it can never return `{done: false}` again?

Dave Herman: Are you sayng that the default behavior should be `{done: false}`?

Allen Wirfs-Brock: Or that there is no default.

Dave Herman: There has to be a default behavior.

Allen Wirfs-Brock: If there is not return method then nothing is called.

Dmitry Lomov: As far as I understand, the intent was to close the iterator for for-of loops.

Jonathan Turner: The iterator consumer should not need to know if the iterator is infinite or not.

Allen Wirfs-Brock: We cannot enforce that `{done: true}` does not change to `{done: false}` in the future.

Dmitry Lomov: The contract of an iterator is that once it is closed you cannot get any more values out of the iterator.

Allen Wirfs-Brock: You cannot enforce that. It is up to the implementer of that abstraction.

Dmitry Lomov: Normally, you create a new iterator every time (in for-of)

Dmitry Lomov: Why is tehre a difference between finite and infinite iterators?

Dave Herman: What Jafar brought to the table was that infinite iterators are not the common case. The common case is finite, like file streams etc. You want these to close by default. We should not optimize for inifinite sequences.

Dmitry Lomov: If you don't want to close then you should not use for-of.

Dave Herman: That is also Jafar's view.

Allen Wirfs-Brock: Generators work this way. They do the right things.

Yehuda Katz: Was persuaded by `{done: true}` meaning that I accepted your hand shake.

Allen Wirfs-Brock: Say that this is just a hand shake is a completely different thing than what we have.

Dave Herman: Pretty persuaded that we do not have to catch this bug.

Dmitry Lomov: You could throw in the return method.

Dave Herman: If you have an infinite sequence that you don't want to be able to close, throwing in return would signal that the user is using it wrong.

Ben Newman: If we adopt the policy that {done: false} would throw, it is not very different from than throwing in return.

Erik Arvidsson: Using throw in your return is also a clearer intent.

Waldemar Horwat: How would Allen Wirfs-Brock's original motivational example (of a reusable infinite iterator) be done?

Allen Wirfs-Brock/Ben Newman: The user would have to wrap the iterator and/or use try/catch

Ben Newman: Do we need to consider generators and try/finally and make sure we are not missing an opportunity to prevent bugs?

Ben Newman: no compelling argument to throw for iterators that do not have a return method, since they can just implement .return() to throw if they really want that behavior.

Conclusion/resolution

- Remove the truthy check for Get(result, "done") that follows the call of "return". (The value continues to have to be an Object.)
- Relevant bug closed with the following comment: ecmascript#3395#c2

4.5 @@toStringTag

(Jordan Harband)

Mark Miller: Do we check the natives before checking the symbol?

Allen Wirfs-Brock: Step 17b, for natives we get both and then if the tag is different a tilde is added.

Jordan Harband: We could make all builtin instances have own non configurable property

Mark Miller: Or skip the override for the built ins.

Jordan Harband: Not looking for complete security. Just want to error if the user does crazy things.

Mark Miller: In es6 classes, if you extend a RegExp the instance is going to continue to be a regexp instance.

Jordan Harband: One use case is for a module that wants `new Boolean(false)` to be same as `false`. Not saying that is a good library, just explaining how it works. With ES6 and someone overrides @@toStringTag these tests no longer work.

Yehuda Katz: If someone sets the toStringTag to Boolean they probably did that because they want to go down the Boolean code path.

Domenic Denicola: If you want to do it the right way there are other ways. Allen posted this to es-discuss. esdiscuss.org/topic/tostringtag-spoofing-for-null-and-undefined#content-59

Kevin Smith: There are builtin methods that throw if the internal slot is not present.

Waldemar Horwat: Allen Wirfs-Brock's brand checks are optimized for the true case. That's ok for error checking (I expect x to be primitive Foo and don't care how slow the code is if it's not) but not good if you just want to reliably test if x is primitive Foo.

Domenic Denicola: Believes that there is a method that does brand checking for all the builtins.

Mark Miller: Checked Caja for brand checks. Found 5 cases. Not clear if they would be safe with alternative brand checks. I'm convinced that Allens proposal is safe enough.

Allen Wirfs-Brock: Then we can step 17 (the tilde case).

Mark Miller: Just to be clear, there is no way to change what typeof returns.

Mark Miller: Issue is ES5 libraries coexisting with ES6 clients.

Jordan Harband: What if someone claims that a String is an Array.

Yehuda Katz: Then they get an error which is what you would expect.

Mark Miller: Requests that browser makers wait for at least 2 months before releasing a browser that drops step 17. The integrity of Caja is at stake.

[agreed to Mark Miller's request]

Resolution/conclusion

- Remove step 17b in the alg (the tilde prefix requirement). 19.1.3.6 Object.prototype.toString ( )
- Agreed to wait 2 months before releasing a browser that does this

Super revisited

Yehuda Katz: Happy with the outcome.

Conclusion/resolution

- `super.property` is available in all concise methods (including accessors) in object literals.

Experimental new directions for JavaScript at Google

(Andreas Rossberg)

[Slides: www.mpi-sws.org/~rossberg/papers/JSExperimentalDirections.pdf]

Andreas Rossberg presenting

Allen Wirfs-Brock: [On methods doing instance brand checking.] Why cant you make the methods non extractable instead?

Andreas Rossberg: Then you probably need [[Invoke]] which is problematic too.

Waldemar Horwat: What about calling a funciton with too many arguments?

Andreas Rossberg: We discussed this. People want it for API evolution. Allowing it makes sense with subtyping.

Dave Herman: Does this mean that V8 would be slower for non sane js?

Andreas Rossberg: Of course not.

Peter Jensen: How is this going to make the VM code simpler.

Andreas Rossberg: It doesn't, but it might avoid making it yet more complicated in the future.

WM: The goal is to make performance more predictable.

Waldemar Horwat: Trying to probe the interaction of sane and non-sane modes. I assume that the sane mode will have strings and a string generated in non-sane mode code will be usable in sane code and behave and be typed as a string.

Andreas Rossberg: Yes.

Waldemar Horwat: Then what happens if in sane mode I take strings s and t and call s.concat(t). Does sane mode statically determine that the result is a string (which would cause issues if String.prototype.concat is monkey-patched), or does it do the conservative thing and assume that the result of the concat call is any, in which case I can't assign it to a string variable.

Andreas Rossberg: Use any<string>

Dave Herman: Better-is-better vs. worse-is-better

Domenic Denicola: I see this as an alternative to asm.js but writable by people.

Dave Herman: I see building on asm.js as an alternative way.

Andreas Rossberg: We are talking to the asm.js team. Both valuable, goals are mostly orthogonal. user-facing vs compiler target.

Dave Herman: Believe we can add parts of this in one js.

Waldemar Horwat: [In response to OneJS assertion that we should wait for us to make the entire language more optimizable instead of making high-efficiency subsets.] We've been going in the opposite direction. We are making things harder to implement and reason about in ES6. For example Proxies.

Dave Herman: would prefer implementors' time be spent optimising existing Jaswanth Sreeram instead of optimising this subset

Alex Russell: False dichotomy: subsetting means that it *is* existing Jaswanth Sreeram.

Brendan Eich: Think that it might be hard to pull this off but it might be informative for future Jaswanth Sreeram features.

Sam Tobin-Hochstadt: You cannot go from an unsound type system to the system you have in your vm.

Brendan Eich:

Sam Tobin-Hochstadt: Unsound type systems are not usable for performance. If we want types to help the VM it is not going to happen as an evolutionary type system on top of TypeScript or Flow.

Brendan Eich: Concerned that it is taking resources away from other more important things.

Jonathan Turner: Even unsound types can be used as hints to the VM.

Andreas Rossberg: Yes, but not much more. You only save a warm-up cycle compared to the current state of affairs. The real perf issues we see is when the type assumptions we make are wrong and we get into deopt/opt cycles. It is very hard to reason about when this happens and it does happens when your application gets larger. Also, more aggressive optimisations require global invariants, which you cannot achieve without globally sound types.

Yehuda Katz: As engines gets smarter the risk of hitting the pathological cases is getting worse.

Yehuda Katz: We hope the engines are not changing their heuristics because we are currently targetting one heuristics.

Andreas Rossberg: Exactly. The smarter VMs get, the less predictable performace will be. Types are the only way out we know of.

Yehuda Katz: I hope that you are planning to write a transpiler.

Andreas Rossberg: Yes, we start prototyping in Traceur.

Brendan Eich: Why was flow not consired?

Andreas Rossberg: Flow is going in the opposite direction. It is doing more inference. It is a non starter for doing inside a VM.

Andreas Rossberg: Typed racket great, but occurrence typing can potentially get expensive.

Sam Tobin-Hochstadt: That is not accurate. THere are ways that you can make things slow but there is no inherent reasons it needs to be slow.

Brendan Eich: If you want adoption. Peopke will go the extra distance to get the predicable performance.

Yehuda Katz: Can imagine a world where a language significantly compatible with Jaswanth Sreeram has usages.

Jeff Morrrison: Flow would want to be compatible with whatever we develop.

Andreas Rossberg: It is complementary. Write in Flow style and have Flow insert type annotations for you.

Brendan Eich: ... Java failed, ActionScript failed

Sam Tobin-Hochstadt: I would not say Java failed. People like static typed languages.

Waldemar Horwat: If Java failed, then what succeeded? Objective C?

Jeff Morrrison: There are several large Jaswanth Sreeram code base projects and a lot of them came to the conclusion that static types helps them manage their code.

Sam Tobin-Hochstadt: The difference between Andreas' proposal is that there is a concrete plan for VM optimization.

Brendan Eich: Afraid that this will be designed in some corner inside Google.

Andreas Rossberg: No. We just had a meeting between all parties interested in types for Jaswanth Sreeram, presenting this. And we totally intend to continue those.

Yehuda Katz: It is important that interested parties are part of the designing this.

Andreas Rossberg: Someone has to make this experiment. We cannot spec a type system without a working implementation. We need to explore and validate options.

Dave Herman: Interop layer. How do these things interact? How smoothly can new and old code interop? In your part of the space you need soundness and you need to drop some of the constraints. It is not an end user target. It is either a target for compilers or for code where you really need to squeeze out maximum performance. What can you do in a type system and keep it smooth.

Waldemar Horwat: This has different goals from asm.js. Asm.js does manual storage management instead of gc; this effort doesn't go that far.

Andreas Rossberg: We did talk to asm.js people to make a version of asm that would support gc etc.

Sam Tobin-Hochstadt: If someone had said 3 years ago that types would be the next big thing in js we would not have believed them. It happened (TypeScript etc)

Brendan Eich: Try to make it a name that impies that it is natural progression to js. You should try to deemphasize the mode.

Domenic Denicola: Calling it "sane mode of Javascript" instead of "sanescript" would be much better.

Brendan Eich: What about "use types"?

Kevin Smith: It is not clear to me what the dependency is between the new mode and type?

Andreas Rossberg: Neither am I. :)

Dave Herman: What matters is how far you are diverging from the language.

Andreas Rossberg: Some traditional patterns you do not need any more with ES6. For example Maps makes usage of Objects as a map obsolete.

Dave Herman: You are better doing these things incrementally. One thing at a time. If one of these succeeds or fails it does not bring everything down or up.

Waldemar Horwat: Little incremental steps cause too much complexity due to the constant arms race to fool existing code while introducing new features. Look at the evolution of ways of defining properties.

Dave Herman: We can all list flaws with Jaswanth Sreeram.

Waldemar Horwat: This proposal is one of the few that reduce entropy.

Dave Herman: Not true. It sits on top of the existing language.

Yehuda Katz: Makes the way the language feels very different.

Domenic Denicola: Jaswanth Sreeram stays within the bounds. It is like a really strict linter. Don't think there are that big of a difference.

Allen Wirfs-Brock: Great to do experiments. New mode. Substantial different language. Skeptical to get this rolled out in the short term.

Andreas Rossberg: One design goal is that a "correct" program that works in sane mode engine will work in all ES6 engines.

Mark Miller: Plausible route to success. Frozen classes can be adopted. Almost identical to const classes as previously proposed. Want to bring this up after ES6. For things that do not fit that model, the train wreck you are going towards is sharing primordials. You end up duplicating all the primordials. The realm api we are working on is an alternative way to approach this.

Andreas Rossberg: There is no intention to have to duplicate all the primordials.

Mark Miller: As you continue towards this goal you will continue to duplicate more and more primordial.

6.3 Decorators

(Yehuda Katz, Jonathan Turner)

jonathandturner/brainstorming/blob/master/syntax.md, jonathandturner/brainstorming/blob/master/README.md

Jonathan Turner: TypeScript wanted annotation/decorators. Talked to Yehuda and agreed to try to come up with a common proposal.

Allen Wirfs-Brock: Is using @ going to cause grammar issues with using @ for private state.

Erik Arvidsson: @ is widely used in other languages for annotations.

Mark Miller: would rather talk about semantics than focusing on the token.

Yehuda Katz: Want to resolve it now because wants to move forward. Wants to move to stage one

Allen Wirfs-Brock: Does not matter. There is no commitment on the syntax for stage 1.

Yehuda Katz: We are reaching a point where people are using ES6 class syntax and run into issues where it is not expressive enough. They end up adding their own syntax.

Waldemar Horwat: There seems to be an ambiguity with the syntax for Decorator.

```
@x instanceof (y) ...
```

Waldemar Horwat: is instanceof the name part of a MethodDefinition or a continuation of the decorator expression? An ambiguity like this can also cause the lexer to go off the rails if you concoct a more complex counterexample with a / inside it and you wouldn't even be able to lex to the end of it to see what follows.

Waldemar Horwat: One option to solve this is to limit decorators to primary or simple expressions. Binary operators cause issues. If you want a complex expression as a decorator, just parenthesize it.

Yehuda Katz: So we can solve by restricting the grammar

Brian Terlson: Or using a delimiter like colon after the annotation

Yehuda Katz: Basic idea of decorators is that there are 2 things that can be decorated. Classes and class methods. Give the decorator a chance to intercept. They take the same params as defineProperty and can return the descriptor that is finally installed.

```
({
  foo: observes(function() {
  }, prop)
})
```

Mark Miller: It is really just a desugaring of wrapping the defineProperty calls.

Mark Miller: Do super work after the method has been decorated.

Yehuda Katz: There is one quirk with getters and setters. They are installing a single property descriptor but there are multiple instances in the syntax.

Erik Arvidsson: We have issues with duplicates

Allen Wirfs-Brock: We have tried to solve this in the past (to bind the get and set pieces together).

Mark Miller: The idea that you are decoratign a descriptor and not just the value is powerful.

Domenic Denicola: You could pass in a flag.

Jonathan Turner: Lets pop the context and get a good overview before we start diving into the details.

Yehuda Katz: Noted that there are quirks.

Yehuda Katz: @readonly is a simple example of decorators.

Igor Minar: Annotations doc goo.gl/42o1oo (requires permissions) Decorators vs Annotations:docs.google.com/document/d/1QchMCOhxsNVQz2zNvmzy8ibDMPT46MLf79X1QiDc_fU/edit# (requires permissions)

Igor Minar: What do we want annotations for? Collected use cases in document. For example for documentation, ORM, DI, tooling aspects

Waldemar Horwat: What is the syntax for annotations?

Yehuda Katz: At this point it is the same as decorators.

Igor Minar: Biggest difference between deocrators and annotations is that annotations support static/ahead of time tooling because it has guarantees you can build on top of.

Waldemar Horwat: Big issue is whether you use a separate sublanguage for static annotations or try to use the same language. Separate language using separate name scopes results in solutions and problems such as C's preprocessor. Using the same language causes severe time-of-evaluation issues, as seen in Lisp's eval-when or C++'s template and class-vs-constexpr gotchas [example: a C++11 class constexpr reference can't use static constexpr functions defined earlier in the same class]. The problem arises in doing compile-time scope lookup and variable resolution in scopes as dynamic as ECMAScripts's.

Mark Miller: Do you expect to be able to have identifier expressions in your annotations? Do you expect that to look up the value in the current scope?

Erik Arvidsson: The original proposal used expressions. There has lately been talk about scaling this back to JSON so that there are no side effects.

Mark Honenberg: Using JSON only seems like it might not be suitable enough.

Allen Wirfs-Brock: There is a middle ground here. Classes could expose an extensible meta-object protocol. Essentially dynamic extension points that you can hook in to. So as part of defining a new type of class definition you might add meta properties and other things. Might be quite dynamic but have a static representation in the source code.

Mark Miller: If JSON is almost expressible enough, with the exception of the scope issue, and if you add IdentifierExpression

Igor Minar: We might be happy with that.

Igor Minar: One other reason for having these side effect free annotations is the VM perspective. Decorators needs to be dynamically invoked.

Domenic Denicola: The performance impact would not be worse than calling the functions yourself.

Dave Herman: Meta programming is generally the case where you do not need to optimize for performance because you expect there to be some penalty.

Mark Honenberg: If you only have decorators and the common use case is annotations ????

Domenic Denicola: If you are in a static environment you can already treat decorators as a special form.

Mark Honenberg: The order of decorators matter. The order of annotations does not matter. Therefore it does not compose.

Yehuda Katz: No one is saying that you cannot prevent people from using decorators for anything other than attaching meta data.

Mark Honenberg: Looks like decorators can do anything but in practice they cannot do much without using global state.

Yehuda Katz: Disagree. They can use the object as the coordination point.

Mark Honenberg: Lets assume I'm creating a component.

Yehuda Katz: You can use a weak map.

Mark Miller: Can you clarify.

Mark Honenberg: I think it is a good property that you can bootstrap the same framework multiple times. The problem is that a decorator needs to register itself with some context.

Yehuda Katz: The decorator can add meta data. Then you call the register with the relevant context.

Mark Honenberg: Yes. But the extra power is problematic.

Mark Honenberg: Depends on when you are running the code. Would like to wait until the framework has enough context.

Allen Wirfs-Brock: Maybe have two different proposals, both at stage 0. Then try to unify them before they can progress to stage 1.

Mark Miller: Having one mechanism that satisifes both needs is preferable to having two different mechanism.

Mark Miller: Maybe you can have a statically verified subset of decorators and if that restriction is fulfilled your tooling can treat it as an annotation. If not it falls back to a dynamic decorator.

Igor Minar: Sounds promising.

Conclusion/resolution

- None?

https://esdiscuss.org/notes/2015-01-29

2015-01-29 Meeting Notes

Brian Terlson, Jonathan Turner, Allen Wirfs-Brock, John Neumann, Jeff Morrison, Erik Arvidsson, Dave Herman, Waldemar Horwat, Domenic Denicola, Kevin Smith, Michael Ficarra, Jordan Harband, Chip Morningstar, Adam Klein, Igor Minar, Misko Hevery, István Sebastyan, Rick Waldron, Ben Newman, Yehuda Katz

Not present?

Peter Jensen (Peter Jensen), Dmitry Lomov (Dmitry Lomov), Sam Tobin-Hochstadt (Sam Tobin-Hochstadt), Mark Miller (Mark Miller), Brendan Eich (Brendan Eich),

PLEASE UPDATE THE PARTICIPANTS LIST!

ES6 Introduction text

(Allen Wirfs-Brock)

[Ask for slides]

Allen Wirfs-Brock: The spec has an introduction text. We need to write what ES6 is adding to the picture. Right now it is empty.

Allen Wirfs-Brock: Asked for volunteers and no one has done that yet.

Possible new text

ECMAScript is now one of the world's most widely used programming languages; it has been adopted not just by browsers but also for servers and embedded applications, and has become a compilation target for other languages. The 6th edition provides the most extensive set of enhancements to ECMAScript since the 1st edition.

It is now a comprehensive general purpose programming language.

Some of the major enhancements provided by the 6th edition include modules, class declarations, lexical block scoped declarations, iterators and generators, promises for asynchronous programming, destructuring patterns, and proper tail calls.

The ECMAScript library of built-ins has been expanded to support additional data abstractions including maps, sets, and arrays of binary numeric values as well as additional support for the Unicode supplemental characters in strings and regular expressions. The built-in library is now extensible via subclassing.

New uses and requirements for ECMAScript continue to emerge. The sixth edition provides the foundation for regular, incremental language and library enhancements.

List of contributors?

(Allen Wirfs-Brock)

Allen Wirfs-Brock: In ES5 there is a list of people who contributed to the spec. For ES6 the list is really large and I'm afraid of leaving someone out. Suggests not having a list of contributors.

István Sebestyén: Non normative. Non mandatory. ECMA leaves it entire to TC39 if they want to have a personal and/or organizational recognition of those who have significatntly contributed to the standard. This can be from case to case different. E.g. in ECMA-402 (ECMASCript Internationalization) all contributors and those who even commented were acknowledged, but of course that standard has a much more focused scope. So it is up to TC39. If there is a risk of forgetting someone who has significantly contributed to the work on ES6 maybe it is better to leave it out.

Brian Terlson: Is anyone asking for this?

Jeff Morrrison: Does anyone oppose to leave this out?

Domenic Denicola: It feels nice to get acknowledged, and we should consider doing it in the future. Especially for e.g. ES2016 when we're on GitHub and can just look at the contributors list. But it's just not feasible for ES2015.

Allen Wirfs-Brock: Suggests not having the list of contributors.

Brian Terlson: Seconded.

John Neumann: Would it make sense to list the companies that contributed.

Waldemar Horwat: That would be worse.

Erik Arvidsson: Agreed.

István Sebestyén: He noted that in the past in Ecma we even had a few cases when companies have been acknowledged, or persones (with their companies in bracket). Up to the TCs how they want to handle this, because this part goes into an informational part of standard anyway.

Conclusion/resolution

- ES6 will not have a list of contributors.

March meeting

(John Newman)

John Neumann: Next meeting is in Paris. Today is the last chance to object.

Waldemar Horwat: Objecting, but will not veto it.

Conclusion/resolution

- The Paris meeting has been confirmed

István Sebestyén: note after the meeting: We have to take into account that like in the SF meeting there will be members who will want to attend remotely via conferencing facilities. Therefore the host should be prepared for that and should provide a well functioning remote conferencing facilities. In the SF TC39 meeting the video part and the remote audio participation worked well, but the audio from the conference room could only be understood if one person alone close to the mic spoke.

const in sloppy mode

(Brian Terlson)

Brian Terlson: Found one site that depends on the legacy const semantics.

Yehuda Katz: One site?

Brian Terlson: It is also a demo site.

Brian Terlson: While it is true that a site might serve different code to IE and Chrome.

Dave Herman: Let me talk to Jason Orendorff (SpiderMonkey engineer) and see what he thinks.

Adam Klein: At the moment you cannot use let in sloppy mode.

Brian Terlson: It is worrying that Google thinks that this is not web compatible and that it has not been brought up before.

Brian Terlson: We could change the semantics and make const in sloppy mode scoped like vars.

Allen Wirfs-Brock: I would be opposed to that. I'd rather drop it.

Dave Herman: Opera used to treat const exactly like var.

Waldemar Horwat: What is the issue?

Erik Arvidsson: In legacy Chrome and Firefox (sloppy mode) const has the same scoping rules as var.

Allen Wirfs-Brock: If we run into a brick wall we can revise the spec in the future.

Brian Terlson: Time is of the essence. The longer you hold out the higher the risk is that IE would fall back to the legacy behavior.

Brian Terlson: IE11 shipped the correct ES6 semantics for over a year and only encountered one problem report.

Brian Terlson: Want to see plan to turn on proper sloppy mode semantics from Chrome and Firefox within the next few months.

Erik Arvidsson: We'll make a plan on how to get to standards compliant const in sloppy mode.

Conclusion/resolution

- Google and Mozilla will move towards a standards compliant const in sloppy mode.

Object.observe

(Erik Arvidsson)

(Presents slides) docs.google.com/presentation/d/1VHdL37sZSLBZuizzSfiiSd9zSy9zNYXiNEjDJeTjOSA/edit?usp=sharing

Allen Wirfs-Brock: Is "types" the right word in "acceptTypes"? These are actions, not types.

Erik Arvidsson: These correspond to the "type" field, but agree that, if we're adding types to the language, also using the term "type" in this way would be not so good.

[...discussion of skipRecords: what should be passed to the observer callback in that case? undefined or null?...] (Domenic to file a bug)

Yehuda Katz: Advance to stage 3?

Erik Arvidsson: Not just yet.

Yehuda Katz: I would decouple Array.observe from Object.observe

Yehuda Katz: I would like to get more implementation feedback (which would help prioritize it in libraries), but don't necessarily think that should block Stage 3 forever.

Erik Arvidsson: Still details to be worked out over scheduling of observer callbacks

Domenic Denicola: ECMAScript-wise, just creating a job and enqueuing it should be sufficient

Yehuda Katz: That doesn't seem like enough, because the callbacks still need to run at end-of-microtask

Dave Herman: This sounds like maybe a seperable work-item

Domenic Denicola: Not clear whose court this is in, between Web specs and ES

Yehuda Katz: It seems like a small, self-contained note somewhere explaining the interaction of ES Jobs and HTML/DOM event queues

Dave Herman: It sounds like a WebIDL thing

Allen Wirfs-Brock: 2 things: 1) Relative ordering of observe notification jobs.

Arv: The ordering is well-defined.

Allen Wirfs-Brock: 2) Any ordering dependencies between your spec and promises. In ES6, promises have a well-defined ordering but other job queues can interleave between promises. So if you expect to have a defined ordering between observe and promise then you probably need to use the same queue.

Yehuda Katz: Needs a user-facing API for managing job queues.

Allen Wirfs-Brock: Yes, quite useful.

Mark Miller: Why is Array.observe a separate API? Shouldn't the default be to observe splices when arrays are Object.observed?

Allen Wirfs-Brock: Then you have to define what the extensibility story for other types.

Mark Miller: Understood.

Discussion of decoupling Object.observe and Array.observe

Mark Miller: You wouldn't want to ship Object.observe without Array.observe, though having two waves of specs is a separate issue

Conclusions

- null is preferred over undefined, more explicitly representing an absence of records
- No stage 3

Brian Terlson: I have no idea what is going on, for the record.

ecmarkup

(Brian Terlson)

Brian Terlson: bterlson/ecmarkup

Allen Wirfs-Brock: what is the difference between ecmarkup and ecmarkdown?

Brian Terlson: Ecmarkup is HTML markup, such as `<emu-clause>`, `<emu-alg>` etc

Brian Terlson: Ecmarkdown is a markdown syntax for specifying algorithms steps. Ecmarkup uses ecmarkdown inside `<emu-alg>` and a few other elements.

Brian Terlson: bterlson.github.io/ecmarkup

Brian Terlson: abstract operations from external specifications may be referenced

Erik Arvidsson: There currently is no way to write BNF. We should add that.

Waldemar Horwat: I have built tools for validating grammars

Brian Terlson: I believe Allen would like to continue using Word

Dave Herman: No, we want to get away from that

Allen Wirfs-Brock: ISO requires the document to be submitted in Word

Dave Herman: Jason has a tool for converting some spec sources to Word

Dave Herman: it is important to me that we work in markdown on github for the next version of the spec

Dave Herman: we may be able to leverage academics that want to represent semantics of Jaswanth Sreeram in a similar way to the spec

Brian Terlson: html imports are used, so a document may be split up into pieces

Interfacing with the Loader Spec

(David Herman)

[Link to slides]

Allen Wirfs-Brock: How do you get a module record from a different realm unless you are using imperative APIs?

Dave Herman: I expect this to be pretty nitty gritty code. It is not going to be common to get cross realm modules.

Dave Herman: The basic idea is that every loader belongs to a realm and that when it process modules it uses that realms policies.

Allen Wirfs-Brock: Why can't the records have all the fields but the fields are not filled in yet?

Dave Herman: Some of these fields make no sense for reflective modules.

Dave Herman: Reflective/dynamic modules are always leaves. Makes things a