Minutes of the: ***56th meeting of Ecma TC39***

in: ***San Jose, CA, USA***

on: ***24 – 26 Jan. 2017***

1 Opening, welcome and roll call

1.1 Opening of the meeting (Mr. Wirfs-Brock)

Due to the absence of **Mr. Neumann (TC39 Chair) Mr. A. Wirfs-Brock** has welcomed the delegates at PayPal in San Jose, CA, USA. He has replaced **Mr. Neumann** at this meeting.

Companies / organizations in attendance:

Apple, Mozilla, GoDaddy, Google, Microsoft, JS Foundation, Facebook, Netflix, PayPal, Shape Security, Airbnb, IBM, Salesforce, Bocoup, Yahoo!, Igalia (Guest), Brave (Guest)

1.2 Introduction of attendees

1	Michael Saboff	Apple	Member
2	Istvan Sebestyen	Ecma	Secretariat (part-time, by phone)
3	Allen Wirfs-Brock	Ecma	Secretariat
4	Brian Terlson	Microsoft	Member
5	Bradley Farias	GoDaddy	Member
6	Jeff Morrison	Facebook	Member
7	Brendan Eich	Brave	Invited Expert
8	James Snell	IBM	Member
9	Mark Miller	Google	Member
10	Jafar Husain	Netflix	Member
11	Jordan Harband	Airbnb	Member
12	Chip Morningstar		Invited Expert
13	Daniel Ehrenberg	Igalia	Invited Expert
14	Michael Ficarra	Shape Security	Member
15	Keith Miller	Apple	Member
16	Kent C Dodds	PayPal	Member
17	Dave Herman	Mozilla	Member
18	Chris Hyle	Apple	Member
19	Waldemar Horwat	Google	Member
20	Shu-yu Guo	Mozilla	Member
21	If Bastien	Apple	Member
22	Jamund Ferguson	PayPay	Member

23	Maggie Pult	JS Foundation	Member
24	Gabriel Isenberg	GoDaddy	Member
25	Kevin Gibbons	Shape Security	Member
26	Franziska Hinkelmann	Google	Member
27	John Lenz	Google	Member
28	Diego Ferraro Val	Salesforce	Member
29	Myles Borins	Google	Member
30	Bert Belder	IBM	Member
31	Tim Disney	Shape Security	Member
32	James Kyle	Facebook	Member
33	Caridy Patino	Salesforce	Member
34	Juan Dopazo	Yahoo!	Member
35	Sebastian Markbage	Facebook	Member
36	Adam Klein	Google	Member
37	Tyler Kellen	Boucoup	Member

1.3 Host facilities, local logistics

On behalf of PayPal **Kent C Dodds** welcomed the delegates and explained the logistics.

2 Adoption of the agenda (2017/001-Rev1)

Ecma/TC39/2017/001 Agenda for the 56th meeting of TC39, San Jose, January 2017 (Rev. 1) was posted in the TC39 documentation.

The final agenda was approved as posted on the GitHub as reprinted below:

Agenda for the 56th meeting of Ecma TC39

Agenda Items

1. Opening, welcome and roll call
    i. Opening of the meeting
    ii. Introduction of attendees
    iii. Host facilities, local logistics
2. Find volunteers for note taking
3. Adoption of the agenda
4. Approval of the minutes from last meeting (see draft, empty agenda outline will be repaired before publication)
5. Report from the Ecma Secretariat (15m)
6. Other administrative matters
    i. Confirm logistics for March meeting
    ii. Who should be the chair?
    iii. Code of Conduct (proposal): seeking consensus for adoption, format and enforcement. (Leo Balter, Tyler Kellen, Daniel Ehrenberg)

7. ECMA262 Status Updates (15m)~~
8. ECMA402 Status Updates (presentation)(15m)
9. ECMA404 Status Updates (15m, Chip Morningstar)
10. ECMA414 Status Updates (5m, Istvan Sebestyén)
11. Test262 Status Updates (15m, Brian Terlson)
12. Timeboxed overflow from previous meeting
    i. 15 Minute Items
    ii. 30 Minute Items
    iii. 45 Minute Items
    iv. 60 Minute Items
    v. Timebox Not Yet Selected (champion, please select one)
13. Timeboxed agenda items
    i. 15 Minute Items
        a. Seeking Stage 3 reviewers for RegExp lookbehind and RegExp Unicode properties (Daniel Ehrenberg)
        b. Discussion of some oddities around classes extending `null` (slides) (Adam Klein)
        c. Seeking stage-1 for Null Propagation Operator (Gabriel Isenberg)
        d. Seeking stage `-0.0` for IEEE-754 sign bit (presentation) (JF Bastien, only attending Wednesday / Thursday).
        e. Discussion of Annex B.3.3.3 EvalDeclarationInstantiation overwriting bindings (Shu-yu Guo)
        f. Seeking Stage 4 for SharedArrayBuffer (spec PR and test262 PR) (Shu-yu Guo)
    ii. 30 Minute Items
        a. Proposed Grammar change to ES Modules (Bradley Farias)
        b. Function.prototype.toString updates (Michael Ficarra)
        c. Seeking Stage 2 for named capture groups (Daniel Ehrenberg) (slides)
        d. Seeking Stage-1 for Realms (Dave Herman, Mark Miller, Caridy Patiño)
        e. Seeking Stage-1 for "do expressions" (Dave Herman)
        f. Promise.prototype.finally to stage 3? (Jordan Harband)
    iii. 45 Minute Items
    iv. 60 Minute Items
14. Non-timeboxed overflow from previous meeting
15. Non-timeboxed agenda items
    i. Web compatibility issues / Needs-consensus PRs (Brian Terlson)
        a. Make [[Delete]] on module namespace return false for @@toStringTag (pr)
        b. Normative: do not call super constructor when ThisBindingStatus is already initialized (pr)
    ii. Stage 0+ proposals looking to advance
    iii. New proposals
        a. Error stacks (seeking stage 1) (Jordan Harband, Mark Miller)~~
    iv. Discussion and updates for Stage 0+ Proposals
        a. Progress report and request for comments on 64-bit int support – links to come (Brendan Eich)

   b. Update on web compatibility of `global` (Jordan Harband)
v. Discuss outstanding cross-cutting issues on public and private class fields proposals (Daniel Ehrenberg, Jeff Morrison from remote) (slides)

16. Overflow from timeboxed discussion items (in insertion order)
17. Closure

Agenda Topic Rules

1. Proposals looking to advance must be added to the agenda along with necessary review materials 7 days prior to the first day of the meeting.
2. Timeboxed topics may be 15, 30, 45, or 60 minutes in length.

Schedule constraints

Dates and locations of future meetings

Dates	Location	Host
2017-03-21 to 2017-03-23	Portland, OR	Mozilla
2017-05-23 to 2017-05-25	New York, NY	Google
2017-07-25 to 2017-07-27	Redmond, WA	Microsoft
2017-09-26 to 2017-09-28	Boston, MA	Bocoup
2017-11-28 to 2017-11-30	San Francisco, CA	Airbnb

Documents discussed:

Ecma/TC39/2016/046 Agenda for the 55th meeting of TC39, Menlo Park, November 2016 (Rev. 1)

Ecma/TC39/2016/047 Responses to the Ecma Contribution License Agreement (CLA), 1 December 2016

Ecma/TC39/2016/048 GitHub archives, 1 December 2016

Ecma/TC39/2016/049 Voting results regarding ISO/IEC DIS 21778 - the JSON data interchange format

Ecma/TC39/2016/050 Fast-track explanatory report for ECMAScript Specification Suite 2nd Edition (ECMA-414) submission

Ecma/TC39/2016/051 Venue for the 56th meeting of TC39, San Jose, January 2017

Ecma/TC39/2016/052 Minutes of the 55th meeting of TC39, Menlo Park, November 2016 (Rev. 1)

Ecma/TC39/2016/053 TC39 RF TG form signed by Bocoup

Ecma/TC39/2017/001 Agenda for the 56th meeting of TC39, San Jose, January 2017 (Rev. 1)

Ecma/TC39/2017/002 Responses to the Ecma Contribution License Agreement (CLA), 24 January 2017

3 Approval of minutes from November 2016 (2016/052-Rev1)

Ecma/TC39/2016/052-Rev1 Minutes of the 55th meeting of TC39, Menlo Park, November 2016 (Rev. 1) were approved without modification.

4 Status of "ES6 Suite" submission for fast-track to ISO/IEC JTC 1 and IT issues

On behalf of the Ecma Secretariat **Mr. Sebestyen (only part-time, on the phone)** and **Mr. Wirfs-Brock** attended the TC39 meeting. He explained:

4.1 ISO/IEC JTC 1 fast-track

ECMAScript Suite (ECMA-414 2nd Edition) has been fast-tracked to JTC 1 as ISO/IEC DIS 22275. There it has been registered but the DIS ballot has not started yet. The latest information is that they will start it soon.

As described in the explanatory report that Ecma has submitted to the JTC 1 Secretaraiat, ECMA-414 2nd edition should become a new IS with a new number and the current Standard ISO/IEC 16262 (equivalent with ECMA-262 Edition 5.1) should be withdrawn.

ECMA-404 (JSON) has also been submitted to ISO for fast-track (the 2013 edition). The DIS ballot has finished in December 2016. It is approved, but because of one negative vote from Japan after resolution of comments an FDIS ballot will be needed. Regarding the fast-track process of ISO/IEC DIS 21778 (ECMA-404) "The JSON data interchange format", the TC39 JSON standard Editors should prepare an answer to the Japanese comments. This was discussed in the TC39 meeting. There is still the plan when there will be a similar IETF Standard (not FRC) we will issue a new ECMA-404 Edition when we will take up that and get it synchronized with ECMA-404.

4.2 ECMA-262/402 2016 editions status

Nothing new on that.

4.3 TC39 IT issues

Ecma/TC39/2017/004 GitHub archives, 14 February 2017

Since the technical work of TC39 currently is carried out on GitHub the Ecma Secretariat's policy is to make in regular periods a complete snap-shot of the TC39 GitHub activities and storages. The above is the 4th such snap-shot. The plan is at least to make such snap-shots before each TC39 meeting.

5 ES7, ES8 and Test262 discussions

See the Technical Notes in Annex 1.

6 Observation from the Secretariat

TC39 has been growing during the last 2 years to a size where the old working style meets its limits. Therefore we are looking at our working procedures and practice very carefully what could be improved due to the changed situation:

In the TC39 meeting we reviewed again the issue of the vacant TC39 chair and Vice Chair chairman position. The current mode of operation with **Allen Wirfs-Brock** works fine, but unfortunately that is only a short term solution lasting for one or two more TC39 meetings. Therefore this issue has to be solved soon. Reflection of those discussions it can be found in the technical notes attached. The current solution with multiple note-takers looks promising. See Technical Notes of the meeting.

Generally it has been decided to carry out such discussions on the GitHub between the meetings, in order not to take away too much time in the face-to-face meetings from the technical discussions.

7 Date and place of the next meetings

See above under 2.

8 Any other business

None

9 Closure

Mr. Wirfs-Brock thanked the TC39 meeting participants for their hard work. TC39 thanked **Mr. Wirfs-Brock** for chairing this TC39 meeting.

Many thanks to the host, **PayPal** for the organization of the meeting and the excellent meeting facilities. Many thanks in particular to the host and Ecma International for the social event.

Annex 1

January 24, 2017 Meeting Notes

Allen Wirfs-Brock (AWB), Waldemar Horwat (WH), Jordan Harband (JHD), Brian Terlson (BT), Michael Ficarra (MF), Adam Klein (AK), Chip Morningstar (CM), Dave Herman (DH), Kent C. Dodds (KCD), Kevin Gibbons (KG), Tim Disney (TD), Daniel Ehrenberg (DE), Shu-yu Guo (SYG), Michael Saboff (MS), James Kyle (JK), Franziska Hinkelmann (FH), Anna Henningen (AH), John Lenz (JL), Sebastian Markbage (SM), Bradley Farias (BF), Jeff Morrison (JM), Tyler Kellen (TK), Gabriel Isenberg (GI), James Snell (JSL), Maggie Pint (MPT), Chris Hyle (CH), Gabriel Isenberg (GI), Bert Belder (BB), Zibi Braniecki (ZB), Jamund Ferguson (JXF), Mathias Bynens (MB), Leo Balter (LB)

Adopting the agenda

- Start with Istvan's items to get them out of the way
- Jeff Morrison more free on Tuesday, preferably
AK: This may take less than three days.

Adopted the agenda

Adopting the minutes

WH: Technical issues prevented us from approving the minutes: ECMA's web site is down. Their Pydio substitute has an invalid SSL certificate. If you ignore the errors and try to access it anyway, it runs scripts that time out.

IS: No one has reported that back to us directly... We have also another substitute which is not yet announced. Allen has I think access codes. Maybe that one works better. Please try it, but not everybody at the same time ;-). It is only a substitute.

JHD: Why not make the notes the official minutes?

AWB: The notes are appended to the minutes, but there is more in the minutes.

IS: Correct.

JHD: Do those on our own without relying on ECMA?

IS: To my understanding the Technical Notes of a Meeting are for you, and one can access it via this address.

WH: ECMA act as librarians for a long-term record. Our many past attempts have not lasted more than a few years; see, for example, what happened to our ecmascript.org.

IS: Exactly. We have a long-term archival obligation. We are trying to archive the most important documents of the standardization process and this in a way that is accessible and understandable to every ECMA member. Many of them simply do not know the present and past TC39 tools

AWB: There is room to work with ECMA to improve the infrastructure for taking notes--if someone wants to do that, everyone would probably appreciate it.

I.S.: We could upload the ECMA minutes on the GitHub to the TC39 Reflector part, would that help?

WH: I was able to eventually somehow get the minutes.

AWB: We'll send this around to get token votes; we can make a PDF of it

Side Discussion

WH: What's happening with es-discuss? I'm getting bounces.

AWB: The list administrators of this are Brendan Eich and myself. If someone else wants to be an administrator on that list, I am sure Brendan and I would be happy about that. There isn't a lot of work but occasionally something does come in.

I.S.: I think this is an example for another TC39 tool, that was popular for a few years, and maybe now is slowly disappearing. It is a real challange for us, how to archive all this for long-term.

ECMA262 Status Updates--countdown to ES2017

(Brian Terlson)

BT: The key dates for our schedule are, at the end of this month, on the 31st, we will stop taking any normative updates besides bugfixes. we can fix bugs but we can't add features. Correct me if I'm wrong but the only proposal I'm aware of that is pushing for stage 4 is shared array buffers. Are there other proposals if I need to be tracking?

DE: There may be some for 402

AK: Do you mean big proposals, or do bugs count?

BT: Any needs-consensus PRs after January 31st becomes a part of ES2018 (if at all).
BT: For those new to the committee, we're not a technical committee, we're a royalty free task group (of the technical committee). All that means is that there is a royalty free commitment to the content we create in the specification. If you would like to patent something that is in ECMA or if you think you have a patent that is pertinent to something we are standardizing in this version. You should talk to your lawyers and potentially opt out of the royalty free agreement. you'd be the first person to do this. We've never had an opt out. Talking to Istvan, I'm not sure we even know what happens.

I.S.: Brian is right. In the current TC39 RFTG everything is expected to be RF. At an "opt out" - which we did not have so far - we have to see how that part of the policy plays out. But it will be difficult because a) who decides and how if the claim for the opt out is valid? b) If valid then either an alternative method has to be found or the RF project cancelled... So, if TC39 wants also "RAND" components in the standard, maybe for some special case, options that must be done in a new, different TG working with a RAND based policy.

WH: If someone opts out (or makes an unexpected patent claim), which happened in other TCs, then we have a standard that someone claims a patent on, then chances are we'd end up voting no on the proposal. We approve the proposal pending it not having an opt-out.

AWB: If anyone does opt out, we are unlikely to approve the specification. Starting in April is when the ECMA governance folks start pushing our work through their pipeline (the executive committee).

BT: In March, the ECMA secretariat will start work on approving the document.

BT: In ES2017 is:
- Async functions
- Trailing function commas
- Object.values/Object.entries
- Object.getOwnPropertyDescriptors
- String.prototype.padStart/padEnd
- new.target and eval reform
- Agents, SharedArrayBuffer and Atomics
- Latest Unicode version--affects all parsers

BT: Several bug fixes and minor changes. Implementors: if you do `git log` on the spec, you will see all

normative changes with descriptions starting with "Normative:"

AWB: Do we still have Annex E listing incompatibilities? Seems like all observable changes in. The document is the real source of truth.

BT: Yes, but I'm not putting every little thing in there; there are tons of minor things. You can look at GitHub, which is sort of a source of truth, even if this doesn't quite align.

BT(?): There is a big one on this page that will definitely make it into the annex which is the alignment of typedarray, arraybuffers and dataview. I think we have implemented all of this so I'm not worried about web compat for this change, so that's nice. Huge number of editorially and front end changes this year. The spec is much more full of links where you can find improved auto-links for cross-references. Tons of rendering improvements and bug fixes. ES2016 was kind of buggy in the rendering of it. A lot of that stuff got fixed. Some quality of life stuff, 100s of editorial updates and improvements. The community, I think we had this year 7 people who contributed for the first time to the spec. A lot of it was not technical, but like, hey, I found this confusing when I was reading it, here, this is better. That is so awesome. TIs' great to have people who are not immersed in this making it better. It's really showing how github is a major win

MPT: Question: this is my first time here. Must the spec be an HTML document?

WH: Actually no, the spec is not an HTML Document. ECMA's requirements used to be that the specs had to be a Word document. Now they also allow a PDF. The HTML is just a courtesy we provide.

I.S.: TC39 is special in that area. Actually we are offering TC39 standards in HTML format too, because we see that there is a high demand for it. This is a special case, the GA is discussing if we should generally move into that direction.

BT: Actually, as far as ECMA is concerned, we went through this last year; They are okay as treating the HTML document as the normative document. They updated the wording on the website for this. What you say is true for ISO, they have a spec for the spec process. They require a PDF.

WH: Since our specs are now automatically ISO specs, that means that we still need to use PDF for the normative format.

I.S. Actually if the Fast-Track goes through then ECMA-262, ECMA-402 will only be Ecma standards. Only ECMA-404 (JSON) and ECMA-414 (ECMAScript Suite) will be also ISO standards (with appropriate ISO numbers). Also the Test Suite remains Ecma only product.

BT: That means that ISO gets a lower fidelity document

MPT: Where this comes from is a day to day life thing. I work on moment. I get people 100 times of day, when I do this piece time ... why is it that way? I spent a lot of time digging in the spec to find that stuff.

BT: You're saying the rendering of it would be slick

MPT: The rendering is slow, following the history of spec changes on date (which there have been quite a few) is non trivial. It's a practicality thing. If you broke that into multiple documents, that would help
From a practical standpoint this comes up at least once a month, I'm fighting around this huge HTML Document looking for changes.

WH: It's annoying that GitHub doesn't know how to show differences in specs over 10,000 lines long, which our spec is.

BT: I've heard that point of feedback before. I've gotten a lot of push back on making a multi-page document. Lots of folks use ctrl+f for no good reason

BT: There are trade-offs.. what are the gaps between pages? If each clause is its own page, some sections

would be a quarter of the spec, others would be a paragraph. It's a hard problem. please contribute something! It will get used. Also, we do have something that unifies the multipage sec already and it's... search.

DE: The WHATWG spec has single and multipage versions.

BT: It's just a matter of finding a good enough way of displaying the spec. On the editing side, originally my proposal for the ecmarkup source document was split across numerous files. I got push back on that basically say I need to be able to ctrl+f in my editor, which is an important thing. If we want to change that too, I still think multi-input documents would be handy. I'm pretty comfortable using grep, though.

AWB: These are future development things. They will not make it into this year's edition.

BT: The number of normalizations we've applied to the document in terms of the syntax of algos, the placement of commas, inline forms like when you say, if blah, then do x on one line, format it this way or that. It's actually very rigorous now. We're approaching machine-readability for algos, with some minor exceptions. So, my request for people out there is start thinking about what you want for tooling. We could for example consider syntax highlighting that doesn't require marking up your variables with underscores, or, um, maybe some kind of intelligence trace to help implementors walk through the spec step by step with a sidebar with a list of algorithms.

DH: That would be amazing for anyone who is trying to understand a deeply nested series of methods.

BT: With machine readability comes a lot of options. Think about what you want. In terms of markup, we've improved performance by like 3 or 4x. The search relevancy should be good now. "." isn't considered a word boundary though. Since the last meeting I implemented find all references. Anything in the spec with an ID, you can find anything that links inbound to that ID;

MF: There are certain things which are not aggregated, "early error", for example. That is probably solvable by aggregating the usages of those kinds of terms as list or some function.

BT: That's possible. The other big one is the polymorphic abstract operations on environment records and the essential methods of objects, where we don't have an algo id for it because there is by definition like 5 of them.

WHB: If you thought about how you would do a find all implementations of an abstract algorithm?

BT: I can handle that in some way. If you click a reference to delete, for the ability for the editor to say, I know you mean this one.

WHB: It would be nice to get a list of those so you could see them all.

BT: You can do it with search but I don't know the right way to do it yet. I had a proposal pulling out... I won't get into that. So anyway, for spec editors, note there is a new type of note, the editors note. You should use this when you are making notes to the committee when you don't expect the editor to include that in the final spec. Easy way to add some stuff that will be removed by the time the proposal is accepted. You don't need CSS in JS files. Please stop hotlinking the ECMA262 css file. I made changes and external stuff changed.

WH: Can you clarify your last point?

BT: Previously ecmarkup would dump a css and js file next to your built spec and you would have to include that. Because that's a PITA, what people did was not generate that CSS or JS and just link to the github version. Problem occurs when I update the ecmarkup verison for 262, that makes a new CSS file that breaks the old markup. The new version of ecmarkup creates an inline script and style tag. If you do that, everything will just work.

BT: That's everything for 262 tooling this year.

AWB: Any decisions we need to make?

BT: Yes, we have a decision to make about atomics and array buffer and some needs consensus PRs I'd like to discuss but not now. I'd like to do some socialization first. Minor stuff. Not on the critical path, to put it that way.

[Needs Consensus PRs](https://github.com/tc39/ecma262/labels/needs%20consensus)

Discussing [tweak to module namespace objects](https://github.com/tc39/ecma262/pull/767)

BT: I just want to talk to the people who know about this to see if it is still what they want. Basically, delete is lying. Basically, we just delegate to the ordinary delete when you're deleting symbols.

DH: Didn't we already talk about this?

AK: Yes, this is a tweak and it turns out we didn't fix it entirely

BT: This is the situation

AWB: So, delete for a nonconfigurable property normally returns false, is that right?

BT: Right

AWS: This makes it align with an ordinary object. makes it look like a nonconfigurable property

AK: Yeah, that's what I missed the last time

BT: Anyone disagree with this change?

Discussing [not calling super multiple times](https://github.com/tc39/ecma262/pull/762)

BT: I can see how people would be nervous to make this change. The problem is that today, we actually print 'a' twice in this case.

??: This is nasty

??: My concern about this is that it seems like a patch over a small... tries to reactivity something about initialization but there are all kinds of other cases that can come up.

BT: Let me be clear here, the second super throws an error, always. But it does it AFTER calling the super constructor the second time

AWB: That was intentional. To do it the other way requires redundant tests, it requires a test both before and after. This only requires a test after. Since you're going to get an error in either case, I went with the computationally simpler algo

JHD: Is invoking user written code less expensive?

AK: Yes, and potentially the generated code. We don't care how fast the error case is.

DH: 99% of code tha thas this bug... they are doing extra test at runtime all the time

AK: Right, if they have one super call, they have to check after

DE: You could inline it but often you haven't inlined the entire class hierarchy

AWB: So an error is going to occur, why do you care if there is a side effect?

BT: I"ll tell you why. you will be very confused when you see this happen. I got an email from an internal team who was very confused. It seemed like super was both succeeding and failing at the same time. They had logging in the super class that was confusing.

AWB: Was it confusing because the error message wasn't precise enough?

BT: No, the error message was precise, it said you can't call super or something. Clearly super was working though, they were seeing the logging;
After a second super call, an error will always be thrown, it'll be called, but it will error.

AWB: Maybe a better error message would help

JHD: The message would have to ex

WH: What happens if user code in the second super call throws?

AWB: In the current specification, if you captured this object, and then you do a second super call, but you throw and catch it, before the binding happens, then you can precede as normal, without it disrupting construction.

JHD: If you're doing that with a try/catch the error wouldn't allow the second super code path, right? The only impact that Brian's change is on the spec algo and the fact that there is a second observable.

AWB: The real point here is that you can do all sorts of crazy stuff. The primary thing we're trying to make sure here is that implementations always do the same thing when you do this crazy stuff. The actual crazy stuff, what it is, who cares, it's crazy.

JHD: Is that any reason to not minimize it?

BF: My concern with removing the second call, if you can catch the error already, I'd be surprised, just personally if the function didn't run but it generated an error. In the other case if I call super the first time and it DOES throw I'm presuming the second one would throw too because it's been set to initialize.

AWB: That seems to be in this space of crazy stuff can happen depending on how they structure their code. We just want consistency.

BF: Do you know what happens if you throw on the first super but not the second?

AWB: The second one would run and you would get a bound super and it would be fine

AK: That's with this change

??: But then we have called super twice

??: Super might not complete successfully more than once

AWB: There is no constraint (with or without this proposal) on calling super more than once, the constraints is on binding super

KD: So, i just have a question, maybe a different discussion. Why do we allow this at all?

BT: You can't know statically if it's allowed, consider if/else/etc

MPT: It's very unsafe because that super call is not idempotent, if you're doing it twice...

AWB: as long as implementations follow what the spec does, we don't care that it does crazy stuff, as long as it's the same cray stuff

DH :Clarifying question. I missed why Brian's proposed semantics require extra tests. I took it on faith but didn't grok it.

AK: The title is pretty clear, you have to check before you call to see if it initialized.

DH: Why do you have to check again?

AK It's not that you're checking super is not called multiple times, you're checking that you initialized the object by the end

DE: There is a check for the super call at the beginning and end. The super call can escape and you can use it in an arrow function if by the time super returns has it been bound to something else?

AK: the super call itself has a check in it, when you assign it to this it checks to ensure it hasn't already been iniitalized

DE: no way to eliminate that second check

DH: Two different kinds of check right?

AK: No, it's the same check: you check the receiver slot before the super() call, and then after the super call when actually doing the binding

DH Oh, so the idea is that you make sure this isn't in a TDZ

Hazard case: https://gist.github.com/bakkot/66b515f402b8cdbf3e405f8509ad3525

WH: Normally people don't call super from an arrow function. It's allowed, so you should be able to statically eliminate the second check if you statically see that super is not ever used from an arrow function

BT: There is no known case outside of the arrow function

WH: I am responding to the argument that it would be inefficient to check twice. Most of the time people do not call super inside an arrow function. If they don't do that, it's efficient for the implementation to check just once before the super call.

BT: That's true

WH: If they happen to call within an arrow function they lose an insignificant bit of performance

AWB: The analysis of th etwi cases is more expensive.

BT: It's not expensive for us to check this.

WH: I don't like the user surprises that invoking super the second time produces.

BT: We have real world evidence that people are confused. I think we should fix it.

AWB: I think it is adding overhead to the 99.9999% case

WH: No it isn't. It's only adding a second check to the super-in-arrow case.

AWB: No, it still is adding overhead in the analysis

DH: I would be skeptical about adding analysis

BT: We're not suggesting that, Waldemar's point is you can avoid the static check if you can know super isn't in an arrow function

DE: It wouldn't work for code that uses eval but that's probably not going to come up

BT: My feeling is that this check is so fast that any analysis is going to drastically outweighj ust doing it

AK: FWIW implementation in ES6, a lot of checks are required. it's slower than ES5 in the baseline implementation. If your'e accessiing a variable in a loop it's noticeably slow. I'd rather not add checks where we don't need to.

DH: I think i lead to Adam's and Allen's side here.

BT: Why don't we get rid of the check entirely? Just let super run twice.

AWB: I would be happy with that but part of my grand compromise on how we figured this out in the last two weeks of ES2015, included this error.

BT: Okay, so we have rpecende tfor doing checks in the name of, i guess, usability, or actually, i don't know

AWB: Yeah, it's a usabiilyt argument

BT: Very suprising how the second super is called

AWB: the suprrise partially comes from, probably maybe not full comprehension for this within constructors, that you don't have a this binding before youc all super. if htey understood that, they would understand why the second super was an error.

BT: How would you feel about making the second super jset work?

DH: Feels radical, feels like it needs more than "sure why not?"

WH: Wouldn't work once we implement private fields.

DE: Eliminating the check doesn't add any basic incompatibilites with private fields here, adding a private field twice will fail, so even if we eliminate this check, it'll throw an exception later

AK: My worries, my implementation feedback. I haven't looked to see what a static check would be. If it is easy, I don't care about this, no concerns for this proposal. I have a guess there are some edge cases that could be hard to detect. This is a case where my moderate opposition might go away.

DH: I still worry, even in that case, 70% of code doesn't ever put super under a lambda and can have the checks optimized away. Maybe it impacts your performance.

AK: The super call can't be in a tight loop

DH: The single super call has the additional check, if you're instantiation stuff in a tight loop it could come up. Maybe it's not likely to every be noticeable.

BT: I'd argue the performance would not be a noticeable performance difference

AK: Have you not found the checks for let to be noticeable compared to var?

BT: In some cases, yes, but that will create more checks

AWB: If the primary issue is user confusion, give them a paragraph of text

DH: I'm not sure that's accurate. It doesn't matter what the error message said, It's that they got the error after the super ran and din't know ecma backwards and forwards. Nobody should be expected to memorize these semantics. Their experience was the constructor ran, but i got an error saying it couldn't run. It's one of those up is down kind of experiences where you question yoru grasp of basic logic. It doesn't matter what the error messages says, the confusion you draw is that you are both running and not running this

JL: I think you could word it properly, this rebinding attempt, this or that

DH: That's true, I guess you could say you could make it clear that wafter we ran it a second time we realized we ran it a second time

DE: We've been discussing this topic for awhile, we didn't atlk about timeboxes, it seems this is a little contentious

AWB: Maybe, the way I would approach everything for the 2017 edition, the timebox is the meeting itself.

MPT: This code shouldn't exist. We shouldn't silently rebind this. The language isn't helping people when we allow code that shouldn't exist and failing silently instead of failing loudly.

DH: That's not really an option.

MPT: At the end of the day it doesn't really matte,r i hear you, does it really matter, the code is broken either way. You could probably just do the single later error check

BT: That slows down the non error case, potentially

MPT: So, all implementations are doing the same thing. We agree it's bad source. What's wrong with this?

BT: What's wrong is that it is confusing to people.

DH: I don't agree with the philosophy if there is a bug in the code it's the programmer's fault and we shouldn't help them. I think we have a role in the design and implementation of the language in helping them be more productive in their process. If there are things we are doing that will make it subtly confusing that's bad. In this case, there is a global cost calculus, how much of a cost is this confusion, vs what is the cost to everyone else in additoial checks, more complicated performance model It's conceivable a better error message could help. With chakra where this happened, "multiple calls to super are not allowed" that leads you to the conclusion that something can't happen that just happened. Now your grip on reality is sliptting. You should say just multiple calls to super have occurred. Chakra/Spidermonkey and JSC have reasonable error messages.

(_TODO: Paste in error messages_)

AK: To be fair, at trunk, V8 has a better error

AWB: I think spidermonkey is the right error. Super was called twice and that's an error.

JHD: You can call it twice. You just can't call it successfully twice and bind this. Are you going to put that into an error message and say it's going to be less confusing? If one of them throws you can call it as many times as you wan't, you can only call it successfully once.

SYG: I don't think the performance cost here is too bad.

MS: I haven't looked at the implementation.

BT: Do you think there is a problem here?

MS: For me it's a cost benefit analysis. There is performance implications to make the check earlier, I don't care that it got called twice. You did something wrong, we're not likely to see this in real life. I disagree with Allen, at least three of those errors seem reasonable.

AK: I fixed this. I can't remember what ours says. It used to use the same error as let in a TDZ . It was bad. I agree this shouldn't be timeboxed. We've put a lot of time here, can we push this somehwere?

BT: Let me ask you this. If you investigate and find that the perf implications are neglible, which I suspect will happen, would you support this change or would you still be against it?

AK: I would actually be OK with it was easy to see. I feel like it might be easy for us to detect. I'm not as worried about someone moving that super call into an error function I don't think that's a thing people do unless they are intentionally messing with the interpreter.

SYG: Independently, we already have some checks scheduled around super calls

AK: We check in the parser for methods whether they contain super property access and need the home object, but we don't do the similar parser analysis for super construct calls

WH: There are worse surprises you can get. For example, if the super call counts how many instances you've made, you'll bump the count by two while creating only one instance. I'm strongly in favor of doing the check before.

DH: Just to be clear if we do the check before we do it after also.

WH: Yes. The check after can be optimized out in most practical cases.

BT: I think we should do due diligence though. If you can check in advance of the next meeting or next couple of days.

AK: I can set aside 15 minutes to look into this

MS: This needs to be resolved by March

AK/BT: It's been there for a long time

BT: Interop isn't a big concern here, we're getting an error

AK: Nobody should be wrapping their super calls in try/catches

BT: Most people are depending on side effects of a super call

AWB: I'll leave this to you Brian, but it sounds to me like it should probably be pushed past 2017.

BT: IF Adam looks at this and finds it's not a problem. If it isn't in this meeting it's in next version.

DE: In general are we really aiming to get these in by the cutoff? We have a draft spec.

AWB: The goal right now is what goes into the cutoff. We do have these checkpoints every year. People do pay attention to this in the outside world.

BT: It's important what goes into that document. Everyone in this room only cares about the edge version of the spec but it's not true for everyone. We should just be intentional about this.

BT: Skipping PR 641 for ES2017, are we okay with this?

MS: Yes, I'm fine with that

BT: I have some talking to on 673 so that wrap this.

DH: Would it be possible after this session would it be possible to take a few minutes for face to face scheduling conversation?

BT: We still need to work on a few layering fixes #673 and #688, which we should leave for later

DE: There's also some new timezone changes

AWB: That's a fairly big semantic change; there are historical bug threads that we should look at for clarification.

Conclusion/Resolution

- No change for ES2017
- Awaiting feedback from implementations about the difficulty of implementation of a static conservative check for possible multiple super calls

6.i Confirm logistics for March meeting

DH: We are confirmed for Portland for March 2017? We could put down a deposit for a venue in March 2018, if we can decide on that right now.

BT: As far as I'm aware, i think it's just renaming it. Putting it in annex b. Maybe needs consensus is false on this (PR 673) actually. I'm not sure we actually got consensus on this one last time.

DE: We also got a new pull request to clarify about timezones, but it's kind of new.

BT: I don't want to mess with that right now.

AWB: It's not just a clarification, it's non trivial

BT: It's a normative change. It makes sense but i'm conflicted about how much I want to try to "rescue" date. We might just stop bailing water and get a new API.

DH: First of all, I believe we are confirmed for Portland in March. We were not able to get like plan a through plan q in the venues we wanted. No convention center. Tilde is really good at this and they have come up with a good venue for us. It was eye opening about how fast Portland books up. We've had good experience there in the past. We can put down a deposit for 2018 for a similar setup, and we know we wouldn't lose that venue. But, we'd have to commit to a date to the tune of $500 that i'd lose from my budget. I can do that. Do we think it makes sense? Our schedule is stable, generally in the 20s week of a month. I could commit us to that now.

DE: How will the VC performance be in the Portland venue?

DH: The space we have this time, I don't know a ton about

DE: When we me tin Portland in the convention center last time, we had folks call in and it was hard to hear

DH: You're right the VC was bad

DE: For 2017, we're in a pickle and we can't be choosy, but if we're picking for 2018

DH: I can find more details about the the 2017 venue.

AK: Are the dates on the agenda correct? Is it the 28th-30th (conflicting with EmberConf) or the week earlier (21-23rd), as we later decided to switch to.

DH: Oh dear. I don't know. I will double check with Crystal. That would suck.

AK: There was concern about conflict with Ember Conf.

DH: Given that they were the organizers of the conference, we can likely assume they took that into account?

DE: What were the dates?

DH: If the VC was enough of a concern about the venue we shouldn't put a deposit in?

BT: If they can solve that...

DH: If they can fix that issue, is it worth committing to a date

AWB: If we're going to commit to a date that far into the future, why not a better date? March isn't great in Portland :)

DH: May could be good, July is always Redmond, July is great in the pacific northwest (covered by Redmond). If not March then what, May/September?

AWB: If we're going that route, we can just say January... March is a little better?

DH: Is it okay if I just pick what makes sense?

BT: I've already started on booking catering for 2018

DH: I can't tell if you're joking!

DE: How far in advance do you really need to do this?

DH: Let me find out.

Scheduling aside

BT: Is Istvan calling in? I have urgent ISO plans to discuss

AWB: The plan is a Thursday morning call. Someone suggested we do this tomorrow rather than Thursday because they think maybe we don't need to be here. We should decide that soon. We have the room for three days.

BT: Should we say tomorrow for Istvan's update? I need to communicate that with Brendan.

AWB: Is that so Brendan -won't- have to be here?

BT: I won't claim to know! Dave, you have what you need?

13.i.f Seeking Stage 4 for SharedArrayBuffer

(Shu-Yu Guo)

- [spec PR](https://github.com/tc39/ecma262/pull/769)
- [test262 PR](https://github.com/tc39/test262/pull/839)

SYG: Last time, we were missing a mergeable PR and test262 tests. Now we have both of those. The memory model is now a top-level clause.

BT: I suggest that even people who are not familiar with memory models could read the notes which are specifically targeted to the developer intuition for the memory model.

SYG: Notes: 1. For programmers using JavaScript 2. For implementers trying to decide which transformations are valid 3. For code generation. These are informative, and describe things that are implied by the memory model, but the memory model is arcane.

BT: This is a new thing, including extensive non-normative explanatory text. It would be great to have feedback on whether you like it, don't like it, etc.

SYG: There haven't been any big normative changes since the last time; @@toStringTag, etc. The initial PR tests the coverage of the ECMAScript parts, like that the properties are hooked up correctly, the methods have the right property descriptors, etc. The difficulty is that it requires that test harnesses can spawn agents, which is provided by the embedding. The PR includes a description of what harnesses should provide, but the harness isn't part of the PR.

BT: If test262 tests exist but can only be run on one engine, due to lacking the time to implement the harness for every engine. Does that meet the stage 4 requirement?

SYG: It's also not all functionality; wait and wake need agents, but other atomics can be tested on one thread.

AWB: We don't have any strict requirements for what the Stage 4 tests consist of

DE: In this case, you're working with V8 to make sure the tests are compatible with the harness, so it should be especially not a problem.

LB: I have been reviewing the tests. They seem to have reasonable coverage so far. About the harness support, this is not the first time that the harnesses are expected to provide functionality (also for TypedArray detach). As far as reviewed the PR, this has my +1.

AWB: Any objections to Stage 4?

Conclusion/Resolution

- Stage 4 for SharedArrayBuffer! *applause*

AWB: And that's it for ES2017

Other time boxed items / Looking for stage 3 reviewers

DE: This is very short, I had a few proposals reach stage 2. I have not yet asked for stage 3 reviewers. Would anyone be willing to look?

BT: I'll sign someone at MSFT to review.

MF: I could be a reviewer

WH: I have reviewed these already, would you like me to review again?

AWB: What this really means is that during stage 3, if there are further reviews, these people will take a look

again

JSL: Unicode properties or myself will take a look.

DE: Great, the more the better!

WH: I will look at any changes!

Conclusion/Resolution

Named reviewers for Stage 3
- RegExp lookbehind:
  - Someone from Microsoft (nominated by Brian Terlson)
  - Kevin Gibbons (nominated by Michael Ficarra)
  - Waldemar Horwat
- RegExp Unicode Properties
  - James Snell
  - Someone from Microsoft (nominated by Brian Terlson)
  - Kevin Gibbons (nominated by Michael Ficarra)
  - Waldemar Horwat

Scheduling aside

DH: Just a quick update, I did verify that this march is the week earlier, the 21st-23rd. I've corrected the date. I don't know if there is an incorrect date anywhere else, but if anyone sees this please fix it or let me know.

AWB: Do you know yet where it is?

DH: I can get more details. Obviously we'll fill out the agenda page.

AWB: Just curious!

MF: Did you say you've checked the previous agendas so ensure they don't conflict? I want to make sure nobody gets confused

DH: Will check

13.i.b Discussion of some oddities around classes extending null

(Adam Klein)

- [slides](https://docs.google.com/presentation/d/1makDPBGA3fo-jjOpjhD97_GqlKRQHvh9Mya-B2Spxng/edit#slide=id.p)

Slide: https://docs.google.com/presentation/d/1makDPBGA3fo-jjOpjhD97_GqlKRQHvh9Mya-B2Spxng/edit#slide=id.g1c264b0f9c_0_58

AK: A little background on constructors in ES2015. There are two kinds, Base class and Derived class which has an extends clause. There are certain requirements about super for these things. Base class doesn't make any sense to call super. Basically Base class uses ES5 style allocation and then you call the constructor. In the grammar, super is an early error for base classes. In the derived constructor you have to use super, if you don't call it we'll throw. This is what the state was in the ES2015 spec. you can tell the difference looking at the definition, if there is an extends clause.

Slide: https://docs.google.com/presentation/d/1makDPBGA3fo-jjOpjhD97_GqlKRQHvh9Mya-B2Spxng/edit#slide=id.g1c26b27b69_1_28

AK: There was a recent change to allow extending null to "do something". (Previously, calling the super constructor from a class extending null would throw, rendering the normal constructor pattern unusable.) A class which extends null is a base class, according to ES2017. At class definition time we check to see whether the extends is a null or an object. null means it's a base class. This looked like it might be a derived class. Prior to this change, the presence of extends was that it was always derived. Basically, there was a spec bug that if you extended null it didn't handle derived-ness correctly. It used to be that you detect a derived class syntactically but now you have to confirm if it is null.

Slide: https://docs.google.com/presentation/d/1makDPBGA3fo-jjOpjhD97_GqlKRQHvh9Mya-B2Spxng/edit#slide=id.g1c26b27b69_1_55

AK: This leads to some funny syntactical things. If you have a derived class, you cain always change what something extends from later. The super call is dynamic. The first time you instantiate a thing, you get an object (a subclass of object), mess with it a little bit and then.

AWB: Actually, to change the super construct call change, you want to modify Derived's `[[Prototype]]`, not `Derived.prototype`'s `[[Prototype]]`. Then, you'd observe the behavior changing. (Note: This was to correct a previous version of the slide, which has now already been corrected)

DH: A few points of clarity. For a base class, there is a base.prototype, it is an object that stores the methods for sharing with instances of this class. It's internal prototype slot is null? Base.prototype

AK: Confirmed, base.prototype slot is null

AWB: The difference is, if you say extends object, the prototype internal slot of base, the constructor is object. For class-site inheritance you get the object.

DH: Base.prototype, do we care, do you care, do you know, if, base.prototype is an instance of base?

AWB: No, it isn't. Anything defines as class, prototype objects created by class declarations are

DH: That was just some groundwork I wanted to be sure of.

(Discussion about the details on the slide)

Slide: https://docs.google.com/presentation/d/1makDPBGA3fo-jjOpjhD97_GqlKRQHvh9Mya-B2Spxng/edit#slide=id.g1c26b27b69_1_62

AK: Derived classes which inherit from Object have a mutable super() call, but this isn't the case when inheriting from null, since those are marked as base constructors when they're created, under the new ES2017 semantics.

Slide: https://docs.google.com/presentation/d/1makDPBGA3fo-jjOpjhD97_GqlKRQHvh9Mya-B2Spxng/edit#slide=id.g1c26b27b69_1_43

AK: Under the ES2017 semantics, if you want to make a class factory which inherits from a parameter, then you have to be very careful to not call the super constructor if the base class is (dynamically) null

JHD: And the issues from the two slides may happen at the same time!

DH: I posit that this seems expected, that a mixin function is in the domain where it is thinking about whether it should call super or not

AWB: It seems like base-ness should be driven by the syntax, and it's not

AK: I was trying to implement the ES2017 semantics and I ran into this; that's why I am presenting this.

AWB: Yeah, this is all what led to our ES2015 state.

Slide: https://docs.google.com/presentation/d/1makDPBGA3fo-jjOpjhD97_GqlKRQHvh9Mya-B2Spxng/edit#slide=id.g1c26b27b69_1_79

AK: Proposal is that literal "extends null" is a special syntactic form to get at the base class with null prototype. If you extend an expression that evaluates to null, then it is a derived class.

AWB: Anyway, mutating the prototype in the super call is not a mainline use case; it's largely for consistency with the way that __proto__ has always worked. As a separate program, we may look into adding immutable prototypes.

JHD: This is a pretty core part of the semantics, and we expect everything to be analogous. What if, if there's no extends clause, that could always be dynamically mutated to be a derived class?

WH: That's a hard thing to make work, since if there's no syntactic call to super(), you don't know when exactly to call it.

AK: The syntactic form gives a few advantages:
- Permits all alternatives
- Early errors for calling super() in an "extends null" class

DH: Does this not bother you at all? If you use the literal null vs some other expression that evaluates to null with dramatically different semantics? What about (null)?

BT: I think it was ES6 draft 23 that we discussed this.

MM: if you're going to create a new special form you shouldn't do it by giving special meaning to something that's already an expression

AWB: When we did this in 2015, if you say extends, and you don't have an explicit constructor, it does a super call.

AWB: If you say, class extends null, what it evaluates to, the prototype of the prototype that is created is set to null and the `__proto__` internal slot is set to function.prototype. So the question is then, is function.prototype constructable?

MM: Isn't it right now?

DH: It seems unlikely to me you could change that without breaking the web. Let me check the console.

JHD: What happens if I make a class that extends an expression that evaluates to null, and later change it to extend something non-null?

WH: Can't do that. The super call will fail.

JHD: Conditionally call super.

WH: Still can't do that. A class that's not a base class will fail at the end of the constructor if doesn't call super.

JHD: Things that I instantiate from class foo extends null, if i get the prototype, i get null.

JHD: What if I want to dynamically determine whether the superclass is null or not?

AWB: If you want this dynamism, you could use the syntax "class extends Dummy" and then mutate it

afterwards.

AK: Anyway, it sounds like we're over timebox. It sounds like we have some objection to the "extends null" syntax behavior

MM: Because it is special behavior for a valid expression in an expression position.

AWB: What about just "class extends { }"? That would be invalid.

DH: OTOH if it feels sufficiently new, we have to think about whether it pays for itself. The cute cheat of "extends null" is nice in this way.

AK: What about the problems I've pointed out with the status quo; what do you think about that?

DH: That doesn't upset me as much as null behaving differently from (null).

AWB: Since this is a 2017 change, I am wondering whether we should roll it back before rolling forward with something new.

AK: I would be fine with that.

DH: BT mentioned, there's a mismatch between user intuition and spec machinery. The mutability seems like the biggest mismatch here.

MM: I'm more comfortable with the ES2015 way of doing things.

BT: I don't have a strong opinion, but I did get some feedback from people who tried to extend null, and found that they can't do that straightforwardly and have to implement my own constructor.

AWB: Well, the constructing-up-through-Function.prototype possibility could address some of this. Web compat issues?

KG: Chrome and Firefox have bugs preventing classes extending null from ever being instantiated.

AK: I can go offline and think more about this.

BT: Can we get consensus on if we want what is in the draft now or ES2015 semantics? Now is the time.

AWB: I would say we should revert it. It's better to go back to something that is already out there and is not perfect than to replace it with something else that isn't perfect that will be replaced again.

WH: I think we should revert it too.

AWB: It sounds to me like it just may require making Function.prototype constructible

MPT: The common knowledge for devs is that class is just syntactic sugar for doing it in ES5. With that in mind, it should behave as closely to the way it used to. Because you're thinking of it as syntatic sugar that's how you think of it.

AWB: But the old way doesn't actually delegate that way either. The old way is actually closer to saying, "it's a base class". Mark, how did we arrive at Function.prototype not be constructable; was it based on your concerns?

MM: I don't think I made a distinction there or see a reason to.

AK: I'm happy to do the work here. Brian's question is good. Do we want to revert that change? I agree there are use-cases for people extending null but I think there are a few reasons to be unhappy with what we

wound up with.

AK: Anyone opposed to reverting and doing more work?

BT: I just want this fixed. I don't have a vested interest how it is fixed, if Function.prototype being constructable works, I'm happy to do that.

AK: I'm happy to try to have this resolved for March

MF: Does reverting it involve making changes to browsers?

AK: Correct, we go back to the current state of shipping implementations.

AWB: This is all part of the January 2015 scramble and we missed a case there.

DE: I'm happy with that whole change overall.

AK: Okay, i'm 3x my timebox!

Conclusion/Resolution

- Revert the ES2017 change for now;
- AK to investigate a new solution for a follow-up spec change.

13.i.c Seeking stage-1 for Null Propagation Operator

(Gabriel Isenberg)

-
[slides](https://docs.google.com/presentation/d/11O_wIBBbZgE1bMVRJI8kGnmC6dWCBOwutbN9SWOK0fU/view#slide=id.p)

GI: I'm Gabriel Isenberg, here to talk about null propagation operator. This proposal is also know as "elvis operator" ?.

All: having a hard time reading the screen

All: Scaling in the browser is hard. Booo at the results!

GI: The common pattern today is using && operator, which is very verbose. We want to support message.body?.user?.firstname || 'default'. If any object is not defined along the way, we should return null/undefined. There has been a lot of discussion around this in the past. We have spec text that speaks to what happens when we hit null or undefined in the chain. This has been proposed in Typescript but was ultimately punted waiting on EcmaScript support.

BT: Typescript has similar syntax already.

JM: Hey, flow doesn't have the elvis operator. We added a really hacky callback oriented workaround. If that's compelling evidence that people really want this.

GI: I'd really love to see this in JS.

DH: I'd give strong support

BT: I'd support this

DH: It's not an accident that this is in high demand is that most applications read data that comes in from a dynamic source. Very often you can't predict statically whether all the pieces will be there. Maybe the data is ill formed, or you have a format with multiple options. Either way, this need to dynamically dig into nested structure happens all the time.

TK: This pattern has been in every application I've ever written, all over the place.

JL: I think this is needed, but it shouldn't be the only place it's added (support for brackets, calls, etc)., as Claude Pache's proposal does. We should also avoid encouraging ||

DH: There is an important sense for which this ties into the specific semantics of this operator. How does this operator treat falsy values. If we want a cohesive overall story. None of that is a stage 1 concern.

JL: Before we finalize it, even before we, we should consider additional operators.

AWB: Relevant to stage one is what are we making stage 1? Is it just this operator? Or is it the general problem dealing with accessing and invoking properties. Is it the dot operator, is it new? The scope of this item should be large enough to explore this space.

DH: It's full of cross-cutting concerns. We want to think about this. It's the usual dance, the totality of the problem is large and we want to consider it but we don't want it to stop progress.

AWB: As long as we define this item in a way that is broad enough to make it clear that we want to consider

DH: I think at stage one we shouldn't care too much. Null propagation operator or operators is the concern.

AWB: Agreed

DH: An example of the kinds of crosscutting concerns we do want to think about What's the special case of ?. with a number following it is a ternary. I don't care much about putting a float literal in a ternary but an array is definitely there.

JHD: Totally support saying this is stage one. I would expect at stage two questions about this would be resolved. The entire JS community thinks of these as null or undefined / skip. For default params, it's ONLY undefined. I think it should be null/undefined.

GI: seems like a lot of conversation would be warranted for stage two.

DH: I'm happy to help resolve that offline.

WH: I like the proposal, but the free grouping section seems odd. From the document:

```(a?.b).c().d   // equivalent to: a?.b.c().d```

I understand how that came about but I don't think that parentheses-overriding weirdness should be included.

JL: I'm not sold on the weird short circuiting.

DH: I'm not sure the call case is so important.

AWB: Claude's proposal addresses a lot of the edge cases; whether we like the particular decisions or not is something to look into.

MF: Would you be interested in moving this forward without syntax? I'm worried about the explosion of notation. After this proposal is in, when new language features come in, will we need to give them associated question mark forms?
```

DH: I'm not at all convinced that we need completeness. There is a strong argument to be made for the computed bracket case. There are hard syntactic questions there. its' for sure tricky but I think there are diminishing returns when you get into other areas.

MF: If this is done as a library, we can get a lot of these things free. We could put it in our standard library, so you don't have to bring in lodash as a dependency.

DH: We live in a world where bringing in dependencies is not unthinkable, people do it because it's convenient. I'm a big fan of it. But, you have actual static cases where you need to do string based indexing. The bloat and loss of clarity in saying what you mean is quite huge. WE're talking about a space where minor differences in syntax can make all the difference in comprehensibility.

DE: From an implementation standpoint it's easier to implement efficiently if the property is represented syntactically vs function calls that you pass strings to a function.

DH: I think we should always be careful and conservative about adding syntax

MF: Potentially a lot of syntax

DH: I especially don't think we need to treat this as a combinatory operator. WE don't need this operator for everything in the entire syntactic space.

MF: Do you think that the committee should suggest that we limit this to member accesses?

DH: I think the advice is good that we should be thinking about the over all space. Part of the strategy of getting this to success is charting a course through this space--there is a meaningful chunk we can ship/solve but that is not so path dependent so hill climbing that we end up diving into a morass of other syntaxes. I don't know the exact answer but the final answer doesn't need that we need something for every form.

WH: I have the same concerns Dave has. ?. is very useful, ?.[is useful but has awkward syntax, and ?.(is weird.

MF: I don't want to bikeshed on particular syntax issues

WH: ?.(is not a syntax issue. I don't think you should be calling functions on things that might not call a function.

AWB: Isn't that your classic, "I want to call this method if it is there?"

JL: I agree that the .? or whatever syntax we use, the property accessors we use are the most valuable. The rest of them seem like they have reasonable alternatives.

DH: To my mind, when you're adding syntax to the language it's for the most common idioms that you're trying to bless with the most concise notation. That you might ever want to say something is not the bar, it's that it's very common and idiomatic. I could be convinced otherwise, but pulling out a function from something that might not be there is much much less common than accessing a piece of data. In the cases where you might need to access a function you can handle using the existing && syntax but accessing field data is so common that we want concise syntax.

JL: My motivation for the call was basically lots of setter/getter chains (getFoo()/setFoo(), not actually accessors). Maybe i should put together an example.

MM: Concerned about semantics like `a?.b.c().d`, where the first "?" changes the meaning of all subsequent operators such as the () and . attached to c and d. In this example c and d don't have ? operators and it's surprising to find out that, for example, c might not be called.

DE: A nice thing about the surface syntax is that if you are accessing a private field on something other than this, you'll likely want to be able to do obj?.#x.

AWB: I'm not sure we need to support this for private state.

DE: If we went in the maximally general syntax, we want to consider

DH: I think the chances are high this will be impossible

MM: I'm trying to focus attention on a?.b.c().d does there need to be a?.b?.c().d? Should a single question mark color all further dots?

BB: I agree, if you look at Swift, they support this and they require a ? after every optional. Docs: https://developer.apple.com/library/content/documentation/Swift/Conceptual/Swift_Programming_Language/OptionalChaining.html

JHD: Now that we've considered this stage 1, I think we should table this until stage 2.

GI: I will champion this in all forms

AWB: Different threads went through different alternatives

WH: I like it, but I don't want short-circuiting and free grouping.

Conclusion/Resolution

- Stage 1, champion for investigating the whole space, not tied down to the details of Claude's concrete proposal.
- Areas to investigate for next time:
    - How should null be treated?
    - Should we have a better default operator than || ?
    - How should computed property access, function calls, new be handled--as in Claude's proposal, or another way? Are they needed for syntax--do they pay for themselves?
    - Short-circuiting--does it need to be as general as in Claude's proposal. Also, free grouping.
    - Interaction with private state--included or not?

8. ECMA402 Status Updates

(Caridy Patino)

[slides]: https://docs.google.com/presentation/d/1w-dhlVfbstl8MO0onMc04julUkk3WtKzTuQLN2Vz6-Y/edit#slide=id.p

CP, ZB:
    - Editorial changes
    - Legacy constructor compromise (ready to land in SpiderMonkey)
    - NumberFormat.prototype.formatToParts will wait pending evaluation by Google; DateTimeFormat.prototype.formatToParts is at Stage 4 and makes it into ES2017

AWB: Is it OK to have DateTimeFormat.prototype.formatToParts but not NumberFormat.prototype.formatToParts?

DE: I don't think there is a situation where you have a formatter and you don't know what you're formatting. It's not polymorphic. It's good for the APIs to be analogous but I don't think we'll discover meaningful differences.

ZB: Intl.PluralRules is at Stage 3; we are awaiting a second browser implementation

AWB: A broad question... in both of these cases, where we don't yet have two implementations. Is this a reflection that these things have gotten to the stage relatively late and there hasn't been time? Is there feet dragging going on among implementors? What's the level of interest in extending thes features?

ZB: It is somewhere in between. Over the last year it was mostly Google and Mozilla (as browser vendors) interested in expanding intl support in JS. I think Mozilla got a little more active because of our internal strategy / higher pressure. Google is supportive and helpful but has a little less resources to implement it. Often we are pushing the spec and the tests, but Google is taking a bit longer. Is that representative Dan?

DE: Yes, that's accurate.

CP: Yes, this spec is only been around for a few months.

ZB: This is what we consider internally on the ecma402 group is done. Userland can follow an implmenet if they want to. It's been reviewed, we think it is a sound and safe API. Once a second implementor steps up we should put it into the spec.

CP: Three more formatters that we introduced about a year ago that have been through discussions and writing.
- intl.ListFormat
- intl.RelativeTimeFormat
- intl.UnitFormat

I'd like to move these to stage 2. My understanding is that stage two means we'll be looking for more feedback from implenetors, but not putting pressure on them.

AWB: The move between stage 1 to 2 is saying, here is an area we think will be useful, let's explore it. Stage 2 is saying we should start getting something like this into a future version of the spec. I would expect at that stage some sort of review presentation. To move to the next stage would be appropriate to have a review of the stage. Here's the API we arrived at, here's why it's good, we should advance this, etc. My sense is that waiting two months is not going to really hold you up any.

ZB: Happy to delay to the next meeting. Two new proposals to put forward. Deprecate `hour12` option and introduce `hourCycle`. Proposal: https://github.com/zbraniecki/proposal-ecma402-hourcycle. I don't think we need to seek any stage now, I'd just like to introduce to the community that we are using it.

DE: About this proposal, I'm wondering what the appropriate vehicle for it is. The spec text is written very well, thanks for responding already. It's small and we may add more thinks to this. I'm wondering if each one should go through a 4 stage proposal. We might move through the stages quickly, but getting implementation feedback on these things would be good.

AWB: I think it could be streamlined as one. As a starting point, I'd like to understand what you mean by deprecate.

ZB: The proposal means that we will still fully support the current implementation (interface). If someone creates a dateTime format with hour12, we'll internally convert it to an hourCycle. If both are present, we'll use hour12, because we'll assume it's an old implementation. In the spec we recommend users use hourCycle.

JHB: Why support both in new code?

ZB: I misspoke, we are doing what you said already. Out of all the things in our current scope are additions.

CP: It's not really deprecation, we're still supporting it.

ZB: I would like to express to users of this API to use hourCycle, not hour12.

AWB: Yes, you're trying to establish a pattern here. If there is a newer option, continue to support the older option but in the presnce of both, ignore the old options. You tend to follow that pattern for future things.

DE: This might not be the last one. There are all sorts of things that fall into this bucket... maybe we'll have more possible values for a key.

ZB: This is the one that we know if we try to get to extension key to options, we hit a line.

BB: Are there options other than 24 and 12?

ZB: Yes, 23 and 11.

BB: Don't forget Mars time!

ZB: This is academic right now, we only support Gregorian calendars, but as we start to think beyond those, we shouldn't carry these limited features.

BB: I have no real objection. I've never been to a country or seen a 17 hour clock.

ZB: If you look at a unicode

DH: Swatch invented internet time in the 90s, like a 10 or 100 clock

BB: I could imagine some metric time system but it seems a little extreme to anticpate something tha tmight be invented in the future while standardizing.

AWB: The real question is, if this is a pattern, does this pattern apply to other specs. Do we have option objects in any of the shared buffer or atomic stuff? We don't have them yet in 262.... but we might someday. So, if we had any... would we apply the same principal? We probably should.

DE: This sets some other precedents. AS a part of this deprecation, if you instantiated something with hourCycle, if it can emulate the old hour12 format, it will, then you can read off the old property or it will be missing if it can't. I don't think this is that academic. In the US midnight is 12. In many places in the world midnight is 0.

ZB: I agree it's not that academic. It has the value of closing the gap and being more consistent. I wanted to test the procedure of this kind of change with this community. I wasn't sure what the reaction would be. I'm not sure if you have a precedent of this.

AWB: No, not exactly. I think it would be good to write up this pattern. I'm not sure where we put it, but if we think there are a set of steps you'll follow here. The next time this comes along it would be good to write that up rather than having to reverse engineer it from what we've done.

MPT: On the record, we have an extension point in moment.js for this and it gets used all the time (extending hour cycles for time display). It is not academic at all. Tons of locales use this.

AWB: I think we're fine.

CP: Are we going to do a pull request on this? What is the consensus here? Is a PR enough?

AWB: I suggest treating this as a bug that needs consesnsus.

DE: Could we get especially much feedback from implementors before landing it?

ZB: I am seeking feedback

DE: Yes, but as a part of process

AWB: I don't think this is any worse than tweaking we were talking about with extends null.

AK: To be fair, this is something that Dan called out as saying this might be problematic when we landed it.

CP: This is the same as a the PR you have open right now, the compatability bugfix. It's about the same. We wait for feedback and we merge it.

AWB: Don't we have a tag for this?

CP: Yes!

JHD: In general for this, we're trying to adhere to a different spec. There is a general loose goal of trying to match that long term. Over time those changes are going to come into play. Implementation feedback is important.

DE: 402 already normatively references BCP47. This is really a new feature. It's not a bugfix. It's exposing extra functionality. Other things that we might pull in from CLDR are also even more straightforwardly new features. Maybe we can add small new features like this by PR.

ZB: I think I agree with what you are suggesting. I just don't understand what the formal way of doing this is. getting feedback takes time. I'm okay waiting. I would like to do this sooner rather than later, but I'm happy to wait for browser vendors if we need to.

DE: As I suggested at the last meeting but it was rejected. Maybe we could have a protocol to get one or two implementations just to like, make sure the idea is fully articulated before we accept the pull request. That will give us more information for new small features.

JHD: The rejection last time was for -any- normative change. This makes a little more sense.

ZB: My only question is, if I land the patch in spidermonkey next week... is it enough? I woudl like to think we can hope for more.

AWB: It's potentially kind of self-serving if the proposer of the change is th eimplementor and nobody else is doing it. But, we all know here what is going on. We can decide if this is sufficient. How do you get others to try it?

ZB: Our experience is kangax compatibiulity tables give browsers strong incentives to implement

AWB: Sometimes, you need to nudge the implementors, the spec says this, fix it.

ZB: Also happy to say we announced it today. We can see if in two months there is any feedback. Then we'll land it.

AWB: People here who represent these implementations should be able to report back.

ZB: Last item, adding a "style" value for date time format. Also an idea from CLDR. Trying to have parity with how operating systems manage this. How can we make it easier for webapps to follow these preferences? We would like to introduce this change in some way. I'm not seeking any stage, I'm just announcing this is a conversation we are having. One open question i'm seeking feedback on: any thoughts about adding style to datetime format?

MPT: There is not community consensus on what they should be. It's a never ending source of pull requests on moment. The community doesn't accept what Unicode has decided.

ZB: My thinking around this is that CLDR is a great place to have those conversations. Second our API is granular enough to allow userland to do whatever they want. Internally at Mozilla, if we want to match how OS's create a date. We need to follow how Windows/MacOS/Android etc do it. This allows us to close the gap, the OS does this, we'll do it too. I get that it is a heated topic.

AWB: It sounds to me like this should go through the stages process, and that this should be at stage one.

ZB: I'm not quite ready, no draft yet.

AWB: Not needed, this is enough. Unless somebody thinks this is not clearly described.

DE: The distinction here is that there is more complexity?

AWB: it sounds like there isn't necessarily a simple obvious solution that everyone agrees to.

ZB: I think there is. The solution is based on CLDR consensus. Which, as the name suggests is just the consensus. There are a significant number of poeple who dsiagree CLDR, but that's okay. We seek to confirm to CLDR, regardless if everyone agrees 100% with CLDR. We don't forbid people do other things, though.

AWB: Is adherence to CLDR a recorded decision and we have consensus on within tc39 or the 402 subgroup?

CP: Even if we adhere to CDLR, there are still questions to answer. We just don't know. We don't have a solution yet.

AWB: For me, that sounds like stage one. I'm not sure what the scope of that items is. Is it just datetimeformat style?

MPT: I would suggest hard adherence to CLDR. The moment you're willing to support -anything- but CLDR you open the door it never, ever ends. My advice from the trenches if nothing else is hard adherence. If it's not available in CLDR, shy away from it. Let people do custom patterns, but the community is going to fight you on that.

AWB: I think it would be quite in scope for 402, if they in fact could agree on such a decision. If you brought that here you'd get a rubber stamp and we'd move on. This is the responsibility of the 402 subgroup. Solve these problems and make decisions where appropriate.

ZB: Okay, i'll rely on my editor!

CP: So, stage one?

AWB: It's stage one for something!

ZB: Okay, stage one for datetimeformat style.

AWB: If you guys really want to consider saying yeah, CLDR is your guiding light for a lot of these things. You ought to say that and do it--see if you get any push back.

ZB: Dan, so far we haven't explictly stated supporting CLDR right?

DE: We have non-normative text stating it and historically Microsoft pushed back on it.

ZB: That was because CLDR at that time was much more bound to ICU, if I rememeber.

CP: Brian, do we have an update on that?

BT: No update.

ZB Would you oppose normative binding?

BT: As a rep of MSFT I can't say yet. There is an important decision coming I am waiting to resolve.

ZB: I'm happy to say that if it isn't in CLDR we won't support it. We expect you to have as much data as CLDR.

BT: The concerns aren't like... in some cases windows data is better than CLDR data. In other cases we have different fields that accomplish the same end goal. Using different fields could be non-standards compliant. That would mean we'd have to maintain two databases with different structures. I think we have to be clear that if we go down the route of standardizing CLDR, in practice no implmeentaiton will use anything other than CLDR. I don' tthink that's a bad place to be personally, but we should be cognizatn of the fact that the more stuff you add stright out of CLDR the harder it becomes to deviate.

ZB: I think that's a good point. I think the current CLDR proceudres are open enough to have everyone resolve their issues there.

BT: I haven't seen that participation with MSFT

ZB: I know we recently sent our localization changes upstream to CDLR. We're testing the system

BT: How is it working?

ZB: So far everything we've sent has been approved. The volume is likely lower than what MSFT would send though.

ZB: One other thing, we have an internal debate we haven't resolved yet. We have two option formats for datetimesformat. Does the community have an opinions on what looks more like JS?

??: The single format style matches CSS, so that makes sense.

ZB: Thanks for your time!

Conclusion/Resolution

- Stage 2 was proposed for these topics, but we didn't get an explcit "yay" or "nay" from the committee, and didn't appoint reviewers:
  - intl.ListFormat
  - intl.RelativeTimeFormat
  - intl.UnitFormat
- Stage 1 for the DateTimeFormat "style" option
- "hourCycle" can proceed as a needs-consensus pull request

13.i.e Discussion of Annex B.3.3.3 EvalDeclarationInstantiation overwriting bindings

(Shu-Yu Guo)

https://github.com/tc39/ecma262/issues/753

SYG: No browser is spec-compliant with this Andre Bargull test case. Reviewing the issue contents.

```
var f = "x";
eval("print(f); { function f(){} }")
```

```

SYG: This should print undefined, based on calling CanDeclareglobalFunction followed by CreateGlobalFunctionBinding. This is a strange discontinuity

AWB/BT: It's a bug.

SYG: IF we have consensus that this is a bug, what should we do?

BT: Can we use our normal rubric of remove eval and do what that does?

KG: In this case you're creating a function binding after you've already been writing to the binding. Not performing the CreateGlobalFunctionBinding technically changes the semantics. So, CreateGlobalFunctionBinding isn't quite identical to creating a var declaration.

AWB: Changing to the consensus implementation of what is historically done.

KG: This is like a weird... this isn't intersection semantics. Nobody is depending on this. We can do whatever we want?

AWB: Nothing might be a possibility. At 2015 we didn't actually specify at the top level, particularly for eval my recollection now is that we starting running into a bunch of hair in all of this. In fact it's not clear there is a valid implementation intersection semantics here. We didn't specify anything. We left it up to implementations. For 2016 we said "oh oops, we forgot about eval and global". Then we started defining things like we understood the situation but we didn't really. You know better Brian.

BT: We had a giant table of all of these cases with browsers and what they did We tried to weave through what they did. There were some pretty crazy things there.

SYG: All browsers agree on this now. We aren't spec compliant but we all agree on what the observed behavior is in the wild.

BT: I thought you said your slide said someone disagreed?

SYG: I think that's just... incorrect?

BT: I guess either it would be undefined or....

AWB: Okay so, that's certainly a quite reasonable legacy semantics. Simply say there isn't an intersection that covers that case.

BT: Yeah

AWB: It's just hoisting across all the blocks. That's close to what MSFT does.

BT: Yeah we used to always hoist out of blocks.

SYG: Are you suggesting we rollback the eval function hoisting semantics spec?

BT: What? No. Let's remember when we got here there was no spec and all kinds of wild behavior. Now we at least agree on sane behavior.

MM: I want to double check that none of the changes or non-changes that we are considering will have any effect on strict mode.

SYG: Correct. But i don't know what we're considering.
```

KG: I would like to propose that we check to see if there has been a variable declared. So, okay. Two potential things you might smash, properties and global objects. I'm not entirely sure what to do about smashing the second kind of thing or what browser agreement is in that case. If you say function array, does that overwrite the array constructor? If there is already a variable of that name declared, we should just not CreateGlobalFunctionBinding

AWB: That's consistent with how it's handled in the global case

KG: Yeah, it is consistently. Functions don't have that other case, though.

AWB: That's why we went through the trouble in the spec to defining this global distinction vs properties that just have appeared.

KG: It's really funny. I don't know what to do about that case. At least if there is a variable declaration of that name, don't create the binding.

AWB: if it's already there, you shouldn't do it at all. imagine you removed the eval. In that case we don't do the hoisting, do we?

KG: What do you mean by "the hoisting"

AWB: B.3.3.3 hack for that function.

KG: Which part of the hack

AWB: The leakage of the declaration in the block out to the global scope

KG: The leaking still happens. The assignment that occurs at the position of the function declaration still occurs.

SYG: without the eval, still at the top level, but it happens at the very beginning of the global script execution instead of executing the eval script so we don't see this undefinedness.

AWB: I have pull out the spectext. I thought we had a list of conditions if this and this and this and this apply these semantics.

KG: The condition is if there are no outer lexical declarations, but you would overwrite a var declaration or a top level function declaration.

AWB: For function declarations this was a set of rules for when you do this. For eval this has to be turned into dynamic texts for the direct eval semantics. They should essentially be the same tests. If it's in a function

KG: Yes. The difference is the same set of tests on whether you should perform the write. At the point you're actually executing the function statement, I agree those should be the same The question of whether or not to CreateGlobalFunctionBinding.

BB: What I find weird about this case is that overwriting happens through eval, but it only happens at that later time.

AWB: So, it still seems to me, whatever the semantics of var foo, block function foo, at the global level, whatever they actually are, if you put an eval in quotes around that block, the static semantics that are applied for the case without the eval should be the same as what is dynamically applied.

KB: The problem is that the declaration occurs later with the eval. With eval, creating the binding occurs after someone else might have created that binding. It's only observable in this case because creating the binding happens in a different place. It has to.

AWB: You're saying the difference is that within the block if you don't have the eval you have block level hoisting.

KB: At the beginning of the script

SYG: So Kevin is saying for Annex B there are two phases.

AWB: In this example, when we do the eval we are clobbering f, even if we never execute the block. That's the semantics of Annex B.

SYG: There is no dynamic counterpart in a function binding when eval is not present

AWB: Isn't there a similar thing going on when we do this in a function. We aren't really doing anything with the outer scope.

SYG: Because the spec mechanism for creating a new var level binding is not this CreateFunction thing...

AWB: It sounds like we're using the wrong spec mechanism.

SYG: No, we need to make a new one.

KG: Can we agree on the new semantics?

SYG: What Kevin has proposed: https://github.com/tc39/ecma262/issues/753#issuecomment-267171024

(quick discussion)

SYB: We have consensus that it is a bug. We want it to print x and we may need a new spec mechanism to do this.

MF: The small mechanics around it can be resolved in a PR

JL: Random question. There are no bugs around global properties here, right? So what, if you have window.f, what happens?

SYG: I think the same thing happens. Properties of window are just like "var f" in the global scope

Conclusion/Resolution

- Consensus on new semantics to print "x"
  - This is a bug.
  - This should print x
  - We may need a new spec mechanism to do this.
  - To get in ES2017 (not too late, as this is a bug fix)

13.ii.b Function.prototype.toString updates

(Michael Ficarra)

[proposal](https://github.com/tc39/Function-prototype-toString-revision)

MF: I don't have materials or anything. I recently made changes to the stage 3 proposal. We discussed last time the line terminator normalization. This is a process that happens at the toString algo to replace all line terminators with a standard one. That was added as a part of this spec. No implementation actually does this. I remind you that the proposal's original intent was just to specify existing behavior. Sort of like an

Annex B requirement. I've removed that normalization. I just want to make implementations aware of that. The reason I've done this is because I don't think it's valuable to try to specify something that all implementations have said they would not implement. We're just really trying to spec reality as closely as possible.

AWB: Just to clarify. Not everything in this proposal toString is done by every implementation;

MF: Where the implementations differ I try to use the simplest possible mental model of how the method should work. We've agreed on this being from the first and last token for the given grammar production. For functions created dynamically we have a very simple string building operation that tries to get as close to the overlap as possible.

AK: To Allen's point, V8 has done some implementation work to match the spec. Thanks for this change. it does make our implementation significantly more performant.

MF: One thing I want to add is that we did look into how difficult is to just do it yourself if you want to. It's a lot easier if you want to go from an unnormalized string to a normalized one if you want to. It's a pretty simple regex.

AWB: The motivation for this sort of normalization is to eliminate variations

MF: There are a lot of good reasons for doing this. IF we were defining this as a new feature I'd be fighting harder

MM: Since I was a strong advocate for normalization last time we met. I want to go on record as saying I do not object. I care much more to see this go forward than to see normalization.

MF: I feel exactly as you do.

DE: Our implementation in V8 has a patch, are there others?

MF:I know Mozilla was working on this when they realized the performance implications

BT: I implemented this in chakra but i didn't do the line termination. I only did it to fix another bug, it's not close to the spec.

MF: I've updated all the test262 tests to not enforce this. I can test this in the wild to confirm the current state of things. The new tests actually enforce the opposite. I'll do some research tonight.

DE/AK: We aren't shipping, our patch is not checked in.

MF: Just wanted to let implementors know.

WH: I raised the issue with the lack of normalization being user hostile, but if implementations are against normalization I'm not going to block the proposal.

AWB: that's it!?

MF: Yup :)

Conclusion/Resolution

- Proposal stays at Stage 3, going forward without line terminator normalization.

What to discuss next

DH: What's going oNNNN? The universe makes no sense anymore!

AWB: If anyone has to wander off before 5, the intent is at the beginning of the meeting tomorrow is to handle all the non-technical issues we have remaining. In particular that includes the issues around diversity goals and code of conduct. If you're interested in those particular areas, the beginning of the meeting tomorrow, you'll want to be here.

BT: Can we ask people to stick around for the CoC issue?

AWB: I have done so!

AWB: Capture groups?

13.ii.c Seeking Stage 2 for named capture groups

(Daniel Ehrenberg)

[slides](https://docs.google.com/presentation/d/1fx5S4DpuD7z4K6ItFW7BjbvYPiuQNxU2769F5YhGP3o/edit#slide=id.p3)
[proposal](https://github.com/tc39/proposal-regexp-named-groups)

DE: A quick overview of the feature: it allows you to access matched groups by a name in the match object.

AWB: Can the same name occur more than once?

DE: No, that's an early error. I'll double check that.

WH: That's an early error.

DE: There are a few issues I've been flip-flopping on. Should we allow this outside of unicode mode? the initial proposal was unicode only, then Kevin pointed out there is no ambiguity at all. The new spec text reflects a method of handling this.

AWB: Other than the annoyance of having to put a u after the slash, is there a reason people wouldn't always use unicode mode

WH: Lots of reasons. For example, if you're looking to match the ASCII a-z in case-insensitive mode, you'll get subtle and nasty surprises if you turn on unicode mode. It's quite common to run regular expressions on such programmatic or protocol data.

DE: The other thing I flip flopped on was whether the results should go on the match object or a nested groups object. I heard complaints about what we we added more properties?

MF: If someone writes a function today that accepts a regex, runs it, then accesses properties of the match that are supposed to resolve to existing properties may be masked by a capture group.

JHB: Sure, if you try to capture forEach or map in my capture groups.

DE: Another argument on the contrary of why we shouldn't expect more properties, given that we do have regex subclassing, anything exec'able should be able to pass it in and those might not be updated for the future version of ecmascript that gets added to the match object.

BT: Do you have a slide showing how destructuring and pattern matching differs?

DE: See the explainer; it looks like `let {groups: {one, two}} = /^(?<one>.*):(?<two>.*)$/u.exec('foo:bar');`

MM: Is there a reason -not- to use a map?

DE: It's not ergonomic, having a separate groups property is already some overhead, but maps are much more; but you can't do destructuring on a map.

BT/DE: Examining potential issues on destructuring assignments when exec returns null.

DE: There is spec text for all of this.

MF: Is groups possibly null?

DE: The match object is null if there are no groups in your regexp. If there are groups but the groups are not reached, they'll be undefined, just like with captured items in an array.

JHD: Groups could always be an empty object

BT: Yeah, that makes sense

AWB: That's adding another object instantiation for every...

JHD: It lets you do Object.keys

BT: Object.keys is a proxy for any operation you might use to enumerate. The only scenario that's important is dotting into the object.

DE: I'm saying that when would you try to do anything with the groups property from a match object when you don't use named groups?

AK: I don't feel like you want to pay for an object for every regex ever.

AWB: Can we safely return the regexp result object to an instance of a subclass of an array?

DE: It still doesn't answer the motification of having groups always there.

AWB: If you have groups as a getter on the subclass....

DH: If we're always returning a unique object when you call the getter when you normally access the groups property.

AWB: It could be the same value, an internal slot that we keep returning.

DH: We have lots of situations like this! Does it have to not exist? Could we just set it to the value of undefined to allow destructuring?

DE: I don't understand what code would require these. If you use destructuring, you know what groups you're looking for. If you write a regex, you know what groups you're looking for. Why would you introspect keys for groups of a regex that you don't know?

KG: There are two completely separate concerns, making destructuring work, and making object.keys work. It doesn't sound to me like anything in this change, having groups or not, would affect destructuring. Having the match object return undefined vs null would have mattered but that ship has long sailed. Having groups there or not doesn't affect destructuring as far as I can tell.

JHB: If the property isn't there, the destructuring will throw

DE: We know, the groups protocol would be very straightforward. Why would you not know what the groups are called.

DE: What is the object keys concern?

BT: Yes. You do a match, you see you matched something. You want to get a list of the groups that matched.

DE: Okay. So, then, making an empty object be in the groups property makes it so you can always introspect which groups exist.

KG: Or just do || empty object.

BT: The reason we are discussing this is to determine if it is reasonable for us to make it easy to work around.

AWB: is there really any question about it being separate? it affecting the prototype chain is a problem.

??: I'm actually concerned about that. People could referential integrity stored in a weakmap. That's an odd scenario I don't want to ever exist, where we sometimes have the same object and sometimes different objects.

AK: We already have a situation where something like this... template strings with a frozen array. The first thing looking into is what are the use-cases

(bikeshedding)

JHD: Currently, because I can't know the number of named captures, the way I've seen replace used, it depends on the number of arguments, you ignore the last two. With named groups being combined into one this could be a compatibility concerns.

DE: There is direct applicabliity to this, you could use the named captures as a legitimate application to replace.

??: In this proposal it's possible to mix named and position groups in the same regex. Is that something you really want at all? You could just say if you're going to use named groups you have to name them all.

DE: In this proposal, named groups still come back positionally. You could refactor an existing regexp by incrementally naming some groups, but leaving the positional ones with no change in their numbering.

WH: You know that somebody will come up with a feature request to have named capture groups that don't use up positional numbers .

BT: I have been in the weeds implementing my own regex engine because in Ecmarkup, I have effectively [a giant regexp that I produce dynamically](https://github.com/bterlson/ecmarkup/blob/f8128632e765960ac55b9853315244ca639bc9b8/src/autolinker.ts#L66-L93). There is no way, right now, for me to know, which part of that pattern I've matched. If I could give a unique name to each alternative that would solve this.

DE: Why can't you do that with numbered captures?

BT: In order to know which you've found you have to iterate the whole thing to find out which one isn't defined. My guess is that the other group names won't be present if they aren't matched.

DE: The semantics in this proposal, in order to match the semantics of numbered captures, all the groups are always present.

BT: Sure, so it's the same problem here. You'd have to enumerate all the keys.

DE: OK, interesting. Sounds like we could potentially make the change to leave out the undefined keys, for this introspection; thanks for explaining. Seeking stage two

MK: That's fine

JHD: Sounds like there are just editorial issues

AWB: Okay, we call this stage two

AWB: Annex B regex grammar, do we fit within those constraints? The non-unicode case is pretty much what browser implementors use.

[bikeshedding about how to address this]

AWB: Are you suggesting if you use the named things, that kicks you out of Annex B grammar.

DE: No, no, it just kicks you out of that production.

AWB: Why not kick you out of Annex B?

DE: I would worry that would have weird refactoring hazards, if we said that if you invoke this feature, then you step into a different RegExp grammar

WH: Describing entertaining problems we'll encounter when modifying the Annex B grammar to add named capture groups: That grammar is unique among the grammars in the spec in that the grammar is highly ambiguous and the order of productions matters. The production that's listed first wins. For `\k` there'd be two productions, one for parsing `\k<id>`, and the other one being a `\k`; if the parse of the first fails, the second one would be picked, which is undesirable.

DE: OK; we'll disable the `\k` identity production in `+N` mode in Annex B.

WH: That would work.

AWB: What if we just kick the regular expression to the non-Annex B if named capture groups are present?

MS: It makes implementation very difficult. I have existing regexps, Unicode ones, named I'm going to go partially into, but if I'm not in Unicode mode, then this other behavior is invoked

AWB: The biggest issue could be that you may be way at the end of a parse before you realize you need to change.

DE: That's already going to be the case in some places. (NB: See related blog post https://hackernoon.com/the-madness-of-parsing-real-world-javascript-regexps-d9ee336df983#.kup9nqgd8)

Conclusion/Resolution

- Moves to stage two
- Groups are segregated into a separate object (to avoid conflicting with prototype changes)
- Reviewers: WH, someone from MSFT
- Issues to look into:
  - Interaction with Annex B grammar
  - Tweaks to groups object (always present? include only properties for groups that are found?)

###

AWB: It sounds like we now have several regex related features to land in 2018. I think it would be beneficial

for users to let them know about this theme.

MB: https://mathiasbynens.be/notes/es-regexp-proposals

January 25, 2017 Meeting Notes

Allen Wirfs-Brock (AWB), Waldemar Horwat (WH), Jordan Harband (JHD), Brian Terlson (BT), Michael Ficarra (MF), Adam Klein (AK), Chip Morningstar (CM), Dave Herman (DH), Kent C. Dodds (KCD), Kevin Gibbons (KG), Tim Disney (TD), Daniel Ehrenberg (DE), Shu-yu Guo (SYG), Michael Saboff (MS), James Kyle (JK), Franziska Hinkelmann (FH), Anna Henningen (AH), John Lenz (JL), Sebastian Markbage (SM), Bradley Farias (BF), Jeff Morrison (JM), Tyler Kellen (TK), Gabriel Isenberg (GI), James Snell (JSL), Maggie Pint (MPT), Chris Hyle (CH), Gabriel Isenberg (GI), Bert Belder (BB), Zibi Braniecki (ZB), Jamund Ferguson (JXF), Brendan Eich (BE), Istvan Sebestyen (IS) (part-time, on conference call), Keith Miller (KM), Brendan Eich (BE), Myles Borins (MB)

Istvan update, items 4-10 on the agenda

4. Approval of the minutes from the last meeting

AWB: We are having trouble getting the minutes to approve

IS: I prepared the TC39 November meeting minutes (16/052) before the 2016 December GA meeting, but we had technical issues with the hosting. The old Ecma private website is down and we have to replace it. That will take at least two months. In the meantime we have put all the data on a NAS storage on the office network and that has been released to all Ecma members (this is the Pydio system on a Netgear box). We have also a parallel implementation on a Synology NAS that will be released soon. For approval, I could upload the minutes to GitHub and send them by email.

AWB: If you email me the document, I can make sure everyone here can see it.
IS: This has been done immediately after the completion of the conference call.

IS: 16/052 has two parts. The general part of about 6 pages that includes list of participants, list of companies participating in the meeting, various ISO/IEC JTC1 matters, like status of TC39 Fast-Track projects, dates of next meetings etc. For the TC39 participants , these 6 pages should be relatively uninteresting, but generally, for other Ecma members and the GA this is a summary information they are most interested in and less into the technical details of the TC39 work. This is included in the Annex, which containes all the Technical Notes (like this one...) of the entire meeting. Generally, we need to work on passing around documents in a more reliable way. We also need to improve feedback mechanisms--I only learned today that Waldemar encountered technical issues in accessing the minutes.

WH: A couple different problems: There is an invalid (self-signed) SSL certificate. Even if you accept this, the scripts to surface the links time out.

IS: From a content point of view, these minutes is the same as what is mirrored elsewhere. We just have to make sure the details are approved.

AWB: We'll take care of the review tomorrow

5. Report from the ECMA secretariat

IS: Explains the current status of the TC39 Fast-track projects to JTC1. We have various components in ECMA to standardize in ISO. For ECMA-262 (the main ECMAWScript standard), we now have an agreement with ISO that we will not fast track it anymore; instead, the ECMAScript Suite ECMA-414 will supercede it (with normative references also to the needed Ecma standards), and the other redundant standard ISO IS 16262 (which is out of date) will be withdrawn. The only standards in common will be ECMA-414 (on the way to fast track), which happened after the Dec 7 ECMA GA. As you remember ECMA-114 has undergone some minor revisions for ISO requests. We have also provided an explanatory report to ISO (what has been published also as an TC39 document). We have published the document TC39/2016/050. The ISO number of the suite will be ISO-/EC IS 22275. Currently it is registeredasa DIS, but the voting has not started yet. When we change the suite standard, then it will get a new Edition number. ECMA-402 will also remain an Ecma only standard and its latest version is just referenced by the ECMAScript Suite.

BT: Question: The document ISO-IEC 16262:2011, we want ISO to withdraw that document, not "stabilize"?

IS: Defacto, it will be withdrawn. The usage of the word "stabilize" is very funny in ISO. It's possible that they'll call the withdraw number "stabilized". I have to check that. But it was ISO that started calling this "withdrawal". Originally we, Ecma wanted that the ECMAScript Suite should get the same ISO number: IS 16262, but that is not permitted. So the Suite will get a new number. The DIS voting has not started yet; we don't know for sure how that will go.

BT: On February 7th, the PL22 working group is voting on what to do on the 16262 document. Until now, they have been voting for reaffirmation; instead, we want them to vote for withdrawal, and in place, we are fast tracking the ECMAScript Suite document

IS: [Some people, like the SC22 Chair, Rex Jaeschke] are in the loop already

BT: I think there's a different group of people in this other group; I'll loop you in in an email

IS: On JSON, there is some work to do: It got the DIS number 21778. The DIS voting finished in December and it is positive--it is approved by ISO. But the Japanese national body had some comments. Mostly editorial, but some possible technical changes; CM and AWB can follow up. The Japanese national body has voted 'No', and then would change it to 'Yes' to reflect incorporating fixes. Therefore now we have to do a second ballot (FDIS) because there was one no-vote. We need to prepare a proper fixed version of the document with the disposition of comments to start the FDIS voting and to get finally this "Yes" vote.

IS: In the ECMA GA, Waldemar was there, so he can fill in any details and corrections of my verbal report. People in the GA are happy about the work of TC39. Participation is growing to-40-50 people each meeting, which is good, but can cause organizational problems..

6.ii. Who should be the chair?

IS: We discussed the TC39 leadership--we have to put into place a leadership body to ensure that it is functioning long-term; in the short-term, Allen has been very helpful, but this is a short-term solution and we need to find a new leadership structure. The best possibility is to have the chairmanship position held by an ordinary ECMA member, but ECMA bylaws also now permit other categories of ECMA members to be the chairman. We have had the strange situation in ECMA TC39 where we had a chairman who was financed by three member companies; this has finished. We also had the strange situation where the chairman did not understand much of what was going on on the technical part (this had entirely historic reasons). The chairman cannot be an impartial adjudicator of disputes if they are not following discussions, but anyway, we have a cooperative spirit on TC39. Strangely enough we did not have in TC39 a Vice Chair, which is in most Ecma TCs the case. A Vice Chair really can help in carrying out the Chairs function. IE.g. f there is a vice-chairman, and the Chair has a company proposal which he has to present then the two chairs can switch off, as one of them is acting as chairman and the other is presenting, so member companies in a chair position can still present their own ideas. So, we can have a Vice Chair, but in addition we may also put a management layer below the chairman. TC39 has the freedom to organize its management structure.

AWB: Would it be possible to have two vice chairs?

IS: Absolutely. ECMA is very flexible on this. If you suggest three, I am sure I can even that get it through.

AWB: The thought I am having here is perhaps we had a chair and two vice chairs that collectively formed a management team but shared the burden of doing that work and, running the meetings, and managing the agenda and other items that came up maybe it would be easier to get three people than it would be to get one.

IS: Absolutely. You are free to set up any sort of management structure. We are very flexible on that. For instance I noticed that people are extremely bored about this subject in the TC39 meeting. So It is possible that e.g. you install a separate small group just for "procedures," on I don't know, "strategy planning". Such

groups then could report back to the full TC39 committee who could listen to the recommendations and make decisions. You have a lot of freedom in shaping how you want to do the work in TC39. It is really not our goal from the ECMA point of view to bind you in any way. So this is really just think about--we don't have too long time to do this. Our temporary solution with Allen (which I think is good) unfortunately cannot last forever, because he has other priorities . We want to get rid of this issue as quickly as possible.

AWB: I'll do the next meeting in Portland because I do not need to travel. From my perspective it would be ideal if we could have stuff resolved so maybe by the May meeting we have a different structure in place. At the latest the July meeting. I wondering, going to throw out something here. The question is how are we going to make progress on this decision?

DE: My employer would support me being a chair or vice chair.

AWB: Could we maybe form an ad-hoc committee here to do work between this meeting and next meeting so candidates like Dan could talk about the job? But also to think about doing some recruiting.

DE: I've been asking around for many months.

AWB: I can work with you guys, but it is you guys have to collectively step up and figure out what you're doing to do. A group of 30 isn't going to do it. A group of 3 or 4 try to pull something together and bring it as a proposal to the next meeting or the meeting

AK: Leo was going to ask the JS Foundation

AWB: I'm not asking for volunteers for these roles, but people to essentially be the recruiting committee. Okay, we want have a chair and tow chairs and why. Somebody to pull it together.

DE: Presumably I shouldn't be in that group?

AWB: I wouldn't want to exclude anyone, so.

??: I could look into the JSFoundation
(MM: This was attributed to me, but I don't remember saying it. Hence the "??". If someone remembers who did, please edit it in.)

AWB: It would be helpful if someone

AK: If there is a lack of pushing for it I can help. We've had trouble finding the right people for it.

AWB: Okay.

DH: We can have some discussion about next steps offline.

AWB: Okay, not here, but some discussions... however you do it is fine. Is that commitment from Dave?

DH: Yes

AWB: Okay, MSFT is the only other ordinary member.

IS: Today there is MSFT, Google and PayPal.

AWB: Okay, PayPal?

BT: I can help.

IS: Usually anyone who is a member is free to join in adhoc groups and normally you ask around in the meeting when you install the group who wants to participate, some people tell you right away, then usually

we set a one week additional deadline to decide who else wants to participate. And then the work of the small group can start.

AWB: I think we need to put together an effort and make some progress

BT: What do you want exactly? Nominations by next meeting?

AWB: Here is what I can imagine at the next meeting, here is what we are recommending as leadership structure, we'll have this or that structure, do we go forward or not?

BT: Okay, a management strawman if you will.

IL: Also, we want roles defined for what people are doing

AWB: Think of yourselves as a school board hiring a superintendent

AK: A self appointed school board

AWB: Okay, that's probably enough for the chair, do you have more IS?

IS: In the last meeting this WH has reported to the GA about what discussion took place in the TC39 meeting on "diversity" of people and group behaviour etc, WH said that maybe there would be further discussions in TC39, maybe there would even be a proposal on a document about the Code of Conduct. TC39 may pick it up or not pick it up, it was about oh, I don't know equality or different communities, about the discussion culture of TC39. I remember I thought a couple of meetings ago that TC39 had a little bit of, rather aggressive discussion culture but I didn't mean it in a negative way. I wanted to say, actually each of the standardization groups, and this is not only true for ECMA but any standards group. Surprisingly each of them have little bit different cultures. You cannot say really which is better or worse but certainly to have an agressive fighting culture which in my opinion TC39 has, someone who comes completely from a different culture might have difficulties with that. I understand those people need some kind of advice or protection or whatever. So those types of things. I don't know WH do you want to say anything re this point?

WH: At the GA I presented what we are doing with our diversity statement and discussions. I also mentioned what happened at the Munich meeting with the bullying which resulted in a member leaving.

BT: We have a concrete proposal, at least the initial parts of the proposal for a document called a Code of Conduct. Are you familiar with these kinds of documents?

IS: A little, I have seen two versions from Allen. One was more complicated than the other. The goals of those documents, yeah, there is nothing to say against it. certainly I think TC39 could accept that and adopt that as a goal that you try to achieve, then install it. I also told Allen that at this point in time this desire only came up in TC39 so there is no such kind of desire at the moment in other TCs but this is nothing bad or good, just a statement of fact. If you feel that something like this is useful within TC39 then I would like to fully encourge you that you should have such kind of document and you should tell newcomers that we are trying to adhere to this document and maybe also to appointment some kind of manager for (me or someone else) if there are difficulties, please contact me and I will try to help or whatever.

BT: That was the question that was coming next which is usually parts of the Code of Conduct speak to how we effectively enforce the sorts of things that the codes of conducnt say we have to do> it's unclear to me how exactly we're to enforce the document, it sounds like one suggestion is sending your complaints to IS and they'll figure itout.

IS: The discussion style can come across as aggressive to newcomers. If we just tell people how TC39 works, in my opinion, yes, the way how it works and the style is acceptable. Sometimes we have aggressive and difficult discussion if we have members for certain cultures, you know then they would really need some kind of help. They are more sensitive

WH: The policy that will be proposed includes enforcement mechanisms with the ability to eject members. How does that jive with ecma rules? Do we have such power?

IS: This has happened once, maybe I don't know 15 years ago, before my time. This happened in TC31. The former Ecma SG just threw him out from the meeting. No we don't have any provisions for this type of thing. We try to talk to people but I have to tell you that also in my practice, at least in ECMA, this did not happen so far. We don't have anything on this in the bylaws or ECMA rules which is formally defined. We can expel under certain circumstances some member companies, if they do something terrible, I don't know. But nothing for individuals in a meeting.

BT: If we adopt a code of conduct... it sounds like you're saying we can't just decide among ourselves that we can eject a dues paying member. If you got an email from someone on the committee saying hey this person has violated the code of conduct, in practice what will you do?

IS: This would be a very sensitive situation, which I would try to avoid. I would just call the guy and I will just speak to him to change his behavior

BT: And if he doesn't?

IS: Then well at least for a specific meeting I can send him out. There are no bylaws for this. I could imagine it could happen but it has never happened before.

AWB: Presumably the next step would be to talk to the member org and say you're sending here a member that is violating the code of conduct can you deal with this

WH: That works for large member organizations, but we also have some small organizations that are members.

TK: Typically, with a Code of Conduct, the committee sets up a group of people, elected by the committee, to investigate complaints and recommend actions.

IS: That's what I'd recommend--better if we start with someone in the meeting. Please go ahead if you think that's a better solution.

BT: Do we need to make changes to the bylaws to be able to enforce the code of conduct?

IS: Whatever you decide within the group, that should be fine; I will just report this to the general assembly. The GA may come back and recommend some changes, but I don't expect that.

BT: To be clear, this means TC39 could disinvite one of its members on its own volition, without more action by the GA

MF: How would you get consensus on ejecting a member with that member included in the consensus decision?

BT: Enforcement would be delegated to a subcommittee

AWB: Should the chairs/vice-chairs be the enforcement subcommittee?

MPT: It's typically a group of people who include minorities, because what looks offensive t one person doesn't look offensive to others. It's completely nessesary, YOu can enforce a code of conduct with a group fo white men, sorry to tell you that but it's true.

JM: Our experiment within the node community absolutely backs it up

AWB: The best solution to that would be to ensure the management committee is diverse

MPT: One problem you have is that if you have the chair and vice chair holding this, that puts too much power in a same place. it can create weird situations of conflict. if anything your code of conduct committee could be outside the tc39 group

AWB: i don't think we'd want to do that

WH: People outside the room don't see the events that happen and cause problems, like what happened in Munich.

MM: That's true, I won't argue that

JSL: So, within, node, as an anecdote. We had an effort where we tried to do some kind of moderation. We had a working group that was answering to our TSC. It didn't work out because there was a degree of mistrust if the policy would be applied equally to leadership. We're moving to a situation right now where the authority of moderation and enforcement is actually being moved out of the TSC and into the foundation. In this way they can take a more objective view. That doesn't exclude people who are a part of the conversation but there will be people who can take a more objective viewpoint to apply the policy equally. If your review committee is made only of the leadership, does that policy apply to leadership?

MM: I very much like this idea of separating the committee from leaders for separation of power. On the thing about outside vs inside, I also want to say that because many of us are, at all meetings or almost all. Having a group such that for any dispute for it is probably the case that many of the people on the committee saw the incident would be great. A lot of these disputes really rely on the types of subjective judgments you could have if you were there. It doesn't mean we can't have outsiders but we want to be confident that there were witnesses.

MPT: A mix seems reasonable. There are three women in the room, this is the first time there are three women in the room. I don't think all three of us would be here.

AWB: one way to address that would be to say the ECMA management team could be an outsider. there certainly are women on the ECMA GA.

WH: Yes, the ECMA GA has women in leadership positions.

-

JSL: You need to be sure that the people who are on that have the mandate to be able to enforce. So, having someone who is already on the management team at ecma is good to find diversity. Actively going out to find individuals in the community is a good idea.

AWB: I'm thinking of this as someone who comes in when there is actually a need for enforcement, distinct from promoting diversity or what-have-you

CM: My experience with these things in other organizations, first of all, having a code of conduct in place drastically reduces the chance that you'll need it and second that it also serves to functions. One as an enforcement function but also a signaling function. it offers people some reassurance that they can participate without problems. Once again that doesn't need to involve exercising the enforcement mechanism

AWB: Yeah, that's my position. You don't ever want to get to enforcement, hopefully you can handle it informally

CM: Yes, hopefully you can set ground rules to set behavior

MS:I agree with chip that some policy should have disciplinary measures. Just like we have the stages document, if we see someone stepping outside the policy and say okay, stop, what's the criteria here? The policy for participation, we would bring it up and say you're out of line. I think it makes sense to have more

formal sub-committee that would be responsible for that. Like Chip says I hope we never have to use it.

BT: I would be very surprised if we use it in the committee. I would not be surprised if we use it on Github, that's the primary venue I think it'll effect.

AWB: You mean non-contributors?

BT: Yes, even non-contributors would be bound.

AWB: We can kick them out any time we want, they have no rights.

BT: WE don't want to start eject people for no reason

KG: Just because we can doesn't mean it's a good idea to do it. Having the COC in place is valuable. It's better to do it via a set of rules.

JHD: constraining ourselves to rules grants folks with no rights some rights,

MS: We warn people once if they've done something egregious

MPT: For what it's worth I work on MSFT in this area and we had one CoC violation reported in 6 months and we have a lot of OSS projects. It's not something that is happening 500 times a day. I wouldn't get too much worked up about how much effort it takes to enforce. If we have a CoC violation we will create a committee that will address this including some people in the room and some minorities as interested. You wouldn't have to have it pre-set, it hardly ever comes up.

WH: It came up this year.

MPT: But when was the last time before that/ it's not like you have to be ready and prepared or people to spend hours and hours of their live on it.

WH: I don't want code of conduct to be used to influence technical discussions. My fear is that it will be used to shut down technical contributions.

JHD: that's exactly what it should be doing, if you can't follow this while making a technical argument you can't participate

CM: What WH is saying is concerned about a CoC argument being used to win a technical argument. It being weaponized. That's something people involved in the process just have to not put up with.

JHD I agree with that sentiment, and I think if and when that happens people see it for what it is

WH: It has come up here before.

AWB: trying to combine threads here, we don't need a standing committee because we don't need a daily or per meeting situation here, this question of how do you raise an issue? anyone can raise an issue. I might suggest one way to approach that is, when that happens, if we have a code of conduct in place. So when a violation happens, the chair group could identify a committee to investigate the issue.

TK: I have a proposal for a code of conduct talking about these details, at https://github.com/tkellen/tc39-code-of-conduct-proposal/blob/master/Conduct.md . Can we identify reviewers?

AWB: I think everyone should be reviewers; this will require a consensus decision of the committee.

TK: How should that work?

WH: I have been reviewing this proposal and making comments. For the most part, I'm OK with it; there are

some wording issues. I have concerns with conflating friendliness with respect. Conflating coworkers with friends seems to me to be a bit creepy.

TK: How about we proceed by opening issues on things like this and working together going up to the next meeting?

WH: One litmus test I use is how it would work against a previous instance of poor conduct resulting in losing a committee member.

DH: That instance had many factors and poor behavior from many participants; I don't think we should be litigating the details of that instance here.

MPT: Not having been there, right now you have a member who seems to be on the verge of tears. Maybe right now this conversation is a violation

MS: What started this discussion is that he isn't sure this would have been effective in Munich.

BT: I think it would have been

MS: To raise that a little higher, any CoC we draft should be useful in situations we actually encounter.

JM: What will make it most effective is our enforcement policy

AWB: I wasn't in Munich and didn't observe what happened. It concerns me if we conflate too much the sorts of things CoC polices are about vs poor meeting management. I don't know exactly what the situation is, but I wonder if that wasn't a failure of the running of the meeting. In many cases well run meetings prevent these things happening.

BT: Specifically I think our failure was that we didn't realize when discussions lost any amount of decorum. We didn't have strong leadership or anyone watching out for these kinds of issues. Had Maggie been there to point this out maybe we wouldn't have gone down that road. This document is encouraging because it actually does say in many words effectively how communication is expected to be handled and I think we didn't follow that in Munich.

AWB: We should all play well together and if things start to break down the first line of defense should be how our meetings work and hopefully some chairpersons could be in place to prevent these things from occurring and escalating to the point where someone is screaming CoC CoC

BT: Talking about any specific actions by individuals is missing the point. We got way too far down the road of emotional discourse and it burned us

JSL: This is my first meeting, i have to imagine this isn't the-first- time it's ever happened.

WH: That's correct.

JSL: Litigating any particular situation that happened at a past meeting is not going to get us anywhere. Let's focus on what we'll do moving forward. The role of the chair is to enforce the code of conduct. Let's make sure that's the way things are run. But we have to focus on what is the policy moving forward. We can't focus on previous instances or we'll get nowhere.

JXF: Would it make sense to have a policy vs in meeting vs online conduct?

JSL: In my experience there isn't enough of a difference.

AWB: We have a proposal here. Normally the process in the ideal world, the proposal is available far enough ahead of a meeting to file comments. I think we're in that situation now where we have something and we should be providing feedback on that proposal for the next few weeks. It sounds like it's far enough ahead

that it's up to the people who are putting the proposal on the table, perhaps at the next meeting we. I would encourage feedback early rather than late.

TK: I would agree

MS: So everyone is charged with looking at and reviewing this document for march.

AWB: Yes, for consideration in March. Don't wait until March.

KG: Before we move on, we were talking about the possibility of having no standing committee. This proposal states we'll have standing committee. Should we discuss that now? I'm not committed to having that discussion.

MF: I think we should also outline the specific goals we're trying to achieve with this so we can evaluate if we are accomplishing those goals.

AWB: I would hope the goals would be stated in the introduction.

TK: The document has six headings which describe the basic thrust. It came from the JSFoundation, and that is based on the Django CoC and other documents. It comes from pretty diverse perspectives, and so I think it's a good starting point.

BT: One often controversial aspect is that we are bound by this Code of Conduct outside meetings in public spaces where we are wearing our TC39 hat; I know this is controversial.

AWB: What does that mean?

BT: For example, say you are invited to give a talk about TC39--you are bound by the CoC.

JSL: This has come up in the Node community, where people have used their Twitter accounts in unproductive ways while acting in a Node-related capacity.

WH: What about the case where you have a personal Twitter account on which you do nasty things unrelated to TC39, and an unrelated third party then publicizes the link between that account and you? Does that turn into a TC39 CoC violation for you?

?: No, because you didn't represent your personal Twitter account as related to TC39.

WH: Are we all agreed on that interpretation?

KG: It would be nice to be more specific about when it applies and when it doesn't; I'll file an issue for this.

WH: Here's a real-world instance of this related to TC39: A long time ago Brendan had made some purely personal political contributions. Pretty much no one knew about them until someone outed them. Would such personal political activity be regulated by a CoC?

AWB: Istvan, anything else?

IS: I'd like to encourage participation on GitHub between this meeting and next one. I'd be interested in seeing this document as well.

AWB: If you think there's anyone on the executive committee or in the leadership who would be interested in giving feedback, that would be great.

IS: We will discuss this at the next executive committee meeting in April. Generally, this is rather remote to them which is basically isolated to TC39.

AWB: I think this is likely to come up in other groups too.

IS: At the beginning, do not expect too much. There are a number of cases where we are flexible to account for TC39's further advances, e.g., the HTML specification as the authoritative copy.

Conclusion/Resolution

- Everyone in the committee is charged with reviewing the code of conduct proposal at https://github.com/tkellen/tc39-code-of-conduct-proposal/blob/master/Conduct.md by the March meeting

11. Test262 Status Updates

DE: You should share what Bocoup is doing with Google

TK: Several of my coworkers at Bocoup are responsible for a large portion of the test suite. One thing we're trying to do is to provide long-term governance and support for tests. To get proposals advanced, we need tests, but we are seeing some things not come through with tests. We have an arrangement with Google to write more tests and some sponsorship; if other implementers find it useful, then you can join in on this.

DE: Actually you've also been continuing to identify and fix coverage holes as well.

BT: For example, the double super call test was missing

BF: The module tests that Bocoup wrote is awesome

SYG: Are you importing V8 tests?

AK: In the past, Bocoup has imported V8 tests, but later wrote new tests

SYG: Would you be interested in fuzzer-generated tests that might be hard to read?

DE: Fuzzer tests often can be reduced to something readable and intelligible, at least the ones that have come up for V8. In my opinion, we should be relatively liberal for what we put into test262.

AWB: I could imagine a failure mode for the tests where they become difficult to maintain because of too many tests for obscure optimizations

KM: For JSC, we try to wire in all the tests from all the other browsers to find any bugs. It can be really useful.

DE: web-platform-tests is very liberal in what it accepts, and it's more like an upstreamed version of what KM describes JSC has; this is way on the other end of the spectrum from test262's review, and it provides some value, but I understand if we don't go that far.

SYG: Could we use wpt more for testing JavaScript implementations?

TK: I'm not aware of any plans for this, but if implementors want to support work for this we would love to do it!

KG: Embedders come up for things like detaching an arraybuffer

SYG: Event loops are more complicated, and so it's even more relevant there, whereas the harness is not as taxing

DH: Maybe this is a heuristic signal that this is something missing in the ECMAScript standard.

KG: Announcement: when you pull in the latest test262 tests, you'll find that $ changed to $262 for harness

hooks

BT: There are new tests for SharedArrayBuffer, requiring extensive API surface to run the tests; you have to be able to create an agent, sleep an agent, etc. Expect to do some implementation work in your console hosts.

13.ii.a Proposed Grammar change to ES Modules

(Bradley Farias)

[_Unambiguous Grammar Javascript_
proposal](https://github.com/bmeck/UnambiguousJavaScriptGrammar)

BF: So, at the last meeting, Dave was talking about use module, and this probably that node has with detecting if a given source text is in the "script" grammar or the "module" grammar. There are certain situations where it can be ambiguous. These are somewhat rare but they do exist in the wild. We are seeking, either to disambiguate source text, or use a new file extension for loading ES modules currently in node. The vote to accept file extension as a possibility was last march or april. The vote to seek a grammar change was last august. The grammar change is a little simple, but it's not syntactic. [explanation of readme of repo]. Node would like to have a ECMA-262 blessed way of detecting the difference. The web is not supporting interop by using an attribute on a script tag. Node must support interop. I'm asking if it's feasible to ask for a parsing change.

AWB: One primary concern about this is that it precludes the eventual replacement or migration to a module-only world.

BF: Correct. There is always the possibility of someone accidentally deleting an import/export statement and then discovering this behavior change.

AWB: It adds friction in places of modules in place of scripts.

DH: I don't think module adoption would fail because of this. It's a big loss of ergonomics that you there are just realistic use cases that netiher use imports or exports and they require this boilerplate. Removing this bilerplate causes the other parse goal to be used. Not saying that node to get to the place where there is no script target, but you could operate as a node user without knowing what non-module content is.

JHD: You mentioned some actual use cases for a module with no imports or exports, what ar ethey?

DH: Some of them are just like, I have a module that serves only as a polyfill on standard globals. When you import it, you're just looking for side effects. Maybe it's not a polyfill, maybe it sets up the state of the DOM for your app. Basically, you're importing it for side effects, not bindings. There is also a usual workflow of programming story, it's not about here is the final artifact of what I'm delivering. The practice of programming is you open up an empty buffer and start typing. In this world, starting with an empty buffer means you writing in a different programming context until you type the boilerplate. Also, you're editing your program and you decide you don't need that last dependencies. You remove an export or import and suddenly the program operates drastically different. There are various ways in which you could unwittingly change the type of document you're working with. It's so far from what you're trying to do that it won't be top of mind.

JHD: I have a clarifiying question, if that's okay. In this world, without this change, if you type import, what happens? It would still switch into a module. It's dependent on the host environment, it culd ask you what

DH: The changing of the contents of the source file don't change which file type/name you're writing to; this is completely managed out of band.

JHD: Out of band you still say this is a module. In the use case you're talking about I agree there are potentially two sources of truth. In all those cases though, in order for usability the environment will make a

choice.

DH: One difficulty here is that it's not my call to make a design for Node. We wrote that proposal that didn't go very well of what it might look like to manage by out of band information. It seemed like we got most of the way there, but there were a couple places where it wasn't possible to signal; you'd need command-line flags or something. My goal would be to find a design that's close to that design as possible.

DE: To clarify, what are you proposing?

DH: I would offer several proposals, the ideal state in my mind is that node will design something where you can completely manage it out of band. Failing that, actually, orthogonally to that, it would be nice to signal in source that I really do intend this to be a module. In those cases, I offer the proposal of use module because it more clearly states intent. Export curly looks like boilerplate, "use module" communicates intentions more clearly. It's up to node, not me, to decide if they want to mandate "use module". Node could decide that you need "use module" or one import or export, or a combination. Or they could have what I am proposing is zero changes, manage it out of band. File extension is "out of band" but it's not my call.

BF: To clarify, you think file extension is not out of band?

DH: It's out of band, but I don't like that solution.

M: I struggle from the standpoint of saying you have a module that is side effect only. I would never allow that through a code review. I would never allow this. If you're scaffolding an express app, and you find that adding an 'import' statement makes these non-local changes to whether it's a module...

WH: Modules can be used for small top-level things such as a hello world program.

JHD: A poylfill would never use module code to do it, no browsers would support. For the next decade no polyfill will be written in module code.

BF: Would any of your polyfills break if they were swapped int module grammar?

JHD: The ones that were written in ES5 would break, they are intentioally in sloppy mode. The ones that are ES6 and later tend to run in strict mode and would not break/

JL: Anything that is declaring a global to communicat between scripts will break if they are modules.

AK: Jordan your polyfills are extensive. If you have a targetted one, that is ... for instance Polymer is targeting very spceific APIs. They know they are focused on some baseline of browsers. I don't think it's crazy to say modules in the near term woud be use for polyfills

WH: Two things:
1.When we adopted modules in the first place we had a big conversation about having an introductory sigil. We decided that no, we shouldn't. We are revisiting past decisions.
2. If we have 10,000 line program, having an import or export at line 3042 shouldn't change what it is, it should be obvious at the top.

JL: I just want to say, from my experience with our internal migrations from different module systems, there is a LOT of confusion when you have to start loading module code in a different way. If you accidentally load a module as a script and it just works, it's going to be a big user issue, more than oh, i'm ediing this module.

BF: are you against changing the grammar?

JL: no

AWB: For a ecma262 perspective, it's a false dichotmomy to say we have scripts and modules. It's higher than that. We have ESScripts, ESModules and CJS modules. Those are all source files that have distinct

syntax and semantics

BF: I would agree

AWB: Distinguishing all three is useful to a tool. It would be wrong for ecma262 to only address one of those vectors, the difference between scripts and modules. I haven't heard any discussion to how to tackle the broader issue other than the fact that external metadata doesn't. Any solution that is strictly saying this is about distinguishing for the purposes of node distinguishing modules from scripts is not a complete solution and we shouldn't adopt it because that. I agree with Dave that it would be perfectly fine for node from a standards perspective to say is the way We distinguish for modules is import/export boilerplate. I don't think that will cause issues cross-platform. I don't think we should be addressing this here. It's implementation or platform level concern.

BM: I'm guessing you're still against

AWB: Absolutely

JHD: Clarify, are you against ANY change that would make scripts and modules not overlap?

AWB: Yes, unless it also addressed CJS modules and made them not overlap

JHD: if a solution that was presented that was not this that allowed cjs/scripts/modules to be disambiguated with parsing only. Does anyone reject that out of hand?

WH: I don't know what that means

BF: I'm skeptical

DH: The idea of the pragma would be that in the core definition of the language it is an early error to try to parse a script with a "use module" pragma. It becomes a way that programmers can use this defensively or specific platform like node can decide to mandate in the cases they think it is good to mandate. Both of those are outside of the spec. The spec just says if there is a pragma it's an early error to parse as a script. I want to give an early argument for a complete distinction between cjs/modules/scripts., i don't think it's a universally bad practice to have a module with only side effects. You shouldn't do it willy nilly but in particular there are certain kinds of things that you would do at the top level of your app and they are standard use case for that (polyfills for example). Something setting up your ambient state, you don't want it happening everywhere, but if you keep it all in one place for you rapp that's legit and those use cases happen. If we just reject it because we don't think it's valid....

MM: It's easy enough to export the function and just invoke it

BF: there are other usecases for a module that sets up state, there is nothing to export, we're out of time and it looks like there are people vehemetly opposed to this currently.

DH: The point I want to make to JHD is that we also have set up a world where your top level script are modules. LIke, the top of your app is a script. In those cases, you defienitely, it doesn't make sense to have exports

JHD: You'd like have imports though

DH: right, but one of the things we've been moving through the stages is dynamic imports, it makes sense to have those, and no other type of import, for some code. We also allow that in scripts. That makes it even more plausible that it does lots of importing but not top level importing unless we force those things to say export nothing. That to me says you suggest that using modules as the new script is a legit new thing and therefore wants to have the same "lightness" as script modules have today. If we have to add anything, we're adding a boilerplate.

BF: Just to be clear, i'm going to mark this as being rejected and take it to node CTC and fall back to a file extension under the presumed .mjs and this will be dsicussed at the next ctc meeting next wednesday. if you wish to join feel free.

MF: Just to clarify, you weren't looking to move this through as a staged proposal?

BF: people are already shipping technical previews, it would need to be fast-forwarded

MF: you said something was rejected, I don't think that is the case

BF: No, but node CTC's request of inquiry about this, we're not going to make a grammar change. People of TC39 are opposed to a grammar change

CP: We are opposed to THIS one, we are open to see other proposals.

JHD: you mentioned a phrase as a tax on modules that don't have exports or top level imports. It sounds like if someone came with a proposal that didn't impose an ergo cost nobody would be opposed but we have no concept of that.

DH: The fact that I presented this last month might have screwed us over. What really i as trying to present last month was to get started a conversation we'd start here. This is a strictly better option for you than a grammar change. you in node can do...it's effectively the same thing as export {}, just use the "use module" pragma.

BF: This becomes even more questionable and this has been discussed before at node ctc, we want a uniform way to manage this that moves from two or three ways of doing it.

CP: You could mandate "use module"

BF: But that is secondary citizenship to the web.

JSL: We have users who want to use that on the same code on Node and web; if the web doesn't require it, it seems strange to require it in Node.

JHD: In the same way one browsers doesn't want to break a page in one browser, node doesn't want to break compat. with code that works in browsers

JSL: We don't want people to have to include this

AK: This is the opposite of this, it's saying use module in node but it will still work in the web

DH: he's saying there will be the whole problem, there is a lot of web code that won't be able to work in node without it

JSL: all that content would have to have use module added to it otherwise, and we don't want to require that to the rest of the world

AK: we agree there is an ergonomic problem

AWB: We are not rejecting this, we just don't have consensus.

JSL: What I would reiterate like Bradley said: If we cannot have an unambiguous way of determining this, file extension option is the only one that we have identified as being, the least bad option.

DH: I really think you've defined the problem, uh, in a way that's unnecessarily limited your options. These aren't the only two options. It's not my job or right to tell you what to do, but I just don't buy that dichotomy. I think your making a mistake to chose the file extension route. I would certainly be happy to work with you

guys on this

JSL: My intuition is that there is a path forward that is reasonable, we just haven't identified it yet. We don't know.

AWB: Just to be clear, the idea that there is a file extension being problematic, that's not a TC39 consensus either.

DH: It's just not our call

AWB: It's not even... we don't deal with files.

BF: I will say if time goes on eventually there will be a true cutoff and we ship something and that will be it. Whatever we do, we do need it to be the same on the web. WE can't have the web not require something that is required in node. I can have myscript.php

AK: The web's not going to require file extensions

BT: Wouldn't it be incompatible, you'll have modules

AWB: Filenames are out of band

KM: What's the difference in effort between adding "use module" and renaming to .mjs?

MPT: As a library author, I don't care either way

DH: I felt like the proposal I had last meeting was promising; I think it deserves another conversation

BF: Let's do that offline.

CP: And let's also explore the pragma

BF: There's a practical time limit if browsers start shipping something that we can't, and shipping seems imminent

JSL: We are having increasing pressure from users for delivering something with modules.

ZB: Given that we have browsers here, we could get agreement to hold off in shipping to make this change first.

TK: Is Node's only objection to "use module" that the web doesn't require it? If the web required it, then would it work for Node?

??: I would not be in favor of that.

JSL: Let's take some more time later to work through this issue.

TK: If we can't work out a solution, after good faith effort, could we agree to live with the pain of "use module", requiring it on the web and add it to the specification?

DE: Seems like we've already heard that the answer is "no" from this discussion.

JSL: If we can't come to an answer for in-band, we will need to put something out-of-band.

MPT: Multiple extensions

Conclusion/Resolution

- Some members of the committee are strongly opposed to this particular syntax change
- There is also some interest in various strategies for disambiguating
- We do not have consensus to move forward rapidly on this proposal

15.iv Progress report and request for comments on 64-bit int support

(Brendan Eich)

Int64

BE: Last meeting I presented Int64 Uint64, in a heck of a draft, trying to show how to do them in a future proof way, as new values types in the language. They got to stage 1. Since then, I think Benedict and LittleDan have talked to me on Twitter and privately about an alternative which I think should be considered. Which is, not to do Int64 or Uint64 now or even try to get them on the agenda in anyway. Instead, we should do BigInt/BigNum. The argument they make is that is what crypto libraries want and it's more ideal as a machine type. If they could be optimized enough they could resolve the use cases for Int64 Uint64. I want Dan to confirm this?

DE: V8 already does similar optimizations, sometimes we tell that something is a smi (small integer) and doesn't need to made into a double. I think determining whether or not BigInt vs Int64 is the first thing we should add is an empirical question. if there is a built-in bigint it can be optimized much better than something implemented as a library. You have a machine-dependent scalar types that should be used in the implementation. You can do compiler analysis on lowering the BigInt which would be not possible for one that is as a library. There are a few questions that could determine which is higher priority: one is, does WASM subsume some of these optimized use cases? Leaving that aside. Starting closer, if you put an explicit mask after every operation, when you want it to be in Int64 mode, then the compiler should be able to figure it out for optimized code. Do we need a lot of ergonomics for the Int64 cases or do we need more for BigInt?

BE: Let me restate this again because I think there is a lot there. Dan is talking about the use cases for new value types competing for optimization effort in the edges. If you optimize BigInt enough, you can indeed sometimes use machine registers, to hold some of the BigInts that are floating around. If you do that and have only bigints you still have a good match with crypto use cases that want it. If you do what I proposed to add Int64 and Uint64 now and optimize them as well and then ave BigNum library on top of them, BigNum won't get opimized as well. There are needs for Int64 and Unit64, OSes have those offsets and sizing, its not all about the crypto libraries but there is no cry from the Node.js community for BigInt like there is for Int64. The standing widespread request doing server programming in code is for Int64 to match system calls.

BT: Why BigInt vs BigDecimal if we're looking for one type?

ALL: laughing

BI: long history here, mike from IBM and Sam Ruby spent some time on this in the early harmony era.

WH: What do you mean by big decimal? BigInts are base-invariant.

BT: I mean arbitrary precision large numbers with arbitrary numbers to decimals. I claim nothing about a specific spec I'm talking about a sepc that can represent something other than numbers.

BE: I want to limit the scope to the discussion of BigInt going forward. There has been a fair argument that if you could have BigInts you can optimized well enough without Int64 and Unit64.

AK: I'm hearing something that is slightly pejorative, you have to do the same optimization for Int64 as for BigInt.

BE: I'm trying to say that either approach can be optimized to avoid boxing. The question is, do you have to do two anyway? Is there enough ground up demand for Int64 in TypedArrays, WebGL, because of Node, which is the #1 driver, you might need to do both anyway.

AK: Question about the node... if you have a 64 bit file descriptor, what's the problem with storing that as a BigInt? What goes wrong?

WH: The thing that goes wrong is when you introduce types of things and you have a function which expects and Int64 and you pass in a BigNum; the type restricts to only certain values. As an example, you can expose a function that contains a check, is this a value within a certain range? The check can pass.

AK: this seems like a weird place to be the one place where we inject types?

BE: We have them already: TypedArrays for one

AK: you can't ask for those in a function signature. People today expect to get an int32 in their function and they might be surprised if they don't get it

BE: When people want float32, they might want this because of TypedArrays. We might have people deoptimizing these paths if they typically take one but are passed another. plain javascript always has to treat numbers as plain doubles, without more speciailization

BF: We have plenty of random hacks where we have arrays of two numbers for things that need to pass around a 64-bit integer

BB: The issue, we kind of get around with sort of, using doubles for the most part, file offsets, very rarely are they out of 2^50. We have to check if it's an int unless it is out of that range, etc...

BE: There is a legit use cases for 64 bit ints. It cannot be ruled out. you have to say, the tradeoffs look like BigInts could absorb those use cases closely enough.

AK: I also want to point out that just as an implementor that's not my only concern. V8 engineers have been pointing out that BigInt should be better for users as well. As a user, if we add all these different types and you have multiple 0s that don't equal each other.

BE: I thought we talked about value types a ton, we're going to have a number of them and they'll have 0s.

WH: IEEE Decimal has a *lot* of different zeros.

BE: I think we all agree they should be ==, ===, etc. We already have IEEE double, so we're talking about going to the next island of safety which is bigint, or should we do both?

MS: If we add these other types we're talking about. I could see 8, 16, 32, 64, 128, signed, unsigned and none of them compare to each other.

BE: I'm glad they don't. That is something Dan and I worked out.

DE: Yes, we've gotten positive feedback from other members of the team about no interoperation

AK: Yes but at least BigInts of the same size would be comparable, right?

BE: No implicit conversion.

AK: The hope that would be this island of BigInts that are compatible with each other at least

DE: There are two parts to no implicit conversion; + would throw but === would return false

BE: There are fine points here, whether or we have explicit or implicit conversions, the BigInt vs Int64 step is still a question.

AK: Bigints would allow you have some semblance of understandability between these different types

BE: you mean you'd be absorbing more use cases with a single type?

AK: Yes, while absorbing those you'd be in a better state than having 8 types

BE: After going through SIMD which is stalled at stage 3.... nobody is here to impose all the ISO C++ types, today. But, we have a problem. The twitpocalpse three, there are latent bugs

MS: Besides file offset, getting values back from stat.. why can't a BigInt handle that? You know you're going to get back a value you can do logical operations on to see which bits are set.

??: I don't think anyone is saying BigInts won't work for node. I think BE is saying we have a use case for smaller integers.

BE: Lets argue apples to apples, if the optimizations for BigInt don't require boxes, there will be slow paths for network i/o

DE: stat is a system call it's going to be more expensive than allocating

AK: it's not significantly harder to optimize a BigInt vs an Int64 that take 64 bits of space.

BE: I think there are always optimizations available. We're talking about doing one or the other or both (which I don't propose personally). I'd like to get a sense of the committee on BigInts as the better evolutionary hop. Who thinks we should do Int64 Uint64 in preference to BigInt.

BF: We just want something bigger; these alternatives don't matter to us

BE: Who wants Bigint?

(the straw poll was stalled in the middie due to objection to the idea of a straw poll, but most implementers seemed to prefer BigInt)

- both: 2 votes
- bigint: 9 votes
- int64:
- either:

BB: The cases where we need 64-bit ints are not in the hotpath. stat is expensive enough that boxing it is not in the hotpath.

BE: I'm not going to champion BigInt. I sunk the cost in Int64, Uint64. My time is limited. Someone else needs to do this. This is why we're five years down the road.

AK: V8 cares very much about the use cases for 64 bit.

??: If the choice is between like, okay, we have nearing completion for 64bit. If we have to start from scratch we might start over again.

BE: That's exactly the issue. I'm talking about who is championing it. Are you gonna do it?

AK: Depending on what the outcome of this discussion is.

BE: I don't have time for it, it's actually more work in my opinion.

WH: This is like 4th or 5th time we've tried to do this in TC39. I first proposed Int64/Uint64 in about the year 2000. We generalized it to BigInt. We generalized it to value types. At that point it became complicated enough that nobody wanted to deal with it. So we dropped it. Next we proposed Int64/Uint64. We generalized it to BigInt. We generalized it to value types. That had too many dependencies and we dropped it.... It's now 17 years later. We proposed int64/uint64....

AK: I haven't been at this as long as you. JS as a language that is moving forward is doing pretty well these days.

BE: I hear it's about to be replaced by WASM. I heard Dart was going to do that too, through. We need a champion.

BE: 64bit is easier. That's why I'm able to champion it, but not able to champion bignum.

BT: We could break the cycle right now by forming a consensus around the specifying some big number type before any other

BE: I tried that last time. That's a given, but that doesn't help We're facing a crucial issue, 64ints as the next step or BigInts?

BE: The big breakthrough was finding no implicit coercions. BigInts are just hard, they have arbitrary precision.

SYG: Do you need to change your value representation from one to the other?

AK: This seems to the V8 folks

SYG: outside of optimization work, it seems like it is just as hard or easy

AK: it's not an implementation concern, it's about users

BE: I heard about optimization early on on twitter, that's good to hear today. that is' really about language

AK: it's big, i can't speak to the twitter conversations, the bigger issue is one of feeling that this fits better with then language, not doing scope creep helps somewhat.

BE: Everyone wants the ideal but who is gonna do the work? That is the concrete consideration that will guide our decision. Plato vs Aristotle, in the real world obviously bigints.

AK: Do you think it's impossible to specify BigInts

?? If nobody is going to do it we'll be here 5 years form now. I feel this very much too. We should move froward with a good proposal, I think int64 was, or, you know, very soon find someone who is going to do the actual work instead of saying oh yeah that would be much better. If your team is willing to put up a concrete proposal I would be okay with that.

CM: To make this a little more concrete. Is there any reason not to proceed with the 64bit proposal until we have a solid bigint proposal?

BE: No, I think the committee today should decide to go forward on 64bits with bigints with a champion for it

DE: Give potential champions some time...

MM: No matter what we decide it's certainly the case that BE can keep working on it

BE: I'm not going to proceed if we have a split committee

MM: What's the MVP?

ALL: 64int

MM: The use case here is obscure. It's valid, it's very valid. No matter how you slice the 64bit integer, it's an obscure use case for most of the community. Nobody is making a rallying cry for it.

DE: Neither

JSL: Node would use whatever works. Our natural preference would be whatever requires the least overhead.

BE: I think it's quite true, they don't know what trouble they are getting into until they hit a binary rounding problem and then they blame me.

MM: I people are clamoring for this, just ship an MVP

DE: One thing you want is an MVP that evolves into what you want; I'm not sure 64-bit ints have that evolution path. I think if we can give until the next meeting ot find a new champion,

BE: I do want to work on it but I don't have time for it.

??: If we aren't adding 64bit integrates, we're adding infinitely big integers.

WH: That's just bigints.

BE: The spec has to describe how it works

WH: A lot of lisps have that, it works fine.

BE: You cannot parameterize division by the exponent, it doesn't work

??: Ok well

BE: Someone has to write down how operators work and it depends on the type

WH: For 64bits or bignum

BE: I'm saying there is work, it's more work for bigint. If you were writing the spec would you say it would be the same effort?

WH: Yes; in fact, the effort to write a spec is a bit smaller for bigints than for int64/uint64.

BB: That's how I think too, your'e still talking about how division works with integers

MM: There is still a mathematical concept that bignums reflect clearly. 64 bit signed integers do not. I hope we don't do 64 bit signed integers, but I hope if we do, despite the runtime costs, i would have it error rather than silently wrap when it overflows.

BE: I still have this concern having done some of the work, bigints are not just a generalization of signed 64 bit ints.

WH: In the spec language they are, it's a very simple spec. If you look at all the basic arithmetic operations, bitwise logical operations, shifts, they all have obvious definitions for bignums.

??: Maybe bitwise binary operations are

WH: Bitwise operations are merely operations on half-infinite strings of bits in infinite 2's complement notation. They all just work. (A positive number conceptually has infinitely many leading 0's. A negative number conceptually has infinitely many leading 1's. Only finitely many least significant bits need be represented for any particular number.)

BE: It's not bitwise I'm worried about. What about divide by zero?

WH: That doesn't change between 64 bit and bignums. What happens when you divide by zero with uint64's?

BE: it does because 64 bit hardware exists on many computers that run JavaScript

MM: What does the hardware do?

??: It wraps, usually

MM: presumably your'e going to throw an exception.

BE: maybe i should reconsider if it would be more work. let's not worry about the champion right now. i just want to confirm we don't have a split committee.

AWB: There are two aspects to this, one is the spec time, the other is time to implement. Particularly, time to implement across all the major implementations. Do people actually think they can manage to implement BigInts? What are the differences?

FH: Shouldn't we value users over implementability?

AWB: Where there is a usecase for this, and there are usecases, it's becoming a relatively critical need, this is not something we should be looking for 5 years from now. If we said it was bignums, when can we reasonably expect that implementations would have produced viable performant implementations in browsers?

AK: From the V8 side, we got momentum internally on BigInt because the effrot required seems similar for what people would expect for Int64.

AWB: What do other engines think?

MS: Same for JSC

BE: Just to be clear, do you mean expectations for users on int32...

AK: I don't know what the details of the staging plan...

BE: Does that bring up signed vs unsigned?

DE: bigints are always signed

??: It's only interesting when shifting

BT: Do you think a general value types is in the cards for JS?

BE: When we got IBM to back off on decimal as a hardcode extension. WE used the same argument that WH cited, value types. Decimal is still out there. I've talked to some people lately that have a need for decimals.

BE: Value types are still on our agenda and we've done some work that everyone agrees with that is constructive. There are all sorts of possible uses, like for the CSSOM; also custom literal suffixes, etc.

BE: Let's talk about this constant rumor that WASM will replace JS. It may happen but not for a long time. We'll have pressure for new value types. Everyone will be happy in number-land until they aren't, and then maybe they'll use bigints when they are available? To me, it almost doesn't matter. We've done enough work on Value types, pressure from the edges is big enough. I'd like use to get there so we don't have to keep hard coding. When I extended the spec, I future-proofed a meta protocol for value types. Realm specific auto wrapping of primitives in objects, that's important. I'm not going to summarize all the learnings we've had. Its useful and could be fruitful. Having been around this current wheel so long it's not worth worrying about. If we need it we can do it.

MM: I've been doing this now for 8 years, which I know is nothing compared to you. Leaving the door open for something to be done cleanly at such a time in the future that someone has the energy to do it. To me, that is a big deal. The difference between decimal without value types in the old days vs your proposal now: You've done the work to understand how value types could grow in such a way that what you are proposing now will be retroactively rationalized into the value type proposal. We don't know if it'll turn out that way but you've paved the way. The idea of holding the door open for something often works. If you're going to stay on the standards committee you have to play the long game.

BE: Good point. THere is some pressure with game developers with mutable vectors and matrices for operators. That's separate proposal. The committee could do operators, literals and value types.

DE: Literals could be done totally separately. CSSOM could actually use suffixes to construct objects

BE: Tab wants implicit conversions.

DE: There are all sorts of intermediate things.

BE: Maybe bigints are no more work. I'll go weigh that. I think we still have people raising our hands on both sides of this.

WH: I'd be happy with either one but want to see it happen now.

AWB: What does it take to get this in ES2018.

BE: If we're being real, I think we should talk about WASM.

MM: I want bigints a LOT more than I want 64. I would really like to see... if we had them, we would never bother with int64 and that would be fine. If we had int64 we'd never do bigint and i think that would be terrible.

BE: We have agreed here that JS isn't a language where we're trying to limit types. We have typed arrays, even with WASM coming up... it might accelerate. We might need 64 bits soon.

WH: Are there implementation concerns with bigints? If there are any, please state them sooner rather than later. We can throw the switch and go the bigint route but I'm concerned about it getting shut down nine months from now due to engines' efficiency concerns, and then we'd be back to square zero.

AK: I wanted to find out what the temp from the committee was before I did any work on it. it's clearly warm and that is enough fuel to do more work. Can we say people who are interested in bigint can look at this for the next two months and we'll talk about it then?

AWB: If you add two months, you've now taken out a major chunk...

AK: I object that if you don't say bigint now....

BE: You need to be de

BF: I will be on the hook for spec work if we agree that if I come next time we're not going to complain about implementation problems. I don't have that issue. I don't think in two months I can get up to feel speed on the entire history of this thing. I do not expect it to go to stage 2 in that meeting.

AK: Michael and I were having some backchatter, and might be interested...

BE: I actually want someone on an engine team to take it. Someone needs to actually commit though and say they'll do it.

BE: Dan would be ideal

DE: I'd be happy to do it, unless AK and MS want to

BE: I think we need to be decisive.

DE: I can sign up to be the champion for this, if that seems...

BE: Until someone says I'm doing it, it won't happen

WH: I'd be happy to review it.

AWB: And the stage one item we have transforms into a bigint approach.

BE: You did an earlier version that I wasn't aware of.

DE: I did some earlier work to figure out the semantics for operations for Int64 in the past. Stuff like division is not totally simple.

BE: I don't think we have marks glitch with overflows.

DE: That's a legit design point we can consider. I'm not going to rule that out

MM: I would be amazed if what happened was we got int64 first, with overflow throw.

BE: Fair enough

MM: I think of all the options, that's the least likely

WH: int64 with overflows throwing is a really bad idea.

BE: Right; so we don't even need to talk about it.

DE: That's a very awkward path to bigints

BE: Okay so, the other thing that needs to be done, and this is where TC39 needs to take the hit collectively (used to be mozilla).

BE: a bunch of people are going to say you bozos why are you not doing int64

DE: Okay, the job for me will be to write an explainer giving that. They'll probably say the same about division. (Explainer: https://github.com/littledan/proposal-bigint)

BE: You can write an explainer but there will be a reaction.

AK : We can't be making decisions about what people say on twitter, can we?

BE: PR is a part of the job, we do need to do it well.

DE: It's hard to communicate through twitter, explainer docs and threads on github are a better medium for the people who are really engaged.

??: One question about this proposal. Whether it is bigint or int64

etherpad went down for approx 5 minutes

Conclusion/Resolution

- DE will champion BigInt

13.i.d Seeking stage -0.0 for IEEE-754 sign bit

(JF Bastien)

[presentation](http://jfbastien.github.io/papers/Math.signbit.html)

(Presenting link)

JFB: Math.sign doesn't let you differentiate 0 and -0; it has five possible outputs. I want to be able to tell about -0 and +0 in an intuitive way.

MM: Just to clarify, you're not proposing to change Math.sign and the operation is named specifically

JFB: Correct, I'm boring from IEEE 754 that has an concept called sign bit. They don't affect the number in any way. If you look at C++ or Go, there are also functions that do exactly this. It's shift and mask, just give me the bit that indicates sign.

??: What does signbit do with respect to NaN's in IEEE 754?

WH: It's complicated.

MM: It matters in the sense that we have to specify something.

JFB: As a user there is no concept of signedness. Although C++ allows you to observe the sign of NaN, sorta, it's not consistently exposed; Go is not mentioned in the standard. We have to make sure that we continue to work well in both NaN-canonicalizing and non-canonicalizing implementations. This proposal just chooses something

WH: Why are we doing this? What is the impetus? Why do we care about the difference between -0 and +0?

JFB: It's very unintuitive?

WH: Why would you care to distinguish ±0?

JFB: It's not an open question, I want to distinguish it, done.

WH: Okay, so do 1/x. Done. If you like, write a function that does it.

JFB: That's hostile to a user.

AWB: Which user? Who are we protecting?

MM: Just to flesh out the argument these guys are getting at. In order for a user to be in a position to care, they are probably enough of a specialist, enough of an expert about what floating point numbers mean that the reciprocal thing is not hostile to them. The people to which it's hostile, have no need for this.

JFB: They have math.sign for that. The bar was high enough that Math.sign was a good thing for TC39

MM: Math.sign is understandable in terms of the mathematical numbers that are represented. Both zero and minus zero are neither positive or negative. They are mathematically zero.

JFB: Yes, but how you got to the zero matters. In some cases it matters to differentiate how you got there.

MM: I understand that it EXISTS, the point i'm trying to elaborate, there are users that care about floating point numbers enough such that the difference between 0 and -0 is significant to them. THat's a very small population. THere is a very large population where Math.sign mapping 0 to 0 is fine.

JFB: That's fine.

MM: The point is that the user hostility argument fails because the users that would care are ones that would find the reciprocal not hostile

ZB: A real, recent case comes from my field of localization. One of the formatters can have a zero for a unit, year/month In this case the unit we may want to say zero seconds ago or zero seconds. The way we'd distinguish that we use 0 vs -0. 0 means "in 0 seconds" and -0 means "zero seconds ago"

WH: That is user hostile

ZB: I still thing it is esoteric, I just wanted to provide an example

AWB: If there is a distinguishable feature in the language people will find a way to to misuse it

MM: I do not think this is positive argument in favor of the proposal.

ZB: I'm not saying that, I'm just providing an example

JHD: So you're talking about the usability of diving by infinity. At this point you could say (describes a way to do this with existing apis). The usability of Math.sign(x) is vastly superior to that form. That's the burden of proof I would see.

JFB: Let me make another argument, when you're doing something math based things, why are you talking about objects. Sign bit is a thing for a person with a math background.

JHD: In that problem domain it makes sense, sure

JFB: It's totally a niche thing, but it's trivial

AWB: We have a complexity budget. Yes it would be slightly better for a small number of users but it is an addition to the spec, our workload, an implementation, etc. Anyone can trivially define a one-line function to do this. What is the overriding... we don't have an infinite amount of complexity we can add. Someday we need to stop

JFB: Sure, but you already have this complexity. IF you are going to quack like a duck quack like a duck.

JL: Are you proposing to add a series of operators.

JFB: I'm feeling the waters... can we add this or should we each roll our own?

JL: I would be inclined to see a series of additions rather than single function

JFB: 23 mathematical operations were proposed for C++ 20 years ago, and they were only just now added to C++. It's hard to get these right, and it takes a long time.

JHD: So you're seeking stage....

JFB: Stage -0!

All; ha

JHD: Realistically this is either stage one or the committee decided that this will never happen. Are people saying they will never allow it?

WH: I'm willing to be convinced. But signbit is not what IEEE 754 specs. Instead, they have a copysign function.

JHD: It's in the proposal.

WH: [looks again at proposal] It isn't. The proposal isn't proposing what IEEE 754 specs.

JHD: The timebox is about to run out. If nobody opposes it being stage one, let's do that.

AWD: Is Rick's proposal is in stage one?

JHD: Let's investigate if they should be separate

AWD: I suggest merging this into that proposal

DE: The issue with Rick's proposal the committee was unsure about the motivation. Putting them together just makes something less convincing.

JHD: the question of it being merged or not is a stage one question

AWD: How many proposals we have floating around in different areas? Things in the same general area should be considered together. I'd much sooner be considering a set of related math functions or just math functions in general than a bunch of individual ones.

DE: What if we say the champions should work together? Look at Rick's proposal.

AWD: Happy to say, sure, you should talk to Rick

JHD: Is appropriate to have them both be stage one?

DE: For this particular case, we want to have a consistent math library but there is no cross-cutting thing that associates the two proposals.

AWD: Is this actually higher priority than anything in Rick's proposal?

DE: The thing about Rick's proposal is that we introduced these operations for degrees.

JHD: I'm suggesting that putting these both in stage one is the right place to determine if they should be considered.

Conclusion/Resolution

- Stage one, but discuss with Rick to consider merging / relationship between math proposals

- Concern about the motivation for users (is this too trivial, and any interested users can implement it themselves?)

13.ii.f Promise.prototype.finally to stage 3?

(Jordan Harband)

https://github.com/tc39/proposal-promise-finally

JHD: The overall state. There is still information being gathered by chrome in particular about whether or not .catch can be modified to not observably call .then. If that's true, prototype.finally would do the same. This feedback wil affect this spec (potentially).

DE: Even if we find that nobody or very people do this. I'm not sure we have agreed to change .catch's semantics.

JHD: we need to determine first if it is even possible. The straw poll we took last time said it was okay because people generally want less observable things.

MM: How are they checking whether the change to catch is web compatible?

JHD: It would only be incompatible if someone subclassed promise and overrode then.

MM: No, there are two other ways it could be incompatible. THey could replace promise.protoytype.then or they could add a then property to the promise instance

DE: Or you could .call the method

JHD: So i'm even less confident now on this change being web compatible.

MM: I want to know how you -determine- that

JHD: I am not sure how that works. I'm not seeking stage three this week. I just want to establish that if .catch will be changed .finally will change. It seems likely that this won't happen, but if it did, I would have a PR that makes this compatible.

??: Whats the motivation for not calling it?

JHD: Just so the .finally call can have less observable specs. I've been told that the V8 team <lost it>

DE: If we look at the value of the .then method and it's the original builtin value, you can avoid allocating the closures.

JHD: That was my primary concern to make the change. It seems in the majority case that seems like the right way to go (to be consistent with .catch). I'm going to merge this PR, get spec review, await the outcome of the .catch investigation and renew my request for stage three. I don't see future changes.

JL: I'm pretty sure that angular and some others depend on .catch calling .then or that they can override .then

JDH: And then the native .catch calls into that? If that's the case, it's not compatible to make that change

AK: WHat's the status of V8 looking into this?

JHD: There is an open bug on your bug tracker.

AK: And we're looking a this?

DE: We checked in a use counter and we'll have data in a few months. Even if the use counter isn't hit, that doesn't address the design constraints this addresses. We don't generally hold nonzero to be the threshold for compatibility.

AK: I'm concerned about holding this feature up... the longer it isn't there, the longer it isn't there. I'm objecting this being stage three because this spec might change depending on the outcome of this work.

AWB: Basically we're splitting hairs about observable vs non observable changes.

JHD: okay, I'm going to merge the pull request where it observably calls .then, look into angular, then confirm it's not compatible.

MM: Given the topic let me say that I approve that when this issue is settled you can go to stage 3.

JL: Do we do conditional Stage 3 approval?

MM: Given the topic it felt poetically appropriate to share this ;)

JDH: Last question, can I have some volunteers to be reviewers? Mark, Dan

Resolution / Conclusion
- MM and DE will review, based on proposal in a PR that observably passes callback functions to the .then method
- Still at Stage 2

15.iii.a Error stacks (seeking stage 1)

(Jordan Harband and Mark Miller)

https://github.com/ljharb/proposal-error-stacks

JHD: I'm looking for stage 1. Spec details don't need to be bikeshedded here. The goal is to lay a foundation for error stacks, such that future proposals can resolve outstanding wishlists. The general gist is that error.prototype.stack is going to be an Annex B accessor on Error.prototype. The goal is to have a union of behaviors across browsers in a way that allows for extension which I'll get to in a moment. The reason the accessor is annex b is to allow

BT: Why is that a value?

MM: It's of value because the stack leaks non-local information that can be used for attack (I provides non-local information that is normally encapsulated). Also, everyone sharing the same realm necessarily shares the same error.prototype.stack, so they have to share the same view about where the stack maps to. So, SES for all of those reasons is an example of an environment which will remove this accessor. Having removed it, you're still a compliant environment.

JHD: If it is error.prototype.stack then the stack information is directly attached to the error object. That can be sent across realms. If you do not have the stack accessor and you have something like System.get.stack

MM: No, you understood it except the use of Realm. If you are in different realms those could have different "Error.prototype.stack"s. For residents of one realm running with different global scopes, because of the different get stack string functions they see, they could have different views of how they map to source positions. If it is on error.prototype.stack, you cannot deny access to it diferentially across realms. You also can virtualize differently for residents of the same realm. It's also a security concern, an adversary can look at how the stack was populated. <lost it>

MM: In all four browsers, if you create an error object and you say error.stack you'll get a string that gives

you information about the call chain.

WH: Am I the only one bothered by this exposing hidden data from closures? MM describes a workaround to hide this data in sandboxes, with a lot of effort, but the basic information leak problem remains for non-virtualized scripts.

MM: No, the reason you make it removable, and Annex B, and deletable is because we're worried about that.

WH: Fine you address this for virtualized environments, what about regular ones? Modules trying to protect themselves from each other?

JHD: Those libraries can't hide that information at this moment. Not specifying it isn't increasing security, it is pretending the issue doesn't exist.

WH: We had the same issue with browsers showing the arguments to functions as a property of functions.

AWB: I know, for strict mode functions in particular we tried to prevent stack walking to observe the callers.

MM: And we were able to get rid of it due to the (strict-mode) opt-in. If I thought we could simply remove the stack trace information, not codify it, but get all browsers to remove it. If I thought that was possible this proposal would be very different. Since all browsers currently implement it, I'm not going to try to argue for that. I'm looking for the least dangerous thing that is compatible with what the web currently does.

JHD: There is huge ecosystems relying on stack traces, we can't ban these.

MM: WH's issue is not with banning this way of accessing. In fact we're specifying that all of them must put the accessor on Error.prototype. It's removable, the remaining privileged operations are on the System object. (WH: that's the opposite of what I said)

BT: What are your actual plans for standardizing Error.prototype.stack? It seems at this point we're all pretty entrenched as implementers on our stack format.

JHD: Yeah, let me get into that. I'll explain the API. The normalization that this is doing is compatible with what all implementors are doing.

BT: More related to the previous discussion. I understand now... I guess the answer I was looking for is that you want this in Annex B because you explicitly want to require browsers to have this property be configurable and you don't want to see error.prototype.stack show up in say, node, or other non browser environments.

JHD: It's already there

MM: We want to allow other implementations to NOT put it there

BT: Annex B is only required for browsers

MM: Yes but it's normative optional elsewhere, it allows non-browsers delete it

JHD: Some implementations do it as an accessor, some do it as an own property.

BT: This is beside the point. I'm trying to understand what you're trying to require of whom so I can understand. I'm extremely skeptical. I don't like adding things to Annex B.

JHD: The first thing in general for adding/codifying it, is not allowing it to be an own property. The reason it should be in annex b is so that it can be deletable without the environment no longer being ecmascript compliant.

DE: What if we just put the whole feature in Annex B? Do we really need the feature dually accessible through this error getter and these properties on the new System object. Non Annex B environments could sort of either not use the future or pull it off to store somewhere before destroying it.

JHD: that's separate, I hadn't considered that

MM: I hadn't either

BT: I would object to that completely. I don't see why you couldn't put this in the main text of the spec along with the presumably the spec for System.getStack

JHD: I guess that's true. Either way you could... either way the accessor has to return a string if you want to deny access to the stack trace information, you'd have to return a string or object representing it.

BT: That's going to be true regardless.

JHD: If it doesn't exist, the only way to get to the stack is through that function

CM: If i understand correctly the reason to put it in annex b is to make it allowable to get rid of it. why not just say that's allow?

MM: No no, it's just saying that once it is disabled it's an environment that's spec compliant.

CM: Just say that, then. Annex B has other implications.

MM: No, we're saying it's optional to implement the feature, but if you do implement it, it must be like this.

AWB: I don't believe Annex B says feature-by-feature thing can be omitted

BT: You can't pick and choose pieces of Annex B

JHD: it sounds like if there is a mechanism for it to be deletable and still be compliant

BT: Making it deletable just makes it writable. We're saying don't put it there at all

MM: If Annex B is take it or leave it as a package. It needs to be normative optional on a per feature basis. If it is only normative-optional as an indivisible bundle, I've been putting things in annex b for years with the wrong assumption.

MK: I had the same intuition you had mark. I'm reading the text and it's not clear.

AWB: o, these are the things that you need to include as a browser

MM: I thought it meant that each thing was individually optional. Michael just read it and said he couldn't tell

AWB: If it doesn't say it is individually applicable then it applies to the second

MM: No, it applies in the section

WH: No, it's a logical fallacy. If you're a browser you must do this; if you're not, you may do this or not do this. There is nothing that states that you can't do something else, like just a part of this, if you're not a browser.

MM: You may do part of it. They are individually normative optional.

AK: I think if you're building a browser you want this feature.

MM: I'm fine with it being what I Thought annex b as, which was mandatory in a browser, normative optional otherwise.

AK: I don't think it should be an optional thing for building a browser.

BT: That's fair

BT: Mark would you want a web browser to not ship this? Are you comfortable requiring it?

MM: I'm comfortable requiring it.

AK: Parts of the regular spec will say parts are allowed?

BT: Before I thought i was optional in browsers as well, was that a misunderstanding?

MM: Yes it was. I understood Annex B to be mandatory in browsers, otherwise normative optional on an individual feature basis

SYB: are we just discovering that English is ambiguous? :)

AK: I don't think there is normative text in the spec to describe what a browser is

AWB: When I wrote this language I was not thinking about piecemeal application. It's plausible to interpret it, but that isn't what I was thinking about.

DE: I think the discussion would probably be helped by more concrete current nonannex-b implementations. We know that SES is one non-Annex B environment; other possibilities could help us refine our ideas for what we should have in spec text. The other thing I want to say (shameless plug) I have a PR for ecma 402 and I have inline annex b stuff and I have different formatting for it. I would be interested in anyone's review because apparently there are a lot of feelings about what annex b is...
https://github.com/tc39/ecma402/pull/84/

MM: I would like to review it

DE: Thank you

WH: I see this as a problem that creates non-sandboxed information leaks. Just saying you can delete this if you're building a sandbox doesn't ameliorate the leaks.

MM: the issue WH is bringing up is that it *can* leak information

JHD: It's already doing that in every implementation.

MM: That's the reason why we are pushing to codify it because we don't think we can kill it

WH: I think we should kill it or require special permissions

JHD: ordinary scripts already require this sort of thing. NewRelic for example, there are billions of dollars in around this. i can't see how it would be possible. I think all member companies here rely on that information.

WH: It's unfortunate that we're required to codify a security hole just because of existing practices.

MM: Do you believe there is something we could propose that could be accepted and be implemented that would plug the hole?

WH: Hmm. Approaches would be either to not build this or to proceed along the way you (MM) suggested of

finding some way to obscure or hide the information it provides.

MM: As far as the first one is concerned, if we don't propose anything, browsers will continue with existing practices and not solve the issue

JHD: I would claim this lays the groundwork for opening the possibility for allowing some sort of global config to say anyone using this System.getStack information to remove all the security leaking information. Currently fixing this is not possible at all

WH: Sure but that would break those billions of dollars of scripts....

JHD: maybe it could be opt in?

MM: That's precisely the difference between what WH is proposing and we have here. Things which have not opted are able to now

JHD: That is the reality of the web, unfortunately. That ship has long since sailed.

AK: it might be cross origin iframes issue

MM: The issue about cross-origins stacks and what information we reveal there is important, fortunately because ecmascript doesn't understand the concept of origin, because of the freedom this proposal allows, this proposal specifically is able to sidestep that. We as responsible creators of the web infrastructure can't sidestep that. it's a hard problem. i don't know what to do about cross-origin stacks.

JHD: You can postMessage an error object

AK: We've had security bugs with stacks

MM: I want to signal with everyone that is an issue we need to wrestle with

JHD: Error.prototype.stack is an an accessor that is somehow normative optional. There are two functions that are stored on a privileged granted namespace, currently called System. Mark has a number of opinions on why it must be a separate namespace. WE can also address concerns of compat about the name system. I'm sure that anyone using that is just using it for a namespace to hold .import.

AWB: Why not use an import to access it?

JHD: Because there is no current feature for builtin modules. As we've discussed every time this comes up, it shouldn't be a blocker.

AWB: Maybe we should change that.

JHD: There is no consensus on that.

MM: In order to change it, you need to do two things, one is you need to solve the problems we've already discussed. Because this is a privilege-granting operations, you basically have to alreayd have a multiple loader environment so you can say that some priv code says you can import the object. normal code cannot. If you had a built-in module system that satisfied both of those requirements I would have no objection

JHD: Okay, if anyone lands built-in modules, a number of existing proposals would be.

BT: WE need some proposal to be a motivating use case to drive built-in modules

BM: We have initial bikeshedding from browser side that we could expose the current url of your module via an import of some kind. The assumption is we'd have our own URL scheme, either the about scheme which seems kind of weird, or maybe js:

BT: What?

BM: Clearly we can't use JavaScript:, we could try to present that in two months if this matters to you at all

BT: It matters to me!

JHD: I don't want to get into this bikeshed. I don't think things should only be provided by builtin modules right now. That's not something I want to get into until now.

BM: So you're not locked in.

??: What if we ignored the container of you proposal.

AK: What is the motivation for providing additional things?

JDH: WE want to create a scope in which System.getStack does something different in one scope or another

AK: I'm curious about what it does. Are there users that want to provide getsack but not Error.prototype.stack

MM: Yes

JDH: I have a use case. I want to filter out any part of the stack trace that is in react.

AK: Has anyone written code doing this?

MM: Causeway, which is a postmortem debugger for distributed JavaScript. It does a post-filtering of the stacks it gathers. It doesn't do the post filtering by bundling the result of the post-filtering as a virtual getStack. It does the filtering basically in order to present, in order to filter out things that are known in that context, likely not to be significant.

AK: I understand the use of working with stack traces and strings and even objects and filtering them. I'm looking for a use case online.

JHD: Okay, so, the API. [Describes the spec as linked above]

JSL: V8 has Error.captureStackTrace.

MM: this proposal doesn't support that

JSL: We are making use of this in Node

BB: That's because in V8 you have to do that

JSL: It basically lets us truncate the stack at a particular site. It basically takes the error and a particular constructor.

JHD: Two responses: On es-discuss, some modifications due to feedback

MM: What we shouldn't do is expose prepareStackTrace, which exposes all sorts of function objects

JSL: We use some additional information to get stack information

JHD: We might want to add an API to configure that in a future proposal.

BB: What would the performance impact be of having this available? {unclear}

AWB: How is [[ErrorData]] represented?

JHD: This needs to be added in a future revision

{What is the motivation?}

MM: There is currently a lot of cross-browser divergence. The only way to get error stacks cross-browser currently is to convert to a string and scrape the string. The SES shim has to do this. The proposal is purposefully vague, to permit omitting frames, etc so you at least get the basics in a cross-browser way.

AWB: But the hard part is actually looking through these cross-browser issues in a comprehensive way.

WH: How useful would this proposal be without those things nailed down? The claim was made that there are lots of scripts in existence that depend on this proposal's functionality so we must include it in the spec, but how do said scripts deal with frames omitted or inlined at implementations' whim?

JHD: They break once in a while.

JL: You should remember the problems with stack frame elision from the tail call discussions.

JHD: again, during these discussions, stacks came up. Mark or I, one of us said we are not intending to normatively require or prevent omissions

MS: I think I asked specifically that not all frames would have to appear. Thank you for doing that in this proposal.

JHD: Thank you, I intend to do that.

MF: I like this proposal, I'm very much in favor of it. Early in the discussion we discussed a convenience API where a function is given and all appearances of that function in the stack trace are censored.

JHD: I went back and forth if i would include that in this proposal. This is hard enough that I think it should be a follow-on as options to the System.getStack call. I'd like to be able to black box react or jquery, etc.

MM: It seems to me that's a perfect example of something you'd want to do with getStack virtualization.

JHD: That's true, you can do that yourself by virtualizing this.

MF: Do the stack frames actually contain the function object? We need to compare against the function object and don't have the source position.

JHD: I agree it's useful to just take the function object, that's a possible extension in the future to this functionality. Nothing bout this makes that impossible

MM: We must not provide access to the actual functions. If we want to provide for this filtering, we must do it another way.

MF: I think it is required that this proposal must provide this. The source position isn't enough, we could just create functions in a loop. this is not sufficient.

JHD: I'd love to discuss this with you offline.

MF: That is a very important API to me, I hope to see it here.

JHD: I wasn't sure how important it was

BB: I am still a little uncomfortable that we're not specifying what is captured when and we know that elision

isn't going to happen in the future. I'm concerned we'll find other things, and we'll progress on this spec and find issue where this is not very practical, too much information, hard to obtain or whatever. A glimpse of what it might look like would be helpful. The structured stack format, can you show it in a JSON-like format?

MM: Yes! It's in the original README

JL: I would really like to see this as the maximally minimal. I would object to adding the additional capabilities.

JHD: Would you be content if we could add this later?

MF: I would be fine if you can do it at the user level or provide an API.

MM: I will not expose actual function objects at the user level.

JHD: we'll discuss this after

MF: Presence in a WeakMap, perhaps?

BB: It would be great if the data were JSON-like

Schema https://github.com/google/caja/blob/master/src/com/google/caja/ses/debug.js#L47

(MM: Since it might move, I provide a copy here with the one correction we talked about)

```
stacktrace ::= {frames: [frame*]};
frame ::= {
    name: functionName,
    source: source,
    span: [[startLine, startCol?], [endLine, endCol?]?]
};
functionName ::= STRING;
startLine, startCol, endLine, endCol ::= INTEGER;
source ::= STRING | frame;
```

MM: It's all expressible in JSON

MM: I'd like to imply that we'd continue to leave this to the implementation and pin it down in further specs.

MS: With this specification of what the object contains, what do you do for anonymous functions, etc?

JHD: Whatever you do now

MM: For anonymous functions, what the shim does is use a question mark. I'm not suggesting we do that, I think we should decide.

MS: <annonymous>?

MM: That would be fine with me

MS: What do you do when someone just wrote in the code using document.write?

MM: Once again, open to suggestions. Browsers do very bizarre things that differ from each other here.

JK: This is a bit of a jump. Right now this doesn't affect the stack in JS. Chrome does have an async mode of viewing stacks, to see stacks that happened async.... that async operation needs to be represented

somehow? Is there room for that?

JL: That isn't visible from the code

MM: This format came from causeway, and causeway is all about stitching together each of those separate turns and async operations into an overall causality graph. (https://github.com/cocoonfx/causeway http://www.hpl.hp.com/techreports/2009/HPL-2009-78.html) So, that's one of the reasons there is an outer object here. The identifying of what turn you're in, and then for an async operation, identifying what turn it is causing, with separately logged information, not in this schema, those are all follow-on things that i'm hoping to propose. When I took a look at overall, all the information that causeway logs that allows us to stitch this back together into a causality trace, this was definitely the place to start. As a place to start this is also the closest to all things browsers agree on. That's exactly why I've reserved some space there. FF by the way, already does deep stacks, though in a different way.

JHD: Any information can be exposed in debugging tools.

JK: It's not observable today but I wanted to know if it was

BB: Chrome has inferred names for method calls; they would not show up in the stack.

JHD: We could add additional properties to these objects. They are frozen but they don't have finite list of properties.

BB: It calls into question... what's the point of having a way to format the textual stacktrace if in fact you have much more information

JHD: Currently that information is not observable

MM: I was not thinking it would be up to the implementation to populate with more data. I was thinking more properties would come from later specs, not from implementation dependencies

JHD: My assumption was that like any other object, implementations can extend them. It seems useful to me to have that information available. If we can't currently get that information, it's fine to not be there.

AWB: Why is it important that these are frozen? They are a snapshot of data you have now, let people hack on it.

MM: if you generate two error objects which have a common stack at some state, basically a common caller, and we have a call tree and we have joins with a common caller. You don't want to obligate them to be made separate. But you don't want the sharing to be a communication channel. It's observable that they are sharing but two entities that otherwise can't communicate, with common access, can't use them to communicate with each other. When we haven't specified that something is frozen, like getOwnPropertyDescriptor, that's a great example. I desperately wish we had specced those as frozen. We now obligate the implementation to generate a fresh one every time.

AK: Why not?

MM: Because, for example, once a property is non-confgurable, non-writable data property. It's descriptor is never going to change. You'd like an implementation to be free to have a little cache and reuse them.

AWB: You'd like it, but this is JS

AK: Why do you assume it would be better to have them always cached? That has its own problems.

JHD: You're not required to cache them

AWB: They are either the same or they aren't.

JHD: Yeah, if you want it to be consistent.

MM: If I had to choose, I would not REQUIRE them to be the same object, not require caching.

AWB: Different from template objects, which are statically generated.

MM: yeah, I suppose, I can't find a killer reason off the top of my head. Maybe I don't have one. This one we can revisit.

MS: Is your concern that someone would use property names you'd want in future versions?

JHD: If it created a fresh object every time, it would just be shadowed. If the concern is implementations... they can currently add a property like they can anywhere else. We might say you can't add new properties that aren't in the spec but leave it unfrozen.

MM: We will need to leave it as an open question whether the produced object is frozen or not.

MS: I can imagine implementation extensions causing compatibility issues. Say an implementation adds an arguments property, but later TC39 standardizes it differently. But this is all about implementations, not about whether the object is frozen at runtime.

AWB: Well, if you're asking for enhanced information, why not use an implementation-specific API for everything?

JHD: Maybe they would be implemented in terms of the other, and this could be affected by our decision about freezing.

AWB: I'd assume they'd be implemented primitively.

WH: How does an implementation censor the result of GetStackString? It can call GetStack and possibly remove or alter some frames, but how does it render those back into something that GetStackString would have produced?

MM: Good point. We need a way of rendering for virtualization. This suggests including another part of the SES shim api we omitted from this proposal: stackString, which is an unprivileged operation that takes the object structure returned by getStack and produces the rendered string returned by getStackString. Then, when virtualizing getStack you'd just reimplement getStackString as stackString(getStack(error)).

Conclusion/Resolution

- Stage 1
- Open Questions:
  - Determine if this belongs in annex b or if other normative-optional text is allowed
  - Add a separate method to convert stack frames to a stack string?
  - should getStack return frozen things?
  - should implementations be prohibited to extend the `frames` array/objects?
  - should we include in this proposal an API that takes a function object and allows for eliding stack frames from within this function?
  - what does the Error constructor text for populating [[ErrorData]] look like?

Discuss non-technical long-term vision for ECMAScript

(Brian Terlson)

JFB: Kill it with fire?

Straw poll, should we kill it with fire?

BT: No, the consensus is to not kill it with fire.

BT: Are there people who would be interested in listening and participating in such content? I was thinking we could debate format, 5 minutes to 30 minutes level. No 2 hour disserations or anything. We could do it at the next meeting if that is sufficient time to create decks.

MF: Will we produce a goal statement or some kind of document?

BT: I think it would be awesome for this committee to have a consensus roadmap.

MM: I would prefer not to use the upcoming meeting. The kind of thing you're asking for, I would really want people to think about what they want long term and let that gestate for a while. With all of us having many things in our schedule for the next two months, having a bunch of unanticipated time over the next two months is going to be too tight for contemplating in this way.

BB: Are we saying everyone who has an idea would say this during the same day? We can do it staged

MM: I would be perfectly fine

AWB: I think it would if we could focus and have like, a vision day, and not have a haphazard here is a piece, there is a piece. We should block off a day and we aren't going to worry about, you know, the current proposals and the process and the bugs and just think about the next 5 or 10 years or whatever seems appropriate and start to form a common vision about where it is we are going.

BT: To answer MF's question, in an ideal world we would have a shared vision of where we are heading. I think some kind of vision day... something that would come from upper management. It felt horrible to say that. Someone give me some other name.

MS: Long term direction day?

BE: Everyone read the harmony email from 08? Long term direction happened, ES6, etc. It's time for a new one. link: https://mail.mozilla.org/pipermail/es-discuss/2008-August/006837.html

BT: I hope that by everyone sharing what their vision is, that we can appreciate the different aspects of experience and interest that people bring to the table and have another rev of that document. We should keep the top 5 long term goals of the committee available. I'm not sure what it looks like, but a reasonable starting point is us sharing what we think?

MS: Brendan can you share that?

BE: You can search "ecmascript harmony mail" it's the 2nd hit

MPT: i started a conversation about vision last night, where I was coming from, the community et al has very little idea of what goes on in this room. For the most part it doesn't occur to them that they would have any influence on the proceedings here. If they don't have a say, I would say they ought to. I don't think they should be here commenting on every little thing. But it does make sense to release a long term vision and accept their feedback and yes you're going to get flamewar and 500 miles of bikeshedding. Inside that you'll find valuable feedback of what people are looking for. As the JSF rep, I'm happy to try to get that some press. Make a doc that says where you're going.

AWB: I assume whatever we discuss would be public and call people's attention to it.

BT: We don't wan tot just dump a bunch of text on people.

AWB: Ultimately we want to get something unified, but the first step is probably "Decks"

MS: Part of our diversity issue, we don't actually have a lot of users here. We talked about it making sense for those in the committee to go to user group meetings small or large and represent the workings of TC39 including what's currently going on. We didn't create any actions but we did talk about doing that.

BB: I think what is confusing for a lot of people is that TC39 itself does not have or state anything that we produce the spec. The spec is pretty unreadable and uninteresting.

BT: hey now

BB: To most people. It doesn't tell you what can I expect from TC39, what do they think is important, how do they approach problems? In a way we are just a group of random people arguing about things. We may not have a shared opinion but we could have a shared message.

??: What is the forum for this discussion? We have esdiscuss, we have github, twitter isn't very effective....

BT: I would hope we could put something on the TC39 wiki.

DE: We produce the minutes, a lot of people read them, but they are kind of hard ot read. When I was working at Google, I would produce a summary of what happened at the meetings. Now, I could do that externally instead

BT: I would love for you to do that; I do that at Microsoft

MPT: It's a stated goal of JSF to produce exactly that for the community. We don't do it , but we could! I was slack chatting with Chris today about this.

DE: Great, let's all work on this together then.

WH: We have had, quite often, meetings with community user groups as an adjunct to TC39 meetings. We had one last year in Munich, in Boston, etc. They were for the user community to meet us and for us to meet them, chat, and listen to their stories. We should continue doing that. That's part of the reason I was pushing for geographical diversity in our meetings.

BT: I promise you have a JS meetup in your area.

BT: Back on communicating of vision....

AWB: Before we communicate it we need to create it

BT: Communicating amongst ourselves, we all agree that would be good. Does it seem unreasonable to spend an entire day on this?

DE: Sounds great.

AWB/TK: not enough

MM: if we spent an entire meeting on it

MPT: you'll want to spend a day on it, give to the community, get feedback, and then do it in the next meeting. You'll never get it in one meeting.

CP: I don't think we're talking about getting a draft of anything. The first step is getting a sense of what the highly varied opinions are

DE: We could write a summary of the visioning session on the newly legible minutes that might happen and

get feedback on that, even if we don't get a consensus vision among the whole committee.

AWB: we need goals, he meanings fo the goals, the specific actions we'll accomplish

BT: The question is, what .... i guess without doing an actual call for presenters. I imagine we all want QA periods too, we have at least an hour per. WE could just cancel the agenda for the May meeting. We should decide that now. People should get stuff ready for that.

JSL: In may we're just ratifying 2016, right? That's where we have the final vote?

MF: Can we just do proposals ready to advance? Let's do the stuff we have to do

JSL: We travel in NY we end up not using three days

BT: What if we have three presenters, and the schedule doesn't fill up?

CP: Who will present?

BT: Not many hands were raised, but I hope to make the compelling case for more people to present.

TK: Good time to bring in the community

BT: I think the two-day plan is reasonable; I'm sure we'll find other content to fill the day with. This is all for the May meeting in New York.

DE: Will the CFP be directed at just members, or also at the community?

BT: Let's have a meeting the Friday after TC39 to bring in the community and talk about the vision.

WH: Friday would be an issue for those who travel to the meeting. An evening during the meeting would be better.

MPT: Lots of people have travel restrictions, so maybe Wednesday evening is best.

WH: In the past, we've already done these things on Tuesday/Wednesday night

TK: The visioning should be done *before* the community meeting, rather than afterwards, so maybe Tuesday night

MPT: How about splitting it up further, e.g., I go to Seattle JS. Anyway, most user groups would hold a special session for this sort of thing.

JHD: It will be good for us to be on the same page

BE: In the past, various people wanted to control the narrative.

DE: Chapters.io has some resources

BT: I'll put a reflector post for figuring out community outreach

Conclusion/Resolution

- BT to make a post on the Reflector saying that we'll spend the May meeting
- BT to write and send out a CFP for internal vision presentations
- Another thread to discuss attempts at community outreach

January 26, 2017 Meeting Notes

Allen Wirfs-Brock (AWB), Waldemar Horwat (WH), Jordan Harband (JHD), Brian Terlson (BT), Michael Ficarra (MF), Adam Klein (AK), Chip Morningstar (CM), Dave Herman (DH), Kent C. Dodds (KCD), Kevin Gibbons (KG), Tim Disney (TD), Daniel Ehrenberg (DE), Shu-yu Guo (SYG), Michael Saboff (MS), James Kyle (JK), Franziska Hinkelmann (FH), Anna Henningen (AH), John Lenz (JL), Sebastian Markbage (SM), Bradley Farias (BF), Jeff Morrison (JM), Tyler Kellen (TK), Gabriel Isenberg (GI), James Snell (JSL), Maggie Pint (MPT), Chris Hyle (CH), Gabriel Isenberg (GI), Bert Belder (BB), Zibi Braniecki (ZB), Jamund Ferguson (JXF), Istvan Sebestyen (IS), Keith Miller (KM), Myles Borins (MBS)

Looking at the proposal list

(Dave Herman)

DH: Let's reexamine https://github.com/tc39/proposals . Can we think about what proposals are still possible for the future?

AWB: What things are likely to be there for 2018?

DH: Well, with the train model, we try to not pay too much attention to the annual release, and let things land when they're ready.

JHD: I'm maintaining this page. The status of the proposals should be visible from the proposal page, if you click on them. For my proposals, I have bugs open about 'the path to the next stage', etc

DH: Maybe we should put more links to specific per-project pages explaining status up.

JHD: If I put "when I expected my proposals to advance", that would be aspirational and possible to change

DH: What if we were "rigorous about being not rigorous", and explained that it

JHD: What if premature expectations of staging led people, e.g., browsers, to think that things are more stable than they are?

AWB: When writing the agenda, we could reference the aspirational advancement dates

DH: The biggest value might be putting pressure on champions to move things forward

DE: I'm a little hesitant to put down these goals; for example, I didn't have time to fix some Intl.Segmenter issues this time so I wasn't able to present, and other times, there may be issues where implementers give feedback or the committee has disagreement that takes longer than hoped to resolve.

DH: I don't think it is a good idea to talk about "what edition' something will land in. That goes against the idea of release trains. When am I hoping to propose stage advancement is useful though, if i twere a hard line, if there were anything in the process tied to it formally I would consider it a bad idea. If it's only a soft requirement will it just collapse in a few months anyway? It'll end up like the rocket ships, we don't actually stay on top of it and it doesn't bring value. I think it is a bad idea to make this ahard requirement and as a soft requirement it doesn't have enough teeth to be maintained

JHD: There is more to it, you're talking about hard/soft requirement for the committee but people outside th ecomittee refernence this as their source of truth. Proposal champions receive the flack for that. I don't wan tto put something up there, front and center. I intended global to be stage 4 this meeting, but it isn't. I don't want people reading it and having perceptions in their minds.

AWB: We can make things private if you think that would be... a private and public version.

AK: What problem are we trying to solve?

DH: That doesn't solve he problem as I understood it. You wanted the public to get an accurate view

AWB: I want us to have a more accurate view as well, what we can expect to be happening

DH: I can say personally I don't feel that as a big problem. Those of us involved in the month to month work generally have a clear sense of what is going on. I don't remember ever feeling suprise that OMG there is an advancement I didn't see coming and he heavens are falling. I could see this as being a thing for a general audience wanting to see what is going on

DE: We have talked at this meeting about a few things we can do for visioning and communication outward yesterday

AWB: I don't think Dave was here for that

DE: We also have the possibility of writing up notes in a more abbreviated form. Maybe these will solve the fundmanetla problem of communicating the vision and status of the committee. That would be better than tying chmapions down to specific dates

JHD: People follow the proposals GitHub closely. There are lots of ways to stay abreast, every time the meeting notes are posted, etc. There isn't one location to get an a real view of this but if anything I've seen it's almost too easy to find out what is going on. Not that I think we should restrict it...

AWB: I think that we have reached a scale where we need more formal structure. More project management if you want to call it that. Everything can't be independent and continue on some track.

AK: Can you say why?

AWB: Yes, because there is no predictability in where we are going and not a lot of coordination between proposals, we have things that are literally sitting for years in stage one

AK: I see a lot of coordination in things moving forward, none for things that aren't moving. At the beginning of this, I thought we were talking about how to take stuff off that isn't being worked on. I agree that should go away

JHD: Right now or anymore?

AK: Right now.

DH: Deferral rather than outright rejection. The idea is to simply cull things that aren't active, not kill them. Just remove them from the list of active things. Don't say this is now off-topic.

JHD: That's a 10 minute commit.

AK: That's one thing. I agree with Dave. People who are moving things forward are doing a good job collaborating. The champion column here makes it clear.

DH: The stages make a checkpoint, if we see cross-cutting concerns, we address them.

AWB: The other thing that might be useful to get more co-item tracking, like, should all regex items be moved along in parallel, perhaps merging or splitting items. These are all things we can do...

DH: What I feel like i could use guidance on as a champion. I'm unclear what the template is for a proposal repo. We have organically grown a template. Some stuff I've been working on, from the "olden days" Do we have an explainer.md? What does the README have? I don't want any more formal process than what is merited, but I do want to know what we're looking for in these things.

JHD: I'm happy to make a proposal template repo. Put your spec text here, create issues for stage advancement, etc. I'd be glad to write that up

AWB: It's almost an issue of... some curation is needed.

DH: Jordan is doing curation, I think there are some little things we can do

JHD: I'm aggressively curating and have been for awhile

DH: Jordan is doing really important work and I think we should acknowldge that

AWB: Can we put that in the minutes?

TK: Getting it all

DH: Thanks Tyler

JL: So are we going to remove more things from the main proposals list?

DH: What I am hearing is an emerging consensus around proposals into some separate file that...

JHD: We already have withdrawn proposals, for things that are abandonded,r jected, etc. WE can add something for things which are potentialy still acive. It's more likely that list will contain stage 0 or 1 proposals than 2 or 3. If that is the status of the simd spec, for example, okay

DH: I'm unclear how backburner SIMD is.

DE: It can be hard for champions to make statements about these thing sometimes

DH: Some people want it to be ambigious, they aren't ready to commit to not committing. Some stuff where it is like... the Date.parse legacy syntax standardization, she (Morgan) has since quit. I'm not sure she has any time or interest. She might not do it, we should ping and ask her.

MPT: I would adopt that. I have a high level of interest in dates parsing correctly

DH: That's awesome!

AWB: If you think about it as a deferred list, if you put it on there, we can say we'll review this, revisit this.

DH: I don't think we need to put a date on it. Is anyone going to champion it?

AWB: I was trying to make the distinction as deferred for the next year, but stay on the roadmap vs it's dead. IF something like this comes up again, it comes up again.

AK: Stage one items aren't on a roadmap

AWB: Well, it's on a roadmap where we decided it's worth exploration

AK: I don't know how you can get less.... any stage between 0 and 1. If a stage 2 item went dormant, a stage 3 item is even harder, even though we have that. This is why i explicitly withdrew Object.observe.

DH: If something is removed, we should remove it....

AK: I don't think we need to cull the deferred list.

TK: It's the backlog

DH: I officially have no opinion anymore

AWB: I'm going to make a a motion that Jordan is now the official GitHub curator

JHD: I will make a proposal template repo over the next month or so

DE: Should the kind of project management that Allen mentioned be part of "curator's" job?

JHD: What I would say is the most efficient way to do this would be to file an issue on the repo for the proposal or go talk to them. The time between wondering what's going on with it and figuring it out should be short. The repo for the proposal should be a way to find out quickly.

DE: IS that the job of the curator?

JHD: If someone files an issue on the proposals repo to do that, I can follow up on it it seems indirect but I'm happy to do it

DE: This is what Allen was saying about general project management.

JHD: I think if we figure out the wording for a deferred list, every few months, I or someone else checks to see what should be moved

AWB: So right now with our agendsa we basically... Jordan you put it together and we kind of leave it up to the members to fill things in.

JHD: Yeah, the only thing I've ever done, or ever seen done is timebox/overflow items i put in a section where i ask the champion to move it to the timebox they want for the next meeting

AWB: At a minimum what might be an interesting thing to do, when we assemble the agenda for the next meeting, send an email for everyone who is on this list and ask them

JHD: A month before the meeting we could post an issue on the reflector and tag all the champions and ask for an update

AWB: Or just decide. You don't have to but...you know, part of this is just pushing people to think about what they, if there is something they need to do, giving them time to scurry around in the last week. Okay! So Dave, are you ready?

Resolution / Conclusion
- JHD to be official curator of tc39 github
- When assembling agenda, cutator to send email to all champions asking for stage status updates or make an issue on reflector doing the same
- create an issue on proposals to make a template repo
- touch base with all people who have proposal to get status to create a deferred list

13.ii.e Seeking Stage-1 for "do expressions"

(Dave Herman)

[gist](https://gist.github.com/dherman/1c97dfb25179fa34a41b5fff040f9879)

DH: There's a babel plugin for do expressions, I think it's pretty popular

SM: We don't use it, but it's pretty popular with JSX

BF: It's pretty popular

DH: So, the basic idea here is to allow more expression forms in the language, a more expression based style of programming, a nice thing FP brough tot the world. I'm not trying to get folks to drink the koolaid about it, I myself have stopped drinking it. Expression are nice though, because they plug together well. Philospohically, that's why they are good. Do expressions contain statements in expression context and evaluate to the completion value of the enclosed statements. They let you have locally scoped lexical variables. Prettier ternary operator. For JSX, if you want to use more expressive control flow when having something that becomes an expression. Long-run, it would be nice to have more statement forms with expression analogues, but do expressions bridge the gap. We should avoid refactoring hazards. No spec written up, but it should just be, "return the value of evaluating |Body|".

(review of linked repo)

JFB: Out of this falls out something weird, like putting a return in there, it would be the same as doing an expression

DH: the philosphical justification is Tennants correspondence principal.

AWB: Out in the lobby a few of us were talking about this. One of the things this allows you do do a return in places where it is impossible currently. Most of the time that's okay but there are a few places where serious considerations are needed about the meaning (argument lists, initialzation expressions of a generator function for parameters and stuff). There is a need to do a semantic review of places this might occur and either decide what the semantics are or to decide in that context.... there are ways to syntactically handle this stuff.

DH: Great question to resolve for stage two, generator expressions especially. My goal is to be free of caveats as possible. There are places where teh existing smeantics of evaluation order is subtle

WH: This also comes up in for loops, e.g., `for (do {break;};; ) {}`

JHD: Your correspondence principal half answers my question... do statements, i like, they make a new form of implicit return, where the completion value is used in a way that isn't other wise like... so like in an arrow function with an implicit return, it's implicitly selection the completion of the thing, so it's an expression. It's a bit confusing since this is the first time in normal code to observe the completion value; people using eval don't usually think about it that way.

DH: This is the first place in the syntax that makes completion observerable

DH: There is an API, eval, that makes the completion visible. Most people don't realize that is what happening. There is a question if people now have to undersatnd what completion is.

JHD: Would it make sense to have some sort of keyword in the do statement return/yield/etc.

DH: I would argue against it. It feels a bit like Stroustrup's rule, when something is new people want it to be noisy so they can feel comfortable with it until they get use to it. In the short term it feels good, in the long term it doesn't. There is a usability concern. How do we teach this? I would argue that it is intuitive enough that it doesn't have to be explained by teaching people rigorous notions of what a completion is. Maybe it's more like, the value of this statement. Oh, what's that mean? It's worth testing that, the early user testing we have in the babel community at least indicates that some people find it appealing. That could be selection bias, though.

JHD: I'm not pushing one way or the other. If you type something into the console it's as if it's doing this already. Whatever you type into a console and pops out that's what happens. My concern is THAT it is teachable, i don't mind which way

DH: It would be nice to get to a point where, I don't think we'll get there or the complete route there should

loc, but it would be nice aspirationally if people didn't have to think about statements much because more of them could be turned into expressions. There is grammar trickiness to get there, but it would be nice if people didn't have ot hink about this. Maybe that is too naive of me

AK: I have a concern that is not about the semantics, it has bad semantics. var declarations. what are your thoughts? I think it would be a mistake to allow var declarations here. I understand that doesn't fit

DH: I disagree; var declarations should be hoisted for the same reason that var already hoists up regular blocks. The thing that you're saying is weird, in my mind is the thing that has always been weird about var. To make var not weird in this case just because it's trying to put lipstick on a pig, it feels to me ... that it doesn't actually help anything. If you want to have understandable lexical block scoping, use let. Var is the thing that wherever you happen to declare it it's in the local function everywhere

AK: Var declarations in expressions in arrow function params happen to be relatively hard to get right (For us) we have an implementation of do expressions in V8. If you are open to arguments where it's hard to understand what the var declaration is...

DH: No question that's hard, that's already the case. Var and catch, var and let and var and eval and let and catch and all of these things in combination can be very confusing. I'm open to everything, right, open to discussion about anything but i think i'm wary of us adding special cases that make it confusing. In practice what people will have to then learn from that is don't use do expressions or don't use var. Not using var is a reasonable outcome, that would be my best case outcome. Maybe we'll look at the corner cases is there is no easy way we can handle is and just disallow var.

AWB: You have to drill down into nested blocks if that is a concern, it's just not just var in the statementlist of the do block, it's contained vars

AK: And var in sloppy eval in those contexts

DH: That's always true about var

AWB: the semantics would be strange. it's really clear, if it is easy to implement those is not clear

DH: Ease of implemention vs ease of understanding. Out the gate it's bad...

AK: I would challenge the ease of understanding

MF: I don't see a reason to allow break/continue/return in do expressions. Can you give me examples why that would be useful?

DH: Yes, The way you expect it to work is that it, has the same non-local control flow effect in a statement context. Inside a loop you have a do expression and you have a break it corresponds to the do, not the loop. The reason that this is powerful is that it, the kinds of... early return patterns that you can use become richer.

MF: I don't think that's true.

DH: You can do them locally and not hoisted them out. If you are in a nested expression, today you have to take the whole expression and refactor it into a bunch of statements. This takes what can be concise, i'm just composing a few expressions here, one particular control flow path wants to exit early. Suddenly the rest of those parts have to become much more verbose because they have to be hoisted into local vars and statements. it would be helpful for me post a efw examples

MF: That would be helpful. I think there might be a general transformation we could apply that would allow us to not require this breaking inside the do.

DH: There definitely is a general transformation but it's prickly to the outer statement. That's the key to the expressivity, it's not a local transformation, but you have to rewrite the whole expression. In practice your

expression nesting might not be that deep.

WH: If you have a labeled const, then you could have a do expression in the initializer which jumps back to the label. What happens to the variable in that case? If it gets created, what happens when it gets reinitialized?

KG: You can't label a const. That's the point that I wanted to make, you can label var declarations but you can't label lexical ones. You can early exit from a var dec but not a const one. That's very strange.

AWB: That's kind of arbitrary, we didn't understand why you would want ot label any of the new declarations. That could be relatxed if we thought it was needed for consistently

KG: Sorry, I didn't mean to interrupt. I like this feature. I have a bunch of things to say about it. I kind of agree with MF that I'm not sure allowing non-exceptional control flow in an expression introduces a huge amount of complexity of the language. There are all these places where you couldn't have that control flow and now you can. You can have an expression which contains a do expression which conditionally returns from inside of it. You cant' determine that from the context of the return anymore. Sure, you could throw conditionally, but not return.

DH: I hate to well actually you, but that is true today. Return can be nested in a finally and it can do whatever it wants

KG: Yes i know, i've implemented try finally, i'm familiar. It's not sort of resuming right after the return, it's doing this other thing. The finally wrenches control away, control might resume after the return or the finally. That's all very strange. The return itself is conditional, it's conditional, but the conditionality is in finally rather than the statement, in the right hand instead of its context, that seems very strange

DH: This has always been a challenging topic, one of the real regrets I have from ES6 is that the lambda syntax we added didn't adhere to this principle. I want a way to finally have a way to nest expressions. I would actually not even want this feature without this aspect. What makes this challenging is that there is a blood principal. Coming from FP, i'm used to control flow operations being expressions. There is nothing unexpected about this... coming only from JS, that's not a familiar concept. Nonlocal control flow is only in statements. My goal is to get JS to a point to where you can do that, you don't have to think of statemnts as something special. If you add this new form that retains the diea that statements are special, it defeats the purpose. That's not.. proof of anything, it's just my goal here. That's what I WANT to acheive for JS

WH: It's a lot of work. A lot of sticky cases. The const case. When you jump out of a const initializer back to the const, does the variable get created or not? If it gets created...

AWB: It would have to be created but its TDZ is not satisfied

WH: ... if it gets created, what happens when you rerun the const statement?

DH: I'm willing to DO work to fill in these cases

WH: Okay then, you do what? There are all kinds of sticky cases. What about a switch statement if you break out of the switch expression?

DH: The goal here is to have something guided by the semantic transparency. That will help us answer each of thsee. Yes it is work, I'll do the work. There is also implemention work which I wont' do rfor every eimplemtnion. My claim here is that the user value of having more flexible expression in JS is worth the work. We have signals from particlarly the templating context, where, templating is all about expressions, you're putting values into other values. My claim is that it is worth doing the work.

WH: The other thing is you have in your proposal, we currently have that an ExpressionStatement cannot begin with an open brace; we'd have to also prohibit an ExpressionStatement from beginning with a do, to avoid ambiguity with do/while loops.

DH: It is premature to have this conversation. Let me explain. Disallowing do expression to be in statement context is one solution that works for sure. I think it is not dieal. I think we may be able to do better but more explaration is needed. All I put here is that there are interesting avenues to explore but I don't ave anythign to rpesent eyt. THat is why I say it is premature and this isn't needed for stage 1

AWB: A completely different semantic direction there for a moment, as youv'e said this is really the first way that we would broadly expose the completion value of a bock other than eval which most people ignore. What i have found from people asking me questions is that when people look at the sematnics of completion value through eval these days, the very few who look, one of the things they find confusing is the fact that declarations dont' produce new completion values, they just pass through. Msot of the people who come to me about that are very suprised and confused. I wonder if you've thought about jus like we werwe ablt o reform completions in ES6 whether we could consider changing declarations so they also have completion values.

DH: If you were asking me to put my life on the line I would bet that isn't web compat. The fact that you get undefined which is a value that you can test and do things with and you can use eval

AWB: you don't get undefined it propagates to the previous statement that might not have a value.

DH: If all you have is a statement or a sequence of statements. If that means you get a value and your program can call eval and it can do suff (epople use eval all over the place) this is a strong hint that the internet probably depends on that being the case. It'll subtly break things. That's just my guess, though. I guess it isn't web compat

JHD: so this is getting the completion value of the block, what if i have a return in there?

DH: The reason why a single line.. why i believe this is sufficient, completions are more than just he normal value, they are an exception monad, they are a nonlocal signal that we need to propagate up the control flow, both expressions and statements use that, if your statement does a return, the completion value is a returning completion.

JHD: If I return three, that evaluates to 3

DH: no, it evaluates to "i am returning 3"

JHD: I can't capture that an an expression? I can interrupt in expression position i can interrupt the function progression that I can't currently do?

DH: That's right, that's what we were just talking aout

JHD: So the same thing happens when I throw, in the same way I can eval a throw, if i do a throw expression

AK: You already have may expressions that throw now

JHD: but the do doesn't catch it right?

DH: Correct, it just propagates them

TK: So i think we are over our timebox. I think the original idea behind this was to get to stage 1. It seems clear it's worth taking the time since we're already doing that, maybe we should just move on.

KG: One more thing, async do is really interesting idea. It would be worth making sure there can be symmetry. Nonexceptional control flow makes no sense in an async do context. An await 0 and a continue the loop the do expression is in the loop has already exited. E.g.

`for (a of b) async do {if (await f()) continue; } // by the time the continue is reached, the loop has exited`

WH: Explain what you mean by async do.

KG: Equiv to IIFE modulo some concerns about nonexceptional control flow. AT least, that is the idea i have in my head

DH: That is the idea I have too but i fi go there it will expand in to more conversation. That's an important constrain for what needs to be considered before this could be considered a finished feature. It's an important crosscutting concern. Thank you. Any objections to moving to stage one?

Conclusion/Resolution

- Stage 1, champion DH
- Items to look into more:
  - Should non-exceptional control flow (break, continue, return) be allowed inside? What are all the edge cases?
  - Does the completion value make sense here, or are its semantics too unintuitive (e.g., for function declarations, which have no completion value)

13.ii.d Seeking Stage 1 for Realms

(Dave Herman, co-championing with Mark Miller and Caridy Patino)

[proposal](https://github.com/tc39/proposal-realms)

DH: Realms was sitting at Stage 0; I've been looking at it with Dean, and Chip, and others. Previously worked on it through ES2015, just the core JavaScript builtins, nothing else fancy. If you create a new Realm and instantiate an Array, it's a different one. Alternatively considered, this is a reflective reification of the ECMAScript realm concept. [going over presentation at https://gist.github.com/dherman/66627aac764a9795dc3875e270f55918]

DH: It forms a sort of sandbox, or you could think of it as a worker with synchronous communication.

MM: That seems odd; Workers are tied to concurrency/processes

DH: Well, people actually use workers this way, and run into the lack of synchronous communication.

DH: Generally, this can provide security isolation, text editor plugins not affecting the state of the rest of the editor, or server-side rendering running in a fresh space that can be set up properly separately.

BF: Another thing is using ServiceWorker to avoid redundant evaluation of module loading, and transpilation

DH: When you have a realm object, you have super privileges over the realm: you can get its global, evalScript

WH: What's on the realm's global object of a realm you created in your example via new Realm()?

DH: By default, what's on it is the standard library of ECMA262--Array, parseInt, Function, not window, document--none of the web-specific stuff.

WH: What about Annex B libraries? In particular, things like the error stack introspection that we discussed earlier in this meeting?

DH: We can discuss; either could work.

AWB: You could also have Annex B realm

DH: You could create a subclass of Realm that's a truly vanilla global object. Anyway, new Reflect.getPrototype(Realm().evalScript("(function() {})")) is the inner realm's Function.prototype

BT: I think I have an implementation of this Realm API in eshost. I arrived at the same API! You can use eshost if you want to play around with this. One difference is that if you're making a Realm in a browser, it's an iframe, so it comes along with all of that stuff.

DH: This is deliberately not the point, so it can be a portable API. In the browser context, you shouldn't get a whole bunch of other stuff. It could be the DOM of the parent, but the goal is to be sandboxed and isolated.

BT: How is that different from saying you shouldn't have access to the DOM in any case, since it's not portable? Just trying to understand the argument.

CM: You could easily put the DOM in, but not take it out easily.

DH: We could actually easily go either way.

MM: If you go the other way, you have to have the expense of adding and removing it.

BT: DOM is a bit of a red herring. What about WebCrypto--shouldn't web programmers have access to it?

DH: You can make a subclass that manually adds the properties from the web platform that you want to add (or remove).

AK: Should init() be called somewhere in the code sample?

DH: init() is called by the constructor, in this design. About the global object, another option is to start with nothing and add each thing from the start. Let's go through the examples further. For subclasses, there's an init() method you can override, as part of the object creation protocol. init() populates with the standard globals. You can override it to prevent things from being added, or manipulate things before or after that. This is one way to do it; let's not go into too much detail about the exact details

WH: When does init() get called?

DH: From the base constructor.

WH: So init() user code runs before the user super() calls return in their constructors?

DH: Yes.

WH: What's the rationale for separating init() from the constructor?

DH: There needs to be some magic in the actual creation of the object, since globals in JS semantics are special. There has to be a separation between creating the object and setting state on it.

WH: To make the question more concrete, what happens between when init() finishes and when the super constructor finishes?

DH: Nothing. init() is the last thing it does.

WH: Why not just put the init() user code in the user constructor after it calls super()?

DH: That wouldn't let you do your own default initialization. Going further, symbols could be a nice way to add some more hooks, e.g., direct and indirect eval, for live transpilation. One powerful use case is language polyfilling. Could also be used for security restrictions disallowing some language forms--eval is a potential hole, and hooking into eval can address it. Then, hooking into module loading could be part, but I don't have a concrete proposal.

BF: Should the job queue have hooks?

MM: This is per-agent, not per-realm

DE: There's a new V8 API for PromiseHooks

BF: I want something from JavaScript

DH: This is a clear cross-cutting issue to look into

AK: V8 PromiseHooks are for agents

AWB: We could clarify in this document that Realms aren't agents, aren't workers, etc. Is the idea that the standard global object is *only* the standard, or could it include Section 16 embedder extensions?

DH: I believe either one is fine, neither is fatal; my preference is that it should not include extensions, and that it should only include those things, just in order to design with portability in mind. Maybe I'm missing something, open to changes.

AWB: In your DOM mocking example, it looks like you're setting up leaking of mutable objects; if you want to prevent that, you have to do things differently

DH: This is an advanced API

AWB: An expert feature

DH: If what you're trying to do is to build a carefully airtight sandbox, no amount of API design will make that a full-proof process. If you want to make a sandbox, this will be hard. Currently, today, you can only do it through various hacks, e.g., hidden iframes.

MM: To be positive, this API makes it much less hard than it could have been

DH: But the fact that you can share objects gives a lot of power. Realms plus Proxy gives you a lot of power to enforce boundaries, but you still have to be careful when setting it up.

AWB: You'd probably want to instantiate global.Proxy rather than Proxy/

BF: is the global going to have Proxy hooks?

DH: You could set a handler to the global object, yes, that's part of the design.

BF: The global object has a proxy

MM: the Window vs WindowProxy thing is actually orthogonal from EcmaScript Proxies. It is an unfortunate name coincidence. A so-called WindowProxy is not a Proxy.

BT: I'm curious what your thoughts are on what it would take to do on-the-fly transpilation. Would this be a combination of the Loader's translate hook plus

DH: I don't think the Loader API will be necessary on the web platform at all. I don't think I'm ready to announce that I'll stop work on the loader, but I think we can get all the benefits of the Loader with a radically simpler API. But today's not the day to talk it through.

MM: I am shocked; we'll take it offline.

BT: But you do think the Realm APi can be used to support the on-the-fly transpilation scenario? Which will

be used when you run a top-level script?

MM: it has to be a different one, as the spec provides different rules for "eval code" compared to "script code".

DH: I don't want to talk through all the details, but agree that hooks are an important part of it, and that we should investigate the hooks as a way to look into module loading as well.

BT: This makes a lot of sense to me, that any source text is associated with just one realm--putting hooks here seems feasible

DE: Would it be reasonable to make an MVP Realm and pull off some of these features into a follow-on proposal?

DH: All of these features are necessary to fully virtualize an execution environment, which is the goal of this proposal. Does that seem reasonable?

DE: I don't understand what's required to virtualize an execution environment exactly.

AWB: It's important to think through the cross-cutting concerns first.

BF: This ties into the global module registry; I am not sure it's needed.

DH: I think the registry could be in userspace

AWB: The Realm API was previously in the ES2015 spec, and it was pulled out due to exactly this issue, interactions between module loading and realm hooks.

Conclusion/Resolution
- Stage 1

Meeting Minutes

AWB: By now everyone should have had a chance to review the minutes of the last meeting. Approve the minutes?

WH: Yes, but we need to fix the deleted agenda lines.

AWB: Yes, we'll fix those.

Approved.

global

(JHD)

JHD: We faced a number of web-compatibility issues for `global`, including flikr and several enterprise JIRA intallations. So it seems like the name `global` is dead. What should we do? Abandon this? I'd like to find another name, with global in the name

MF: How about gl0bal?

MPT: globalThisPascal

JHD: Global, $global, __global__, anyway, I'll open a bug and we can come together and make a list of possibilities. This will take a while, since each time we try aname, we'll have to find a browser willing to try it, if it breaks, we'll have to roll it back, hope that trust hasn't been broken enough to try again. Both Safari and FF are willing to try again but I don't want to just try again every two months.

WH: What was the cause of the flickr breakage?

JHD: If a global variable named global was present moment would attach to the wrong thing.

AWB: Where I saw it was detecting what platform I am on.

JHD: No, they were all trying to find the global object to attach to. The JIRA breakage is moment being wrapped with a this value. This allows them to extract moment in a non-global way. Every other usage of conflicting code is looking for a global to attach to.

AWB: The real thing we'll get to is that I think we should realize that feature and platform testing is a particular hazard for any name we know exists in some platforms and not others.

JHD: I totally agree

AWB: In choosing names for anything, not just this, we should be really aware of it.

JHD: For example, the word "process" is forever going to be banned in the spec, it used to detect node. The word "window" is being used to detect browser. The same goes for a long list of other names. That was always a concern with the "global" but I was hopeful people would just find global faster that way

DE: On the other hand, web APIs are frequently adding new properties to the global object without any of the concern that we are having. WASM add one.

JHD: If the web platform groups, standard groups, don't already share something, when we all pick a name we should pick names that are meaningful but not upset generic to avoid collosion. As the language, there are some names... it is likely, using a small english word is going to make more sense for the language than the web but maybe it goes the other way

DE: In the web tey haev to consider these things too. For example for web components we had to change the name of a bunch of stuff. Some code shipped was looking for a name that was an earlier name. These things happen, we don't have to conclude that it would be impossible to add this. If it is specific name added by a platform it's ...

JHD: I agree we'll find a permutation that works

AK: From experience of being a team wthat shipped names like "Entry" and "Fetch" the problem won't e findinga short name, it'll be finding one we are happy with. I have less concern about any short nam ebeing compat.

WH: We will have the same problem with BigNum, trying to find a web-compatible name for it.

MM: You said there were three approaches, so far we've run two

JHD: I make a github issue, we bikeshed
AWB: We do have a long afternoon here with a number of interseted parties

JHD: I want to give Dan some time.

MF: I'd rather not have a public github issue on this

JHD: We can do it on reflector. If there is something we can achieve today and it doesn't conflict with Dan's

later presentation is good. The last option was System.global. Which namespace does it go under? One of the primary objections voiced by Domenic who isn't here today. He didn't want this to create a namespace that doesn't exist yet and may never exist otherwise. If we aren't going to have System.import every, that's a good point. Because the error stacks proposal needs a namespace, it makes sense that this could be shared. That is our third option. My expectation of the web compat for that... I'm sure people are already creating System namespaces and attaching import to it. I'm guessing the way people wrote this code they check for it before they create it. If they have written that sensible code, it won't breakif we create a System object iwth a different property. I don't expect it to be problemantic, both this and the stack proposal would need to find something web compat anyway. Regardless it would be under a namespace. Once we have a web compatible namespace, any property we add is safe

DH: Can I just say, we don't need to have the discussion but I never felt convinced by the outcome of the discussion. Domenic felt really strongly and I kind of backed off. Oh wait, no, it was you that was concerned that Reflect coudln't <missed this>. I was the one who came up accidentally when I created Reflect.parse in spidermnokey years ago. It got hijacked to mean the MOP operations

JHD: When we removed the enumerate proxy trap we removed Reflect.enumerate

All: talking over eachother

MM: As co-other of the proxy proposal, i don't consider that to be a constraint. The privilege separation is a serious issue

JHD: That constraint is the reason wy we removed against my arguement the enumerate method very useful.

AK: We removed it because we didn't like what it did

JHD: No, we removed it because we removed the corresponding Proxy trap

AWB: Yeah, and it seemed like it was included for that. Anyway, is it short enough as System.global?

JHD: I think it is. You often have to do a whole bunch of checks to figure out which of several places to get the global object from. So it could be Reflect.global. Anyway, Reflect sometimes has a little privilege, right?

MM: Everything defined as the primordials in ES2017, zero of them have privilege.

AWB: Reflect is an object, it has a prototype, which is Object.protoytype (of some specific realm). If you give someone Reflect, in addition to the priv of the properties you pass to it, you also give them to the privs of Object.

MM: There is the immutability of primordials, JS makes all of the primordial states highly highly mutable. Everyone sharing access to those can mutate in ways others can see. The constraint we got right in ES6, we got almost right (with two exceptions) in ES5, if you use Object.freeze on all of the primordial objects it so happens that there is no mutable state left. Since we sort of backed into, that was not a necessary conclusion.

JHD: Are you saying that is true in ES6?

MM: There are two remaining exceptions that will remain forever: Date (both Date.now() and new Date()) and Math.random(). I'm talking about the primordials on the global object, not instances that are created later. There are no Map instances in the primordial state. So the fact that Date.prototype was a date and Rgexp.prototype was a regex, there was a bunch of stuff that you used to be able to mutate but were fixed and removed. Date is a read-only channel for finding out the current time. It's access to mutable state, but not the ability to mutate. There is also Math.random, if it were cryptographically strong pseudo-random number generator, wouldn't be a communication channel. Object.freeze is enough to prevent shared

primordials from not being a communication channel.

MM: Well, it seems like a survivable mistake to put global as a property on the global object, but it seems like something that we'd always regret.

AK: I want to argue that we shouldn't be talking about this, I think that Mark was convinced a global object was special enough that we could put global as a property on the global object and it wouldn't cause an issue.

MM: Yes, it doesn't, Yehuda made an argument, he was right, that in terms of the hard requirement, we can survive not putting global on the System object. I always felt it was a mistake because when you do grant priv in a way that emulates the way hosts do, the global object will generally be an object that carries a lot of priv even though its a survivable state that would be something we will always regret. That is why I come back to this issue. It really would be good to have it on System. it IS a hard requirement NOT to put it on an existing primordial because then you mix. Yehuda was right that if you make your own separate global, his argumetn does not defend making it Reflect.global

AWB: The priv you're worried about putting it on Reflect would give access to the global object. If you didn't want to do that you could sensor global on reflect.

MM: If you censor or virtualize it, you have to create a different Reflect with a different identity, which means different code in the realm sees different reflect items. by doing the priv separation we enable all code sharing the same realm to all share the same identities for the primordials. There is no unsafety in the sharing. That means you don't have the identity discontinuity problem. My array is not an instance of your array among separate code in the realm

AWB: Why is it special?

MM: It isn't.

DH: This is a key thing that I missed in the past. Why is Reflect special? it's actualy true of almost everything. Reflect should not be specail and suddenly have power tools attached to it. There are none in the stadnard library at all

MM: There are zero in the standard library

AWB: Why wouldn't some new namespace object like a hypotehical system, why isn't that an isuse?

MM: The reason is because thats the purpose for it to exist. Its not unexpected, it's not a problem for it to be differently privileged.

BF: I'm curious, we have spoken about this before, about importing a powertool on my side, curren tscript identifier. Would this suffer the same problems as an import you could never mutated?

JHD: If you want to lock down a realm oyu need to change waht the global this eappears to be. you can't do that if you import unless you can virtualize it or change what the import is.

MM: this goes back to the same issue we had with the stack thing. if we do have a way to have a way to have built-in imports such that it ... we can distinguish in an ergonomic way with priv imports and non-priv imports, global could go into a priv one.

JHD: As long as you can virtualize it (or anything that needs it)

DH: That requires not only that we have worked out all of our mechanisms for putting std libraries in or systematic changes to how imports work.

BF: If there are well defined specifiers that shouldn't be true

DH: Currently the spec says nothing about the string you import. This is changing modules to indicate some are special

DE: There is an idea to make an object literal that is associated with the import, you could put a crypto hash of the file you're going to download, you can put other tokens there. if you want to put some more key values associated with an import there are other reasons why this would be good to have, e.g., for CORS or SRI

DH: As the champion of modules t would be nice to be looped into that

AK: We did

DH: not once

AK: We're looking for syntax for various things.

DH: so you're talking about CORS. It would be nice to be looped in here

AK: The reason we're looking for a thing that sounds reasonable to us before we try to make it reasonable to you..

DH: That's fine.

DH: I apolgize, that's legit. We should not block this on path dependence

BF: Domenic is only not in favor of System, he did not say veto it.

JHD: He's opposed but won't die on a hill for it

BF: He says it is minor and he can talk about it later

JHD: I am not seeking a stage change. I am looking for a direction from the committee with the hope in a meting or two we have two implementations that are web compat and we can move to stage 4.

AWB: I think whack-a-mole option is the best. This case, a simple global name that doesn't conflict with current use would be the right thing. this is a special thing which has essentially in all implementations already been given a simple name. We can come up with a non-conflicting global name, that is preferrable to a namespace that will never be used again.

DH: Afer I said let's use System, I changed my mind. Will we really create a namespace for something that will only have one thing?

MPT: Can i propose a technical solution to this problem? I think I have the infra, but has anyoneever craweld all the files on github and ...

JHD: I did that as a part of the initial search, where I looked through 20 pages of GitHub search results, that gave me the false confidence about global. Github doesn't search commit histories.

DH: There is an enormous amount of closed source too.

MPT: We can do this.

DH/JHD: that's just a smoke test

MPT: I was curious if it had been done because we have the resources to do it; we have better infrastructure than just looking through GitHub search directly in the web UI. It takes infrastructure.

DH: Mark has a rebuttal for the namepsace for one item

MM: The thing is, there are five existing proposals that would populate the system object
- weak references
- default loader
- getStack getStackString
- global

MM: The abstract point that this is an example of. Over time, these things have previously been held back, waiting to have a conventional place to put things that grant priv. Of things in this category one of them has to go first. Whichever one goes first, with the other ones not yet being accepted the natural criticism is that we'll end up with a namespace with one thing on it.

AWB: Doesn't this create a catchall place? it's not clear to me for the things you listed would have the same priv.

MM: it's not that they have the same priv it's that everything else has none. If you want to differentiate between the priv of things that are otherwise in the same realm, this is a reflection of things you need to virtualize or censor
AK: That is three that exist

DH: it's helpful to talk about what they are concretely. When you think about what they are, System doesn't make sense together concretely. The name System was about querying the system to find what modules are installed.

MM: Well, I don't care about the name

JHD: If we agreed that there's a privilege-granting namespace, we just need to choose a name.

DH: Mark has been arguing for this for a long time, so we should respect this and not reopen it. There's a lot of security-critical code on the web, and it would be helpful to have a way to segregate the things that you can and can't expose to those things. There's work in maintaining that constraint. Given that that's the constraint driving this, we should consider this as part of the name--the name should reflect that it's the power tools.

AWB: Ring 0

DH: Let's take a name that includes this.

DE: If the web is adding random properties to the global object that grant privilege, don't we need a whitelist rather than a blacklist, unless you depend on the Realm API?

MM: The Realm API is a core part of the plan

JHD: Should we start putting new Web APIs in a privileged namespace?

AK: I don't think it's likely that the web would do that.

DH: We have restricted privilege in JavaScript, so it makes sense to separate this.

JL: It seems like this privilege doesn't correspond to what users would think.

AWB: It seems strange to say SuperPrivilegedObject.global--who'd want to say that

MF: The global object isn't typically being used to say like, i for iterating over an array. It's a system level thing it is seen as a priroity.

JHD: People access the global variable a lot because historically we had no module system.

DH: I think this is the right word: internals. It is in a way exposing some of the more internal... it has a hint of power, but not off the usual path of programming.

JHD: Sounds good.

DE: How about Util. No, joking.

AK: I don't think the global object is internals, it's right there

JHD: it's privileged enough to coordinate

BF: Why can't I freeze it then?

AK: That's a separate question

JHD: It is somewhat internal, it gives you direct access to things...

AK: No, no, i can access everything on the global object without touching it

MM: I do want to correct a historical thing about what Dave said. 8 years ago when JS was ES3. The almost perfect separation of priv between JavaScript and Host defined things was an accident. After 8 years of defending and strengthening this separation, t is no longer an accident.

DH: Sorry, I meant an accident before you came to town.

JHD: Let's say we go tentatively with the name internals, with the understanding that we can sitll bikehed.

AK: I think we should find a name for this thing, not separate the discussions. I think the global should be a global name and we should put the discussion for the "thing" for stackTrace is a different discussion.

MM: I strongly want it there

MF: Why not put it there?

AK: First, we have to agree on the object. I want the global thing in sooner, the longer we don't ship it the less purpose there is.

MM: I think we should skip shipping it earlier vs putting it in the right place

JHD: I would wager to say if we can agree on the name for the object t will not slow this down

MF: It will speed it up, we won't have to be so careful

AK: I'm not talking about the spec text

DE: none of the proposals up there are anywhere as far along as Sytem.global

MM: something has to go first

DE: it's more than that if we have two proposals at stage there and they were both not moving because System wasn't there yet. the others ones are much, much less, they are stage 1.

AK: except the one that isn't going to be a sthing?

MM: I want to hold up global until the other ones got to stage 3

AK: That is exactly why we need to have this discussion. It happens that for whatever reason some of these discussions around priv granting...

MM: even if none of these make it, this will not be the last thing that will need to grant priv

JHD: we can in a practical sense we can say in 5 years we will have this and we need to reify it.

AK: If that was true it would be in stage 2 now.

AWB: So we have been talking about System for more than 5 years, youc an't just...

JHD: Speaking a sa champion for the stack proposal, I didn't ask for stage two, major semantics are not specified, it's not ready. I want it to be ready. I want Mark to take this up. My goal is to get that done quickly. In some number of years it will be finished and something we agreed upon will ship. it's already compat.

AK: it may not include the stack or getstack trace. We may expose the stack trace in other ways that don't include System. I'm partially making these arguments for Domenic.

JHD: I hear what you are saying. There is a slim change that getStack and getStack string will not need a priv granting object. The real question is.. if we likely will have a priv granting object, let's assume its going to happen and some stuff will be on it. At that point will we regret not putting lgobal on it?

AWB: no, global doesn't need to be there, it would be perfectly fine

DH: A global is already always accessilble to you. The idea that we woudl take this thing that is already available and is the one major thing you need to blacklist to get attenuated power and pretend you were hiding it under a power tools namespace is silly

MF: Global is not always available to you, environments without eval don't have it.

AK: it is available to you there as window

JHD: Browsers with CSP there is, but there are browsers that restrict eval. AK is right, it is available as window.

DH: The top level this is available as window. The other point is that CSP is not the normal intuitive mode of JS. I'm not saying it's bad, though I have my issues with CSP. Naming is about what is the appropriate intuition for people. This makes sense in an environment where global has been censored but thats just not normal.

AWB: If you can write a script you can script script that will access the global object without eval

DH: the idea that the power tool could have a reference to itself without needing to have a reference to a special power tools namespace to intermediate it seems natural.

MM: So this is Yehuda's argument. I think the argument does work in the sense that there is nothing fatal about that decision. it's survivable. it's one we will regret because it means when you are virtualizing you have two global names you have to start with instead of one?

DH: no, because you don't actually have to prevent access ot it, you just have to prevent that object from having power tools, you don't need to censor that name

JHD: you're saying you have to strip things off it

DH: you create a realm stripped of all power and giving reference to it doesn't grant power

MM: You want to create, when you have multiple degrees of priv in the same realm they are going to see different global objects and different "System" objects, if global hangs off System you start with System.

DH: You create a new realm and determine what's on ..

MM: I'm not talking about creating a new realm, i'm talking about creating a new priv level, separation, within one realm. We want to avoid the my array is not your array problem. We want multiple priv levels in one realm is to avoid these identity discontinuity problems. The plugin scenario is the problem where you don't want that identity discontinuity if there is no other reason causing you to do it.

DH: In those situations how are you censoring it? Are you deleting things?

MM: No, you're.... you're creating an evaluation context for the separately trusted code such that what it sees different objects as the global object and the system object,

DH: you're doing this without a realm api?

MM: this goes back to the discussion of separating the realm conversations from the scope conversation. you create the evaluators in the same realm that saw a different global object at the end of their scope chain

DH: In both cases are you not giving them a restricted global object? the case where you have untrusted code is where you restrict global. global is no longer powerful in those case.s why do you need to censor access to it at this point?

MM: You're making Yehuda's argument, I agree.

DH: I'm saying youre NOT doing. Your describing a use case where they do have access to it. The argument you made a minute ago was restricting TWO things. I claim that is false. YOu will not remove access to the .global object because it is a self reference ot this power restricted thing

MM: In the case of the namespace, you still have a self reference which would be System.global refers to global, it's two step self reference but it's still a self reference. The... by making th self reference a separate global variable from System it simply means that you have to, when you're making this new priv thing you have to hook up both self references directly rather than uniformly replacing it.

DH: I'm saying that if you design these APIS to hook up the references. Lets talk about the realm api and you can tell me why its not going to work. one realm proposal is that it initialized the global object with "all th usual things" for this new embedded realm, but not <mised it>. One of the things that does is hook up a global.global object and you have a power attenuated thing now, it just by default does what you want. The only thing you have to remove is still access to "System" or whatever it is called. WE could go further to say Re

MM: that's very interesting... I think you've changed my mind;

DH: That's awesome. I think I agree with Domenic's perpsective. It's nice to have a global

JHD: Does that then mean we have consensus and we'll bikeshed on the choice of names and once we find a webcompat on we'll use it?

MM: I don't want to say consensus on the approach yet. Dave having convinced me is less than a minute old. It took me weeks to think about Yehuda's argument. I want to live with this for awhile.

JHD: In the meantime, are you comfortable with a reflector issue.

Conclusion / Resolution
- We will look for a web compat simple global name
- We will make a reflector issue to bikeshed

Public and Private Class Fields (Daniel Ehrenberg, Jeff Morrison and Kevin Gibbons)

[slides](https://docs.google.com/presentation/d/1yXsRdAJO7OdxF0NmZs2N8ySSrQwKp3D77vZXbQOWbMs/edit)

DE: Kevin is joining the champion group for private state. There are a number of things we are approaching consenus on. I don't want to say we're at consensus but the public and private keyword, Allen you've expressed interest in this. At the last meeting we discussed if we should have this in addition to the sigil we have to have. A few arguments for this were that we don't have it for methods and you really want it when you're learning but after you learn you don't want it. The second one is the stronger argument. For the token we talked about different possibilities, it could be = or :=. It seemed like equals was pretty agreeable.

WH: By "token" here and on the slide I assume you don't mean the token that starts a private name.

DE: Correct. That would remain `#`.

MM: Mark laughing :)

DE: (Reviewing slides linked above)

AWB: One of the things here you're implicitly saying as you go down this path, when we designed classes we decided in tc39 about whether class bodys were statement list like or object literal like.

J: We went statement list like, but we do allow comma delimited to declare multiples

AWB: I undertsand, you have contacts where you have keywords where a comma separated list is very natural in the statement context. if you don't have a header on the statement context it looks like somethign else.

JM: I just wanted to point out also that I know the class bodies are originally designe around statementlists. They aren't aren't that now though

AWB: They are statementlist like, in particular one of the things we wanted to allow for was the potential for future inclusion of declarative satements like let/const or even function. That could have useful meaning, particularly now that we're adding lexical contours to these things.

DE: This syntax wouldn't preclude any of those issues

AWB: It makes the syntax less....

DE: YOU have a bare thing but i tisn't followed by a brace? What's the issue?

AWB: You have bare things that look like assignment statements;

JM: You're suggesting the alternative is a keyword prefix interspersed?

AWB: The alternative would be to follow the pattern we've used for essentially all other <missed it> except concise methods.

DE: This is really about the first point in the three (no public/private keyword)

AWB: They all tie together.

DE: How?

AWB: Common declarations would naturally follow, they already have comma separated lists, it would be

surprising if you couldn't

DE: I don't know if I call public/private keywords are like var and let, they modify a declaration.

MM: Allen is a a short summary of your position, no comma unless we also use the leading keyword?

AWB: I think that might be less confusing if that was the case. I like comma separated lists.

MM: I just want to understand. Are you only saying that we should have both the leading keyword and the comma or are you saying the two decisions are tied together we can have neither or both?

AWB: The decision space includes all possible options. A set of consistent decisiosn without even looking at something, making something good an intuitively guessed, i would say yes.

DE: You would be okay if you could do class x=1 and anew line and y=2.

AWB: I will always be sad that there isn't a keyword in front of those things but I would prefer if you're going to go that route that yes, it not include commas.

DE: I think we could defer commas. If you do put a comma it's... the real motivation is for the decorator proposal to have a more convientint syntax. Yeah, you could do this but I don't see it as being really important.

AWB: YOu've had public feedback on this, right? I believe the public feedback, my sense is there has been feedback o both sides.

DE: Most people said please stop the proposal, we can stand the sigil

AWB: YOu have to have it

JM: To be clear, to drop the sigil is specific to private state.

AWB: The problem is weather for example you have a declarator or public or for that matter "own", we've had quite an extensive conversation about why you might want that keyword.

DE: IS there an issue on the public repo about that Jeff? I have feedback on the private state.

JM: I don't think there has been much discussion

DE: In the private state proposal there is a whole issue of having the private keyword and the sigil ... some people are kind of supportive but most people want the whole proposal dropped. Adding the private keyword would maybe help, but there hasn't been a lot of constructive feedback.

AWB: I have seen people talking about this, the readability of it. Just readability

WH: *If* we have a keyword, it should not be public or private — it should just be something like `var`, `let`, or `own` — but I don't think we should have a keyword here.

JM: I think var/let will not fly. They imply variable bindings/lexical scoping/too much misguided familiarity.

WH: I'm just saying that if we're required to have one it should be one of those but I don't think we should have one.

J: I thinkw e have a consistent design without them

AWB: Final though, if you could go with own you could use it for both purposes as the sigil. It's private and otherwise a rpoperty.

BF: I'm hesistate because we have all of these getOWn properties.

MM/AWB: It's the same meeting

BF: it's not about privacy

DE: the own is not about private properties

AWB: own means it goes on the isntance withou tthe sigil it's a property, with ti, it's a field.

MM: I did misspeak before. I was confusing the concept of being per-instance vs the concept of being an own property. Private state is literally not an own property but it is a per-instance property. I think the way people understand own is the major intuition is that it is directly on that instance as opposed to being something that is inherited. I still think own is plausible and it actually fits will with the story for why we introduced static. The keyword is distinguishing which object are you declaring something about. It also fits with the idea that static could be equally orthogonal. If you said <missed it>. The idea that the sigil is orthogonal to the keyword...

AWB: Things without it meaning normal concise methods are on the prototype

DE: I have in the next slide, even more meaty issues that we havne't resoled. Maybe we're nowhere near consensus. Maybe I can go through the presentation and get to the even more difficult questions and we can come back to these afterwards.

AWB: One of the reasons I pushed you on this and making this explicit across the things is that it really comes down to a basic cross-cutting thing. how are we origniing class bodies and other things we 'll have in the future?

DE: About the corsscutting design, if you want to talk about htat, having a function declaration inside a class, i think a nicer thing would be a private method.

AK: let us move forward with the presentation

DE: comma declarations could totally be dropped to deal with these issues.

DE: Fields would be added after super. We talked about maybe you should create the instance with all the fields in them up front already added regardless if they are public or private and then fill them in. IN this model whatever the upser constructor returns, that is what gets added as properties or as private state, it gets added ot that thing. The last part of the semantics is that the scope is similar to the scope for a separate method except there is an early error if you include arguments. If you include arguments you get an empty arguments object... the rationale for making it like amethod is because we want stuff like this and super property access to be possible. Methods have all that stuff and this is good model. I was at some point we would want a runtime error <unable to keep up>

AWB: For built-ins or host objects that want built-in like performance, it's still possible for all the really priv fields to be allocated as part of the C struct that...

[bikeshedding]

[set vs. define]

WH: It needs to be define. Set cannot work for const and such.

JM: There are pros/cons both ways: Users might expect to be able to call a superclass's setter by putting a field declaration in a subclass.

WH: No, that's not what one would expect to happen. A declaration is a declaration.

JM: OTOH: Set is incompatible with supporting const in the future

AWB: There were exploits for object literals. We had to change them from set to define.

MM: The thing that was already widely deployed, Dan is right, as an exploit it was narrow. There is a different way of looking at it, at the time that all this was an issue. JSONP was already in common use. JSONP loads text that presumably is in a JSON format and does or does not pass it through validation and then evals it. The issue is not when the source of the JSONp is the attacker. The issue is as dan was saying is the asme origin privacy protection. Where the origin that is loading it is the attacker and the external source is the defender. What they want to do is provide data that the.... it can't be JSONP, i'm not able ot reconstruct it, i'm sorry.

DE: JSONP is already a huge sameorigin violation.

MM: The issue that Allen is raising, with regard to something can be labeled exploit or security hole, i think that the exploit revealed an immodularity, an anti-modular action at a distance that was surprising. this.foo = 9 is not suprising. a declarative initialization acting as an assignment is surprising. It causes a non-obvious coupling with two separate pieces of code. Being surprised that things are separated is less dangerous than being surprised that they are coupled.

WH: It's the same as the `with` problem. We banished `with`, it has a nasty property that things outside the scope could affect the meaning of assignments. Using set would unleash the same nastiness here.

AK: This particular issue has been discussed I can't count how many times in the last two years. Is there something new?

DE: That is a good point. I want to figure out how we come to aconclusion on it. I think the preference of th echampions would be define?

JHD: Unless we ask people to raise hands....

DH: I don't feel super comfortable trying to reach consensus with people missing

DE: I'm not looking for consensus today, i'm looking to the committee about the fact that we have a split committee

JM: I want to make sure that as we talk through this, the reasons we are proposing/objecting to these things are clear. If the need for define is clear and the need for set is clear we need to be careful not to rathole on either. I guess what I am saying is if there is clearly a tradeoff and neither is perfect lets lower the bar on how important it is

??: Has there been any discussion about making it an error to define a property in this way?

DE: We did talk about this at som epoint, we would check if the property already exists and make it an error to overwrite. It would have some performance costs, it would have to be dynamic. I

JHD: it would also mean every public field I added to my superclass was a breaking change

??: True already

DE: it would help for exactly that breaking case, you'd see the overlap. It's not a foolproof check, you could do a property check and defineOwnProperty, but proxys could interfere as well..

MM: I don't think if you are subclassing the superclass and it's attacking you, or going out of its way to be nonintuitive. You are facing a very difficult threat model one that I intend never to face. I think that

subclasses should only subclass superclasses that they are willing to trust. If you can't trust a class, compose with it. If there is something above you on the prototype chain, trying to defend yourself down the chain is very difficult, i just avoid it.

DE: I wasn't talking about security

MM: That's actually the point I want to make. The coupling is the problem I want to point out. As an engineering concern, the fact that you get coupling through a proxy, i'm not concerned about that.

MS: Just to follow up on the irreducible cost side of things. If its only an own field that we concern ourselves with, the object you get back from the super should have a fixed shape and you should be able to bypass.

DE: Sometimes... its an improvement over the status quo

MS: not saying you can always avoid them, but in the common case you can bypass the costs

AWB: how do you know that?

DE: We don't know that but we could have an inline cache that does a combination of verify that has this in class which doesn't include the property from before and then add the property. We could say in spec language if has OwnProperty throw a type error, next define. That allows define but avoids the bad expectations about set being called.

MS: if you can get additional safety from having the define behavior it seems worthwhile

DE: I don't know if that will make people who want set happy. Some web standard had in the explainer or example, it had an onMessage thing which was just using the method and expecting it would call the setter and work.

MM: I think there might be a surprising three way coupling of design issues that i haven't seen before. let me ask for the people who prefer set semantics. I want to raise this question. I want it in the notes. If we did use the own keyword, such that the syntax were own x = 8, would you agree if the meaning of that were set would be confusing. because, if the meaning is set, the result might not be an ownProperty

JHD: I agree. I hadn't thought of that. Would the define folks feel comfortable with own foo = bar being define and ? being set.

WH: No, we should not have two of these

DE: We could have = for set and := for defined

: the semantics i need are set, i don't want a constructor

AK: The best advice I can give you Dan, choose one, go convince the people that want the other way

DE: I have tried to do that and I have failed so far.

AK: I'm not sure this is going to get us there, most of the opposers are not here

JHD: let me clarify my position. if the token is = it must be set, full stop if the only option is define, it's subpar but acceptable. if both are available, i'm also comfortable with that.

AK: its really best not to say things, i will not

JHD: I apologize, I meant full stop as end of statement, not that i wont entertain discussion

JM: Babel has had for about an equal amount of time both set and define because originally the proposal

was set. In neither case have we had any confusion. This is actually rarely a concern either way. WE should probably focus on concrete concerns given that we have tradeffs with pros and cons.

WH: I find these things analogous to lexical scoping. If you're inside a scope and you have `let x = 5`, I would be very surprised if in most cases you got x with a value five, but if there were a setter of x in an outer scope then the setter were called. That would be utterly bizarre to me. This is reason A why it needs to be define. Reason B is that const and types and such will not work with set at all, and there is no way I can see to make them work.

DE: As to the multiple options thing, :=, = or own =

WH: I'm willing to be flexible with the syntax to use for initializers to make it obvious that they are defines.

DE: Multiple versions is not sufficiently motivating, its' obscure.

DH: We would need a very strong justification to have the two. The bar is pretty high for that. You were looking for advice on how to break the log jam? I've often found that getting people back from the statement of their position to the constraints that lead ot the position, taking a pass at those can help. You have to collaborate well, but it might be worth another pass, really trying to get people to state why they hold their position. This isn't a thing that's going to go great on a github thread but it might be useful to get a clear.... if you feel we've reached a stalemate and you can find the criteria to resolve it. Maybe dig a little deeper to see what is leading people TO their opinion. Its' not surefire, it's a practice I use.

DE: Concretely here, I agree, what is the right way to dig deeper.

DH: Take people who have had opposing opinions and try to capture those opinions.

DE: I did this.

AK: For me jordan didn't do a great job of articulating it, do you have a fully articulated version of his concerns? Is it just that = seems like assignment?

DH: That's worth engaging with. We have multiple uses of = in JS and they don't always make sense.

AK: In a param list in a function it doesn't mean that

MM: WheN I brought up the point of own with = i was proposing that the own be mandatory with the consequence tha tthe confusing expectations go away.

JHD: That falls into the other thing I said, if we get a define semantic that isn't confused with assignment, fine. It's not ideal, I want set, if I need it I have to do it in the constructor. That isn't an obstacle to the proposal.

DE: I thought if we only had := and not = it would be unacceptable to you.

JHD: I'm sorry if I phrased it that way, that is subpar but I don't find it unacceptable.

DH: The other part that might be helpful.

CM: This kind of ties with what Jeff said. I'm concerned about our awareness of what users think. We have an availabililty bias by talking on github. Do the typical javasript developers even aware of set and define.

TK/MPT: No

CM: I would submit people think it is a set and end of discussion

JM; Let's acknolwedge that we can go rounds on this speculative and subjective thing for a long time. The

compelling thing for define isn't the tradeoffs that go away. There is a real future compatibility problem with const in presence of going with set, that was the convincer for me that define is the way to go. That seems like it should be stronger than these concerns that are hard to substantiate.

DH: Jus tto follow up, set vs define is a red herring. Nobody including half the time experts of the spec are thinking at that level. They are working in the conceptual space they are in. No matter what, they have no expectation that it will trigger something up the superclas chain. As I am writing my code am I thinking at all about the fact that a field Iv'e described could be triggering a setter?

AWB: We've seen this over and over again in writing spec text, people don't think about it.

WH: This is the `with` problem once again. Users may not know the difference between set and define but don't expect their variables to be stolen by outer scopes.

AK: If your'e tyring to get set behavior you'll have to go learn about it

MS: A lot of the confusion goes away if there is an error

JHD: All of the existing complaint code with the spec will have this.foo = bar in the constructor. If I try to use public fields, if I am subclassing a library using a setter on the protoype chain, that will be confusing. Maybe i'll just move stuff into my constructor, but if I do this.foo = bar brings about the problem again. The mental model is that I think users will hold is that this is something I am taking out of the constructor and putting in the class body.

WH: That argument means you can never improve the safety of the language.

MM: I want to probe that intuition again

JM: Refactoring expectation hazards do seem like a pretty compelling argument aginst define. However, do you really think that concern would be mitiagted by a := or a keyword?

JHD: Yes, that would mean I need to learn new information and this information can describe set and define? Looking at other code that is already doing it won't show me about those differences.

AWB: If you've said this.x = 5 in a constructor where you've inherited that class with a setter that isn't even creating an own field.

JHD: it could be.

AWB: the constructor is probably setting...

DE: Can we move to the next topic?

DH: Can we try to actually... along the lines of what Maggie is saying, take some examples and glean from them what you think the intended meaning is. I fear we too easily get into abstract intuitions.

DE: Not clear to me

DH: maybe the problem is that nobody ever uses setters

DE: We have them all over the place

JL: it doesn't come up in practice much

DH: What I mean is more, look at real code and see... put yourself in the position of the person writing the code and imagine refactoring scenarios. Put yourself directly in the drivers seat of the code.

DE: IT's not clear to me how to do a representative sample of code.

JHD: is this an example? (displayed a constructed code sample that showed the observable difference between Set and Define)

DH: no, this is not a real codebase, that is contrived. If you look at a real codebase you can see where the author's head was when they wrote it.... nevermind.

BT: Also want to say you could look through a bunch of typescript code to look for this pattern. I would not be surprised to find this, TypeScript does set. I wouldn't be surprised to find code that depends on that. That could be interesting data. Typescript has had set semantics for 5 years now and there has not been a single issue filed on it.

JM: I really do think set an defined are pretty equal tradeoffs.

MPT: Can someone in the room?

JM: If we start looking for examples, we'll look for examples that follow the semantics of the platforms they use. It's not going to give us a real sense. especially if we're worreid about refactoring. I don't think this will move us forward on this frankly subjective discussion

DE: Another thing that would be hard to me to figure out from there is that Jordan's intuition would have different refactoring issues

AK: Own is a nonstarter and := also is not because why is that there. In the future, have a narrow set of things, don't ask all the questions all the time. You have narrowed on what you like to do. Maybe you can figure out how to move that forward convincing those we are not in agreement.

WH: I don't think we should dismiss `own` as a non-starter. It could be a compromise solution.

MM: I'm leaning towards own

BT: I like own

MPT: All I was going to say, not that I don't know the difference, can someone write why the average developer would want define over set? Wouldn't people expect setters to run, based on the fact that most people don't really get the JS object model and the other operations that you might do?

BF: I can speak for Node core, that we don't want lots of interception points for setters to be called in places like this

MPT: not for people who know a ton about JS

DH: I don't think it's about an understanding of th eoperations. Its about what I'm thinking about when I'm declaring a class or a field, i'm declaring something on this class, i'm not thinking about the inerhitance hierarchy. It would be really suprising if something someone else code changed what I was trying ot set.

MPT: That's all that matters to the average developer?

AK: this is what Jeff was saying...

BF: We don't really know what the average developer expects. This is not something we should be stating. This isn't about intuition. This is really about what are the advantages of these things. I think it should be define bceause basically working on node core, people do things do their prototypes and it breaks internals because node hasn't fully harded them. It does things in very suprising ways either to the developer or people who are.

MPT: At the end of the day we're asking if people are worried about a setter climbing the inheritance chain.

DE: In this committee we have to make small decisions all the time about which semantics are better, as experts

MPT: Having heard about all of this, go ahead and let it be Define, you have good implementation details. I don't think this whole own := keyword helps the average developer does not care about the difference between those two things. They have to go figure out what that new = is in JS. Just let it go.

WH: You asked earlier how this would affect average developers: When we have const, it would be very strange if we either cannot add it or have to make it work in a different way from non-const.

JHD: If const foo = bar uses define and foo = bar uses set, I agree that would be confusing.

AWB: As was mentioned when you get private into this, it doesn't have the equivalent of set. It's always going to be like a define. Maggie is talking about consistency across the whole thing.

(Topic: reflection of private)

DE: Should private state be reflected? Our current view is "no", but you can define and use a decorator that will allow reflection. Brendan thinks perhaps we should swap them.

WH: No, we should not swap them. Hard private should be the default. I don't want to have to write `IReallyMeanPrivate #privateFoo` to get a hard private field.

BF: Node wants hard private.

DE: Next question, when should static initializers run--top-to-bottom, left-to-right, or afterwards as discussed in Munich?

DH: We need to deal with how this will affect decorators.

JM: we should be able to reference the class name in the initializer

DE: If you have a date class and you want to have a default date as a static property, you could have it be that the date class is visible in the scope where the intiailzers is executed, you'd be seeing an unfinished instance of the class.

AWB: There wasn't a lot of reason do this one way or the other in ES6. We did not want it to be visible during evaluation in the extends clause. Computed names, eventually, if I remember right, we removed all kinds of error checks because we also rmoved them from object literals. I suspect we could bind the name at class definition time before evaluating any computed name expressions.

DH: What you have prototyped in V8, makes it impossible to write `class Date { static epoch = new Date(0) }`? That's basically fatal as far as I can tell

AK: Correct

DE: This is not even fully checked in

DH: I'm just wondering if people are saying it's on the table to say that you can't reference the class in the initializer.

DE: If you can do this, you'd be observing an unfinished version of the class. Maybe for the restricted case of just the static things, since they are properties on a single object. Maybe that's okay? If we made them visible for all the computed property names, then that would allow the computed property names to observe a very unfinished class.

JM: We have talked about this in the past and agreed that the partial initialization is a little weird, but ok for static fields

AK: Is there anything else happening other than adding properties to the constructor?

AWB: If you refer to the class and you try to instantiate it that probably wouldn't be good

AK: To be clear, the function is done being created; the constructor is there syntactically, and it's other properties of the class that would be added later potentially

BT: You can instantiate it fine... it might not have all of the static factory methods on it

AWB: it might not have the methods populated on the prototype

BT: The method should be

JHD: The way we did it last year was the final step was static fields

BT: The Munich plan was you would see the class without all of its static fields, you could observe them getting added as the static initializers were evaluated. Everything else about the class was done, including decorators.

AWB: Computed names would evaluate as part of the instantiation.

BT: computed names was the first step, and evaluation of static initializers comes later.

BT: this is why the class finisher exists, so that static properties are evaluated after the class decorator (you don't get a totally unfinished class that you're using in the initializer) but before the finisher (so the finisher could, for example, freeze the constructor)

AWB: I'm sure you can come up with an ordering.

JL: I don't want to speak for other implementations, but having all of these passes.... reasoning about the code becomes much harder.

AWB: I raised... the intuition of users looking at a class definition that expression execute in order.

BT: That doesn't exist.

DH: I'm not sure how useful this conversation is going to be. We can start. We're about to do the same thing we did last time. In practice a lot of these expression are totally side-effect free, like, look up this symbol. The order in which side effects appear will be totally unobservable because there are none. If we go for the simpler evaluation, of things that in practice aren't going to have side effects (like names). The class is not in an appropriate sate to do common things you're trying to do. Many more programs will break as a result. I can't use a freeze decorator, for example, because I don't have access at the same time. The multi-stage evaluation order, being the thing that allows you to do the most number of things, decorators, as well as being able to refer to the class in initializers, will let things people expect to work. Yes you can do convoluted side effects in those expressions but I claim in practice that is going to be less common than I tried to write static epic = new Date, and something happens because it didn't get the decoration or it was in the TDZ. I was happy with the Munich plan, this allows the class to be in the right state at the right time.

AWB: As long as like thing happen in intuitive order, for example, the decorators in fact fire in order, or the own field initializers or if you have comma... as long as those things locally happen in order I think we're okay

DE: Do you have anything to add to this point?

J: I don't have strong opinions. I don't know about decorators but if people are going to want to refer to the class.

WH: C++ has struggled with this a lot, creating multiple phases of evaluation. It's a very complex set of of problems. After a great deal of work, their solution still generates some really nasty consequences: you cannot take the sizeof of a class when defining a constexpr constant within the class, or you can't call a constexpr function defined within a class to define a constexpr constant within the same class.

AWB: It's why in earlier proposals I chose to exclude static fields, because they start to introduce many of these problems. It's really complicated and the staging..

DH: To be clear, I'm representing my best interpertation of the problems but we do need to take those requirements into account

J: I'm not going to hold things up because I think left to right is better.

DE: You're the expert in the world right now

J: Implementing it with several phases is obviously harder. I don' tthink it will be impos

---disconnected for approx 2 minutes---

DE: So, great, we are sticking with the munich plan but I am glad to have talked thsi through, for a number of months this has been outstanding.

JHD can you restate the plan of record?

DE: As we discussed during the break, one approach would be to use a keyword to distinguish the location of where the defined thing goes: none → prototype, `own` → instance, `static` → class object. To avoid mistakes caused by forgetting `own`, for variable definitions we'd allow `own x = 5` but not `x = 5`.

MM: Allen and I are planning to actually write up this proposal

DH: I believe it would be good to have a future face-to-face discussion about this

DE: What if it was a pull request?

AK: I have a random temperature taking that most developers don't know what own means.

DE: Will it be googable?

MPT: I think that 30% of developers know what own means.

MM: That is way higher than I would've expected, that's great!

(folks talking over each other)

MF: Did I miss it? Did we ever establish on the evaluation order is actually web compatible?

DE: That is a legit question.

BT: There was only one questionable issue.

DE: We have not shipped it, we don't know

BT: It only impacts if we can decorate object literals

J: It also affects consistency

MF: We don't want object literals and class declaration orders to differ would we?

DE: We'd need a champion to propose the real spec text for something driving the change in object literal stuff

DH: Brian and I will look at getting spec text written with Yehuda. We'll do that as quickly as possible.

DE: Thanks for bearing with us on this difficult topic

DH: Thanks for driving it!

Conclusion/Resolution
- Sticking with plan in Munich meeting for order of evaluation (static variable initializers run after computed property names)

- - BT to work on separate spec text proposal for the object literal ordering change (computed property names before values) to match Munich plan

- Eagerly awaiting a proposal from Allen and Mark about the syntax issue. This might lead us to make Define acceptable semantics
- Work is ongoing on the Babel implementation of private state
- Proposals remain at Stage 2

AWB: That is the end of the Agenda.

ECMA-404

CP: The Japanese raised a long list of objections. Allen wrote a diplomatic and brilliant reply. A bunch of editorially corrections need to be made to the document.

WH: Is the list of objections longer than the standard?

CP: Yes.

AWB: Close

CP: Some of the objections were minor editorial mistakes. Some where small changes requested that would break the world

AWB: One or two was like, the scope of this should be something much different

WH: In all the years I've been here the Japanese delegation has consistently been very diligent. I'm very impressed!

AWB: Yes, but not until the very end when you're done

MM: Yeah

AWB: Also with 404, there was a bunch of bickering in IETF that ECMA didn't normatively reference standards enough

BT: Wait, there are politics around normative references?

AWB: Oh yeah

BT: Should I be extracting... we normatively reference unicode. should I extract tit for tat?

MF: At least some new emojis!

BT: we need a sigil seagull!

DH: We should have JS emoji

?: And a WASM one

SYG: TC39 hats!

Conclusions / Resolutions
- Normative references in spec text are apparently political across specification organizations (at least between IETF and ECMA)