Minutes of the:	*57th meeting of Ecma TC39*
in:	*Portland, OR, USA*
on:	*21 – 23 March 2017*

1 Opening, welcome and roll call

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

Due to the absence of a TC39 Chair **Mr. A. Wirfs-Brock** has welcomed the delegates at "The Leftbank Annex" in Portland, OR, USA (Ecma/TC39/2017/005 Venue for the 57th meeting of TC39, Portland, March 2017).

Companies / organizations in attendance:

Tilde, Apple, Mozilla, GoDaddy, Google, Microsoft, JS Foundation, Facebook Intel, Inria, PayPal, Shape Security, Salesforce, Bocoup, Igalia (Guest), Brave (Guest), Deluxe Corp. (Guest), Babel (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	Yehuda Katz	Tilde	Member
7	Brendan Eich	Brave	Invited Expert
8	Sarah D'Onofrio	Microsoft	Member
9	Mark Miller	Google	Member
10	Peter Jensen	Intel	Member
11	Alam Schmitt	Inria	Member
12	Leo Balter	JSFoundation	Member
13	Daniel Ehrenberg	Igalia	Invited Expert (Part-time, by phone)
14	Michael Ficarra	Shape Security	Member
15	Dean Tribble	Deluxe Corp.	Invited Expert
16	Kent C Dodds	PayPal	Member
17	Dave Herman	Mozilla	Member
18	Logan Smyth	Babel	Invited Expert
19	Waldemar Horwat	Google	Member
20	Shu-yu Guo	Mozilla	Member
21	Tom Dale	LinkedIn	Member

22	Jamund Ferguson	PayPay	Member
23	Maggie Pint	JS Foundation	Member
24	Liza Gardner	Boucoup	Member
25	Daniel Rosenwasser	Microsoft	Member
26	Zibi Branieski	Mozilla	Member (part-time, by phone)
27	Matt Johnson	Microsoft	Member (part time, by phone)
28	Diego Ferraro Val	Salesforce	Member (part-time, by phone)
29	Myles Borins	Google	Member
30	Rafael Xavier	PayPal	Member (part-time, by phone)
31	Tim Disney	Shape Security	Member
32	Caridy Patino	Salesforce	Member (part-time, by phone)
33	Sebastian Markbage	Facebook	Member
34	Adam Klein	Google	Member

1.3 Host facilities, local logistics

On behalf of the Hosts **Mozilla Dave Herman** and **Tilde Yehuda Katz** welcomed the delegates and explained the logistics.

2 Adoption of the agenda (2017/006-Rev1)

Ecma/TC39/2017/006 Agenda for the 57th meeting of TC39, Portland, March 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 57th meeting of Ecma TC39

1. Opening, welcome and roll call
   i. Opening of the meeting (Mr. Wirfs-Brock)
   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 (review draft)
5. Report from the Ecma Secretariat (15m)
6. ECMA262 Status Updates (15m)
   i. Review any new or open issue with ECMA-262 2017
   ii. Vote to accept ECMA-262 2017 final draft and forward it to Ecma GA (approval subject to completion of RF opt-out period with no opt-outs)
7. ECMA402 Status Updates (15m)
   i. Review any new or open issue with ECMA-402 2017
   ii. Vote to accept ECMA-402 2017 final draft and forward it to Ecma GA (approval subject to completion of RF opt-out period with no opt-outs)
8. Test262 Status Updates (15m)
   i. Automate deployment is on the way. We need an email acc for management.
   ii. Good progress in >100 issues and PRs in the last month.

- iii. We have tests for Object rest/spread properties.
- iv. Approve TR-104 draft and forward it to ECMA GA. PDF link

9. 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)
10. Timeboxed agenda items
    - i. 15 Minute Items
      - a. Template Literals Revision for Stage 4 (Tim Disney)
      - b. RegExp Lookbehind Assertions for Stage 3 (Daniel Ehrenberg, Brian Terlson) (slides)
      - c. RegExp Unicode Property Escapes for Stage 3 (Daniel Ehrenberg, Brian Terlson) (slides)
      - d. RegExp dotAll Flag for Stage 2 (Brian Terlson, Mathias Bynens)
      - e. RegExp Legacy Features for Stage 3 (Mark S. Miller, by Claude Pache)
      - f. SIMD.js status update (Daniel Ehrenberg)
      - g. ECMA402 Presentation of the three formatters for stage 2 (Zibi Braniecki) (slides)
      - h. Intl.Segmenter for requesting Stage 3 reviewers (Daniel Ehrenberg, Leo Balter) (slides)
      - i. Test262: using root license as default for files (Leo Balter)
      - j. Weak References status update (Dean Tribble)
    - ii. 30 Minute Items
      - a. Dynamic Module Reform for Stage 2 (Caridy Patiño, Dave Herman) (slides)
      - b. RegExp Named Groups for Stage 3 (Daniel Ehrenberg, Brian Terlson) (slides).
      - c. Realms update (Dave Herman, Caridy Patiño) (slides)
    - iii. 45 Minute Items
      - a. Orthogonal Class Member Syntax for Stage 1 (Mark Miller, Allen Wirfs-Brock)
    - iv. 60 Minute Items
      - a. Arbitrary precision integer type for Stage 2 (Daniel Ehrenberg, Adam Klein) (slides)
      - b. Review the Code of Conduct Proposal (30mins) and define the Enforcement Subcommittee (Leo Balter)
11. Non-timeboxed overflow from previous meeting
12. Non-timeboxed agenda items
    - i. Web compatibility issues / Needs-consensus PRs
    - ii. Stage 0+ proposals looking to advance
      - a. Date Proposal - NodaTime as a built-in Module for Stage 1 (Maggie Pint, Brian Terlson)Slides
    - iii. New proposals
    - iv. Discussion and updates for Stage 0+ Proposals
13. Overflow from timeboxed discussion items (in insertion order)
14. 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.

Dates and locations of future meetings

Dates	Location	Host
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/TC39/2017/003 Minutes of the 56th meeting of TC39, San Jose, January 2017

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

Ecma/TC39/2017/005 Venue for the 57th meeting of TC39, Portland, March 2017

Ecma/TC39/2017/006 Agenda for the 57th meeting of TC39, Portland, March 2017 (Rev. 1)

Ecma/TC39/2017/007 Responses to the Ecma Contribution License Agreement (CLA), 20 March 2017

Ecma/TC39/2017/008 TC39 GitHub archives, 7 April 2017

3 Approval of minutes from January 2017 (2017/003)

Ecma/TC39/2017/003 Minutes of the 56th meeting of TC39, San Jose, January 2017 were approved without modification.

4 Status and Approval of “ES2017 Suite”

4.1 ECMA262 Status Updates

i. Review any new or open issue with ECMA-262 2017

ii. Vote to accept ECMA-262 2017 final draft and forward it to Ecma GA (approval subject to completion of RF opt-out period with no opt-outs)

4.2 ECMA402 Status Updates

i. Review any new or open issue with ECMA-402 2017

ii. Vote to accept ECMA-402 2017 final draft and forward it to Ecma GA (approval subject to completion of RF opt-out period with no opt-outs)

TC39 took a single vote covering approval of ECMA-262 8th edition and ECMA-402 4th edition. All attending members voted yes. There were no abstentions or nay votes:

Tilde: yes

Mozilla Yes

Google yes

Shape security: Yes

FB: Yes

MS: Apple: Yes

GoDaddy: Yes

BT: Microsoft: Yes

JSFoundation: Yes

Intel: Yes

INRIA: Yes

PayPal: Yes

SalesForce: Yes

TC39 also mistakenly applied the same vote to TR104 2nd edition because they forgot that that the modified text had already been taken and approved by the GA in December 2016.

The new Editions of the ECMAScript standards is a great achievement of Ecma TC39. Many thanks for the hard work both to the TC39 group and the editors of the standards.

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

5.1 ISO/IEC JTC 1 fast-track status

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 on April 12.

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 again and Allen Wirfs-Brock and Chip Morningstar will prepare asap the Disposition of Comments Document and the Updated draft standard to JTC1 (via the Ecma Secretariat). 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.

6 ES9 and Test262 discussions

See the Technical Notes in Annex 1.

7 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 more TC39 meetings. Therefore this issue has to be solved immediately. There is agreement in TC39 that we need a fast solution to the TC39 management issue.

Generally – due to the recent high growth in membership - TC39 is having same “Scaling” and maybe some “culture” issue:

One possible way to improve the situation could be to prepare training materials and some processes might be developed to introduce new members and new delegates to the world of standards organizations and development processes.

We probably need a "welcome to TC39" document that goes to each new delegate. Actually, we probably need one for new member organizations, too. We have to ensure that all members know how standards organizations like Ecma work, what their rights and responsibilities as members are.

See also the discussion under 9.

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 – which have the highest priority.

8 Date and place of the next meetings

See above under 2.

9 Any other business

TC39 has discussed a draft “Code of Conduct” document. The current version is attached in Annex 2.

Normally development of policy documents like a “codes of conduct” policy paper is not be delegated to a TC or TG level of Ecma. Of course in case of a need by any TC any initiative is welcome. Nevertheless the Ecma Management, including the ExeCom should be consulted and involved, because this issue might be more general in Ecma and not linked only to TC39.

Therefore the Secretariat will forward the issue and the Annex 2 document to the next ExeCom meeting (April 2017). TC39 will be informed about the reaction of the ExeCom asap after the ExeCom meeting.

10 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, **Mozilla** and **Tilde** for the organization of the meeting and the excellent meeting facilities.

Annex 1

March 21, 2017 Meeting Notes

Allen Wirfs-Brock (AWB), Waldemar Horwat (WH), Brian Terlson (BT), Michael Ficarra (MF), Adam Klein (AK), Dave Herman (DH), Kent C. Dodds (KCD), Tim Disney (TD), Daniel Ehrenberg (DE), Shu-yu Guo (SYG), Michael Saboff (MS), Sebastian Markbage (SM), Bradley Farias (BF), Maggie Pint (MPT), Jamund Ferguson (JF), Myles Borins (MBS), Logan Smyth (LS), Sarah D'Onofrio (SD), Alan Schmitt (AS), Dean Tribble (DT), Peter Jensen (PJ), Mark S. Miller (MM), Leo Balter (LBR), Zibi Braniecki (ZB), Rafael Xavier (RX), Yehuda Katz (YK), Caridy Patino (CP), Diego Ferreiro Val (DFV), Brendan Eich (BE), Lyza Gardner (LG), Mathias Bynens (MB)

3. Adopting the agenda

- agenda

DT: WeakReferences. Please add 2 additional minutes.

AK: Please add items 1 week in advance.

Agenda adopted

4. Adopting the minutes

Minutes approved

5. Report from the Ecma Secretariat

AWB: Istvan will do this later.

6. ECMA-262 Status Updates

(Brian Terlson)

BT: Spec is now frozen to any normative changes.

BT: Most new things will go into ES2018.

BT: Ready to approve the draft. Minor editorial changes are ok.

BT: Printable Style sheets not working, we need to improve this. Asking browser devs to make printing support easier :)

AWB: The patent document is out for review. Members should get it to their lawyers.

7. ECMA-402 Status Updates

(Zibi Braniecki)

ZB: I'm not prepared for this discussion. I believe Caridy will be giving this. Will come back in a few minutes.

7.ii Vote to accept ECMA-262 2017 final draft and forward it to Ecma GA (approval subject to completion of RF opt-out period with no opt-outs)

AWB: Voting on the standards is one of the few things that needs a vote. In a minute or two, we will go around and people who actually represent Ecma members should determine who from each organization will vote. Each member will vote yes/no/abstain (where abstain is the equivalent of "no"). We're voting on the 2017 version of ECMA-262 and ECMA-402, based on the final drafts that the editors prepared, subject to the assumption that no one raises an objection during the RF opt-out period (in which case we would not be approving the standard in June, but that has never happened). After we approve it, this will go through the Ecma executive committee, onto the GA in June with the final vote.

WH: No one has ever opted out of an opt-out, but we've had patent shenanigans in the General Assembly before.

MM: Have we had any patent shenanigans regarding ECMAScript?

WH: One of the companies decided to say that they might have patents regarding every standard that was up for a vote.

YK: Only once a feature gets to stage 4 does it get in the spec. What we're doing now is technically a formality.

AWB: I suggest we do a single vote covering both docs.

LBR: We also have the Ecma TR104.

AWB: We will handle that separately.

AWB: Let's go around the room and have those who have voting authority make your vote. Note takers please take note of the organization who is voting.

YK: Tilde Yes.

DH: Mozilla Yes.

WH: Google Yes.

DT: Shape security Yes.

SM: FB Yes.

MS: Apple Yes.

BF: GoDaddy Yes.

BT: Microsoft Yes.

MPT: JSFoundation Yes.

PJ: Intel Yes.

AS: INRIA Yes.

KCD: PayPal Yes.

DFV: SalesForce Yes.

AWB: Someone will count them. We got "yesses" and no "no"s. It will be referred to General Assembly for approval. Our work is done for last year.

All: *clapping*

Conclusion/Resolution

- Resolution to submit the spec to the General Aassembly for approval.

8. Test262 Status Updates

(Leo Balter)

LBR: Need help from Ecma for an email for use by test262 so that private emails don't need to be used.

LBR: In other updates. I was able to do a bug triage on more than 100 issues and pull requests in the last month. There were many many closed issues (around 50) including tests on rest and spread properties. We now have over 100k LOC for async/generator tests. For object rest/spread, we have a new JSC contributor from my country who has been very helpful with the tests. He also implemented it in JSC. We need more support--I'm glad Google is supporting me, but I need more support for test262 development.

LBR: I'd like to discuss removing the copyright line in test262 tests in a timebox.

AWB: Let's discuss TR 104.

LBR: There's still a draft.

AWB: People have probably not reviewed it adequately to vote on it yet. Post a link here for TR104.

link

LBR: Yes, I will post a link. We didn't have any updates on it since last year.

AWB: If we can get the link posted, people should have time here before the end of the day Thursday to do a quick review of that. I'm guessing people wont have any concerns. That also requires a GA vote, so we should have that taken care of in the same package. Is that ok everyone?

LBR: Yes.

AWB: So get the link up there.

Any items for the 2018 edition?

AWB: We talked about the 2017 edition of the standard. Are there any items for the 2018 edition?

BT: One of the big things I've been pushing this year (and plan to push in a big way next year) is machine readability. Specifically reducing the number of terms used in ECMASpeak. Specifically, I'm looking into making ECMAIDL a subset of WebIDL.

YK: We should really be willing to use a subset of WebIDL. There may be something more suitable for webspecs that's more suitable for JS semantics.

BT: That's the path I'm interested in going down. Thinking of calling it JIDL.

WH: What's the context for making the spec machine-readable?

BT: WebIDL is a web interface description language that the web uses to define the shape of APIs. It's a specific language for describing interfaces.

MM: It IS NOT straightforward (laughter)

WH: How much of this is meant to be machine readable? Are you proposing replacing the prose definition of the sort function with a machine–readable algorithm?

BT: Today if you want to see what properties exist on intrinsics (for example) you have to read a bunch of prose. I want something that's easier to be read by spec readers. I'm not talking about the algorithms, but specifically properties and their attributes, and not even coercions. It would be a straightforward small syntax. This way, we could get rid of text like, "and this property has these attributes".

BT: This is not going to replace the prose of the algorithms like the sort function.

MM: A question for Alan Schmitt, who's been working on automating the ECMAScript spec: Is there something that could be extracted from text in this form?

AS: I think so. If there is something you want to extract, you could easily get it out.

BT: Could spec readers easily consume this syntax?

AS: Not sure.

BT: Let's talk, then.

YK: I think that this would significantly improve the spec.

DT: Are you also looking at refactoring parts of the spec. For example `eval` would be good to extract out and parameterize. Are you looking at that sort of thing?
BT: Yes. There is a proposal to add a structured header to clauses that say things like "this defines this method and takes these parameters" That would help with tooling. That's definitely a road I'm going down. I haven't given much thought to any other refactorings.

AWB: In what?

BT: Possibly inline Annex B, although that's controversial.

WH: Yes, that's controversial!

BT: I think it would be pretty nice. But yeah, that's the only refactoring that's on my radar. I'm totally interested in refactorings. But I've been in this document so much that I'm resistant to do any more significant refactorings.

AWB: I can understand what you say about the IDL for properties. But I'm wondering about the other major place where there are prose descriptions for built-ins are on the arguments. It's not clear what you could say about arguments other than "we call the first argument X".

BT: WebIDL has the capability to describe function parameters. It goes further to say that if you define your function as taking in a number that implies that there's a coercion done to number. I'm not going to include this in the MVP. I don't want the syntax to describe the shape of a function. I would leave that to editorial conventions and prose. The automatic coercions... I have two minds of it. There's an issue raised in 402 where it says do all your coercions at the top, and the Ecma standard says do all coercions when you need them.

YK: The ECMAScript way is more natural.

BT: I don't like the WebIDL way of doing all coercions up front. There isn't much value in describing the types of coercions....

YK: The current spec has arguments because function.length matters sometimes.

AWB: The current spec has prose that describes how function length is given.

BT: We do have language in the spec that describes that the function's length is the number of params given.

YK: Given that is a thing in the spec (and it's convention). I think it would be fine to have something that was to solve that particular problem.

BT: I think we'll get there eventually. I'm trying to take babysteps. The structured header for clauses is going to be a [big one?]. I am continuing to push the tooling side of things. I have some features planned.

YK: +1

BT: And then I do plan to continue pushing the tooling side of things. I have some features planned. Before I can do that I need to fix printing. But after that I plan to finally have the ECMarkup tooling to have: "Given any clause..." Wouldn't it be sweet to click on a clause and for any given clause it'll show all dependencies? That is now possible. There's also a VSCode extension one of my colleagues has been working on and I'm working on refactoring to support that as well. And I think that's all I have on my end. There's probably more that I'm glossing over having not prepared anything. Wish list items?

MF: Support for proposals would make everyone's proposal writing lives better.

BT: So I don't need to manually integrate all the proposals?

MF: Yes.

BT: That sounds great.

YK: I would like that as well.

BT: I have not talked to a single implementer that prefers having Annex-B separate.

MM: I would like Annex B to be separate.

BT: If I have the elements for the diff-spec stuff, I think I could add a checkbox to separate them.

YK: I think there are some places you need to monkey-patch things.

BT: And one of the other things I want to solve is this refers to step 6 and step 7.

YK: You want labels.

YK: I would love someone who believes strongly in this to give a presentation on this in the future (not now).

DE: One other thing is color in the spec. In other specs they make a lot of use of color which makes the document more readable and less boring. What if we use more color? And add (to the draft spec) examples. If we have a bug that partains to the draft spec we can have this to cross-reference and have it highlighted red or something.

WH: Color is useful for notes, identifiers, keywords, grammar productions, etc. If we were to use different colors for Annex B and everything, it could require too many colors and get really confusing.

DE: I'm talking about background color.

YK: Take a look at the decorate proposal which does this well. One thing I'd say about color is there's an accessibility issue with color. We should have someone who has an intuition about this (ideally someone who has a11y needs) review our work to make sure that we choose good colors.

AWB: We're getting into document design space. We need to get that as a cohesive whole, not just one step as a time.

BT: I totally agree with Dan's request for more semantic meaning to notes and that side of things. Ecmarkup has some [new tools] that make it easier to notice what's happening. Everyone should check out the decorators spec, because that's kind of where I've been putting my ideas. (Examples: https://tc39.github.io/proposal-decorators/#sec-runtime-semantics-makefunctionuninitialized, https://littledan.github.io/proposal-integer/)

AWB: I don't see substr() as needing to be in Annex B. I looked back and couldn't figure out why it really was put there. Everyone has added it. I don't see why it needs to fall into that category. It's just a redundant.

BT: How do you see substr() as different from anything else in Annex B.

AWB: Some of the other functions are in there because they have limited applicability or Y2K issues or some other bad aspect to them. I don't see anything that bad about substr(). The concern about creating confusion doesn't apply. It is what it is; implemented in all engines, etc.

BT: What's the concern?

AWB: Because none of the Annex B criteria really apply to it.

YK: What are the criteria.

AWB: Things that have issues with them.

YK: I think that we should merge Annex B into the spec. There's a strong sentiment among people (including implementers) on this point and I think we should have someone come with a prepared presentation on this subject.

BT: Step 1 is inline it.

YK: The answer is not "it is normative".

BT: It is not normative. It is in the clause that...

AWB: I was having a discussion with someone about Node using normal sloppy mode using normal function semantics. And I asked "Why is that?" And it's because that's what V8 does. Because you should never actually see that.

YK: There's a lot of JavaScript in the world and implementers all need to implement Annex B.

WH: That's throwing in the towel. Don't want to make Annex B normative.

AWB: In the web you have lots of code that nobody can maintain. That's not the case on Node.js.

AK: In the interest of moving along, I think it'd be good for us to end this conversation for now and push it on to another time.

YK: Can I say one more thing? I think that if people want to advocate for not throwing in the towel, then we should come up with the goals for this.

AWB: This would require evangelizing implementations to say "you would have to switch".

MM: That's not required for my goals. I wouldn't mind a switch, but none of the main implementations needs to have a switch in order to meet my goals.

AWB: That would be required if you say these things have to switch.

MM: None of this would be required with the goals I have in mind.

BT: I'm coming at it from an implementer's standpoint.

MM: We should just have a face-to-face conversation Brian to talk it out.

BT: Ok.

SYG: I have been told Caridy is on the call now.

Conclusion/Resolution

- Further discussion on Annex B

7. ECMA-402 status update

(Caridy Patiño)

AWB: We already have the vote to approve the final draft. Hopefully you will tell us that it's ready to be approved. *chuckling*

DFV: Having issues hearing him. He's joining now.

DH: I am pinging him now. Some issues with audio setup. Is that Diego?

DFV: ...

DH: Our sound person says you are showing up as self-muted.

CP: ...

DFV: I'd say you should move on and try again.

AWB: We'll go ahead and move on to the 15-minute items.

10.i.a. Template Literal Updates

(Tim Disney)

- https://github.com/tc39/proposal-template-literal-revision

TD: Lifting the escape sequence revision to tagged template literals. It is in multiple browsers (Chrome behind a flag and Firefox), Test 262, no recent changes, asking for stage 4.

YK: TL;DR: It makes things easier to implement overall.

WH: Has anything changed in the proposal lately?

TD: No.

AK: The owner of our scanner didn't love how it was layered between the parser and the scanner. And the real question is do we really have folks that want to do things with this?

TD: I have things I would want to do with it.

MF: The violations of the escape sequences that are not uncommon sequences is justification enough.

YK: For what it's worth, I have hit some variance of this bug repeatedly and I've always been: "Oh, I'm glad we're fixing this."

WH: I like this proposal.

AWB: In particular, going from stage 3 to stage 4, unless we've had problems with it then we should just ask whether it meets the criteria for stage 4. So... Do we have consensus that this should be a stage 4 feature which means it's in the standard in 2018?

AK: Sorry, for my memory, did we expect any compatibility issues with this?

TD: No compatibility issues because you'd have to be relying on a parse error.

AK: For cases where there might be compatibility issues, I'd say maybe hold back, but I don't think we have to worry about it.

AWB: Ok. Consensus? This is our first ECMAScript 2018 feature.

YK: A use case would be putting JS code into a template string for documentation.

Conclusion/Resolution

- Stage 4, our first ES2018 feature

10.i.b RegExp Lookbehind Assertions

(Daniel Ehrenberg)

- https://github.com/tc39/proposal-regexp-lookbehind

There's a presentation linked from the agenda.

DE: Just to recap the purpose for the proposal. This is a zero-width assertion that matches a string without consuming anything.There are both positive and negative look behinds now. ... At the last meeting I was asking if we could go to stage 3.

(see slides for more details)

WH: It would be helpful to know what has changed, because I've reviewed this before.

DE: The only thing I changed since the last meeting was updating the explainer to make a correction to the explanation of the numbers in the groups. The spec always said the groups were numbered left to right, now the explainer explains this also. Other than that, nothing has changed since the last meeting.

WH: In that case I'm happy with it.

DE: We have two implementations. One in V8 and one in the Dart VM. It's not a JS implementation, but it does implement this feature.

AWB: 2 implementations are not required for stage 3 anyway.

LBR: Writing a test for this (test262) would be the best thing to review the proposed syntax.

DE: We would love your help in writing tests!! There are some tests already on github, but we would love and plan to do work on these tests, but having started porting them to test262. .... Okay, I think we had a few reviewers, trying to find the notes from the last meeting to be sure. So the reviewers were: someone from Microsoft, Kevin Gibbons, WH. Did anyone from the Microsoft team look at this?

BT: Not thoroughly. I couldn't find any regex experts. I love it, but I suppose it should get a proper review.

MF: Kevin Gibbons is not present and was not able to provide his feedback yet at this time.

AWB: We can advance it to stage 3, or keep it at stage 2 to elicit more reviewers. The first question is does anyone object to stage 3?

YK: I would assume Chakra or V8 would have said something by now if they had issues.

BT: This is not at all concerning. This is a well understood feature that exists in other regex implementations.

AWB: In the absence of objections, we should consider advancing this to stage 3.

YK: I think V8 is happy with the ??? I assume V8 is ok with those changes.

MF: I have gotten feedback from Kevin, he said it looks good. The only issues were typos.

BT: So Dan, do you want to talk more holistically about what you think the state of RegEx will be for 2018?

DE: At the last meeting, Allen said, for ES2018 wouldn't it be nice if we could send a message that shows all the nice features for RegExp. We'd have everyone's work together. We have look behind, ..., and named groups, dotall flag, and match all. So, for match all, that's currently at stage 1. I think maybe we can get it through by ES2018. I was wondering if it'd make sense to go with semantics that are analogous to the current regular expression thing similar to split where it ends up cloning the RegExp. On the other hand, we've previously discussed adding a Symbol.exec method for being the basis for subclassing RegExps. I'm more hesitant about that. But I think we could do matchAll separately from this. So I started talking with Jordan about this and hope we can get all 5 of these things through by 2018 to have a coherent story for RegExp.

AWB: So I noticed there are a number of agenda items related to this. I think it would be a good thing to try to do all of the items for this as a group and then at the end of it all we could review our overall expectations for 2018.

MM: Given we have a RegExp legacy statics proposal, currently at stage 1, would you like to consider that as part of the group as well?

AWB: If you're ready to talk about it.

MM: I can talk about it.

Conclusion/Resolution

- RegExp: Lookbehind is stage 3.

7. ECMA-402 status update

(Caridy Patiño)

CP: So far no feedback on the 2017 draft that was published in the last month. We haven't done any major updates, though we did small fixes and editorial work. We expect to continue with small work in 2018. Zibi(?) and Daniel will present a few related proposals. Also ??? will be updated. We keep waiting for number format, format to parts from Google.

DT: Your voice went out....please say that again.

CP: We want to give updates on 2 fronts. the formatToParts? on Intl.NumberFormat.prototype.formatToParts, some Google update would be good on that. The other thing is the CLDR. Brian will be giving us some update this week. We will wait for decision until we make some adjustments on 402 spec.

AWB: Well, we actually have the thought. We approved the draft to approve the GA (?).

CP: I can work with Brian on the small details we need to update the draft and get it out.

AWB: Okay, so, thank you! Let's continue specifically with regex items. the next one is actually Dan and Brian which is property escapes.

CP: Excuse me, does this mean we'll get to ECMA-402 segmenter after lunch?

AWB: Let's do it now since you are here.

CP: This is the ECMA-402 Proposals.

Conclusion/Resolution

No specific conclusion, let's review the current proposals for ECMA-402.

10.i.g ECMA-402 Presentation of the three formatters for stage 2

(Zibi Braniecki)

- https://docs.google.com/presentation/d/1ddnQB8oUYyv7qtsmRFgcsScAI4uHTj8z9z_cPJxl0e4/edit

Intl.ListFormat

ZB: There are three formatters that we believe now are ironed out enough to request a stage 3. The first one is called Intl.ListFormat. It closes the gap with localized JavaScript data structures.

ZB: (makes presentation). I believe it's mature enough to propose stage 2. Stage 2 requested.

WH: I'm still looking at this.

AWB: What's the use case that this is supporting?

ZB: There are multiple use cases. The most common one is sort of lists. Default is to separate lists from left to write separated by commas. This would support right to left languages as well. Another common example is listing days of the week or names. Another example is a building block for future formatters, which is a listing of numbers or a listing of units. ...

WH: I was looking at the slideshow of the detailed internationalization spec which is quite dense. What I couldn't find is the specification of this from a user's point of view. What kind of things can you put in these things. I'm looking for documentation that someone who would call these things would read to find out what they support.

ZB: Are you talking about options that are supported? Or the values that go in the Array?

WH: If you go back to the Intl.ListFormat slide, what does this last example do?

ZB: We have an Intl formatter which is Intl.NumberFormat. And you can also override toLocaleString. This formatter could also be the same for arrays.
WH: What will it do on an empty array?

ZB: Return an empty string (like it does now)

AWB: The fact that is extending toLocaleString in places *should* be noted, but ... I guess I'm interested in the background of this feature in terms of its usefulness. Are there other internationalization packages that have this thing.
CP: Yeah, I can speak about that. We do have prior art. Java has an API for it. In terms of the user... You're seeing this in almost every social network when you have a list of people who like your tweet/facebook post, you have to list them based on the locale of the user. But we're saying for all your formatters, we need to have data to reliably list an array, the user would have to do this themselves. This just aligns with the existing formatters.

ZB: It may sound redundant. The reason we want ListFormat is that we want to format lists. And JavaScript doesn't have any support for any other kind of lists. In an effort to close the gap. It's any kind of list.

WH: Why is weather measured in kilobits?

AWB: So there are several formatters here. Potentially we could advance these separately.

ZB: That's what I was thinking.

AWB: They are the list format, unit format, and relative time format. They're all fairly different. They're not connected to each other.

ZB: As I said Unit format, would benefit from list format, because it would allow to format compiled units. And I wanted to suggest stage 2 to Intl.ListFormat.

AWB: As a reminder, stage 1 means it's exploratory. Stage 2 means we've explored it and we think this is a feature we want to see standardized even though the details are still subject to change. So a request to go to stage 2 is a request for us to reach consensus that list formatting, roughly based on this proposal, is something we should work on getting into the standard. Any objections? I don't see any. I think you got stage 2 on the list formatter. Let's go on to the next one.

BT: Should we nominate reviewers now?

AWB: Is there something to review?

ZB: I think that's for stage 3.

DE: We nominate reviewers at stage 3.

JF: I know the folks at PayPal were working on something similar and would probably like to review this.

ZB: It really depends on ...???? They are separate spec proposals. I don't think the spec is finalized for them. How about we have ListFormat advance to stage 2 and get interest on the others later.

AWB: Yes, let's do that.

ZB: So PayPal said they may be interested in helping review this?

JF: Yeah, I think we should be able to help out.

Intl.UnitFormat

AWB: So let's go to UnitFormat.

ZB: It's an interesting formatter. We see a high demand for it. But it's a little tricky to figure out what kind of data should be inside it and what kind should not be. What I can say is that we're looking into a very plain formatter. So we'd expect users of the API to specify the unit. But we think it would be very powerful to turning JavaScript into a powerful platform that can internationalize user interfaces. It can be any unit of measure.

WH: Question. You mentioned that you were intenationalizing celsius and fahrenheit. Do you mean translating the word "Celsius" to their language or do you mean converting celsius to fahrenheit if their locale indicates that temperatures should be shown in fahrenheit?

ZB: Absolutely not doing any conversion from the actual number for the unit. It will mean displaying the word Celsius in their language, depending on their script. Same happens with kilobytes.

CP: We don't want to do any transformation of the value. That should happen in user-land. They provide the value and we format it.

WH: So you're just doing a word-translation service?

CP: Not just a translation. It's more complicated than that.

WH: How is it different?

ZB: Two examples of that. We are translating the whole pattern, which may be different depending on the language. (Think "10 years" vs. YR10 or something appropriate to the language). So the symbol for kb for example, 10kb, may be different depending on the language. The intent of the formatter is to allow users to display to the software user in their language.

MPT: I believe there are calendars other than Gregorian that will have units other than seconds. Are your units strictly Gregorian calendar units?

ZB: No. It's not finalized, but we will not be narrowing it down to any single system. We will want to support duration, we will want to support time & date unit for calendars.

MPT: Ok.

WH: I'm looking at the list and my main issue with what you're providing in the list is very ad hoc. You have terabytes but not petabytes, kilobytes but not kibibytes, meters per second but not kilometers per second, etc. etc.

CP: That's stage 3 right? We need to finalize the list of options.

WH: The problem is very pervasive, I'd like to see these defined more generically instead of the few cases which you have selected.

ZB: Ok, that's good feedback. If you look at the proposal itself, it lists the proposal of units that we'd like to offer in the first draft. I honestly think the list is negotiable. And we can discuss this later. Skipping that part, it has a list of units we'd like to offer as part of UnitFormat. I consider the list of units that we would offer a negotiable thing. I would like to get to the point where implementation can choose what scope of units they can support. I don't know how optional it would be. I think the user can determine what the runtime supports, maybe if the implementation doesn't support petabytes, gigabytes will be the highest level of measure they can use.

WH: The premise of the metric system is that this is you should not need to do this with each one. An ad hoc list of which combinations are supported and which aren't is too hard to use. The metric system allows arbitrary combinations of prefixes and base units, and there aren't that many of them. Find a generic solution that supports that.

AWB: Why are these together? I feel like they should be separate. Can you explain the logic behind this?

ZB: It's the same operation regardless of what unit you're using. Are you suggesting a separate formatter?

AWB: I'm wondering why you don't have a DurationFormatter, a MemoryFormatter, other formatters...

MPT: I'm definitely going to put a proposal for a duration type. And if you propose something for time in units it would be good to have these together. That's hard to deal with itself.

AWB: By it being just an option, and the options object, I just have a gut feel that makes it easier for implementations to vary. Adding a new option type seems like lighter–weight and maybe it shouldn't be. Similar concern to when you saying implementations will be able to choose which they support allows for interoperability issues we don't want to have.

CP: Polyfilling in the future is definitely something we'll have to look at in the future.

ZB: Would you like us to stay on stage 1 and rethink the naming? Or are you ok with us going to stage 2 and rethink the naming?

AWB: I think we need more stage 1 exploration.

CP: I agree, we could probably spend more time thinking about this.

YK: I have a general question. I could be wrong, but it appears that while the metric system itself is orthogonal, it might be dependant... It seems that more complicated cases mean the decision of how to format something may be domain-specific.

AWB: Let's say you're working in some domain and that domain has some types you'd like to introduce. You're going to want to supply formatters that do that, it's unclear how those would plug into this UnitFormat stuff. And people would expect that UnitFormat is how you would do that.

YK: We have some questions here that it's worth looking more.

ZB: Thank you for the feedback. I think we can keep it in stage 1. ..But I will have to say. Every library out there has some kind of unit formatter, which says me that this is something that is useful. .. We have some proposal in the spec, but didn't feel comfortable proposing it yet. There is something else about combining units that we may want to use list formatter for. I'm taking this as a "let's keep this at stage 1" to think more about naming and compound units.

BT: I would like to add Maggie's point on how this changes with the introduction of an actual duration type... Having some exploration of whether this changes or not based on that. What are your thoughts?

ZB: I think it should not change. But we may want to have a `toLocaleString` on that type. I don't know if we'd want to have DurationFormat because it seems to me that would not require any computation. One of the specific things about number format is that there are a number of computations that are being performed internally. ... As I said I'm very happy to keep it at stage 1 and see the duration proposal and see if we evolve our thinking in that time.
MBT: I don't have a duration proposal for this meeting, but the more I look at this Intl stuff, the more I think I need to put a duration proposal in for July.

YK: I would love to work with you on that. I did that Duration proposal for Rust and would love to work with you on that.

MBT: Let's talk. I have a few other things going on today, but I think we'll need a duration type.

DE: Can I say something with regards to the current proposal? Some of the points here are similar to the current state of ECMA-402. For example. NumberFormat lets you format currencies and ordinary numbers. There is a list of currencies there and everything is in the options object. To me, an options object, seems like a pretty consistent design. You could still polyfill new data it by over-writing the Intl object--there's nothing special to these objects; not fundamental like built-in modules. The other thing about which units are included or how ad hoc it is. There is a close connection between what's here and what's in CLDR. If we want metric units to be combinatorial, we could ask for better data in CLDR to be able to add these additional formats and things there, though this is a process that takes time.
(see http://www.unicode.org/reports/tr35/tr35-general.html#perUnitPatterns)

WH: I don't know what's in the CLDR, but I know the metric system and it does not have a lot of units. I think it's quite reasonable to include the entire metric system in this.

ZB: We've seen this pain a lot in operating systems. So we can stand on the shoulders of giants with MeasureFormat and learn from them. This is what I'm doing for the proposal. Happy to stay on stage 1. We can continue conversation in the GitHub repo for UnitFormat. I also agree we can wait until the duration proposal comes up as well.

WH: Let's say you're translating things into Greek and you're translating to a shortform. Do you use English letters or Greek letters? It depends. Are you using colloquial or scientific usage?

CP: I believe we'd use Greek letters...

WH: That's not necessarily the correct thing to do.

RX: You can define different numbering systems. You can change the numbering system, but not the language. Is that clear? For example, Chinese using wide numbering system, it could be used the locale zh-u-nu-wide. (Clarifying this point after the meeting: It's possible to use Intl.UnitFormat to display the unit in one language, but the digits in a different numbering system. For example, `new Intl.UnitFormat("ar-EG", {category: "digital"}).format(100, "kilobyte")`

`=> "١٠٠ كيلوبت"`(Arabic using Arabic digits), or `new Intl.UnitFormat("ar-EG-u-nu-latn", {category: "digital"}).format(100, "kilobyte") -> "100 كيلوبت")`.
WH: No, this is not a question about the numbering system, this is a question about the translation of prefixes.

CP: Normally what we do is give the tools so they can decide how they want to format something. If we want to allow them to format in the case you decided, we just need to expose the tools. I'm not sure this proposal is not supporting that.

YK: Allen is right, there's an extensibility issue. I think it's plausible for medical apps to render some things differently and I think this should be extensible so the committee doesn't have to answer all the questions.

AWB: Let's move on to relative time format.

Intl.RelativeTimeFormat

ZB: This is by far the most requested time formatter. It's basically about formatting relative time. It's intended to allow us to present a delta of time in a human-readable way. There are a few ways of doing this. A numeric way, following CLDR and a text type, which for certain values which may return a more human-readable variant (tomorrow, yesterday, now, etc). These are the kind of overlays on top of the system. There's also a short value vs. long.

WH: Are the relative values only relative to now? Are they always "N days before now", or can you ask for a translation of "N days before X" or "N minutes after X"?

CP: You can't do that; it's always in the present.

BT: Can we put a break in our discussion? We're way over our timebox and lunch is ready.

AWB: Why isn't this a category of the UnitFormat?

ZB: Mostly because unit formatter is not a unit of deltas. It doesn't have the ability to format units of time. It also doesn't have the "type overlay".

YK: I assume there are some languages where it's not so simple as adding "ago" to the end of the number.

AWB: What I'm more getting at here is...is this the way we want to define these? Before moving past stage 1, do we need to further resolve the issue of how to address these types of things?

ZB: No we should not hold off. We've spent a lot of time.

MPT: As someone from moment.js. This is starting to catch up to what we have. This is not the common use case. I do have some concerns between the difference between Calendar math and Timeline math. 8 hours ago can be yesterday. As far as different units, twitter is a good example. People want to know how to do things like "1d ago" or just "3d" for example for saying something is 3 days old.

ZB: In the spec proposal there are issues that we are still working out. Please anyone come and help with the proposal. We've been in stage 1 a year ago.....there is an open issue for relative time for unit formatter.

CP: We think this could be what moment uses under the hood.

DH: I'm so sorry Maggie, lunch is here. We can continue after if that makes sense.

AWB: Let's continue now that lunch is over.

ZB: I added examples to the slides during lunch to present use cases. Continuing on UnitFormat. These formatters (plus one more that's in the ecma402 GitHub repository) are the only ones they are currently planning. Every use case they have encountered in the last year can be covered by some one or two of the 4 formatters. It's not that they are trying to port all the ICU formatters to JS, but rather they have the subset that is need for the web. Stage 2 is that TC39 agrees that the problems need to be solved. Is there a reason that we do not want a feature like that in ECMAScript?

WH: Those are not the only choices. We could say "come back when you have improved it".

ZB: Sure, but let's define the API before we move or should we say "this does not belong in ECMAScript" (do we move this forward or do we not)

DH: You will usually know if people think it shouldn't be in ECMAScript. That will be clear.

YK: I think in general, that is a necessary, but not sufficient thing for stage 2. The norms of the committee is that the API is reasonably done as well as semantics.

AWB: I was going to comment earlier. When the consensus is not to advance, it does not mean that it might pass in the future (without changes), it just means that people aren't ready to move along as it currently is. It might simply be a matter of coming back with more complete information.

ZB: We are reviewing the proposal to move RelativeTimeFormat to stage 2.

AWB: The relationship between this and the UnitFormat--I'd like to see more exploration of that. Are we going to do things with time types and duration types, and what might the impact of that be? These questions make me wonder whether this is ready for stage 2.

YK: I think it's not ready for stage 2.

BT: I don't think there's any rush for this for stage 2 either. I'd like these to go into these as a bundle for 2018 for 402, but we have plenty of time for that.

ZB: Can I get clear direction form the committee whether we want to BLOCK the two formatters on duration types, or whether they can proceed without that.

AWB: We're not hearing people say it's a blocking issue, more concern that it's not adequately explored or explained yet.

ZB: What would be the recommendation, should I try to present a refined proposal separately or keep them as a bundle?

WH: Whichever you do, some people will want the other, so do what makes sense.

YK: Work together with the Duration proposal.

CP: Let's keep the proposals separate; it's easy for me to integrate them separately.

ZB: So ListFormat goes to stage 2 and the others stay at stage 1.

ZB: The last proposal I have is a proposal that I'm seeking stage 1 for. I presented it at the last meeting. It's a proposal for simplifying the way of formatting date and times using two formats `dateStyle` and `timeStyle`. Instead of doing them separately, we would actually like to use one of the four lengths of

style "short", "medium", "long" and "full". I think it's a pretty good representation. It makes it easier to use DateTimeFormat, but it also allows one more thing. It is language independent. For example, the "short date" specifiers are relative to the current locale. By using dateStyle or timeStyle, the user hands of control of the formatting to the language. We expect this will be the most common way people use DateTimeFormat.
CP: The data part is the new thing here, to find the default settings for the locale. It should be useful. All other things that you mentioned are not very relevant because this is low level API, and can be achieved in user-land, except the default settings per locale, which is data.

AWB: This sounds plausible, but most of us are not familiar with all the Intl functions. I would find it useful to see "here's what you have to say today" and "with this proposal, here's what you say instead".

YK: There's something that's been bugging me. I suspect that if the entire library existed, it would be used to get at some underlying primitives that are not exposed. I say this as someone who worked on the rails intl library because it's so hard to make the high level story good for all domains.

CP: Maybe an API that gives you access to all the setting for your locale.

AWB: Part of the requirement is that the surface API needs to accomodate different underlying intl libraries.

BE: This is the extensible web manifesto. We might do OK if we design a high-level library, but we might get it wrong, and putting in low-level primitives.

MPT: We want the CLDR data without having to pack it. What would save our lives would be CLDR data accessible through the browser.

YK: That might be tricky because Edge doesn't have CLDR data.

BT: If you are just talking about exposing the underlying data, Windows does use CLDR extensively so we could potentially expose underlying data. I'm not ready to commit to standardizing on CLDR, though.

RX: I do Intl and localization at PayPal, in globalize.js. It's not just a matter of roping a lower-level API into a higher-level one. For example, when you say Date.short, so let the engine to figure things out in the specifics about the language/locale. So it's more than grouping things together--it has localization smarts under that.

CP: What we're saying is that that information is needed to produce the right date format--it could produce options that would feed back into the Intl constructors.

ZB: What the committee is saying is that it would be great if there was an API that would return the underlying data, and another API that would format using that as the pattern.

CP: Or rather options, as pattern has issues.

ZB: One problem is that CLDR is not frozen. Not just in the data, but in the key paths and the way the data is organized. We could either expose something very unstable, or put a stability layer on top. For that reason, I think we have to expose some abstraction of CLDR.

YK: I totally agree. I actually don't think we're going to design a good API on the fly. Exposing the API that's in the browser is not the goal. A shim-layer is the goal. The thing you said about exposing a pattern and then a formatter is one possible primitive factoring, there may be others. Looking at globalize.js and moment.js, the issue is that they're too big. Why don't we first ask: "What could we do to make moment smaller?"

ZB: Our end goal should be to try to make creating libraries like moment easier.

MPT: Can I get a call with 402 and the moment team and we can work some of this out?

ZB: To wrap up this. Are we not willing to move this to stage 1 and we want to keep it stage 0?

YK: No, let's move it to stage 1.

AWB: I think moving to stage 1 is fine.

YK: Stage 1 is: we want to move this to solutions space.

ZB: One other problem that this solves is that we encountered a problem where a user manually changes their local OS clock. There is some conversation about trying to map OS formats to the JS environment and get access to the OS clock style (for example). CLDR is similar. If we have this kind of option, "short", "medium", etc. We can fine tune it with all of the data from the user's OS.

AWB: I think we've covered this. One take–away, as this library grows, there's interest in the whole TC39 for understanding this problem space. Thank you!

Conclusion/Resolution

We will treat the 3 proposals independently.

- Intl.ListFormat
  - advanced to stage 2
  - Reviewers:
    - PayPal
    - Daniel Ehrenberg
- Intl.UnitFormat – remaining at stage 1
  - Issues raised by
- Intl.RelativeTimeFormat – stays at stage 1 pending more understanding of durations.
- dateStyle/timeStyle gets to stage 1.

6. Addendum to test262 item

LBR: People asked me to mention. We are using an integrated test generation tool. That's why we have so many lines of code. It has documentation on test262. I would be very happy to show this test generation tool to anyone who is interested. The automated deployment I mentioned earlier uses this tool to confirm that tests are properly generated.

10.i.c RegExp Unicode Property Escapes for stage 3

DE: There's a link from the agenda. These have barely changed. [DE talks through the slides]. The thing that has been narrowed down recently, is that the list of properties that are supported is explicitly enumerated. Previously implementations could expose anything they wanted, which would lead to needless differences between platforms. It is implemented in V8 behind a flag. I asked for reviewers last meeting. The potential reviewers were Waldemar, James Gibbons, and someone from MS.

YK: Are the new properties...???

DE: The preference was to be restrictive with the number of name/value properties but to include every possible random binary property. We could revisit that.

YK: Does this mean that the committee needs to routinely review the list of properties and approve them one at a time?

DE: I think that should be done in some place. I think the browser should be ??? about where property support is. If it's a bad idea to support a property, then we run the risk of browsers supporting bad things.

YK: I agree with enumerating them. But this does have a list management challenge.

DE: It will take actual implementation work also. Practically speaking, if we can write Test262 tests and implementers are running those tests regularly, and a patch going into the spec will motivate them to do it. It's a harder sell to have them watch for a set of properties. There's a challenge with this. It's a little ad hoc. There's one part of the spec where we can reference that say all the properties that are defined. We can't reference one section that has all of them. It's good for us to maintain a list in light of that.

AWB: How long a list are we talking about?

DE: A list of properties that are supported. For named ones, it's just general ??? there are 30 or so.

WH: There's a list in the proposal.

AWB: My sense is our spec should list them and where they're defined. And if there are new ones, it seems like a reasonable thing to adding new ones.

YK: Is there a lot of code? Presumably you're talking about a RegExp engine that has those characters.

DE: For implementations based on using ICU, ICU has support for mapping one of these categories to a list of ranges. So I don't know how Microsoft's Unicode works, i.e., whether they have something similar.

YK: I can believe it's ok doing this on an ad hoc basis. In practice we'll wake up and notice that we didn't actually do it. Maybe now the committee is healthy enough that we'll actually do it.

DE: I don't really understand the theory of change where we wake up and we want these things in browsers and don't have a process to get them in there. Otherwise we are waiting on multiple browser vendors to add this stuff.

YK: I think the bottom line is what Allen said is we should make it part of the process of annual review. If there's no hook like that people will forget to do it.

DE: I don't think propeties get added all the time. Emoji properties are the hot topic now.

YK: Ok I withdraw my concern.

AWB: I think the general principle here is if there is some other normative spec you can point at and say: do this. If not then we need to have a list for implementers of our spec. It sounds like you're saying it's the latter we need to enumerate.

DE: The current spec doesn't enumerate these properties.

WH: I couldn't find some of them. I could find a few. But I couldn't find `any` for example.
DE: `any` is in Unicode Technical Annex.
WH: I was looking at TR44. It's not in there.

DE: Sorry, UTR-18. I'm not sure it's very useful...

WH: Just searched for all uses of the word 'any' in UTR-18. It's not in that document either.

AWB: I ran around that list a while ago and "any" seemed not clearly defined.

DE: I posted a link in the tc39-delegates channel. I agree with you Allen, each one of these deserves a definition. I'll fix that up by the next meeting. Timebox check. Should this go to stage 3 or should I get more clarity about these properties before going to stage 3?

MB: `Any` is defined here: http://unicode.org/reports/tr18/#General_Category_Property A greed that the Unicode specs could be clearer — there is not a single/obvious source for a list of all binary properties, for example.
AWB: The property seems like a minor refinement you'd expect to be worked out in stage 3.

WH: I'm fine with stage 3, but I want clarity on some of these things.

BT: I have ideas on what we could do to track these properties.

AWB: Sounds like no objections to stage 3.

MM: Any change to reviewers?

BT: I guess we still need to get this reviewed.

DE: Presumably it has been reviewed.

BT: *Presumably*.

WH: I have reviewed it.

DE: Mathias wrote it, and I read it.

Conclusion/Resolution

- Stage 3 approved.
    - WH reviewed it

10.i.d RegExp dotAll Flag for stage 2

(Brian Terlson)

- https://github.com/mathiasbynens/es-regexp-dotall-flag

BT: We want stage 2 for dotAll. There have been no changes since stage 1. It remains an extremely simple feature. For those who haven't heard of it before, it adds a new flag `s` to the expression (like other engines) literal that allows `.` to match *everything* rather then *most everything*. There's a high-level API in the proposal.
MM: Why "s"?

WH/BT/BE: It's Perl, c#, Java, VB.

DH: I know you've presented this before. And I noticed you use the word "character" before and it's not necessarily meaningful. Are you talking about "code points"?

WH: Without "u" it is referring to code units. With "u" it is referring to characters.

BT: I think we're talking depending on the context.
Without `u` we're talking about code units.
DH: `s` is orthogonal to `u`.
AWB: In the regexp spec in ecma262, the word "character" specifically means "the unit you are matching".

DH: What are the historical things that dot didn't used to match?

WH: Only the line terminators: CR, formfeed, NL, etc.

BT: JS devs have written a RegExp that's a cryptic character class (see the proposal examples). Horrible hacks and we don't want to do them anymore.

YK: Ruby bases this on the `m` flag. If it's a multiline regex, then you want `.` to include multiple lines.
DH: So basically "ms" is the thing that you always want?

BT: Stage 2?

WH: Sure.

DH: Sure.

DT: Reviewers?

WH: I've reviewed this before. Unless it's changed and I missed that, I'm happy to say I've reviewed it.

BT: I would be very surprised if we find changes to this proposal.

AWB: Review really becomes important as we approach final spec text.

WH: This is essentially final spec text.

??: What makes this not stage 3?

WH: There's nothing to do to get it to stage 3...

BT: We actually have final spec text. You'll find changes to flags, adding the `dotAll` property and the various other changes that need to happen.
SD: Does it need to go up with the Unicode changes?

WH: It's orthogonal to Unicode. See above "character" definition.

AWB: Is there a reason this can't be stage 3?

BT: Stage 3? Anyone object? We're basing that on Waldemar's review. Based on the spec text I think we're good.

LBR: I've actually reviewed it as well.

BT: Then we're good process-wise.

Conclusion/Resolution

- Stage 3 approved.
- Reviewers: WH, LB
  - they previously reviewed it.

10.ii.b RegExp Named Groups for stage 3

AWB: The next regex item we have is named group.

DE: There's a presentation for this linked in the agenda.

DE: So the named groups proposal also hasn't really changed.

AWB: I think it's a good idea to give a review of what the proposal does... Have at it.

DE: The motivation is that if you want to parse something that has a bunch of pieces and you want to get them out, accessing via index is not as usable as getting them using names. This proposal takes the consensus position of "most" languages that have named groups as syntax. They are permitted in back-references, replace, Unicode and non-Unicode expressions. There was a concern with `\k`, but that's been resolved. The detail slide is missing the new resolution of that issue... The update since last time: worked out the Annex B grammar. I saw no required change. 2 other changes are discussed in the slides.
WH: What function are you talking about?

DE: replace.

AWB: We need examples. [DE adds one]

DE: [describes the example]

```
"foo".replace(/(?<a>o)/, (a, b, c, d) => { /* ... */ })
```

Should the group object be in the C or B position? [discussion ensues]

BE: I would say D is the place to put it.

DE: I believe you. I'll move it back to there.

DE: The other things at the last meeting. We considered some changes to the `group` object. The `group` property is only created if there are any matches. It was proposed to always create the `group` object or share a frozen empty instance that you get on a match that doesn't use groups.
TF: Dan, can you show us again the difference in the patterns when destructuring the groups?

DE: Sure, I'll write it on the slide...

writes on the slide

```
let {a, b} = /(?<a>x)(?<b>y)/.exec("xy").groups
```

DE: If you have a variable instead of the regex, then you could default like you do with anything else in JavaScript (so it would be overkill to create a groups property regardless of whether there are groups in the RegExp):

```
let {a, b} = unknownRegExp.exec("xy").groups || {}
```

DE: For existing regexps, there's no detectable change in behavior. This fits into what would have been syntax errors. The other question is whether all the properties should be created, even if they are not matched (e.g., because they were optional). `"y" in /(?<a>x)|(?<b>y)/.exec("xy").groups` This could be useful in a lexer. ??? You could do this by looking at `Object.keys()`. With the current draft, I've left in place that all the properties are created. This is not a strongly-held opinion.
AWB: And if one isn't matched or processed, its value is `undefined`.
WH: It's created with the value `undefined`.
AWB: I think that's less useful than leaving it out.

WH: I agree with Allen that it's a judgment call. Leaving it out allows one to use `'in'` to test it; on the other hand, the numeric properties are always created and get the value `undefined` if they don't participate in the match. It's easier for users if we're consistent with them.
AWB: You need the numerics because you have to count to get the position.

WH: It (a missing match) could be a hole in the arrays.

DE: Say people learn to do this pattern with `Object.keys()` or `for .. in`. Then they'll be surprised. Instead, if we leave out these features then we avoid confusion for the users.
AWB: My gut feel is that there's a enough difference between the two styles that you won't have a confusion.

WH: Yeah, but if you make a typo... I would rather be consistent with numerics.

MF: It's useful to get a predictable object shape coming out of `exec`.
DE: We would get a predictable object shape if we always created own-properties. That might provide a trivial perf advantage. Things are more optimizable if the shape is consistent.

BE: Yep. Sounds good.

DE: Brian and I have been talking about this issue for a while. I could go either way. I feel weakly this way, others feel weakly the other way.

WH: I like it the way you have it now.

MS: It isn't created right now?

DE: It is created now.

MS: Created, but `undefined`? Just like the numbered ones are. Ok, that's the way I like it too.
DE: The only change in the spec I'll be making is moving back the `replace` parameter to the last position. Given that, is this proposal ready for stage 3?
WH: Yes.

BT: Seconded.

BE: Great.

AWB: Hearing no objections.

BE: Nice, thanks Dan!

Conclusion/Resolution

- Stage 3

10.i.e RegExp Legacy Features for stage 3

(Mark Miller)

- https://github.com/tc39/proposal-regexp-legacy-features

AWB: That was the last RegExp item. Anyone have more to say about that?

MM: These are things like RegExp.leftContext and a bunch of weird static properties on the RegExp constructor itself that have spooky effects. If I could just get rid of these, but of course I can't. There's an agreement between browsers, so we're better off codifying it than not.
BE: I am so sorry...

MM: The goal here is to strike the best balance between acknowledging reality and keeping things codified and nailed down in Annex B (and all that means). It's already at stage 2, as part of this general attention to moving RegExp stuff forward, I would like to bump this today.

MS: Was it research that all browsers do it or just some browsers?

DE: These semantics are different from what each browser ships. In terms of configurability and what happens between realms.

MM: In terms of minimizing damage...

DE: This was reviewed by people working on regular expressions in V8 and was thought to be web-compatible.

MM: Dan's right. The properties are the properties that we gathered from browsers. I don't know whether they're union or intersection, but it's small in any case. This proposal also makes some semantic differences that we expect to be backward compatible. Nothing spooky happens in some cases. They're also configurable and delete-able.

AWB: And it's not just that it's structured, I assume that if you evaluate a regexp in one realm, the results are not visible in another realm.

MM: Correct. No browser has any explicit action-at-a-distance. What coupling might there be between realms unless we outlaw it. That's the coupling that could have been ??

AWB: ??

MM: This suppresses the spookiness both between realms and subclasses.

DH: From the description, I can believe the changes are web-compatible. They sound like the kind we would take a chance on. My biggest question that I don't think affects stage 3: what about Annex B? We need to review the goal and its role and judge what its future is.

MM: I enthusiastically agree.

BT: Motion to append Annex B discussion to the agenda.

YK: +1000

BE: The question is whether it should be there?

DH: Should Annex B be in the standard? Why should it be there? That helps determine whether something should be in Annex B.

BE: Yeah.

DH: Earlier this morning we had a discussion about the mechanics about the spec document itself with regard to Annex B. That discussion bled into the purpose of Annex B. Should Annex B be there?

WH: You just asked "should Annex B be there." The answer is "no" but we *have* to have it for web-compatibility.

DH: I didn't mean, "should that material disappear from the spec?", I'm saying "should it be differentiated?"

WH: Ok.

BE: Ok, right.

DH: To be clear, I'm not arguing for anything. (We shouldn't do that right now.) I'm just saying that we need to find clarity around this question. There are mismatches in different people’s heads here.

BE: I want to frame this question. Part of this seems to hinge on an empirical question: "How much is required for web-compatibility?"

MM: That is orthogonal... Oh, I see what you're saying...

WH: It's not what was implemented, but you need to figure out how much of this Legacy RegExp stuff is required for compatibility. This distinction arose in the very proposal you're presenting now.

MM: Many of the things in Annex B were admitted to Annex B only with the understanding that they normative-optional. Everything in Annex B needs to go through the entire process again to determine if they're normative-optional... ???

BT: We are diverging.

DH: We need to grapple with this question. But I'm happy to leave it at that. But my question is: "Does Legacy RegExp go all the way through stage 4" and then we deal with this Annex B stuff at that point?

MM: The subject doesn't have to do with where (Annex B) stuff appears in the text. It means that a conforming system can omit Annex B stuff.

YK: The point you made about the process is a true fact of the procedure. But if the people in favor of not conforming can be clear about their actual objections, then we can address them directly rather than need to sort out the larger Annex B issues.

DH: You're operating under several constraints with "normative optional." I think it's important that we satisfy these constraints. We should also deal with "forked ecosystem." The normative optional construct is one that says "we're not worried about one JS ecosystem may not support this feature" and that leads to balkanization. Part of our role as a standards body is to solve and prevent problems of balkanization. We should see if there's a better way to deal with this than normative optional. I wonder if Annex B and normative optional allow us to mitigate balkanization issues.

MM: I don't think anything we've discussed affects whether we should proceed to stage 3.

DH: The point of order is that I would like us to have a plan to address this.

MM: I think we're agreed on that.

AWB: I think it's clear that today Annex B is the place we have to put it. There's an orthogonal issue of: what do we want to do with Annex B (whatever's in that bucket). ??

DH: I don't know what the criteria are for Annex B.

BE/WH: Let's have the discussion at the proper time.

BE: We can assume a good outcome and if something goes bad we can discuss.

MM: Dynamically scoped is also spooky action-at-a-distance. Any objections to stage 3?

BE: I'm so sorry.

DE: There's one point I want to make about it. The subclasses restrictions. There are some regex classes in the wild. XRegExp is an example, which is an old-style subclass instance of RegExp, so it doesn't really get touched by the detection in this proposal which doesn't modify new target, but it seems a little bit odd... The justification for restricting it from subclasses is that it would give bad answers for specific subclasses for example… it might leak implementation details. XRegExp will get past that because it won't get detected.
AWB: Well it doesn't not get detected, it just doesn't meet the definition of what we call a subclass.

DE: Well, but this document mentions certain concerns which XRegExp should qualify for. And it could leak implementation details in these dollar sign things (special variables)... maybe it could cause unexpected behavior.

MM: I think the trade-off is highly in favor of doing with Claude codifying (preventing of sub-classing of spooky...). Are you objecting on this basis?

DE: No, I'm not objecting on this basis.

WH: Who were the reviewers for this?

DE: I was a reviewer.

AK: Then it cannot go to stage 3 until we have another reviewer.

WH: There are things in there that I don't understand. Like why it has anything to do with inheritance.

MM: I'm happy to clarify with you offline.

MF: We also agreed that any proposal planning to advance would be available 1 or 2 weeks (I forget which) ahead of the meeting.

AK: It's nice to have it advance if it could. But what we got out of it is that it needs eyes on it.

MM: Withdrawn from stage 3 for now.

AWB: Are you planning to present it at the next meeting?

MM: Yes.

DH: Add it to the agenda for next meeting right now.

Conclusion/Resolution

- No change in stage. Reviewers identified and it will be proposed for stage 3 next meeting.

10.i.f SIMD.js status update

(Daniel Ehrenberg)

DE: I'm no longer championing SIMD.js. It was pursued as the best way to provide to users in JS. But at this point, WebAssembly is sufficiently far enough along. It is hard to optimize sufficiently well for the non-asm.js case, so better is to simply specify it for asm.js. The decision could be reversed, but it's currently on hold.

YK: I'm interested in Brendan and DH's response.

BT/AWB: If we have a proposal that's at stage 3, the champions should write up a status report.

YK: Is Mozilla also abandoning it?

PJ: The main reason it's on hold is WebAssembly. They were involved with the SIMD proposal as well. It remains to be seen whether WA will fill the gap, since SIMD is not part of WA yet. Hence maintaining the 'on hold' status. Also, just like in C++, when people want to do SIMD, they can use asm or intrinsics. We'd like to see something simliar for JS, but it's a big burden on browser developers.

BT: Is it appropriate then for V8 to delete their implementation?

PJ: Yes I do think it is, but I don't have control over V8.

SYG: We still have the code available at Mozilla, but the main people working on it have moved to working on WebAssembly, and basically the JS code for SIMD.js is basically dead.

BT: I'm getting a feeling that there's an unstated position of the V8 team that SIMD in JavaScript is untenable and will not go forward.

AK: V8 does not conclude that SIMD.js makes sense at this point.

BT: I don't see a path forward for consensus on this point.

DE: That's a good point. Perhaps I should switch this to withdrawing.

BT: I'm just getting this out there. I do think that stage 3 is not the right thing for this.

AWB: Sounds like it's losing stages here.

YK: It sounds like there was some implementation problems with making SIMD.js performant, like developers who tried to use it would be disappointed.

AK: The roadmap to get to a performance JavaScript implementation looked long and unmaintainable. Were there a huge push from JS community we may have thought it was worth it, but with emphasis moving to WebAssembly it doesn't make sense anymore.

YK: Does the V8 team even believe wasm is a "real thing"?

AK: The V8 team's perspective on asm.js has changed over the last 3 years. The team tracks the performance of asm.js in the wild, etc.

YK: I guess what makes me worried here is that we're prematurely rejecting potential future use cases.

MS: Apple's point of view is we don't think SIMD.js is appropriate as an addition to JavaScript because of the perf, but also the use cases and how much work it was. We think that wasm is a better way to bring this to the web. I think in the past I made it clear that we didn't want to implement SIMD.js.

BT: I'm personally sympathetic to wasm, though there are some concerns. I have met some people that *did* get really excited about SIMD.js. I can help you find those programmers.

PJ: There are more JS developers than C++ programmers. The places where SIMD improves performance are typically isolated to small hot loops. Forcing developers to bring in the whole wasm toolchain to extract better performance in those cases is a big ask.

DT: Is there a timeframe for whether wasm solves this problem? And if it doesn't should we bring SIMD proposal back?

YK: That could be addressed by moving it to stage 2: "the committee believes that JS should solve this problem". The claim that JavaScript programmers will get enough value out of the existence of wasm is as yet unproven and it's inappropriate to assume it will.

MM: You're saying stage 1 rather than withdrawing it?

YK: Yes.

MM: I think it should be withdrawn if it is championless.

AWB: I think it's either withdrawn or it is stage 1. But before either of those, we need documentation for why we are making the transition.

YK: Yes just writing down that we think it should be addressed by wasm will make it easier to address in a year.

AWB: I'm reluctant to say: "Dan please write up something because there's very little motivation to do the work." What I would say is that: "The attempt to be withdrawn has been postponed".

WORDS...(lol).

AK: Is there any advantage to actually remove this from stage 2?

AWB: It's not withdrawn until we have the write-up that says it's withdrawn. As soon as this disappears, all sorts of people will start asking what's happened, why has this gone away? Part of our job is to document why we make decisions.

MM: The notes from this meeting are made public so people will see why we did this.

AWB: These notes are not a coherent presentation of the reason for the transition.

AK: I think there should be an announcement. When I did something similar it got picked up in es-discuss.

AWB: I'm suggesting to wait because the champions should do the work to write-up something.

DE: I'll write something up whether or not there's procedural pressure.

YK: If someone asks for stage 3 they should do a write-up. I think Dan is doing the right thing.

AWB: I do know that pressure goes away and people have other things to do.

MM: The amount of new information is sufficiently low that we should move onto the next topic.

YK: We should find someone who could champion it at stage 1.

PJ: We would like to see it go to stage 4, but I can be a caretaker champion.

DT: You're owning up to asking the questions in 6 months to know whether we address it again.

DE: If we decide to reintroduce it, it can go straight to stage 3 or stage 4 if appropriate.

AWB: Probably not.

PJ: It has been reviewed. If it is picked up, I don't see why not.

YK: The withdraw claim is quite a strong claim.

BT: If we do withdraw it, the code will be removed from Chakra very quickly.

YK: You can't reuse it in Chakra?

BT: So, I'm still feeling I'm going to have to argue strongly for not deleting this code if we move it back to stage 1.

PJ: Maybe the wasm picture will be clear in a couple months. I'm not sure how far along things are. From that sense it should remain at stage 3.

AK: I don't think Chakra would ship this in Edge and then have the committee withdraw it.

MPT: Nobody in this room right now really wants this. We have an obligation to the development community to signal that this isn't happening.

YK: I don't agree that nobody wants this.

AK: That's why I'm saying we should withdraw it.

MPT: I would tend to agree with that.

YK: That's what V8 decided.

AK: You're guessing that's what happened.

YK: There seems to be diversity among engines. I didn't mean to imply that V8 had a nefarious position.

AWB: It seems clear that this is no longer a stage 3 feature.

BT: That's clear. Stage 3 is not an option.

YK: ??

MPT: If we were to back it up a stage, I could put in a statement that says we want to see what happens with wasm before we keep pushing this.

YK: V8 has made it clear they are betting on wasm.

BT: I think practically speaking, based on what I have heard, it would be extremely difficult to get consensus on this even after wasm has been around for a while.

YK: Imagine a year from now, wasm fails to get traction or adoption. Then would your answer be different?

BT: I think that's an interesting question.

AK: I get the impression from people at Mozilla that they're not excited about SIMD.js either.

BT: You will find people like that on the Chakra team as well.

YK: I think V8 took the plunge.

MS: Just to be clear, before this meeting, we made it clear that we are not supporting SIMD.js. Essentially: "I don't think this is appropriate for the standard, but if it becomes part of the standard we will implement it".

YK: How did we get consensus at stage 3.

MS: ?

YK: You're basically saying that we've never had consensus for stage 3.

DE: Michael was very consistent on this the whole time.

MPT: Regardless of the past, I'm looking at two implementers who say they wont do this. We need to decide from here.

MS: If this does become part of the standard, we will implement it.

BT: Withdrawal is more reflective of the likelihood that we'll get consensus on this feature.

AWB: Do we want to go forward exploring ??? computing hardware.

WH: I have a feeling we will regret withdrawing it in a couple of years.

YK: I think it's a legitimate question that we don't know how wasm will go.

PJ: We have tons of use cases for SIMD.

YK: It seems like engines are withdrawing their interest in it. Engines are making a statement.

DT: I think what we want to do is get a new consensus and the presentation should be "what is the ?? we should get to." ...

YK/BT: We should move this to stage 1.

MPT: Officially this is being moved to stage 1 because browser implementers won't like this as much as they thought they would.

people: No...

AWB: No, Daniel will write up a status report to make this more clear.

DE: I'll write up a status report and we'll talk about it next meeting.

AWB: Put it on the agenda for the next meeting.

AWB: If you're a person coming back in two years and you say "here's a thing that TC39 is working on and put it on a shelf—why is it being revived?"

DE: Right, so I'll write a document explaining that.

AK: Here's the thing, can we make a change sooner or later? There's no point in holding devs in a state where they think something's happening, but it's not.

MPT: The move to stage 1 removes the community concern.

BE: If the implementers and browsers don't want to implement it then there's no reason to leave it in stage 3.

YK: I think there's broad consensus that it should not remain at stage 3. Even if we don't ship wasm, it's fine that...? doesn't happen. I think it's ok to communicate that we think that this will work out in wasm...versus not shipping it at all.

DE: For some broader context, a lot of the engines have shifted to focusing on other types of performance (not numerical perf) maximizing softer factors like dynamic performance. and SIMD is more in the category of numeric performance. That's why we prefer to focus on wasm rather than SIMD.

WH: Both kinds of performance are important.

DH: I apologize that I missed this...this might be a slightly different perspective: I think that WebAssembly has changed the context in which we talk about this...

WH: I don't understand what you mean by changing the context.

LBR: We are over the timebox.

AWB: We have a conclusion. We want some documentation about why that is. Dan will write it up.

DH: So the point that I wanted to say was that wasm was working towards including SIMD in the spec. I think there is a possible route, where the web gets SIMD via WebAssembly. And then at some point that will help us understand if we need SIMD in the JS side. That gives us new signals about developer interest in this feature. I'm fine with demoting, whatever it is that we say, let's not say that it's not possibly interesting and that it doesn't matter. They may be interested in it in the future and we should leave that door open. Why prematurely tell developers that a thing you might want will never happen?

DE: That's a great point about getting more feedback from devs. I want to mention that the feedback this committee has given has been really valuable to inspiring the implementation.

BE: It's not going away? It's going to stage 1?

AWB: It's probably going to stage 1.

Conclusion/Resolution

- Stage will remain at stage 3. DE will write up the reason for deprecation and present it. At that point, it will transition to a lower stage (possibly withdrawn). It is proposed that Peter and Yehuda will become caretaker champions of the high-level goal at stage 1.

10.i.h Intl.Segmenter (requesting stage 3)

DE: Unicode also defines an algorithm for finding breaks between words and sentences, which CLDR tailors per locale. On slide #4 there are new methods that allow you to jump to the next segment. Previously it provided forward or reverse iteration through a string. We decided to wait for users to ask for an improved API. So I changed the API to have preceeding and following. Also to define break type. It's at stage 2 right now, and now that the API is closer to being finalized I want to ask for more reviewers.

CP: I can review.

JF: I think I can get someone from PayPal to review.

DE: Any thoughts on the change in the design?

AWB: It wasn't clear to me from what's on the current slide what the actual change was.

DE: Oh. There used to be that segment iterators had an internal slot ...forward or reverse segment interators...you could call segmenter...? You can jump to a point, going to either the preceding or the following position. You can access the index and break type getters to figure out what that segment was. That's the new API: It's more expressive. You still have the ergonomic iterator (segment iterators) and you can do for...of. So it doesn't change that simple use case. Glad to have reviewers; thanks.

AWB: OK, so we're not actually advancing to stage 3; we're just preparing by having reviewers?

DE: Yeah, I hadn't previously asked for reviewers because I was unsure about this aspect of the API design. So yeah, still at stage 2.

AWB: Sounds good.

Conclusion

- Remains at stage 2. Stage 3 reviewers identified:
    - Caridy
    - TBD from PayPal

10.i.i Test262: using root license as default for files

LBR: This is mostly to ask to defer to discretion of lawyers. (My company doesn't have any working on this.) I'm trying to make Test262 a nice place for new contributors. It's already too complex for new contributors. We struggle with pull requests. We require every test file to be prefixed with two copyright lines declaring the copyright for who is contributing. But today we already have the CLA. Considering that, for new tests, without these copyright lines, we're deferring to the license that's in the Test262 project. I'm not looking for consensus now, because we need to talk to lawyers.

AWB: What you need to do is refer this to Ecma ?? because there is a specific Ecma policy document that specifies that, for Ecma modules, how the software should express itself...so you have to go through Ecma.

BT: That's what he's asking for, is that policy to change. He's not going to ask for that to change until he talks to lawyers. I know what a lawyer is going to say. They'll say you need the copyright header. It makes things more clear and so people can't say they didn't know it was copyrighted.

AWB: But that's OK. With appropriate pushing, we've done this stuff before.

WH: You need the header where?

BT: At the top of the file.

LBR: Every test file.

AWB: I'm confident that if you go, can we get by with 1 line?

BT: He doesn't want one line, he wants 0 lines. Lawyers would always want that one line.

AWB: One line is the best you can get.

BT: Having it at all is the difficulty. We've had people push back because they don't want their real name on the test.

LBR: We've got a lot of contributions without the copyright lines; some of the contributions say I can agree on a copyright as long as I have a CC ShareAlike...

AWB: Strictly speaking you're not supposed to accept anything that doesn't conform to the license.

MPT: When you sign the CLA, you're literally saying your lines of code belong to Ecma. You can't sign the CLA without giving the code away.

MM: Leo, I think what we're all saying is if the reason people don't want to put the copyright lines on there is because they don't agree with the copyright, then removing the lines is not going to solve the problem.

BT: So I think that there is a reasonable argument that it's really annoying to put that thing in every test, having written thousands of tests myself. I've seen the friction it causes, not from philosophical opposition to BSD licenses. I know what lawyers are going to say, I suspect.

MPT: As far as the real name thing goes, if it's a concern of anonymity on the internet, there are ways around having your real name on something.

BT: That's not the major issue here. The major issue is ease of writing tests. We want people to contribute as many tests as possible, so the fewer roadblocks as possible.

LBR: Yes.

BT: We recognize that the copyright header is potentially useful, but is not a requirement for copyright. We just have to make...if we have consensus, the burden of having to put that in every file. [Risk of] not having it in every file is extremely unlikely...

AWB: If the Microsoft lawyers will represent your opinion on this... If you can..

AK: We had a similar case in our tests. We don't have headers in our test and we're keeping it that way.

MF: Are you distinguishing the test files from the source files?

AK: It's just which directories they're in.

MF: Did your lawyer distinguish them?

AK: We didn't ask the lawyer to change the policy on source files.

MS: Our tests do not have the headers and the source files do.

BT: No one here has objections to removing the copyright header: is that the case?

AWB: I don't believe we have the authority to say that.

BT: I'm not saying we have the power to change it. I'm saying we're agreeing that asking for a change is a good idea.

LBR: If I have an objection here, I'm not going to move this forward. I want to confirm: I don't have an objection, so I'm going to move forward.

AWB: I can help you explain that to him if you need to.

MB: In node land, we recently had to reinstate headers in our files (including tests) because we removed them before talking to lawyers. Because we did this without talking to lawyers, it's been really painful. So talking to lawyers is a really good idea.

WH: Can you explain why?

MB: When the io.js fork happened, all the headers were removed, and it turns out... there are laws... We just landed this about a week ago. If you go check the Node repo right now...the first round of the headers took us about a year.

AWB: We have to go through a committee of lawyers.

Conclusion

There was no objection from the committee to removing the headers, and Leo will escalate this with Istvan.

Weak References status update

DT: I did the first round and started a draft. It was left at most people liking it but concerns about what it would be like in practice. Turns out the requires being in large C++ environments. Not a lot of progress has happened there. I'm looking for someone else to pair with me who can do the C++ side so I can focus on what I intended which is the JS side. That will really help in getting an implementation in V8 and node so people can play with it.

MM: Given what you're asking for there's no reason to limit it to those two.

DT: That's true.

MM: Any open source engine.

DT: If there's an engine that's less production and simpler to add this kind of prototyping.

MM: So you're looking for someone who can volunteer to help with it.

BF: I can help, but I'm only going to help with V8, because that's what I know.

MM: And thank you, that makes a huge difference.

DT: I'll sync up with you and we'll see where we're at. Not something we can have by May, but we'll see how far we get. There are minor tweaks to the spec. But none of them are substantial.

Conclusion/Resolution

Bradley will help out and they will report back in 2+ months.

Day ending.....

LBR: I have a link for TR104.

MM: What is ECMA–104?

AWB: It's the Test262 PR that fundamentally .... it points to the github.....

AWB: We don't have any more 15 minute items. So we should call it a day.

AWB: There are hats in a box.

March 22, 2017 Meeting Notes

Allen Wirfs-Brock (AWB), Waldemar Horwat (WH), Brian Terlson (BT), Michael Ficarra (MF), Adam Klein (AK), Dave Herman (DH), Kent C. Dodds (KCD), Tim Disney (TD), Daniel Ehrenberg (DE), Shu-yu Guo (SYG), Michael Saboff (MS), Sebastian Markbage (SM), Bradley Farias (BF), Maggie Pint (MPT), Jamund Ferguson (JF), Myles Borins (MBS), Logan Smyth (LS), Sarah D'Onofrio (SD), Alan Schmitt (AS), Dean Tribble (DT), Peter Jensen (PJ), Mark S. Miller (MM), Leo Balter (LBR), Zibi Braniecki (ZB), Rafael Xavier (RX), Yehuda Katz (YK), Caridy Patino (CP), Diego Ferreiro Val (DFV), Brendan Eich (BE), Lyza Gardner (LG), Istvan Sebestyen (IS)

Report from the Ecma Secretariat

(Istvan Sebestyen)

IS: ECMA-404 received a negative ISO vote. Have to do another go-around and address the comment. Should do it within a month. Ball in our court.

WH: What was the negative vote?

AWB: It was a long list of objections from Japan, both editorial and larger items. Need to coordinate with Chip.

IS: ECMAScript Specification Suite fast track got an ISO number but hasn't started voting. Need eight weeks to start voting and working on translation to other official languages. Will start the ballot on April 12th. It will be a 3-month ballot. Istvan thinks that somebody will likely make some comments, which will necessitate a second round and disposition of comments.

IS: Next Ecma General Assembly is in June. All documents are due two months in advance. We're in the RF 2-month opt-out period as of Feb 14th. Opt-out period closes on April 14th.

IS: What should we do about TC39 management structure? John hasn't been responding. AWB has been doing this temporarily.

AWB: I'm only available until May as temporary chair. Need to work out a permanent chair by then.

IS: Chairman needs to be neutral when wearing chairman hat, but can represent his company's interests when not wearing that hat. This situation comes up often in other standards committees.

AWB: Our next agenda item is a non–technical item about Code of Conduct and it might be good for Istvan to be here.

Review of Code of Conduct

(Leo Balter)

LBR: Last meeting got a lot of feedback; I'm trying to address all the feedback on the current CoC proposal.

LBR: The open issue is whether...

...lots of discussion..

BT: You would like us to approve this document at which point we can at least get Ecma management to take a look at it and make sure there are no problems.

LBR: We can only move to Ecma if we have consensus.

DH: I would certainly move to accept the document with the consensus that TC39 thinks it’s the right document ... we'd prefer feedback than having them reject it outright.

MS: Do we want to include just the Code of Conduct or all related documents?

BT: Are you sure we want to remove the 4th paragraph about how to report problems?

LBR: The question marks denote a place where we don't have an email.

YK: There is no consensus on having a subcommittee.

BT: We clearly need an email, we need a place to report stuff.

YK: There's no process around how the subcommittee gets formed, lots of open questions.

BT: I would like to get consensus that there should be a group of people and then we can figure out how it works.

YK: There should be at least be one person that can receive reports and do something about it.

DE: We've heard some strong arguments for separation of powers, having a separate "HR department".

DH: Have you worked with HR departments?

DT: Obviously you can escalate to Istvan, but formally that's the primary thing for chair: moderating discussion.

MS: It doesn't sound like we're going to be able to reach consensus on the "remedial" part of this document, but not the enforcement.

BT: Document is useless without enforcement.

AK: The message we're trying to send here would be lost without enforcement mechanism.

MPT: If everyone thinks document is toothless without enforcement, can we add some language?

AWB: I'm really uncomfortable with the whole "enforcement" idea vs. the aspirational goals of how we want to conduct ourselves. It seems to take away from the focus on how we want to behave and the focus on correct behavior.

LBR: If I rename it to Code of Conduct supporters, would that help?

MS: Can we add the chair and the Ecma Representative for the time being?

LBR: We've had a discussion around an individual person, I'm not really...I don't think an individual person would be able to solve, for example, sexual harassment. If I'm chair, and someone brings an issue like this to me... I need support to help and/or talk to this person.

AWB: Istvan or the chair in this case would go to the accused person and talk to them.

BT: What if the person involved is self-employed and company policy doesn't apply?

AWB: That's not most of us, but if that's the case I would hear from Istvan.

IS: We have appropriate measures in place; we do not need enforcement here.

SD: I would say it's not just for sexual harassment. But is there ever a reason where the chair may be non-partial in a proposal?

AWB: What should happen in that case, and in the past it has, that's the point where the chair should step away from his "chair hat" and someone else should step into the chair position.

SD: Do we ever talk about that anywhere? It's not really something we talk about. It's maybe something we want to have written down.

AWB: This morning we talked about this. You really want the chair to be knowledgeable but for the most part detached from the tech arguments.

MM: Doesn't the same issue come up with the committee? The "sub-committee" may have strong feelings on these issues.

AK: This is one of the reasons to have multiple members of this body.

MM: Yes, but it may be an issue. All vs. 1 type of situation.

AK: There's no magic bullet here. There's 9 members on the Supreme Court, e.g. This goes to Myles' point about having a robust setup to start with so you don't have to have the process discussion at the same time you have a severe issue. Ran into this at Chromium. How do you deal with something that's happening right now? …a bad situation to get yourself into.

MPT: There's almost no way to pre-select the Code of Conduct committee. I have many personal friends with people like Brian or ??? and because of that I couldn't participate on a committee in that situation.

MS: But recusal, if you had a committee of four, that's easier than trying to form a committee when the violation is reported.

AWB: I don't know what the Ecma executive committee and the GA is likely to say, but I suspect that at the point where you talk about enforcement, they may step up and say "that's not our job".

WH: Istvan said, we can do some things, but the process would have to be approved by them [Ecma]. We can't have a separate patent policy from other TCs. The process is something we need to coordinate with them.

JF: Are we going to punt? Are there a set of things that we can approve on and move forward? If we can agree on 95%, we can move forward instead of punting.

AWB: If we have a set of documents that we're willing to pass on to the executive committee via Istvan, total agreement isn't even needed, we can move those documents up the pipeline and get the initial feedback. There's an EC meeting in April. That's a possible next step.

IS: I think this is a good idea. You will have the first reaction by the next meeting.

LBR: I believe this current proposal without an enforcement sub-committee is not effective. If we don't set the format of how to move forward.

MPT: You will get social-media blowback if you put this out without enforcement. e.g. People on social media right now are pointing out how few women are here. Do not put that document out without an enforcement body.

AWB: I wasn't suggesting putting a document out anywhere except up to Ecma for additional feedback. We can have enforcement in it, just note that they might disagree.

MPT: Can we get consensus for Istvan/Ecma to review? On the document as-is.

MF: Is this without defining any members of a sub-committee or an email address? How are they meant to review that?

AK: We can get that feedback... If it turns out that that's the sticking point, or if there are sticking points.

YK: I think we should move forward with the document.

BT: It would be disastrous for this group's continued existence to not let us enforce this CoC.

YK: If Istvan comes back and says we can't enforce, that would be bad.

AWB: Let's be clear, it's not Istvan who approves: it's the executive committee. It's "us" because it's got members of our own companies on it.

JF: We have a CoC, which most people agree are good aspirational goals. But we're asking what enforcement looks like? Is there a minimal agreement that when there's a violation that there's a set of people get notified?

DH: It seems like we're being sloppy about what "enforcement" means to different people here. What's the meaning here? If the meaning is, literally, that there is a group of people who will meet and decide what to do...if that's a good starting point or if people think that's not enough of a starting point... I hypothesize that there is a set of people who review complaints—might be acceptable to everybody.

WH: I'm fine with that. If that's the only thing that the group reviewing complaints does under the policy. It would be more controversial if, as some suggested earlier today, that group were also tasked with creating additional more detailed discussion and participation policies than what's contained in the policy we're adopting. If that were the direction, we should figure out the policies first.

SD: I just wanted to make a quick closing comment too. We have managed to get through this discussion without causing offense. The focus of terminology has been violators, but actually this is more about helping people feel more welcome. You know I've been terrified of this group for the last month, I'm grateful for how people have presented themselves and hope we can can continue in that spirit.

AWB: What do we need to do so we have lunch?

MS: Are we comfortable presenting this for Ecma [executive committee] review in their meeting in April?

AWB: The Ecma executive committee.

BT: Which companies are on the executive committee? Are they TC39?

IS: Intel is there, IBM is there. Hitachi. Microsoft. Those 4 at the moment.

BT: Thanks.

IS: And of course the secretariat is there but we don't have decision power. We're just the organizer. It is basically the membership.

BT: So it seems like we do have consensus on this document as-is?

JF: No removing that paragraph [4]?

BT: As-is, yes. OK, I move for lunch recess. Reconvene 2PM.

Conclusion/Resolution

- TC39 will present the current Code of Conduct proposal to Ecma's executive committee.

General

AWB: When I said that someone should step up to being the chair, that was really addressed to the member companies. Many member companies have people that are ideally suited to running and mediating the meetings. Please check whether your company might be willing to send such a person to these meetings.

10.iii.a Orthogonal Classes

- https://github.com/erights/Orthogonal-Classes

MM: There's a strong push that when assignment happens it should look like assignment. We had an Aha thought that cuts through this and other proposals: There's a whole bunch of different proposals to enhance classes. Original classes we did the minimum we could agree to, postponed other things. Proposals coming through are lots of those proposed issues. We're wrestling with them individually. Each one has its own proposed syntax, semantics. We try to pay attention to how they cross-cut. But most of these proposals have something to do with adding class members or new thing to classes.

MM: You can divide these into three orthogonal dimensions. You can characterize where they are along each dimension with a different syntactic indicator. You can do this in a way that is 100% upwards-compatible with classes already deployed (obviously a requirement). You can do it in a reasonably upwards-compatible to the spirit and the work gone into these particular proposals.

MM: Rather than combining particular, individual proposals, syntax.

MM: One dimension is placement. All proposals add some kind of member either to class constructor object or to the class constructor prototype object.

MM: Three dimensions:

- Placement:
  - class constructor
  - prototype
  - instances of the class
- Visibility:
  - public properties
  - private fields
- Kind:

- method-like:
    - methods (constructors, async methods, generator methods, etc)
    - accessor properties?
- data property-like:
    - private fields
    - properties

MM: Private state proposal has postponed the issue of "what does a private static state look like?" All of these other things have been postponed because use cases individually weren't strong enough. By adopting an orthogonal framework, anything you don't have a well-motivated reason to prohibit, you should allow.

MM: Particulars: The placement indicator is leading keyword.

MM: The presence of a leading static keyword means "put it on the constructor object." The presence of an initial "own" keyword means "put it on the instance object." I've gotten push-back on the "own" keyword, but it needs to not be "private" or "public" since that's the kind of sematics we are trying to break up into these combinations.

MS: Is "static own" a syntax error?

MM: Yes.

YK: Is that an orthogonality violation? (making speciic rules about "static" and "own")

YK: You don't take the Smalltalk perspective that "static" is just a meta object?

MM: No. It's too late for that in JS.

MM: So, if you just declare a method with no keyword it's already a method on the prototype.

AWB: Well we don't have a meta class......

MM: That's the sense in which I mean it's too late. For those who remember enough history, JavaScript in this regard [is more?] like Smalltalk 76 than Smalltalk 80. Smalltalk 80 had a very carefully constructed meta-braid, which JavaScript does not have. Hence "static own" doesn't make sense without that.

MM: A "protected" attribute would go on the visibility dimension, if we ever decided to do that (which I don't recommend). There's a kind of thing that's data property like..........# sign. This is also compatible with the spirit of the proposed private and public states. The difference between the proposals would primarily be the ....

MM: The beautiful thing about an initial keyword followed by name followed by initial value it looks like a declaration rater than an assignment. This gets us away from the expectation that it should have an assignment behavior.

YK: I think it still hasn't fixed the factoring issue but it has fixed the pedagogical issue.

MM: It also addresses another point of controversy: should class properties be initialized as configurable or non-configurable? I thought they should be non-configurable. But with this orthogonal pattern, it makes more sense for them to start out configurable. "own" means placement on the instance, and similar items start out configurable.

AWB: In the current class design, when we place properties, when we say "own" we are defining a property.

MM: Yes. It's private though, it's not a property so configurability is not an issue.

YK: I'm happy to accept this concession, but don't understand how you arrived at it.

YK: So "own" means "configurable?"

MM: No, own means placement on the instance. It's already placed using defined property semantics and configurable, that should apply to other places.

AWB: In the current class design when we place properties. Like when we say "own" without the private visibility we're talking about defining a property.

MM: I do mean it's specific to properties. If it's private, then it's not a property and the issue of configurability doesn't matter.

YK: But methods are also configurable. ..... I'm happy to accept this concession (which one?)

MM: It's consistent because...let's take "own" followed by method syntax or "static" followed by method syntax or simply method syntax with no "own". The only difference between the three is the keyword or its absence. The only semantic difference should be the placement.

YK: So the consisteny issue is between properties and methods. If you leave off the keyword, the property or method is configurable, it would be surprising if adding the "own" keyword made the item non-configurable.

MM: Trying to stay within spirit of orthogonality. Main value is combinatorial space of things that might get proposed. Most points don't have strong use cases. Enough would get proposed that it would create burden. We get most of the combinatorial space without burden because each element can be understood separately. We get two further benefits for the data-property-like syntax. Comma-separated list of member names with optional initializers in one declaration. And there's also thinking ahead to how this interacts with decorators and what Alan and I are proposing: if you decorate a declaration that is declaring several members, the decorator applies to each of them.

MM: There's some points in the cross-product space that are nonsensical. We can statically reject them without causing surprise, because all accepted programs follow the consistent semantics.

Oh, I'm sorry. One more detail: The constructor!

The constructor has to be specified as method-like.

YK: "Method-like syntax", can you clarify? I've read the document and I understand...I was thinking concise methods might eventually happen. We had a proposal before where you did not have to put curlies (??).

MM: I'm not only trying to rationalize proposal on the table, but also to rationalize how to further extend semantics so that they fit in the framework and keep consistency.

YK: So it looks like a method, what are the semantic consequences?

AWB: In this proposal, we have reserved the meaning of the #constructor because we might want to have something but we're not quite sure in the future.

MM: 3 reasons to prohibit something from the cross-product:

- it would be harmful
- it would be non-sensical
- we have not decided whether to allow but could decide later

YK: So what does it mean if something has method-like syntax?

AWB: A method that is a function that is defined as a property. Syntactically it's defining a function.

MM: "method-like" means it's defining a function in that you can use "super" inside the definition and it's using method syntax.

YK: Static method, it goes on the prototype—

MM: Static means it goes on the constructor.

AWB: Third dimension where methods occur. Two points, syntactic, what is the syntactic form of the thing being placed: a binding list or a concise method.

YK: What is the syntactic difference between things that are data-like—

AWB: That dimension is a syntactic dimension; whether it's a binding list or a standalone single declaration.

MM: And you can't declare an accessor method as part of a binding list.

AWB: There's actually a pull request that Mark hasn't accepted yet, the paragraph on constructors is kind of bogus. Has a whole file of examples taken from some of the existing proposals recast.

MM: Wonderful.

MM: Any objection?

BT: I have a number of points, but let's let Dan go first.

DE: My intuition is that this is one more dimension than what might be the common understanding of objects. This dimension of "own" vs. being on the prototype is definitely there, but the way that we look at classes, we don't have to think about adding the method to the prototype. With the public and private fields proposal, and the private methods straw-man, it sort of retains this property. You don't have to be explicit about the prototype vs. the instance. This is intuitive and fuzzy but seems important. Talking about "owned" vs. "notowned" gets into the weeds and will confuse people. I'm espcially concerned about non/owned methods. People might think they are supposed to put "owned" on everything, which would be hard to optimize.

MM: I understand the point. The philosophy I bring to language design is represented by goals 8 and 9: be understandable in a shallow and wide sense and invite understanding to become deeper and narrower. JS is good at that now. New people learn patterns initially but then over time they can learn the deeper semantics. The dominant use cases would be the reasonable common ones. The regularities suggest to people where they can explore deeper and find additional ways to use the system. [MM/MARKM: review and edit this]

YK: I agree completely with that philosophy.

AWB: ... Irregularities that for example are in the present proposals.

MM: The proposal uses syntax like x = 3 to declare a property on the instance.

AWB: So if you write "method"...

MM: If you write a method it's on the prototype....etc..

YK: Isn't that just the exact same consequence as with Smalltalk: "static" is already inconsistent.

MM: Static in this proposal isn't inconsistent. It doesn't have the regularity that Smalltalk has, but it has a regularity to it that's appropriate to JS.

YK: The way you get the regularity is by syntactically disallowing a field to not own .... ???

AWB: Another difference between Smalltalk and JavaScript, in JavaScript I don't think a programmer, even a journeyman programmer, can live without knowing about prototypes. That entity really is part of the language where in Smalltalk, it isn't. It's a slightly different scenario.

YK: I have an orthogonality spider sense here. ...... lots of comments about specific syntax .....

MM: What I'm saying is that's the non-orthogonality of the current proposals. With this proposal, those non-orthogonalities go away.

YK: What I'm saying is ....

MM: Distinguishing between rude surprises for authors versus rude surprises for readers. Readers do not have any of these non-orthogonalities.

YK: So if you don't see a prefix, it's means the prototype.

MM: Lots of people will ...bah humbug, lost it...

WH: How would you put a data property on a prototype in this proposal? I understand that this is a rare case and you don't want to allow it by accident if someone omits the keyword, but if you do fall into that rare case, how would you do it?

MM: You would create those imperatively, like you do now, instead of using declarative syntax.

AWB: An idea kicked around is possibility of static initializer in a class body, at class definition time. The logic of putting a real data property on the prototype is so rare...that it doesn't deserve having syntax.

DE: At a high level I like the goals about orthogonality. Aside from that big developer intuition thing, I'm not sure about adding "own methods". It seems a potential source of confusion and is redundant with property initializers. They could end up being confusing. The big use case I've heard about is people wanting bound methods, which this proposal doesn't provide.

DE: There's also an example of having an "own private" method. Maybe this would be in conjunction with lexically scoped methods in class bodies. I'm concerned if the first private method thing that we add is "own", it would catch on and be hard to optimize. Might we consider excluding that case?

MM: If we decided to prohibit those cases, that would still be consistent with this approach, it would just knock more members out of the cross-product. I would consider that to be negotiable. I also want to say that at every moment there is something concrete being proposed. Thus, the concrete proposal is that methods stay in the cross product but constructors are not included in the cross product.

AWB: An important thing about the matrix here is understanding the orthogonality. The more holes knocked into it, the harder it is to get the concept. Private methods...the utility is low but we left it in proposal to fill in matrix as much as possible. Was hoping we'd have something like lexically-scoped ...?

WH: Lexical scoped function.... could you clarify what you mean?

AWB: Since classes now have lexical scopes. There's no reason you couldn't have function declarations inside of their lexical scope.

WH: No reason, other than that the syntax is already taken.

MM: The reason I'm resisting knocking out methods is exactly what Allen says. Once you have an orthogonal framework. Once you knock it out, everything has a ....

DT: It's useful like Dan where he's mentioning these things, let's not chase down resolving particular ones. With respect to going to stage 1, these don't have to be decided now. Want to be sure we don't deep dive.

DE: I'm not sure what to do as the champion as the private state proposal if we decide go ahead with this.

DE: There was a subproblem: it could be confusing if you have a property declaration goes onto the instance and a method declaration goes on the prototype. We could solve just that subproblem without introducing this entire matrix by adding the "own" keyword. We could do that along the way to adding private methods. The thing that's good about having public/private support explicitly rather than using lexical methods is that you don't have to change call sites (e.g., by passing "this") in order to change visibility.

YK: There's a general point I want to make about if this is even an ok proposal is this an OK time for that?

Multiple: No.

MM: This proposal cross-cuts a lot of existing proposals and the whole process architecture does not accomodate that. I think that's a bug. We all know as language designers that we need to be raising cross-cutting concerns and have a way of dealing with issues that cross-cut. I'm going to leave those procedural issues to AWB.

DE: Alan has raised this before but the cross-cutting discussions have been happening the whole time among the champions of decorators and public state and private state.

MM: There's been discussion of cross-cutting concerns, but there hasn't been a proposal on the table whose effect is to have cross-cutting impact on proposals that are e.g., farther ahead of them.

AK: Let me restate from Dan's perspective. The fact that this comes as a separate proposal instead of part of what's being done from those other proposals, which seems like a change in process from what's already been happening with the other proposals.

MM: There are multiple proposals that this effects. We could merge with them, but it's more than just private, it's also the public proposal as well. Decorators, etc.

DE: I can write up a doc for the next meeting about the state that we are at for the cross-cutting considerations.

AWB: What we are asking for as this being stage 1 is really figuring out how this integrates with the other proposals. We actually considered just filing this as issues, but it would get lost at the issue level for those proposals for something that is an overriding thing. That's why we did it in this form.

BT: I think on the recent topic of process, it seems strange to me that stage 1 means we are giving consensus to change stage 3 proposals.

MM: I agree that's what stage 1 means. We could propose this as an amendment to the private state proposal, but it seemed to deserve due process.

BT: Right, I brought this up to suggest that stage 1 is not a high bar.

MS: This would subsume private state?

AWB: This would probably combine with private state and public field proposals.

DE: I'm not sure how to proceed as the champion of the private state proposal. I think it would be reasonable to file issues within the proposals to have all of the discussion. Making it a separate proposal. How can we have multiple state proposals is a bit of a weird state to be in.

DT: It seems like it's approximately both. The idea of doing this consistent thing seems good, we should discuss. There are champions of current proposals, championing their use case. We should make things consistent across the use cases. As the consolidated thing catches up, then we can decide whether or not to merge it. We don't have to couple them so strongly.

AWB: To some degree both the public state and private state proposals were stalled at stage 2, but there were enough issues with the edge cases that they were having trouble advancing to stage 3. Take this as an offer and approach to being able to advance them all.

DE: I don't think there are that many issues. At the last meeting I laid out the issues that were open, we disagree about them, but there's a small number of issues. I would like to be able to have a discussion about which syntax we like better: the one that ?? and I are championing or this one so that we can decide and we can advance one of the syntaxes.

AK: This reminds me of the discussion of decorators and annotations. We could have had them advancing together, racing to get the "@" symbol usage. But that would have been bad. Having multiple competing proposals seems like a bad thing.

MM: When it's procedurally possible, I will propose merging the proposals.

DE: We've discussed this multiple times in the past, and had reasons to not merge the proposals.

WH: Have we resolved the cross-cutting issues with syntax?

DE: I believe we have.

WH: Does that match Mark's proposal?

DE: No.

WH: So we haven't resolved the cross-cutting syntax issues.

AWB: At the last meeting we still had intense discussions about the syntax and we didn't have consensus.

DE: But we identified all the issues...

MM: Independent of .... with a proposal for an orthogonal syntactic framework which points out all of these different cases. As long as this is viable ... this should absorb the syntactic issues from the other proposals. Then the other proposals, such as "what specifically does a private property mean...independent of placement" are semantics that can be individually worked out in the separate proposals.

DE: I'd like to hear from people who have actual feedback.

BT: If I can get to my actual points. When you sent the mail about this proposal, I became extremely excited about it. I could see how the syntax aligned with my mental model of the object model. And then I went and talked to some JS developers and universally the reaction was viscerally negative. They partly reacted to the "own" keyword. A lot of developers don't have a...

MM: Was it the "own" keyword or something else?

BT: The presence of keyword was the biggest problem. Coming from current programs where you can do data properties without a keyword, saying "add this keyword for your 99% use case" was not an easy sell. there's an arguent to be made that members today and new members tomorrow have default placement that makes sense. So allowing the implementation to place members based on the kind of member is worth more than the orthogonality. There's other issues, e.g., per instance and per class allocation semantics.

WH: You said you liked the "own" keyword but found users don't like it. But at the last meeting we had users who don't like NOT having a keyword on a definition (because it makes it look too much like assignment) so we have users on both sides of this issue. How do we reconcile them?

MF: Well there are other options like ":=". They weren't complaining about the lack of a keyword, they were complaining about the similarity to assignment.

DH: Can we forget that := was ever brought up?

DH: I have a hard time believing that the similarity to assignment is actually a source of confusion.

DH: There's an expectation about different constructs. When I think about a private data property, I expect that to be on the instance. To have to say that every time seems like it's too verbose for the common case.

BT: I would submit that we have evidence to the counter, given the significant usage in the ecosystem already.

DH: To have to say "something else" to qualify what seems like it should be the default behavior for that construct seems wrong.

??: and you would end up needing a keyword to indicate "put it on the prototype", which seems like a problem.

BT: I also want to make a broader point: I'm not saying that people I talked to represent the broader community, but it behooves us to realize that we can't really evaluate benefits of orthogonality without talking to developers and making sure they can even see that there are axes in place in the first place.

AWB: And when you talk to developers who have already been using some syntax to do this, that's a different sample than talking to developers who have not had a syntax.

SD: From the F12 development perspective, I've had to remove the word "proto" from Intellisense when they type in window....the average JavaScript developer doesn't think this deeply. We've had to remove things because people don't know this stuff.
YK: My concern is part of the way I've been thinking about decorators. And as Dan pointed out working with people working on these other features. Decorators are a descriptive API. If you can do "@own" I'd also like "own static". There is a problem where you may be forced to type a thing that gets over-written by the decorator. In your code you may have to .... have some code to put something on the prototype.

SYG: [Asks BT what objection is about "own" keyword from developers]

BT: I think there's a perspective that I think might be true that default placement of a member should just depend on what kind of member it is. If we have a new member in the future we may have more ... trade-offs?

SYG: I guess the trade-off is, given that the default placement of, say, function bindings is confusingly global if you don't use ?? ...if we could go back and look at it, people would want it to be local. We want to, going forward, have default placement?

BT: I don't think I follow the question.

SYG: To me, I buy the argument that own vs. var or let consistency makes it easy to reason about. Two: existing programmers find that having to type "own" is onerous, so it makes default sense for these to be on the instance. Third: I think that for functions—

AK: He's saying, why do you have to have a keyword before a variable declaration?

SYG: It makes sense for me that default placement for variables is in function not global. Given that we have to live with the need to type var/let/const, it seems—

WH: Are you saying that variable declarations should look the same as assignment? They do if you omit a keyword, so what does "default placement" mean?

MM: If I may. I think Shu is saying that the cases are analogous and that declarations should have a keyword.

BT: I also happen to agree with that but no one that I talked to found that argument compelling.

BT: Since it's in an object literal context it seems weird to have to put a leading keyword.

DJR: Well, yes, it doesn't have the colon. You can make the argument that class methods or instance methods should have a keyword as a declaration point, but they don't. In this case, we just use a different initialization token. We could use a ":", an "=".

MM: Feedback from users is tremendously important. You [BT] have some and I have zero. We almost never have a scientific sample, but. The people that you asked: are they already using a syntax for members? The user feedback that I would value more is given the shape of this proposal vs. shape of other proposal and given a population of JS developers who are first learning classes in the context of these proposals: which makes it easier?

BT: I'd love to see some investigation into these questions. I want us to take a data-driven approach to which is the more easily-learned conceptual model.

MPT: I just took management of a team of 7 entry level engineers. They are actually converted network admins. Let me teach them classes next week.

MM: Please let us know ahead of the next meeting. I'd love to get the answer to that.

AWB: Which version of classes will you teach them?

MPT: I'll teach them classes as they exist today and then say: "If I had added this 'own' keyword, would that help you?"

AWB: You might try putting up two slides to show them both styles as acceptable alternatives and let them choose, that would be great.

MM: If you present them both as pure (peer?) things at the same time, that would be great.

DH: Doing user studies is good. Another form of validation I have found really illuminating. I am an outlier in that I don't always look for data-driven solution to everything...it's easy to over value how much we derive from data. There is a surprisingly illuminating exercise you can do, particularly when you'e talking about fine-grained details which you scale across your code. I'm not talking about canned example like in the slide, but I'm talking about taking some real life code and do the A/B comparison. It's particularly useful when not doing it in a confrontational setting, privately. Do you remember when I did this with comprehensions. It's just shocking how much it changed the way we thought about how these things looked. I feel very strongly that if we did a similar exercise we'd learn a lot. And it might be valuable to share that with people. This is as subject as anything to bias, but it can be very illuminating.

AWB: I want to follow up on that, we actually did that when we did the original class design. The Smalltalk class hierarchy that I implemented and reimplemented in four or five different variants, about 500 lines of code, very useful.

DH: Another quick ...... the shape of the code related to the code around it.

MM: I accept that this is a needed form of validation. Since there is a large body of Babel and TypeScript code...

SD: What's the difference between some of the other related proposals and this one? We need to decide about that before we can judge how this would be for users...

MM: My premise in all of this work —...

SD: Is there some way that we could have these three back-to-back and agree where the difference is and figure out the stop-gaps?

WH: This discussion is an outgrowth of the issues from the last meeting. That's what we're doing.

MM: What remote-Dan was also suggesting, now that this is on the table, that we use the issues list on all three proposals to discuss what it takes to reconcile them with each other.

SD: Sounds like your proposal is an addendum with placement, but then we branched out to think about what customers think. How do we get back to focus about ...moving this to stage 1?

MM: There's a special focus on leading "own" because, concretely, existing proposals are incompatible with this one in this exact way. Except for that, I think that everyone would accept that this proposal is upwards-compatible.

BT: We are well beyond our timebox. I have additional points and there are raised hands. Can we have a conversation tomorrow?

MM: Would people find it acceptable today, while there are 3 people in the queue (of comments) that we take these comments and no others?

MPT: It's going to add 10 minutes.

General: Ok....

WH: Trying to figure out what the various options are. What I see as available options are:

- A. Instance property definitions don't use any keyword.
- B. Use the `own` keyword, as in Mark's proposal. But Brian says users don't like that. There could be a couple reasons for that and I'm curious which one it is. One possibility is user dislike for any keyword. The other possibility is users not liking having to use "own" on properties but not on methods, which I can very much see how it would be annoying and confusing. This brings us to a third choice:
- C. Have a keyword (example: `prop`) to declare properties — all properties. Could maintain orthogonality by combining it with static if you want a static property. For properties you would need to `const` or `prop`. These are the three possibilities I see, I haven't heard any others.

MM: Okay, well umm. So Dean...

DT: We're way over timebox. The commitee has spent a big chunk of time, it's clear the committee is very interested in this. Should this be a stage 1 thing where the committee can really look at this and begin exploring it in more detail? To me, can we go for consensus that this is something that we want to pursue and the rest of these questions become a stage 1 and stage 2 conversation.

BT: It's well-taken that the general bucket of people that haven't learned class syntax is larger than current users. But let's be clear: talking about tens of thousands or more using currently-proposed syntax. Not insignificant. If it turns out that this proposal creates viscerally negative reaction, I think we will have a problem regardless of what we're saying. I don't know how to weigh that versus the other things.

BT: I think also the story gets worse when you ask "What should TypeScript or Babel do with existing syntax?" Very hard to deprecate syntax, especially when requiring more boilerplate. You can throw warnings, etc., but that makes them more angry. You can leave the syntax in, but that breaks the mental model. You've got something without a placement sigil. I don't think that works either. Curious what you think.

MM: I would take exactly the last of the proposals. I would leave in existing syntax, make it synonymous with new syntax. Don't know enough about TypeScript environments and how well they encourage users to change things. I would provide an automatic tool that can transform them from lack of own to own. I can't force people to use the tool. The private state proposal already has a worse form of this dilemma for existing TypeScript users.

BT: You mean rationalizing private vs #? The biggest problem I have with clause16 extension is we have to explain to TypeScript users the change of the mental model. Extremely widely-used existing syntax doesn't follow this new mental model.

MM: I understand the nature of the dilemma: genuine pain point. But our responsibility in committee has to weigh the differential experience of a much larger number of future JavaScript programmers higher. If it's a big pain point for current and small gain for future: that's a hard judgment to make. More data would help.

MM: I would like to ask if there are any objections in moving to stage 1?

YK: My biggest issue with making this stage 1: we have active in-flight work. This is a potential unifying direction, but I'm worried about putting hard breaks on other proposals.

MM: I accept that we did not achieve consensus on stage 1.

YK: I do have a strong desire to see if we can make this fit with what you had in mind.

MM: We agreed that some discussions need to happen whether or not we achieved consensus on this.

Conclusion/Resolution

- Did not achieve consensus for stage 1.
- Mark is going to reach out to the champions of the affected proposals for follow-up.

8.iv. Approve TR-104 draft and forward it to Ecma GA

AK: Can you give a two-sentence summary of this?

AWB: TR-104 directs people to say "yes, test262 consists of all of these things on GitHub. Go look at GitHub."

LBR: Basically a description of test262 with formalities and ready to bring to Ecma.

AWB: Blesses test262 as official Ecma work product. Non-normative.

LBR: There are a few facts. As of September 2016, test262 consisted of over 23k test files...

AK: I skimmed it. I will give it affirmative: yes. I find it hard to imagine someone objecting.

AWB: I do, too, but I want to be fair. This is one of those things that Ecma as an organization: they think it's good for them.

Conclusion/Resolution

- Consensus reached to approve and forward to Ecma GA

12. i. Needs-Consensus PRs

#856 Avoid Duplicate TA#fill value coercions

BT: Issues with `.fill()` on typed arrays. Specifically the issue involves how to do coercion on the values passed into the `.fill()` argument. The fix involves calling `toNumber()` up front.
AWB: Any other Array prototype methods not reimplemented have similar problems?

BT: I looked briefly, did not see, but didn't dig deeply.

DE: I looked. I did not find any other functions that would benefit from similar treatment.

BT: You could argue that we don't need to re-implement the algorithm.

AWB: Could this fix be applied at the Array.prototype level?

BT: No, because Array.prototype.fill doesn't do the `toNumber` coercion.
DE: It got re-factored after ... Shared Array buffers.

AWB: ...

BT: It's only slightly different (than arrays) but it is different.

AWB: No, but I do think you should duplicate the whole algorithm.

BT: It is duplicated.

Conclusion/Resolution

- consensus to accept this PR

#854 Remove Implementation-defined behavior for Number methods

`Number.toFixed()`/`toPrecision`/`toExponential`
AWB: When I looked at it, it looked like it was a superfluous paragraph with those three methods. When I looked at the spec these all just looked superfluous.

BT: But this...I agree with that notion, this PR amounts to deleting those extra paragraphs that you mentioned.

AWB: It felt more like old boilerplate code, like maybe it was just some common practice to add this language.

WH: Where is this language?

BT: Look at `Number.prototype.toFixed()`.
BT: Look at the actual PR.

WH: Ah, it's after the algorithm. It's fine to delete it.

BT: Seems fine? Good. Getting rid of it.

DE: Also, `toFixed` et al support greater precision from when Spidermonkey added it 18 years ago. Should we force all implementations to support greater precision?
WH: The reason we didn't mandate the greater precision and instead left it as an option back in Edition 3 was that we didn't want to require every browser to do this at the time. But if you can support greater precision like Firefox does, you should.

DE: Well we mandate ... and things that are much harder to calculate than this.

BT: I think we should consider that as part of a separate PR, Dan, if you want to whip something up along those lines?

WH: I would not want to remove the extra precision.

DE: Maybe -20 to 100 [precision]

YK: I had an issue tht was similar to this last week. I had a language binding and I had to decide about what a float in langauge A would be in language B.

AWB: I think we should take this input as input to a future proposal or PR.

Conclusion/Resolution

- consensus to accept this PR

#785 Array.prototoype.sort checks for compareFn being valid

Presently spec implies but does not say that the compareFn must be `undefined` or callable. PR enforces that expectation.
BT: `sort` takes a compareFn. Spec does not currently say that compareFn must be callable or undefined. Implementations disagree on handling non-callable/non-undefined. Update to throw a TypeError.
AWB: Still need the language about proper behavior of a sort compareFn.

BT: ...We should really be throwing TypeError early for things that are not callable. Table [in PR] shows various testcases and existing behavior.

JSC: Could you take a look and see if you're worried about changing behavior. Most of these align with existing behavior in Chakra/Edge.

AK: There's a minor possibility of breaking something here.

AWB: It is something that non-interoperable.

AK: I did comment on the thread. I find it pretty low value here.

YK: I think some number of users will make mistakes with this.

AK: My intuition is probably that it's web compatible to change this.

MPT: This seems like a great up-for-grabs.

LBR: No tests for this. This is the most frustrating function I have for test262, everything is implementation-defined. I'm really positive to anything that allows me to write a test on this.

MS: The first row there [in table]: shouldn't that also throw an error...OK, so it's callable but you're not going to call it because only one element.

BT: That change was fairly recent too, not calling the compareFn if only one element.

BT: Any objections to taking this pull request?

Conclusion/Resolution

- consensus to accept this PR

#833 Remove duplicate keys from proxies

BT: All of the possibilities of how....

MM: Why did we change what we did originally to end up in this current place? Do you know whether...

AWB: I'm pretty sure from the issue thread that (tom?) is aware of it. I'm largely responsible for not doing this check before (something) because I was opposed to (computationally large checks).

BT: Yeah, and to be honest it isn't that bad and it's super confusing and anyone implementing it is asking why, why why. In the spec it's not too bad, but in engines it's a lot more (obviously unecessary)

MM: I'm completely satisfied.

WH: Is it possible that the proxy checks involve more proxies?

AWB: Well yeah, sure, if they access an object using any of the meta operations and if it does it will invoke the handler.

MM: This is not even an edge case, a major part of proxy design...it's by the double lifting trick that you can implement the policies that are .... of the track in one place.

BT: Any objections to taking this PR?

LBR: If we accept this we will have breaking tests.

BT: We have a needs-tests label. Feel free to apply it.

BT: I updated permissions to ecma262 so that delegates have ability to add labels to ecma262 issues and to close issues and pull requests and to push branches, which I think will be helpful in the next year.

AK: Need to be clear about what kind of comparison is used when things are added.

WH: This is a one-line change?

BT: Roughly I think so yeah, you could expand that one line into a loop with explicit? checks.

AK: Every other use of the word "duplicate" in the spec is...not these reified objects...

BT: But there's only one way to compare a string right?

AWB: ... symbols?

AK: As the editor, keep an eye on that. I fear that someone will use "duplicate" somewhere and not know what it means.

Conclusion/Resolution

- consensus to accept this PR

#852 Move side-effecty toIndex in TA constructor to before detached check

BT: Current spec won't throw when `toIndex` causes buffer to be detached.

1. Add another isDetached array buffer check.
2. I think moving the toIndex further up is the correct way here.

AWB: Does moving it up change the order in which arguments are examined?

BT: It does not change the observable order of side effects. It should be fine to do and not cause compat problems.

Conclusion/Resolution

- consensus to accept this PR

#853 String.prototype.replace edge case alignment

BT: Currently spec says substitution strings ($1, $2) with indexes beyond the matches groups (i.e. $5, $6, $7) is implementation-defined.

BT: All implementations agree that the replacement is not done for such substitution strings. PR aligns with implementations.

AWB: In my work on this in ES5 and ES6, for things like this I tried to propagate what had been specified in...?

WH: When we wrote this in Edition 3, the plan was to only allow these things when there was actually a capture of that number. Unfortunately on some implementations, if there were only 3 captures and you did `$4`, you'd get a literal `$4` in the result instead of a replacement. We wanted to deprecate that.
BT: So you wanted empty string?

WH: No, we wanted an error.

AWB: So is there any chance that throwing an error for this would be web compatible. The hope to deprecate this has failed.

BT: That is excellent context and it makes me a little sad about removing this.

YK: I think there is a useful discussion we should have about deprecation.

BT: I will add notes with history that Waldemar provided.

MPT: Did you want to do the needs consensus on ??

BT: I'm open to continuing.

Conclusion/Resolution

- consensus to accept this PR

March 23, 2017 Meeting Notes

Allen Wirfs-Brock (AWB), Waldemar Horwat (WH), Brian Terlson (BT), Michael Ficarra (MF), Adam Klein (AK), Dave Herman (DH), Kent C. Dodds (KCD), Tim Disney (TD), Daniel Ehrenberg (DE), Shu-yu Guo (SYG), Michael Saboff (MS), Sebastian Markbage (SM), Bradley Farias (BF), Maggie Pint (MPT), Jamund Ferguson (JF), Myles Borins (MBS), Logan Smyth (LS), Sarah D'Onofrio (SD), Alan Schmitt (AS), Dean Tribble (DT), Peter Jensen (PJ), Mark S. Miller (MM), Leo Balter (LBR), Zibi Braniecki (ZB), Rafael Xavier (RX), Yehuda Katz (YK), Caridy Patino (CP), Diego Ferreiro Val (DFV), Brendan Eich (BE), Lyza Gardner (LG), Istvan Sebestyen (IS), Matt Johnson (MJ)

General Discussion

AWB: RSVP now for the NodeJS CTC meeting on the evening before the May TC39 meeting. Before the TC39 meeting, there will be a meetup with the NodeJS CTC.

10.ii.a Dynamic Module Reform for stage 2

(Caridy Patiño)

- https://github.com/tc39/proposal-dynamic-modules
- https://docs.google.com/presentation/d/1EYOysPhgjXtgmuNoZ_wUCMElZ8GKLxJmCLeF0EvUXkc/edit#slide=id.g1f19d52d1f_0_0

YK: Even if node doesn't want this feature.

MF: In a previous slide you said anything you can do in node modules you should be able to do in ECMAScript modules. Can you clarify what you meant by that?

YK: I didn't hear him say those words.

MF: To my knowledge that is exactly what was on his slide.

CP: There's an example in the README where you can see things you can do today in CommonJS modules; you'll be able to do the same in ESM but you will not be able to do the same when you combine them.

DH: I just wanted to clarify that you meant that in an absolute sense, because there are things you are prevented from doing in ES Modules like mutating the export bindings.

MF: And the module namespace object cannot be callable in ESM.

DH: Okay, this could turn into a philosophical discussion, because umm default export looks like a thing that lets you do whatever you want to do, but I don't think we need to litigate this forever.

MM: I think we can just back off from what sounds like an absolute statement.

CP: What I meant to say is you can do it in ESModules today and transpile that in CommonJS, you should be able to do that same thing. I can rephrase that. There's more detail in the README to describe what I was trying to say. The second piece of feedback was from Allen. The fact that there might be side effects for these bindings... ?? Accessing an importing binding cannot have side effects.

DH: It's not just from an implementor's standpoint. It's from a programmer's standpoint that being about to be sure that accessing a variable doesn't cause random things is good.

CP: So the actual solution that we're proposing (this is the second iteration), Dave came up with the idea that whenever you're trying to run the instantiation processes of modules, you could inspect what modules you're trying to ?? If any of these are dynamic modules that haven't been evaluated yet, you should be able to store information and enforce this connection between the modules. So in slide 12 you'll see that we're doing the regular instantiation order... {references slides}. We'll continue doing the same for ESModules, but if any module is just a dynamic reference, we don't want to evaluate it up front. What we're doing is just adding these bits to a new slot called [[PendingImportEntries]] where we can store information about this binding and where it comes from. Eventually during module evaluation, right before `main.js` we're going to walk the list of [[PendingImportEntries]] and create the bindings for them. At that point `x` and `y` are already evaluated and we're all ready to evaluate `main`.
AWB: In this example you are only using default exports. Can it have named exports?

CP: Yes, it's the same thing.

BF: The current plan is to only support default exports. We're working on other things... Much like this, you ??

AWB: It's clear that it wouldn't be very hard to make default work, but it wasn't clear that you could make an open-ended set of names work.

BF: Correct.

CP: It's the same, we do have the details of the binding. We have the local name for the binding, the import name for the one we're importing from. One more point is that this solution is not just for CommonJS modules, but for any other dynamic modules (like wasm)

MM: What are the other possible values for the internal slot named ImportName?

CP: Let's switch to the spec text. It will clarify. Step 12.d.3 If you go through the module instantiation and creation, we get these ImportEntrys. We walk all the table entries from parsing the source text. If the entry is pending, we put it there. This is the same entry when you parse.

AK: To answer your question, the name default comes from the syntax. If you say `import a from './wherever'` that's where the `default` comes from.
MM: What are the other possible values for that slot?

DH: If you use named exports it would be whatever you asked for.

MM: And the reason why `default` is not mistaken as a variable name is because it's a keyword?
YK: I just want to say, in general, I don't think we need to worry that much about what Node does here. We need to worry that the semantics are flexible enough to give Node options, and I believe these are flexible enough for other options as well. But we shouldn't be concerned about that.

AK: Given the motivation stated at the beginning is based on node, it's interesting that you say that. Until this morning, I had not heard this was meant to support dynamic modules cases other than node.

YK: I think AMD modules have always been a plausible story. I've been talking with Dave about whether this is actually true that there are other use cases and that's when we realized that the future transpiliation use case exists.

CP: To add to that. If you go to the rationale paragraph, one example is CommonJS, but this is about general dynamic modules, not specifically CommonJS.

AWB: I have a question. You're describing this in terms of module declaration instantiations. I'm interested in how this generalizes to what the actual JS programmer sees. So if you have something you've imported with either a default binding or even a named binding, but the module is in its pending state...??

CP: We do have an example in the slides..

AWB: I'm wondering when they're resolved and if it's possible to try to access in the JS code while it's still pending.

DH: I believe that the programming model is that the validation will occur before execution starts.

AWB: That means that the evaluation of the dynamic module (its body or whatever it needs)... In CJS, you need to evaluate the entire module... That will happen before evaluating any code of the importing module.

CP: We can go to module evaluation in the spec and you will see in step 7 right after we evaluate all the dependencies. then we are ready to evaluate the ES module that is importing the dynamic module. We eval the pending bindings, otherwise we throw an error. So before the source test but after the dependencies.

AWB: Ok, you described this as Create Pending Binding, you resolve all these bindings and their values are available.

CP: Yes.

AWB: Ok, any evaluation of import modules still takes place prior to any evaluation of the importing module?

CP: Yes.

DH: Assuming you have cycles. You have to pick one to go first. So you can witness uninitialized variables, but that's the same as in any system.

AWB: Ok.

DH: Do you want to say more Caridy?

CP: I think the last one is just an example. But, that's it.

AK: The last time we talked about this, there was a discussion of circularity and how you can make guarantees. You discussed this but I don't know what changed.

CP: Yeah, we were doing more work on the module evaluation? phase... We settled on the idea of simply making sure all these imports can be created during evaluation and if they're not ready we throw an error. In the previous proposal we were still evaluating the importer source text and then running an extra step to make sure everything was ready. The conclusion is in node today you cannot do those things. You cannot have circular dependencies in this way because the export update hasn't been created yet. We were trying to solve a problem that wasn't really a problem.

AK: Going back to the previous discussion, you basically cause the module to export all the things that anyone tried to get from it, and then if that fails you get an error.

MM: What is "tried to cause the module to export anything anyone tried to get from it"?

AK: For ES modules they export the things they declare as exporting. For dynamic modules their set of exports is defined by their importers. I could say `import atom from underscore` and atom will now be exported from underscore.
BF: I don't think that's entirely true if we go back to module declaration and instantiation, I think it's statically checked.

CP: You can resolve to pending because you haven't evaluated the dynamic module.

BF: So if the loader does not generate a pending, does that mean it's an error?

CP: If you know the shape of it without evaluating it, you don't need to return pending.

AK: If resolution is ambiguous or pending... throw a syntax error? I'm confused...In section 9b.

CP: {references 9b}. It's never declared or created, because of that we throw an error.

AK: Okay I need to read this new spec then, more closely I guess.

YK: I think it would help people to do that, maybe we can pause this.

AK: It's possible it's a waste of most people's time because it is very detailed.

AWB: I have what I think is a higher level question. A lot of this is about initialization order, so I want to verify my understanding. If I have a module and it has 3 imports: a, b, c using default imports. And let's assume their bodies is just a console.log and that a, b, c each don't depend on anything else. The expectation would be that when you load the root module, you would see on your console: a, b, then c and anything the importing module did. Now if b is changed to be dynamic, am I correct that we'd see: a, c, b?

DH: No, the whole point of this is to preserve the original behavior.

AWB: Ok, then my understanding of what the pending modules works..

DH: Caridy is the one deep in the spec. Pending is about these are the names that have been requested by some of the importing modules but we can't yet validate them because we haven't executed them yet.

CP: It's not evaluation, it's creation of the import binding. But yes, you are correct.

DH: And won't have created those until after we have finished evaluation of all the dependencies before evaluating the importing module.

CP: After evaluation of all the dependencies but before the execution of the module. The whole proposal is saying that during the instantiation we need to create the import bindings, but we cannot create some of them yet....

WH: What does the default indicate?

DH: That's about the whole default import syntax: `import a from 'a'`, that `a` is effectively sugar for: `import { default as a } from 'a'`.
WH: So, is it possible not to export a default?

DH: Yes it's possible not to export a default. That's in ECMAScript syntax. In a dynamic module, you can choose to name one of the imports "default".

AWB: I think I'm comfortable with what's happening here, whether the details of the spec are (perfect?) or not.

DH: I think Caridy did a good job talking about the goals. We can make sure the spec is flexible enough to preserve the order of the modules loaded whether they're ESModules or Dynamic modules. You don't want validation to happen at arbitrary times. We want to get, early in the programming model that we can evaluate modules in a consistent order regardless of what kind of module they are...

AWB: So if .. I think the only observable difference here is that if a dynamic module is involved, there are some situations where an unresolved import binding error, where in the current spec that would always be produced bfore any evaluations, there are now situations where some of those errors may occur after some eval has started, but only in the presence of dynamic modules.

CP: Correct.

DH: A way to make that more concrete. A dyn module, since it could have executed before a static module, could dynamically ask for the module and so observe whether it's been initialized, but there are no invariants violated in that scenario.

AK: When you say executing the modules in the body of the source function ??? This adds a strangeness to the lexical environment of modules. Meaning they have their lexical environment changed after it's created.

DH: In other words, is there a soundness hole here? Where a cycle between a source text module and a dynamic module... where the dyn module can see the source text module before it has validated.

AK: Yeah, at the moment that just creates a reference error, but the way the spec is written is strange. The static module doesn't create a lexical binding. From an implementation perspective, it doesn't ??? This is interesting...

AWB: Ideally this all simply reduces to TDZ-like behavior. Certainly the root module that's exporting some things... Before it goes off and evaluates anything, if it's a source-text module it doesn't know the binding values it knows all the names...

AK: But that's not how the spec is written. It says we don't create the lexical binding until after that function has run.

WH: Do we have any idea as to which lexical binding when it gets created.

AK: It could be global.

```
import {a} from 'a'
export function f() {
    console.log(a)
}
```

DH: Oh, the fact that the way the spec is written means it doesn't add it to the environment. That's just a bug in the spec.

MM: Ok, good.

AWB: Yeah.

DH: It should be changed to add the lexical binding. Our goal was to not have it be possible for import binding to ever trigger side effects, including dynamic reference errors whenever they are evaluated. Adam has found a case where before you have started eval of top level code, it's still possible to enter into a function, resolve a binding and get a dynamic error. We got close and "I'll take close".

YK: I think there is a different way that I would...(cut off)

CP: One thing that I want to clarify is that this is only possible if you have 3 modules, where 2 are ESModules and the 3rd is a dynamic module and there is circularity with the dynamic module. The problem doesn't matter because it doesn't exist.

MM: Now that Adam has identified the bug clearly, I don't understand why you are willing to live with it. Now that it's been identified, is there anything especially hard about closing the bug?

DH: Oh, its characterization is straightforward, but the solution is not obvious.

MM: Well if the solution is not obvious then I don't understand the problem.

DH: Well, what's the obvious solution?

MM: You want all the scopes and bindings to be populated (not initialized but populated) before anything can be observed.

DH: Of course, oh, there's a fix for the not setting up the lexical environment correctly. You just create it up front, you can never observe that that changes. However it's now pointing to an export that we can't yet validate if it exists. We can't yet validate if the thing we're importing is a real thing that's being the exported.

WH: So would you do a TDZ?

AWB: It has to be a TDZ.

DH: This is effectively a TDZ.

YK: Yes, it needs to be a TDZ.

MM: If the failure is a TDZ then I'm happy, but I thought the bug was other than a TDZ.

AK: I'm happy to explain why this is....it's not trivial to fix.

YK: So I think that Dave's characterization of the goal (no side-effects, but reference errors). Referring to a variable might create a reference error, there's an invariant there. It would be fatal if we were not able to avoid it with other assignments. I had never lumped those assignments together.

AK: My complaint wasn't this did an error, but that it did something very strange where it tried to access outside of the lexical environment.

YK: Your point last meeting was it had side effects.

DH: You pointed out 2 issues and we are conflating them.

MM: State the one that is egregious and easy to fix.

DH: The one that's easily fixed is we don't even add the binding to the lexical environment first, and we can witness that, then we add it later. We add the same variable reference before and after we can be looking up an entirely different environment. That's a bug and we can fix it.

CP: So I....

DH: We thought it was the case that there wasn't ever a case that you could .... ?

MM: So when you access...

DH: We may be able to fix that, we just found out about it, but we can investigate it. This only happens when you early call a bunch of functions several times. You have no hope of that doing reasonable things anyway.

WH: What would be subjected to the TDZ? Would the call of the function fail because you would get a TDZ error on invoking it, or would the function run and get TDZ errors inside it when it accesses variables?

CP: I created an example on slide #15. It shows the problem that Adam referenced.

AK: I want to explain why the previous version of the proposal tried to do the other way of doing this. Some new machinery must be invented. The other module doesn't have an environment record, so you can't set it up correctly because there is no exporting module thing yet. Where does that get created? The node loader default only solution solves all this. If that's the only dynamic loader you must deal with, then it could never fail because you know what exports you have to deal with.

YK: That's only what they currently think they *might* want to do.

DH: We heard Bradley say they're investigating ways to see how you could state what you're export set is. So let's say, from a user model perspective, there's something that node recognizes that allows you to specify upfront what your fixed shape is, what that could be built on top of is a reflective layer (nothing exists for this now), but you're opting into that where you state I'm playing by new rules that say I have a fixed state that I cannot change dynamically. If you do this, then we don't need to execute to discover what that shape is. In other words, either you have that thing and you don't need to execute, in which case you have to be a default export and again we know the shape of that as well...

YK: I would be happy to explore that.

BF: We have prototyped out exactly that. You can model it with two ES modules right now.

DH: Sure, perfect. And we could make that API nicely. The going assumption we had is that the only way to discover the shape was to execute it. I'm suggesting that instead of that, we could say you have to declare before you execute what the static shape is.

WH: By static shape you mean "names and exports"?

DH: Yes, that's the thing you declare once and for all and never changes. You can state names before your execution.

WH: And that would be constant, it's not computed?

DH: It would be computed dynamically, but it is staged, in the sense that once you have told the system what it is, that's frozen and immutable and will never change.

WH: What does that computation have access to? Does it have access to other modules or is it purely local?

BF: Let me explain what we have right now. Basically when you get a list of exports for some sort of dynamic record, we're generating a static list of specifiers (like we saw earlier), and that needs to be available at or before module instantiation, by that time it is static and frozen. That's when observable effects come into play.

CP: Does that mean that you evaluate the module?

BF: We have a couple of different shapes of modules but we know all their shapes prior to evaluating them currently.

AK: When you say node, this is in some thing outside the language, you aren't evaluating expression to figure it out right?

BF: No, there are a few different approaches (Dave mentioned one: an out of band file), or some pragma at the top.

DH: I want to continue answering WH's question. If you are making it dynamic, that is exactly the right question to poke at when looking for soundness or initialization order problems. I believe that a staged system makes it possible to avoid that problem, but we need to investigate this further.

WH: I'm fine with poking at that problem later.

DH: It's the right kind of question, but now's not the time to answer it. But we should allow you to declare upfront what the shape is in a way where it doesn't have access to the modules before they've been evaluated. This means it now becomes easy to have a straightforward binding setup and evaluation setup...

WH: What does the shape consist of? Just names? Or does it matter what kind of thing? e.g., getter/setter, proxy, etc.

AWB: They're just names.

DH: From the outside it doesn't change with those things. For pre-existing systems, they'll just have the static shape of one export called `default` which is a dynamic binding that can be anything. So unless the author opts-in, they'll get the single anonymous default export. That's pretty close to what Node has planned already. We're just hypothetically creating a flexible system they could build on top of.
AK: This is an interesting area of discussion. I'm starting to wonder though if these dyn modules are going to be modeled as providing a static set of exports. Does that reduce the amount of spec change needed to support them?

DH: Yes, this is what I'm saying is that we've identified a much simpler system here.

AK: Is any spec change required?

DH: Possibly not. There's almost nothing in the spec to support dynamic modules. The combination of the work on Caridy's part combined with the feedback from Adam, we've come up with a much simpler answer. And I'd like to take the learning and apply it to the next rounds in May. I don't want to discuss whether or not this is stage 2 because I know what the next steps are and they fold into what the API for dynamic module creation is. Sorry Caridy, I just said that without asking your opinion... What do you think?

CP: It is true that if we go that route it will be simpler. No changes in the spec are needed.

DH: I'm excited, I think this is a good place to wrap things up unless anyone has feedback or questions.

AWB: I'm going to pile on. I think that the fundamental thing here is that if you say `import foo.js` is there a way to determine the exported names from that module without having to evaluate code?
DH: The quickest way to explain is that Node discovered a key thing. We can make the default mode for the whole ecosystem to be `default` export and we can opt-into named export. That gives us strong backward compatibility. Once you have that it's easier to see your way to semantics that makes sense. Is that right Bradley?
BF: Yes.

YK: One other closing the loop. There's also AMD modules. They don't trivially have a solution. I think that having them default to default export works fine. Because they are eval'd with an outer wrapper and an inner function, you can add to the outer wrapper pretty easily.

DH: Thanks for bringing that up because it's easy to forget about other dynamic modules systems.

YK: Yeah, we use AMD so... :)

AWB: At a pure reflection level, you can imagine that the reflection API that when you instantiate a dynamic module, you are required to specify what the export names are.

DH: Ok, I think this is a good place to end this discussion.

WH: I'd love for someone to go through the example on the screen to determine the sequence of events.

DH: It doesn't matter anymore because the new system means this will never happen.

WH: Why is that?

DH: Because we know the shape before any evaluation takes place.

WH: Does `y.js` have a default export or not?
CP: Yes.

WH: If it has a default export, then what's the problem? You're importing its default binding so already know its shape. Is it because it's not the right value?

CP: The `import x from 'x.js'` happens ....
DH: Can we discuss this offline? It's not important because we're not going to do this anyway. Let's take a break.

[Per discussion during the break it turned out that the example on the screen had other bugs.]

Conclusion/Resolution

- Remains on stage 1

Awesome work, now simplified as a result of awesome feedback. A new version will be presented at next meeting.

10.ii.c Realms update

(Dave Herman)

- slides

DH: This is a recap of the work I've done on the realms API.

DH: Working with Caridy, MarkM, Dean, and others.

DH: Realms are similar to an iframe on the web. Effectively you can have 2 different copies of the JS world that can talk to each other. Historically this didn't exist in the spec, but it was a reality.

MM: Up to ES5, there was this fiction that all JS computation and execution happened in one realm, but with iframes this wasn't true.

DH: On top of that, node has the VM object with similar functionality. That's now (ES6) codified in the spec, but there isn't a formal way to interact with them. You can create a new iframe, but it comes with all this DOM crap. The goal is the Realm

API is to just create multiple realms with nothing else by default. This is useful for building frameworks and structures on top of it. It's a pretty low-level (and complicated) API. This is something you do when you're implementing a transpiler or security framework. So it's ok if the surface API is a little advanced. Goals for today are to present progress and get feedback. I'm not looking for stage 2 at this point. I presented my current thinking in January and mentioned I believe we can get to a place where we don't need a registry API or loader API. We had thought we needed those. I think there are a couple of tiny hooks we want in the browser space, but most of the functionality can be done in user-land. I have some sample code. Please don't nerd-snipe me too hard. I have a plausibility argument and it's looking more and more plausible to me. The first example is the smallest I could come up with.

DE: Example: a tiny, fixed registry.

DE: They have one big switch at the top of the code that said. If the module is jquery, give them a simple version of jquery, if it was underscore, etc. And that was their entire registry. So maybe you can build a tiny fixed registry this way (using realms).

CP: Another point around this is today people use Rollup which is effectively a ???? for client-side.

DH: By calling super I'm subclassing the builtin realm API. It is just a demo here.

WH: You are missing a declaration of #registry.

DH: OK I just got the details of the private field wrong. Imagine there is a declaration. At any given time a "registry" might want you to load new modules. So I'm taking jquery as an example of that. So you have a jquery loaded in one realm, and you want to explicitly share it to another realm. You may want to allow modules to be realm-specific.

KCD: When I worked at Domo people could build little graphs on their own. And a big problem we had was that every single one of these were loading their own angular. This would be a perfect solution for that.

DH: Perfect, yes, I use jquery as an example, but there may be other big things. In order to have total reflective control over the whole realm, you need total reflective control over the import. I'm not confident I have everything nailed down, but the basic idea is that you can specify a hook and by doing that you can tell the system how to resolve a name and how to provide it with a module. So here, I have a fixed set of modules in my registry, if it's not one of those, then throw a reference error, otherwise....

WH: What does `this.#registry.graph().then(graph => graph[name])` do? What's in `this.#registry` hasn't been explained yet.
DH: {explains slide}. This is asynchronous, allowing you to use `import()` as well as top level import.
WH: Who gets to call the [Realm.import] method? Only the system, or can adversarial user code call it and cause trouble?

DH: One way to think about the entire realm API is that it is a privileged API. There's never direct access to that API. There are documented ways in which e.g., the import semantics triggers these hooks, but there's no direct access given to the realm object to code within the realm. The creator of the realm has access to it, but they control whether it's made available to anyone else.

DH: So top level usage of this, we create a realm, we use eval, use `import()` and that produces a promise and that gives us the module.
KCD: So what you normally get, is that going to be whatever is exported from that module?

BF: The completion value?

DH: The result of `eval()` is going to be the completion value of eval. I'm forgetting the details of the import–parens syntax.
BF: It needs to be in an expression position.

DH: So I'd have to put parens around the eval here (in the example)?

KCD: I basically want to know what's returned from a call to `eval`, is it a completion value or is there some mechanism to export something specific?
DH: I'll get to how to dynamically create modules in a realm, which is a key feature, but that's separate ... There's a semantic surface we need to be able to cover in an API. But basically, the code evaluated in `eval` is a script, not a module.
AWB: So you've chosen to expose an eval() method that evaluates ???. You could also choose to expose an `import` method.
DH: I think it's important to expose the minimal API you need first and not mix them in with conveniences. What's the minimum set of stuff that I need? This is an EWM kind of API. It's a low–level API to begin with and people can build abstractions on top of it.

AWB: If you look at it from the point of view of modeling what the language lets you do syntactically, we have two types of top level.

DH: Agreed, but allow me to proceed because there's more.

MM: Of the Script start symbol, those strings that satisfy that grammar can be evaluated as scripts or as eval-code.

DH: Oh we talked about that before, yes.

CP: We have github issues for those.

DH: [continuing demo] This is the registry class. Imagine I have a @memoize decorator... This is not the module namespace, it's a reflective API that we need to have to do all the things you could reflect on for modules. So I'm positing a parseModule operation. It difffers from eval. I want to take some source code and get a module record that has been parsed by an engine so that, e.g., I can ask what names it depends on. I need the ability to go meta on code to setup the environment for running it. So this helper method is just basically to fetch all things in my graph, for this tiny demo I can just fetch them all upfront.

DH: Continuing on.....by the way Caridy did you find the PDF for this?

CP: Yes.

DH: The `graph` method is the one we actually saw the first class using. How do we do the linking? We have to go and fetch them asynchronously. The result of this is going to be a promise. For the module record A it basically has this not-yet-satisfied set of dependencies. Where it asked for the name `./b.js` I'm telling `b` is the thing that satisfies the dependency. For `b` I'm doing it imperatively. This is also at the level that's teetering toward bike shedding, so I don't want to spend that much...
DT: What does `link()` do?
DH: Link is the thing that says .... now what we're doing here is wiring together a graph. As a recovering functional programmer. One of the things I discovered is that functional programming is especially (good|bad)? at graphs. Imperative programming can be very helpful here.

AWB: So link can produce an error?

DH: Yes, that's right. Now I'm not convinced that I've nailed a connection here. Sorry about that. I think it's getting close to a way to tie this all into the semantic operations of the module link and evaluation process.

AK: Feel free to defer this. It seems like you're presenting a bunch of examples and maybe you'll cover the concepts later? Is it a non-goal to allow this to apply to existing realms?

DH: For the time being, it's a non-goal. I think it's a much thornier space because you have all kinds of questions about what the existing environment is depending on...

MM: I think a way to rationalize, to imagine how one would extend this to hook into those existing host frameworks (iframe, etc) is to imagine that this had come first and that the host frameworks were built on top of that.

DH: Should people have the power to modify the environment they're running in? That said, there are critical pieces to configuring my environment that will matter a lot. A lot of the loader API in the space of what you're talking about (configuring the environment in which my core API is running) is satisfied by Service Worker which can do this. Like how do we lookup module names when they're imported. I think it doesn't satisfy all of the use cases. For one, the first-run semantics (the things you can't do when you first run your app). Also what you can't...

YK: You mean illegal URL?

BF: The specifiers are different.

YK: ServiceWorker intercepts network requests, and it's unclear ??

DH: It has restrictions which should be a URL and you ...??

AK: That's a syntax error.

DH: That's something you want to do and you can't right now. That can be solved with some fixes to Service Worker, and ??

AK: So you're not suggesting this is the way to solve this.

DH: That seems like an obvious reaction to have, sorry. I'm considering these separate.

BF: Your comment on being a recovering functional programmer. We have this already implemented and we're moving to a functional approach for reasons we can discuss.

DH: I'd be happy to discuss that.

MM: To elaborate on my response to Adam's question. I think there are two very clear examples. Really one example... That motivates a lot of the separation and the rationalized re-layering. One of the things you'd like to be able to do with this is in user-land create a library that would run on node, creates a multi-framed browser environment. Likewise you'd like to run a multi-vm environment in the browser. This is possible with Realm because of the required isolation.

YK: There's a thing Dave just said is that a big part of what happened here is the original loader API had a lot to do with networky considerations. Service Worker took care of that. So now if we tried to do that it would have composability issues with Service Worker, so now we can defer to that instead of standardize it in the JavaScript layer.

DH: Yeah, in some sense, this is about getting the layering right. The original attempts were eating into the network layer. This really is about hooking into the core syntactic elements of JavaScript and then allowing the hooking code to hook into those other layers as appropriate.

WH: Trivial question... Where did `this.#jquery` come from?
DH: From the constructor. That came from when the registry was instantiated. But the point is that's coming from the outer realm and sharing that explicitly to the child realm. You never get implicit sharing. Ok, concepts: parseModule, import hook, a way to satisfy modules independently, and a way to link those modules.

BF: Is there a reason we used a computed property instead of just a name?

DH: (new slide) Example: a dynamic loader. Here's another TinyLittleRealm class. Uses a TinlyLittleLoader class which is sharing jQuery from the outer realm to the inner realm. Referrer is used to deal with de-relativization. The loader is going to produce a custom type relative to this program. The final stage of loading this module is it's fully linked and here's the finally linked module. So `linked` is a promise of the fully linked module. `module.ensureEvaluated()` ensures that if it hasn't been evaluated so far it will be evaluated now.
WH: Is `[Realm.import]` always async?
DH: Yes.

BF: We can have this discussion, but with async functions...?? I don't want to talk about this now. It looks fine to me :)

DH: Same top-level usage.

BF: I do have a question on `ensureEvaluated` here, in the previous slide I thought `realm.import` returned an unevaluated dependency and this one looks like it ensures it's evaluated. If it's coming from something that's not evaluated, you're hoisting evaluation.
DH: It's a bug in the first demo. You should just call `ensureEvaluated`.
MM: Can module be a regular object?

DH: No I would expect that to be some kind of brand check. I don't want to add a bunch of conversation and other conveniences that confuse the underlying model. {explains slides}. The register helper method will do the work of fetching this thing. Effectively we have a userland implementation of the core semantics of default module loading in ECMAScript. We'll fetch the specify, parse the module, then get all the requested names (specifiers), then do a recursive load on all of those specifiers, once that's completed, then we'll add them to the dependency graph and then call link on the module and set the three promise steps as the entry in the registry so you can track the progress of the whole thing.

WH: Interesting that you didn't write these using `async` functions.
DH: Went back and forth on that.

MM: Is this a case where...

DH: Part of the value of being able to implement loaders in user-land is there's so much value in doing this. There's sort of no end to the different ways you want to do it. There are tons of different things that you might want to do. What's exciting to me about what we're doing is how simple it is. Effectively hooking into `import` is the whole story.
DT: If that was a for loop with async/await instead of `Promise.all`, it would serialize it. So this is a good example of where the lower level tools are very useful. This example may also be a good foil for working out fancier support for async/await without full serialization.
MM: Can you point out explicitly where you're gaining information from the API that you would be otherwise required to parse yourself without it?

DH: `requestedNames` tell me all the import strings that appeared in the source code.
WH: So the registry defined on the last line of the example combines `name` with three promises.
DH: Right, and that's something specific about this demo. So I'm trying to demonstrate that userland registries and loaders are going to be possible. We have those hooks. Caridy's been working on a polyfill.

CP: Yes.

DH: That work continues. Other than that I'm just looking for more feedback.

AWB: At least in the spec, names and bindings might be mixed up. Let's be more specific. (basically)

DH: Name is a pretty overloaded term...

BF: I think there's still a little more API work to be done here (I guess we're at stage 1 so that's expected). I don't think you should expose as much of the low level hooks as you have here.

DH: Oh, interesting.

BF: We should always make sure that ??? is a promise.

CP: We do have an issue in the loader spec, so we will not utilize the name here.

YK: If your opinion is it's fine that the ??? module returns a promise. If you want to access the promise, we disallow you to do that.

MM: Yep.

MM: I just want to mention that people don't read too much into the specificity here.... We are discussing some radical refactorings to this, so this may be very different in the future.

AWB: For your jquery example, you're essentially showing sharing jquery across multiple realms you might want to create here. The footgun aspect of that is it would seem on the surface is that that instance of jquery you're sharing itself is going to be compiled in a different realm than the realm that you're sharing it in.

CP: I don't think that will be a problem. I think you are abstracting the normal jquery object. I don't think it will really matter much, you can evaluate jquery in the new realm, extract the global jquery object and use it as in the example. This jquery will somehow have to become a dynamic module.

DH: I want to call that by design. (Scott Isaac told me JS causing general protection faults was by design.) Here's what realms are. By default everything is separated, for security, etc. B) so you know every time you're incurring sharing. If you choose to share a library that you know was designed to only be used in 1 realm, you face the consequences. OR maybe you design a library that works across realms. It gives you the control as a module writer to do whatever you want.

MM: The frozen realm work is specifically targetting to deal with the issue Allen's raising (it's called an identity discontinuity). Most existing JS code and legacy libraries are not built to work smoothly between realms. And that will remain the case because it's conceptually difficult. Frozen realm is a way to use a single realm where you have multiple protection domains and global scopes all sharing the single realm. Frozen realms is an extension of the realms work which solves this problem.

AWB: So that sounds like one way to do it. Part of what I'm wondering is that you are almost there and people are going to trip over that, and maybe some minor tweaks might address that. e.g., If jquery is shared at the parsed module level, then you get most of the sharing efficiency without the complexity.

DH: I definitely believe there's a lot we can do at parseModule to share the work you do.

YK: I want to reiterate what Dave says about this being a low level API. But the discontinuity problem has an inverse problem is some modules....such as Ember's server-side rendering library works, which is there is a realm which emulates some portion of the browser. Let's call it the Ember Realm. We want it to render some stuff then take the results and serve it over http. In order to make that work, the host realm needs to give Ember a way to get it back. That thing would have a discontinity if you copied it, because that thing's internals expected to be in the other realm.

MM: If both of those were operating under a single frozen realm...

YK: If you think the solution to that problem could be solved at the parseModule by making copies.

MM: So you're not arguing that you cannot solve that problem with a single frozen realm?

YK: No I was simply arguing that you could also solve that with the parseModule API.

DH: Do you mind if I do a quick synthesis. What we're all taking about is sort of the classic problems of sharing and mutation. And there are sort of classic solutions which include mutability and copying. And so I think this is the first trip around the block to explore, and we'll keep exploring that and not trying to solve more than we need at this point. I think making it inexpensive so that you can instantiate a module across realms is (very good).... Frozen realms. I like immutable data structure. I think frozen realms have a lot of promise as well.

BF: I agree with everything you just said, so that's good. Has any thought been given to the behavior of the window.proxy object?

DH: Domenic has been asking useful questions about this on one of the issues. We have some discussion here.

BF: In node, we don't have a proxy object for example so we just need to make sure..

MM: Just need to head off the term confusion. The `WindowProxy` object cannot be modeled with an ES6 Proxy.
BF: Has there been any consideration of job queues? e.g., the promise realm is shared across realms?

MM: Yeah, they're pro-worker per agent. Not per-realm.

BF: I didn't see anything about controlling promises here. Is that separate?

MM: Yes, once Dave revives the old wiki.ecmascript.org pages. I can point you at a bare start at an API to try to re-ify agent.

DH: Even though they belong in separate layers we want to make sure that they are....similar.

MM: They're largely orthogonal. If you wanted job queues and enqueue things on specific job queues. None of that is written down on the agent API, but if we wanted to address them reflexively they'd go in ??

AWB: Mark, the wayback machine has a pretty good archive of wiki.ecmascript.org.

DH: Ok, thanks everyone.

Conclusion

- Stage remains at stage 1. This was just an update.

10.iv.a Arbitrary precision integer type for stage 2

slides

Slide: *Why Integer rather than Int64?*

DE: Basically the strange wrap–around semantics in C is almost always a bug. Anyone who actually wants that can apply it straightforwardly.

Slide: *How?*

DE: New primitive type `Integer`.
MF: I wanted to point out because I didn't originally realise it myself: the `n` suffix is chosen because it's orthogonal to the base representation. You could represent binary, octal, decimal, and hex notation integers.
BE: The suffix "n" is a good choice because it is the second letter in "Integer" and "I" would not be a good choice.

```
// Takes a Integer as an argument and returns a Integer
function nthPrime(nth) {
  function isPrime(p) {
    for (let i = 2n; i < p; i++) {
      if (p % i === 0n) return false;
    }
    return true;
  }
  for (let i = 2n; ; i++) {
    if (isPrime(i)) {
      if (--nth === 0n) return i;
    }
  }
}
```

Slide: *Code sample: asm.js (?)*

```
function Add64Module(stdlib, foreign, buffer) {
  "use asm";
  var cast = stdlib.Integer.asUintN;
  var values = new stdlib.Uint64Array(buffer);
  function add64(aIndex, bIndex) {
    aIndex = aIndex|0;
    bIndex = bIndex|0;
    var aValue = values[aIndex>>3];
    var bValue = values[bIndex>>3];
    return cast(64, aValue + bValue);
  }
  return { add64: add64 };
}
```

DE: It's not clear if we want to add support for asm.js.

BE: It's hard to remove asm.js as it's just a subset. It's not like WebAssembly is the only way people will...

DE: If we did want Integer to work with asm validation. It can work, but it's unclear if we want to make it work.

BE: Let's talk about that later.

Slide: *Library Features*

DE: For library features there's Int64Array and Uint64Array. And functions to get these from array bufers.

- Uint64Array, Int64Array
- Integer static methods
  - Integer.parseInt
  - Integer.asUintN, Integer.asIntN

Slide: *No Implicit Coercion*

MM: *questions about n...*

DE: The current proposal is acually to use "N". Bikeshedding.

BE: I don't think it's the name so much as the variability of the "n".

MM: I was thinking you would say Integer.asInt64, rather than Integer.asIntN(64,...)

BE: If you look at the ASM.js example, you can see that it's cast asUintN and then ...
DE: Maybe we should have asInt64 as a spcific thing.

MM: Now that I see that the number was a parameter, I have no objection.

DE: This is identical to the Integer64 proposal but there's no implicit conversion. This is because there is no type that subsumes the other. Also, if you have a bare 1, it has to stay a Number for web-compatibility. Therefore we require an explicit cast such as calling the Number constructor.

DE: If you do + Integer the current proposal would throw (?)
AK: I don't think it's only asm.js that uses this. People have used addition elsewhere to cast things to numbers.

YK: Yeah, I've done it.

SYG: If plus Integer returned an Integer?

BE: It's not truly backward compatible. If you mix old and new code (and we changed the semantics of +) then we can't change that.
AWB: This is always an issue with ...

WH: It's only a matter of degree. Unary + is commonly used to coerce to Numbers in existing practice. However, unary – can also be used to coerce to Numbers, but prohibiting unary – on Integers to produce Integers wouldn't fly.

BE: Possibly an open issue: what to do with +

DE: It's on the bug tracker (Shu filed it).

MF: Dan do you mind going into some of the issues with comparison operators now?

DE: The current semantics are that `Number === Integer` will return `false`.

Slide 14: Comparison semantics: current proposal

- `1 === 1n false`
- `1 == 1n` is TypeError
- `1 < 1n` is TypeError

MM: I like this. With `==` I'm concerned: what does `1n == null` do?
DE: It coerces null into a Number, do the compare, and then get a TypeError. We could discard future-proofing so we could make the comparison operators work for this case.

Slide 15: Comparison semantics: Allow semantic comparison

- `1 === 1n` is `false`
- `1 == 1n` is `true`
- `1 < 1n` is `true`

MM: I like the prior slide better. It's not worth the special case. The one defensible use of `==` that people have defended is to say `== null` as shorthand for `=== null || === undefined`.
WH: What does that mean with regard to that slide?

MM: The answer should be `false` instead of throwing. Users expect that for any value x, x == null will not throw and is equivalent to x === null || x === undefined.
YK: Does `==` ever `throw` right now?

WH: Yes, that's an argument for defining the values of mixed comparisons such as `anInteger < anNumber` and `anInteger == aNumber` to their exact mathematical results instead of throwing.
YK: There's a specific invariant that `== null` never throws,
but `==` in other cases does not neccessarily follow that same rule.
AWB: Both `==` and `===` are defined by type enumeration of the two arguments. Two of those types are the type null and undefined. Those could be special.
MM: What I'm saying is that they should be special with regard to `null` and `undefined` not with regard to other Integers.
DE: I think that for comparing with `undefined` and `null` the expectation is that it would return `false` and not `throw`.
DE: Continuing the semantics explanation. + would not lose precision.

BE: Can you explain the plus issue again?

DE: The plus issue is that IEEE754 arithmetic is well defined. When you try to combine them you will get outside of the domain of both. There are two reasons why mixed operations are bad. The first is you will confuse users. The second is that you will run into bugs.

Slide 10: Optimization potential

- Use Smi (small integer) infrastructure to optimize as Int64 (sometimes)
- Could work with asm.js
- Multiple browsers expressed optimism about implementing with good performance

DE: For optimization potential currently across JavaScript engines there are .... Could also be used to optimize Integer as Int64 (?). We could make it work with asm.js.

MM: What's Smi?

BE: Small integer.

YK: Fixnum.

YK: I believe the optimizers are optimistic about optimizations. Do we expect that implementors will implement these optimizations?

AK: They will be as good as if we implemented 64-bit integers. The main implementation feedback is that it won't be any worse than 64-bit integers.

BT: I don't think that's what I heard exactly. Conversations from 2 months ago. In order to optimize Integer, we need profile data to show that this value is always in the 64bit range, then we can do the optimal thing there. That's what lets us do the optimization. We would not need to teach our engines to get that data, so I don't think it's difficult.

YK: Is the assumption that once things warm up it's the same? Or are you saying (Adam) that u64? would have the equivalent performance problems to making it fast in cold state.

AK: For v8, to be really fast, you want to fit inside a Smi.

MM: As long as you have any tagging at all, as long as you use nanboxing? or not to do it ...

YK: I understand why ?? comes up. Are there no internal paths that never expose the values to user code that could take advantage of the known representation..

AK: Our runtime code also uses the GC Heap so....you can have it on the stack sure, but if you pass it to some other runtime thing.

YK: I guess that's what I'm talking about. If you know it's u64, then you can optimize it.

AK: But if it's a typed array you know that already.

BE: V8 boxes still right?

AK: Yep.

YK: Do bignums require that you heap-alloc pretty liberally?

AK: No, for 64 bits you have to heap allocate.

DE: Until you know the types that are flowing through the program. I think a lot of implementations will be heap allocating, even 64-bit integers. Because if you have a pointer? on the stack that's a 64-bit pointer, you have to keep that separate in some way from the 64-bit integer.

SYG: We are confusing two optimizations. Int64 and BigInts are going to be the same because we need to allocate stuff. For V8 that's about the same amount of work. For SpiderMonkey, I think it will be more work, but only a little bit more.

BT: That's the general message I heard as well. Slightly harder, but it's not worse enough that our first implementation will be pretty much on par with what Int64 would have been.

BE: E.g., 1.x for some fraction. Where the various cast operations are used in asm.js etc, you could imagine have it compile to unboxed, etc. implementation. Is any implementor looking at that?

SYG: Since our asm.js is kind of ahead-of-time. Any engine that optimizes for asm.js has to kind of have that stuff anyway. The cast operator and the with operator...you realize it may not always be a constant, but it's not going to be a ?? big deal.

DH: If you ask me to write down the type optimizations that I need for primitize types, I don't know if I could do that. But how hard is it to characterize that?

WH: There are really two obvious choices. There are only two I can think of that do reductions of an Integer x modulo 2^{64}: `asUintN(64, x)` and `x & 0xFFFFFFFFFFFFFFFFn`.

DH: Those get you into the type. You also have to know what operations you can do to keep you in the type.

WH: It's the same ones.

DH: Right, but in order to do arithmetic with the type, you have to understand how to compose this.

WH: Integers form a mathematical ring. Uint64's form a finite ring. The ring of mathematical integers maps nicely to the ring of integers modulo 2^{64} in the obvious way. So you can do as many of the ring computations (+, -, *) as you want using mathematical integers and do one reduction modulo 2^{64} at the end, and you'll always get the same result as if you had been using Uint64 modulo arithmetic throughout.

DH: That's great, I enjoyed abstract algebra in college.

BE: It's shown in the example Dan gave for asm.js.

DH: We're getting there. I'm asking how hard would it be to write down the list that these are the sets of operations that we can do that would a) inject you into the Int64 value space and b) eject you from the Int64 value space (??)

DE: TurboFan has a great ??? for modeling those kind of ranges here. I would recommend people who really want to stay within the range to apply the cast operator after each operation. If we were going to stay into ASM.js we would require that.

WH: That's overkill. All you need is to do the reduction to Uint64's of inputs. You can do as much multiplication, addition, substraction, and shifting left as you want, as long as you do one reduction at the end, before you store it into a variable that escapes. You must also do the reduction before any comparisons or divide-like operations, including shifting right.

DH: Let's say I'm writing some crypto code in JS and I want it to be fast. A) How hard is it to write that code? What do I need to do to get that code to work? I'm not arguing against big ints here. I like big ints. Dan you made the claim, most of the time you don't want big ints, but it's good to look at the cases where they do and understand how hard it is to use their goals. People in node have been asking for this for years. So if we optimize for that, how hard is it (for implementors) to optimize for? And then how hard is it for the ...

DE: I did talk to some people in the node community about this. Some people expressed a preference for Int64 over Integer, however the maining worry was about possibly falling into a bad performance path if they left out the cast operator. On the other hand there were positive arguments in favor of interger over Int64 in some of these use cases. ... something about hashes... With Integers we can definitely build integer math faster than we could directly in javascript. We could use SIMD or similar things.

YK: One of my sad Rust experiences is opening the great int debate in Rust. You want 3 things: wrapping (Crypto), an exception if you overthrow, or you might want a bigNum. This proposal targets where you might want a bignum case (and there are a lot of cases where that's true). If you're writing crypto.

BE: I talked to Fedor Induttny and others about this and they want ...

YK: My point is a bit broader than the specifics. This proposal also says we don't want u32. We don't want many types. BigNum is the right thing. But if you look at those types there are use cases outside of BigNum. TL;DR there are cases where u64 is the thing you are actually talking about.

BE: I agree, sometimes the type is different. Sometimes when you want the wrap-around or other things, the type is the most ergonomic solution. That is the trade-off that we are giving up now in favor of other benefits.

YK: I don't think we need to give up those other benefits (or options for alternative types).

WH: Yes, we decided this at the last meeting.

(...)

WH: Last meeting's consensus was more than that, that we should pick one or the other. And that the one we should pick is Integers. Everyone that supported Uint64 decided we should go over to the side of Integers in January.

YK: I'm saying I don't want to object to that consensus. However, I am concerned about a discussion that happened in January, that may not mean consensus (about foreclosing ever adding machine–size types in JS would be a mistake).

MPT: We're doing this for node, let's get this done (paraphrased).

BE: I want to make progress. I view Integer as progress. I don't agree that we should foreclose machine types in the future, but I think that can and should be deferred.

DH: Trying to foreclose is a perfect way to make things run around in circles. It makes it harder to think that we need to try to close the door to other options in the future.

DE: We will work with people to make it futureproof.

YK: Does anyone believe that if they support this proposal they are not allowed to support any other types of numbers in JavaScript in the future?

WH: That's not my position. I was on the side of Uint64 in January. At that meeting I said I'm fine with Integer for now and we achieved consensus on specifying that as the solution for now. I have no particular objections to doing Int64 in the future if it's still needed.

AK: There was a discussion about should we do Integer, should we do Uint64. And it seemed like Integer was a way forward for that and met with some significant positivity at that meeting.

BE: We achieved consensus on moving forward with Integer first and making sure it had the cast functions and other things to make sure it had efficient math using Integers.

AK: And that is the spirit why Dan is presenting the proposal in this way.

DH: If there is confusion about what people are saying it's very good if we get that down. There are many discussions where we have multiple alternatives and we pick one of them and sometimes people take away (incorrectly) that we can't do any of the other ones as well. It's important to get down in the minutes that we are still open to pursuing other paths (with numeric types). I want to make sure that's in the minutes.

BE: The only time the committee salted the earth and said "never will something be added" was named spaces from ES4. oh and: RING HOMORPHISM!!!

DE: In the discussion with people that had libraries, I got a sense that Integer would indeed be a significant help.

DT: Does this help any users that were interested in decimal and things like financial users?

BE: I did talk with FinTech people. Crypto people not use Uint because it's the best they have access to. FinTech people want BigInts. (?)

DE: I've heard interest from FinTech client of Egalia, but don't know details.

Slide 11: Specification status

- Detailed spec text based on Brendan’s Int64 spec
- Open issues
  - Editorial: How to deal with the various spec numeric values
  - Should mixed-type comparison be allowed?
  - Should ToString include the literal suffix?
  - Bikeshedding
  - Bigger standard library, or save more for v2?
- And many more! Please come on GitHub and tell me what’s wrong with the proposal

DE: There is detailed spectext. It has some color issues that I will address. It has open issues.

DE: Currently in the spec there appears to be some implicit coercion between Numbers and mathematical values. Once we have multiple numeric types, we have to be more explicit about how this is accomplished.

DE: *Should mixed-type comparision be allowed?* - We could allow it. It wouldn't extend to user-defined types as well.

WH: I believe I and Mark would be in favor that in order to get `null` to work.

MM: I'm in the opposite position. I want comparison to `undefined` and `null` to work for all types, but do not think that mixed type comparison between different numeric types to work. That's weird but with the spec'd behavior, then there's no inconsistency with how comparison would work for user-defined types. That consistency is good.
BE: You handled that in the `null` and `undefined` special cases. I agree with Mark and I got the committee to first say they liked Uint64 and then this proposal, I think we'll do better if we keep firm on that requirement. It may be a bit tricky, but I'd rather avoid that.
MF: Do you consider mixed type comparison, implicit conversion?

BE: In general yes.

WH: Allowing `5n == null` and not allowing `5n == 0` is untenable.
DE: I have gotten feedback form potential users about the comparison. A lot of users are unhappy about throwing exceptions on mixed operations. This one seems solvable, but some similar ones are not solvable.

WH: The reason I support mixed comparisons is precisely because they do NOT consider implicit conversions, they consider exact values.

DE: I want to leave it as an open issue for now, It would be resolved for stage 3.

BE: I think this issue has to stay open, because in the terms you just raised there may be some conversions unless we hard-code some cases for Integer.

MM: And how do user-defined value types participate in these comparsions?

BE: We don't like these problems any more than anyone else does, but to make progress we decided to defer a lot of these discussions. We said no explicit conversions, no ??? and we'll wait on the other stuff. That has worked in the committee before. But I admit it can do short-term violence.

MPT: I would think the community would rather see no impicit conversions vs. implicit conversions that don't make any sense.

MF: What is an example?

MPT: Dates conversions can end up really weird. `00` vs `000` dates.

DE: Next issue. Should toString include the literal suffix "N". The current spec doesn't put the N.

WH: It's analogous to asking whether string.toString() should put the quotes around it or not.

MM: `toString()` should not be thought of as something to recover the source string for a programming language. What it should be doing is represent using characters something that indicates what the value is.
WH: So `toString()` of Integer 5 should return what?
Everyone: `5`!
DE: I agree too and that's what the spec currently says.

DE: The last question is the standard library. There are a lot of things that could belong in a standard. Should these be in v1 or should we add them later.

BE: What issue number is this?

DE: #20.

DT: Is Number 0 truthy?

DE: No, that's an open question.

WH: The current proposal is that 0 is falsy and all other numbers are truthy.

BE: There's precedence in the committee going back to decimal.

MM: How do you imagine 0 falsy extending to user-defined types?

BE: Value types would have an examplar 0 value.

YK: I have a question about what else we might do here. There is a place in the current standard library with JSON. Which is that the JSON number format technically allows you to express things that are not Numbers. Perhaps it would be interesting to pursue a JSON.parse that gives you arbitrary precision integers.

DE: I think that's a good idea to think about. There's an open bug to think about JSON Integer.

DE: Is this ready for stage 2?

WH: I am happy with it. I did a good review of it with DE. The remaining question for me is how this will work with Array buffers. There are some interesting questions there.

DE: There is an issue on that: issue #23.

Conclusion/Resolution

- Stage 2 approved
- Reviewers:
    - Waldemar
    - Leo
    - Brendan
    - Michael Saboff

PR #778: Make LocalTZA take 't' and 'isUTC' and drop DSTA(t)

(Maggie Pint and Matt Johnson)

Matt on the phone ?? (M)

MPT: Describing a slide about issues with daylight savings time in JavaScript....

MPT: What we've done is said, always pick the first one. The reason is because of a scheduling context. If you have something needing to happen at 1AM and you are told the second 1AM then you're going to be mad. You get a similar thing with the spring forward transition. You get a set of local times that do not exist. Spec says move the ambiguous time by the direction. That's not predictable for the user so we define the spec to always move it forward. Because time never randomly jumps backward in real life.

WH: What do you mean by moving forward?

MPT: So in the United States during this time, 2:30 AM doesn't exist. So it's going forward by the amount of the transition now.

WH: We always have outages when people schedule cron-jobs at 2:45AM.

MPT: It's going forward on the global timeline, not backward. 2:30 is becoming 3:30.

MF: Why do want this to be valid?

BT: We don't.

MPT: We don't, but this already has a lot of implications, but it's better than going randomly in some direction.

BT: The current semantics are that you get a timeout, you don't get a .....

MF: ??

MPT: ??

MJ: The cron job example is a good example of not wanting to throw, because then you'd get no cron jobs at all for that day.

BT: My feeling is to not get into the business of fixing the existing date object and keep things as similar to the existing implementations as possible and create a new one that's better.

MF: On the non—existing date, if you ask for a date and it cannot give you a valid response, then ...??

MPT: You are going to break if you do that, for example with the cron—job example michel stated.

WH: What does it do if you ask for March 32nd?

MPT: It wraps to April...

WH: That's precedent.

BT: Yeah I think in general this date object is designed not to tell you and just do *something*.

YK: Is the intent here that we're always moving forward, or always moving backward?

MPT: Forward, so this is what it *may* currently do, this is what it was doing two years ago (slides).

BT: That's just the spring forward case, for the spring backward case it makes more sense to... ??

YK: If you always go forward, but if you compare the date with the date that comes before it seems like it will always be a positive number. If you're adding a number it should never be a smaller number than you.

MPT: When would that happen?

YK: I don't know.

MPT: Do you mean the "Fall" case when it'll bring back ??? in UTC?

YK: I think this whole thing is trying to make things incrementally better.

WH: The proposal does not satisfy the property you (YK) requested.

MPT: If you take fall forward on the global timeline then things will trigger an hour later.

MJ: So one thing I'll add is Java8 and its standard and pretty much all the platforms that have this bug have been trying to solidify around the best way to handle this and *this* is the same set of expectations that they have standardized around.

BT: But to answer Yehuda's question, I'm not sure that this would come up because when you get a date that's in the context of a timezone in JavaScript... So once you have the date for where you are on that timezone, your time is disambiguated.

MPT: Ok, so we have consensus on this?

BT: I don't think we're trying to make this incrementally better, we're trying to define behavior.

MPT: This is a "needs consensus" pull request.

DE: That does a little bit more than this. I've seen the other bug multiple times. I can explain that if people are interested. I'm happy for this to be merged.

MPT: If you want to take that, go for it.

DE: Sure, the definition of the shift for localtime is based on a daylight savings shift. Moscow times shifted historically by an hour. This patch also makes that local timezone offset a function of the time you're passing in, just like the daylight savings offset.

MJ: I thought that was correct in ES6.

DE: Some things were corrected in ES6, but this part wasn't.

AWB: I believe the reason this wasn't changed then was to do what you're describing requires having the historic data available. There was reluctance to imagine having implementations (keeping the historic timezone data around).

MPT: Matt, I think what you're thinking of is the soft language that says you should prefer the IANA database when possible.

MJ: Correct, I thought it was already formalized in the spec.

DE: I think this pull request adds it. I'm happy for it to be added. I looked for how JS engines implement local date-time transforms and they seem to all do it by calling out to these system calls to get the local time based on unix UTC time and then for engines that implement the spec you get the localTime of now and find its .... time based on that.

MPT: If I remember correctly, what happened is that it used to say you should follow the current daylight savings rules but it never codified that you should follow the IANA database when possible.

MJ: The other thing I was recalling was that ECMA-402 ??? So this has already been implemented based on 402.

DE: Really? Wait? Which article?

MJ: 12.3.2 in ECMA-402

WH: So which way does 402 resolve this?

MJ: To take history into account, and modern operating systems can do that.

MPT: That would only cover formatting, that would not cover date.

MJ: Correct, it needs to be merged back into 262.

BF: Is this timeboxed?

Everyone: No...

MPT: Is everyone clear on the use case?

BT: I think this is more of an implementation question here.

BE: In May we'll have guest speaker from ?? talking about an ES6 implementation excluding eval...??

BT: That's like 95% Intl data.

BE: I don't know how small that database can get, but we should not decide this without talking to people doing very small JS engines that have to embedded in (very small hardware)?

DE: If you look at the timezone file, for a particular timezone it's pretty small.

BE: But for the whole world it's really large.

DE: But the date functions only let you use your current timezone.

MJ: It only says use the best thing available to the system.

MPT: The offset of the local timezone in UTC measure in ms. To me ??

BE: That 48k of RAM is running a JS engine. They're very cheap systems that run for a week or more. They're IoT...

MPT: I'm sympathetic to this IoT issue, so we could change the wording to say: "if it's available otherwise fall back to current."

BE: The problem with the web is you tend to get a de-facto standard even if you don't specify it.

AWB: That's ok.

BE: It'd be ok.

MF: Didn't we have similar concerns when talking about Unicode support? What's the resolution to that?

BT: Yep, we're requiring everything.

AWB: It came up specifically about case mapping.

BT: toUpperCase and toLowerCase require the full case mapping.

MF: So we should probably do the same thing here as well.

BT: Yeah..I'd like to see.

AWB: That's one of the things you'd have to evaluate on each case.

DE: If you're an embedded system you only have to implement a single timezone.

BT: But we require it for Unicode case mapping.

DE: It doesn't take up that much space.

BE: It might be that they'll also show up for the appliances, and nobody wants to make appliances for each region that has the specific dates for that region.

DE: Don't you actually want to have accurate dates?

MPT: You won't find anyone that wants accurate dates more than me. But I don't think we can ignore that we're asking them to take on a lot of data.

AWB: Seems like one of those cases where a strategic "should" is the right answer.

BT: We can't use "should".

AWB: Sometimes we do, it's ok.

MJ: (reading from ECMA-262). The PR removes some of that wording and replacing it with some examples and solidifies the ambiguity. It doesn't say you must use the full implementation.

MPT: Nobody is arguing for the ambguity, but what we're at is.

BT: I was just curious. In light of your concerns around IoT and requirement of data tables. Are we going to be able to move forward with a new date proposal that would require this kind of data to be available in order to have a good system?

BE: I don't know. I think it's an open issue. We should hear from the ??? folks. As usual, people innovating on certain areas don't come to this meeting, they're too busy.

AK: How much do they care about conforming exactly to the ECMAScript standard?

BE: It's big enough. They can't use npm, they don't want to. But they do have a standard library. We'll hear from them in May. I think it'll be interesting to talk with them about this topic then.

MJ: All I'm saying is the PR in question does not state that you must carry the entire timezone database. It tells you to use the best available, which can be different.

AWB: And that can vary among implementations.

BE: Does that achieve the goals that you have?

MPT: The main thing to me is to resolve the ambiguity issue.

BT: We all do it anyway, so it's not like there's a pressing issue here.

DE: This is an important issue for Chrome, and this bug keeps getting filed on Chrome. It'll be great to get this codified.

BE: You mean allowed but not required.

AWB: Is there something that does not allow historically?

MJ: In 6 it is not allowed.

DE: In 6 it is allowed for ??? and disallowed for ???. This is why ?? started working on it and it's now disambiguated.???

BT: So I guess the base question here is: Do we want to say that everyone, all implementations of this, it's OK or do you want to say we're wrong and we should do something else instead.

AWB: We want to allow what you're doing, whether we want to require what we're doing.....

BT: So this does not require, but to allow.

MPT: If we all agree that the proposal does not require, then it's sufficient.

AWB: Or let's make the proposal so that it doesn't require.

BT: We can make sure that it doesn't require. Because I'm fully supportive of Dan and Chrome folks closing bugs.

MPT: Ok, so we have consensus on this to be merged.

BT: Any objections? We definitely want to make it optional to use historic timezone data. Using the best information on the timezones.

AWB: That language of "best available" is...

BT: It's handwavy....

MJ: We should say the best available on the operating system.

BT: But what's an operating system? ;-) Some systems don't have one.

MPT: The more restrictive the phrasing gets the worse it gets given the situation...

BT: Why don't we just use "may" instead.

MJ: I think it's a "should".

BT: I really don't want "should".

DH: I don't want to go down the road of "magic feathers".

AK: I don't think so.

DH: Shoulds and mays are not good...

YK: Isn't this what notes are for?

BT: Umm... No. Notes cannot include normative content.

BE: Dan says you want to allow it.

BT: But "may" is you can if it's available.

AWB: "should" if it's available and "no" if it's not.

BT: We have consensus % this question of the specific wording...

Conclusion/Resolution

- Consensus modulo improving the language

12.ii.a Date Proposal - NodaTime as a built-in Module for stage 1

- https://github.com/maggiepint/proposal-temporal
- https://docs.google.com/presentation/d/1b6gTBphc-QEYE6rqYZ6VkFDRLE8K-EDx464tGfZ-Q6Q/edit?usp=sharing

(Maggie Pint)

MPT: I have this proposal seeking stage 1. Sometimes we like to take stage 1 proposals to stage 2 criticism. This is a replacement for date as it exists today and would be built on top of the date module system that I believe Brian is the champion of. In order to drill on this topic which is huge... I actually pulled one of my conference talk decks... The fundamental problem you have in datetime is perspective. In order to make it work, we assume that we have this global timeline that isn't ambiguous and it doesn't go backward and forward. {continues to give her talk... see the slides}

Slide: Coordinate UTC

MPT: - We assume we have this global timeline.

MPT: Coordinate UTC (not the time in London), basically the same as GMT, but the scientific community prefers GMT.

Slide: Local Time

MPT: On the flipside of this you have local time. It does not exist on the global timeline. It usually not contiguous (with daylight savings, etc).

Slide: UTC Time

MPT: When you have a time in UTC, you know where you sit on the global timeline and you know if this point in time is before or AFTER some other UTC time.

WH: How accurate of time-keeping is this proposal intended for?

MPT: Like how accurate should the API be? I'm hoping for nano-second precision, but that depends on the timeframe.

WH: Super accurate time keeping runs into issues with relativity, etc. It becomes very tricky to keep track of "absolute" time beyond millisecond precision.

MJ: There are plenty of scientific applications that care about nanoseconds in the data itself.

Slide: Local Time: Saturday, April 9, 2016, 9:11 AM

MPT: When you have local time, you don't know where you are on the timeline. And you don't know whether it's before or after another time in a different locality. You limit comparison operations.

Slide: Timezone Basics

MPT: In short, time zones are a region that agrees on an offset from UTC. These change. Egypt, Russia, others recently changed this.

Slide: Time Zone: America/Chicago

MPT: *Referring to a graph of all time zones*

BE: I work with someone who thinks we should all just use UTC.

MJ: No.

MPT: When is the sun out? That's a timezone.

laughter

Slide: code examples (timeline math vs. calendar math)

```
moment('2016-01-01').add(1.5, ' hours').format('LLL') // " January 1, 2016 1:30AM"
 moment('2016-01-01').add(1.5, ' days').format('LLL') // " January 3, 2016 12:00AM"
```

MPT: More explanation of slides.

WH: What's the difference between the two times?

MPT: One is calendar math, and the other is timeline math. You can only make computations in whole numbers with calendar math.

WH: Why is it January 3rd and not January 2nd? I.e. why does it round up?

MPT: Because you can't have half a day. It rounds up. And that's because you can't define that. It's important that you make it clear that you can only work in integer units and you can't convert between calendar math and timeline math. One day is not a fixed number of milliseconds or minutes. Fundamentally in a datetime API, you're moving along the global timeline. But when you're doing calendar math, you're moving along the calendar. You can only have your finger on one day at a time.

BT: Are not minutes variable length as well?

MPT: So, most datetime APIs ignore leap-seconds which is what I think you're getting at and they do that because from the business-domain of datetime, they don't matter. They do in scientific timelines and durations. For our purposes, we're going to ignore leap-seconds in the date API because they belong in a different domain.

AWB/MPT: Are there any minor programming languages that have a leap-second API?

MJ: The biggest problem you get into is internationalization. You have to consult a table to know whether that is valid. Usually if someone passes a leap-second timestamp, it just gets absorbed and rounded down to 59.

Slide: JodaTime/NodaTime Types

MPT: Steven Colburn created the JodaTime proposal. It models the problem domain really well. He's also involved in this proposal.

MPT: If you think about time as an "instant" is a point on a timeline.

MPT: The only thing you get from an instant type is a point, a point in life. From there, you have a typeflow for zoned date type. A composition of a point on the global timeline and a timezone. And a calendar system if you want to get fancy. So now what you have is a datetime in a timezone (that's sort of the god type). Once you have all of that, then you know everything you need to know about that time. In that case a `ZonedDatetime`.
WH: Does the localtime include the current offset or not? I'm asking in the context of ambiguous local times on the fall back day which can be disambiguated by including the current UTC offset.

MPT: The way the ... timeline represents that is an offset from now.

BT: You're talking about the zone datetime?

WH: When you convert an instant to in zone, does the result include the current offset from UTC?

MJ: There's a little bit of variation between some of the implementations that have come about. JodaTime variates due to this. ZonedDatetime based on the instant means there's no ambiguity. It's an implementation detail. If you're starting with local values... ??? We have an open thread on one of these issues. This doesn't have to be resolved right now.

MPT: So... ?? It's pretty common to have LocalDateTime. Another really common is LocalDate, like "July 25th, 1926" there's no time or timezone information with that. The tendency is to zero-fill the time which creates a lot of ambiguity. You don't know whether it was midnight or zero-filled. In that same vein, you can have time only, where you have a day, but you don't know whether it was really January 1st, or they were year-start filling?? So these types should remove that ambiguity. If you have a LocalDate and a LocalTime, the only place you can land is a LocalDateTime, so it forces developer safety. It models the set of concerns.

MF: Is any of this proposal going to contain time and date arithmetic? Or representations of duration?

MPT: Yes. I felt like I needed to get you to here.

BT: There's also more of a text-based introduction, the NodaTime docs have a great primer on the types.

WH: Are dates in Gregorian calendar only?

MPT: That's a debate that's going on in the proposal right now.

MJ: On that, Noda and Joda both said yes, and then Steven corrected it in Java 8 and said no. We want something that works well for ECMA-402 and ECMAScript.

MPT: So Steven's argument is when you make non-Gregorian ....?

RX: Can I ask a basic question. I would like to understand about LocalDateTime, when you're talking about LocalTime, you say "it's usually not a contiguous timeline (DST)". On LocalDateTime, in which timezone, do you assume to make those jumps?

MPT: It assumes no timezone because of that the API will force you to do certain conversions. Say you have a LocalDateTime and it doesn't exist and you try to convert it to a LocalDateTime, you'll have to specify a resolver to say which instant you pick if you have an ambiguous time.

RX: It's confusing for me to understand if it's not going to have a timezone attached to it, otherwise it should be contiguous.

MJ: `LocalDateTime` as a type, is contiguous without a timezone reference at all. If I add an hour, I add an hour, if a day, it's one contiguous day. There's no relation to a timezone.
RX: Just like UTC?

MJ: Correct, except UTC is relatable back to a date in time, whereas here I just have an amount of time, no way to relate it back to a specific point in time.

MPT: Matt, Is it the case... I feel like John makes it you have to do conversion....??

MJ: That's the difference between period and duration in the Noda API.

MPT: So what we're saying to breaking things down... If you have a LocalDateTime type, the author made the decision that you could only do calendar computation, day or bigger. This means you circumvent this ambiguous time problem. If you want to move to hours or minutes, you'd have to convert into a ZonedDateTime.

MJ: The real distinction to keep in mind is that `LocalDateTime` doesn't have a time zone reference.
RX: Ok, thank you, I understand.

MPT: I've presented a large number of types. I haven't even touched the offsite datetime type which belongs in this as well. I didn't even touch the OffsetDateTime spec.....Everyone is probably familiar with the ISO time formatting specification, which can help you understand local and global time, but it doesn't tell you what time it will be in an hour from now, because we don't have a time zone.
MPT: So, what we're proposing for now is that this particular API, fully models the DateTime domain that's correct and helpful to the user. As you have noticed, it is very large. Other types we can hit on, we discussed on Tuesday a Duration Type, a period type, which is calendar intervals, and all of those types together make up the complete domain.

MPT: The first thing I'm looking for from this committee. Is this a road that we want to go down?

BE: We want to do the more minimal approach of the extensible web manifesto and see if we can keep the standards out of big APIs.

MF: You have precedent. ??

MPT: Yes, lots of precedent.

DH: One thing I'm not clear on is how much of this depends on surfacing stuff the OS knows and how much of this is pure library that could be written in JavaScript?

MPT: Umm, most of it is pure library that can be written in JavaScript and the only thing that's being surfaced is the time zone data and the point on the timeline. I'm not completely attached to this proposal. There are other options.

BT: Another option, and maybe this goes to Brendan's point, add an API to expose timezone data and then let libraries...

DH: That's where I was going with my question...

MPT: I think you want an immutable data type for Date. I think the fact that the current Date object is mutable is destructive to users.

AWB: The existing Date object is always and forever going to be the existing Date object.

MPT: The immutable value type has direct user value. The rest of it has great value as well. But to not have people messing around with dates...

AWB: A library could do that...

BT: It could but ... ?? value type...??

MPT: A date is a value type...

BT: This is the biggest reason why I support this proposal over just exposing time zone data.

MPT: ??

AWB: If what you're saying is...

BE: It's not a value type in the sense of n64.

MPT: It's an object.

DH: That was my first question. We were talking about an immutable object, but its `typeof` **is object.**
MPT: Right! Yes, sorry. We fight the fight on Stack Overflow about "you can use a library that works or you can do the thing you're doing." There's this set of people who are going to use the language primitives. Those people aren't making good code. It goes beyond just mutability vs immutability. Date doesn't have add methods, you have to get/set them, they bubble, and you don't have safety when you overflow a Uint.

MF: Does everyone agree that using date arithmetic is a common thing. If so, I think we can agree with providing a standard library to solve this common problem.

YK: I disagree with some aspects of what you said. Cloning an existing popular thing is not correct.

BF: Pretty early on we talked about how it didn't make sense to combine types. Are we thinking we will try to prevent people from trying to add them together.

MPT: You can combine LocalDate and LocalTime, but you can't put it on the timeline because you don't have timezone information.

MJ: So you can't do something like LocalDate and then say at 3:00, what is that exact value in UTC. You can't say that because then you involve a timezone.

BF: So we couldn't for example compare with an instant.

MPT: Correct, you can't compare `LocalDateTime` with an instant, because you can't compare two instances that aren't on the global timeline.
BF: This is what is concerning to me, having people do ambiguous things like what we talked about in the previous discussion (about time zone problems with the current `Date` API)
MJ: Another clear example of why this is neccessary. One of the biggest things that comes up on S.O. is why do I have a disconect between HTML5 dateTime types and the ES6 Date types. Someone will take something from an `<input type='date'>` and throw it into a `Date` constructor.
WH: Is the month number for December 11 or 12?

MPT: Let's index from 1. So it's 12.

CP: My only feedback is that we should create an API???? There's a little overlap with the 402 proposal, I'm not sure it's connected. Either way, we plan to expose some pieces of the timezone. Maybe that's an area we can collaborate on...??

MPT: Sure, there are actually quite a few overlaps with this proposal and 402. If we were to replace the date object no matter how it works, there are overlaps in the form of parsing. Locale Based parsing or formatting should go into 402.

CP: No 402 doesn't do *any* parsing.

MPT: Locale–specific date format, you need CLDR data for that.

AWB: I'm not sure what your ask is at this point.

MPT: My ask is "do we want to move forward with this API module or do we want to simplify it?"

BT: I think in light of Brendan's concern, we should decide whether we should be in the business of making more big APIs.

YK: I don't think Brendan would say there shouldn't be *some* new APIs here.

BE: Remember the min–max classes? If you can find the min–max BetterDate, I suspect there will be more winning all around.

AWB: So I think exploring the date spec here, The date API is something there's a clear need for me. It's not clear to me how it should be packaged up. I'm not sure it belongs in either.

MPT: laughing....

AWB: How are new APIs packaged? We've talked about built-in modules. There may be a difference between built-in and standard modules. You may be able to define a built-in module that's not part of 262 but you could import.

MPT: That's interesting.

BT: And that's kind of a separate... I mean, we should move forward.

MPT: I was not expecting to get a consensus. Could I get a read on people's thoughts?

BT: I want to point out we are doing some max/min here. This proposal include the most useful types of the system but it does *not* include *everything* from Noda or Joda time. It's a small subset of that.

WH: For me this hinges on a few things, Yes it is adding things to the language that are in the language already, though it's doing it much better. The questions for me are: how big is it, how stable is it? And should we provide the full functionality or should we just provide the building blocks so someone can build on top of this?

WH: External libraries can evolve relatively easily and can deprecate mistakes or blind alleys. On the other hand, it's extremely difficult for us to deprecate functionality we add that we regret later. So, how much has the proposed library changed over the last 5 years?

MPT: Not much. Stephen made a few simplifications in Java8, but the APIs have been relatively stable in long time.

BT: The base types have not changed.

MJ: We have some quotes from Brendan about copying dates from Java :)

Everyone: laughing

BT: The committee is not moved by Java.

Everyone: laughing

MPT: We cannot rev-versions. We cannot just get rid of it. Once it's in Ecma spec we cannot change in "Java 8".

BE: The web is so big, you cannot control or deprecate things. I'd love to deprecate the Date object. It can be deprecated by the community and linting tools. There's no "flag day" for the web.

MJ: If you go the module route, can you rev modules?

BE: Modules you load from the server, so of course. But in the browser environment, no.

AWB: That's the point he's trying to make. We should be able to standardize modules. ...

BE: If you say it can be a module that can be hot-fixed, that's fine. If you say it's standardized, it's very hard to change.

MPT: The minute it's in a browser, it lives forever. Here's what I'm feeling. People agree we should do work here. I don't think that people agree on the API.

BT: I'm not sure that's true, I haven't heard a reasonable alternative to this API. We agree that Date is broken. There is general agreement that JavaScipt should have some built-in affordance for doing date operations without shooting yourself in the foot. I think that there is a desire to trying to create giant APIs, but I have not seen an alternative that is reasonable.

YK: I'm trying to avoid injecting random work that I did in the past into this conversation. I would not agree that JodaTime is a de-facto standard.

MPT: I would not say it's a de-facto standard.

YK: I made a strong claim. Let me jump to the punch-line. There are specific details we can extract from JodaTime. I agree that we could make it in some senses like that... stage 1 seems fine is what I'm saying...

BT: The point that I want to make more clear is that we would all like there to be a small `Date` object that would function as the kernel of date-innovation, but I'm not convinced that's possible.
MPT: To the contrary, the developer community in JavaScript has been living with moment.js for 5 years. Moment.js is a ZonedDateTime. You can do everything you need to do with it, it just isn't pretty.

YK: Are you saying there may be consensus that something like this may ??

BT: I'm encouraged by what you said that is totally independent of this effort for Rust that this model is a useful thing.

YK: I found the operations that JodaTime and Java 8 have some footguns that I would like to avoid.

BT: Sure, but it seems like there's a lot of alignment. Seems like this is a good path. We're not just copying everything. We'll be intentionally copying just some semantics.

YK: I don't think tie-breaking without some thought is good.

MPT: Nobody's suggesting that.

MF: These are well-past stage 1 concerns.

DH: This is obvious stage 1.

WH: Yeah, let's do stage 1.

MPT: Let's start with that. Thanks!

Conclusion/Resolution

- Stage 1

Annex 2

Code of Conduct (Proposal)

Home Enforcement Subcommittee Reporting Guidelines FAQ Changelog

TC39 Code of Conduct

Like the technical community as a whole, TC39 is made up of a mixture of professionals and volunteers from all over the world. To avoid communication issues and unhappiness, and to promote inclusiveness, we have a few ground rules that we ask people to adhere to.

This isn't an exhaustive list of things that you can't do. Rather, take it in the spirit in which it's intended - a guide to make it easier to enrich all of us and the technical communities in which we participate and empower others to speak.

This Code of Conduct is enforced within all spaces managed by TC39. This includes IRC channels moderated by TC39, mailing lists such as es-discuss, issue trackers on projects hosted by TC39, and TC39 events and meetings.

If you believe someone is violating the Code of Conduct, we ask that you report it by emailing tc39-conduct@ecma-international.com. For more details, please see our Reporting Guidelines.

Be friendly and patient

We understand that everyone has different levels of experience or knowledge in many diverse fields, be it technical or non-technical in nature. We also have areas of knowledge we are eager to expand. We want to be a community where people can not only contribute, but feel comfortable to ask questions as well and learn along the way. When correcting another participant, respond with patience and try to keep it polite and civil. Remember that we all were newbies at one point.

Be welcoming

We strive to be a community that welcomes and supports people of all backgrounds and identities. This includes, but is not limited to, members of any race, ethnicity, culture, national origin, color, immigration status, social and economic class, educational level, sex, sexual orientation, gender identity and expression, age, size, family status, political belief, religion, and mental and physical ability.

Be considerate

Your work will be used by other people, and you in turn will depend on the work of others. Any decision you make will affect users and colleagues, and you should take those consequences into account when making decisions. Remember that we're a world-wide community, so you might not be communicating in someone else's primary language.

Be respectful

Not all of us will agree all the time, but disagreement is no excuse for poor behavior and poor manners. We might all experience some frustration now and then, but we cannot allow that frustration to turn into a personal attack. It's important to remember that a community where people feel uncomfortable or threatened is not a productive one. Members of the TC39 community should be respectful when dealing with other members as well as with people outside the TC39 community.

Be careful in the words that you choose

We are a community of professionals, and we conduct ourselves professionally. Be kind to others. Do not insult or put down other participants. Harassment and other exclusionary behavior aren’t acceptable. This includes, but is not limited to:

- Violent threats or language directed against another person.
- Discriminatory jokes and language.
- Posting sexually explicit or violent material.
- Posting (or threatening to post) other people’s personally identifying information (“doxing”).
- Personal insults, especially those using racist or sexist terms.
- Unwelcome sexual attention.
- Advocating for, or encouraging, any of the above behavior.
- Repeated harassment of others. In general, if someone asks you to stop, then stop.

When we disagree, try to understand why

Disagreements, both social and technical, happen all the time. It is important that we resolve disagreements and differing views constructively. Remember that we’re different. There is strength in having a varied community with people from a wide range of backgrounds. Different people have different perspectives on issues. Being unable to understand why someone holds a viewpoint doesn’t mean that they’re wrong. Don’t forget that it is human to err and blaming each other doesn’t get us anywhere. Instead, focus on helping to resolve issues and learning from mistakes.

Original text courtesy of JSFoundation, Speak Up! project, and Django Project.

Questions?

If you have questions, please see the FAQ. If that doesn’t answer your questions, feel free to email us.

Code of Conduct (Proposal)

The JS community is awesome.

Home Enforcement Subcommittee Reporting Guidelines FAQ Changelog