Node.js Foundation Modules Team Meeting 2018-07-11

* **Recording**: Coming Soon
* **GitHub Issue**: https://github.com/nodejs/modules/issues/148
* **Minutes Google Doc**:
https://docs.google.com/document/d/1m1yCbb5A6t8KGy9Ov33I7YNKR_fxMPCIJLC3HymGLS8/edit

Present

- Bradley Farias (@bmeck)
- Geoffrey Booth (@GeoffreyBooth)
- Gus Caplan (@devsnek)
- Guy Bedford (@guybedford)
- Hassan Sani (@inidaname)
- Jan Krems (@jkrems)
- John-David Dalton (@jdalton)
- Jordan Harband (@ljharb)
- Michael Dawson (@mhdawson)
- Michael Zasso (@targos)
- Myles Borins (@mylesborins)
- Saleh Abdel Motaal (@SMotaal)
- Kevin Smith (@zenparsing)

Agenda

Extracted from **modules-agenda** labelled issues and pull requests from the **nodejs org** prior to the meeting.

Approving PRs

None this week

Update on Progress (10 minute Timebox)

* Have presentation on loaders. [#135](https://github.com/nodejs/modules/issues/135)
  - 3 minute Timebox
  - not scheduled yet, might need dedicated time slot
  - confusion around loaders & their terminology, presentation meant to clarify the basic concepts and develop a shared language
  - prob. 45, could be 30 minutes
  - conclusion: put it on for next week

* Initiative: Terminology / Historical Decisions documents
[#119](https://github.com/nodejs/modules/issues/119)
  - 3 minute Timebox
  - looking for additional participation
  - initial attempt / draft has been developed but needs more work
  - Saleh happy to keep driving it but wants input on the actual content to ensure it's correct
  - issue has been stagnant and progress has been lacking
  - no volunteers right now
  - myles pretty busy for next week, might be able to jump in afterwards
* Developer Survey [#85](https://github.com/nodejs/modules/issues/85)
  - 3 minute Timebox
  - open question: should we do an open survey or rather focus on specific stakeholders like bundlers and other ecosystem tools
  - could be an important input channel for future discussions

Discussion (45 Minutes)

* Thinking about deadlines [#123](https://github.com/nodejs/modules/issues/123)
  - 10 minute Timebox
  - question: what are other members of the WG thinking about the proposed deadlines?
  - [bradley] current deadlines aren't realistic, especially the actual dates. Due to expanded scopes and ongoing discussions, we move the end date back
  - [ljharb] our goal shouldn't be to shipping soon or fast but to make sure that we ship the right thing. We do need to decide on what our first implementation should be and what our constraints should be. Discussion around specific dates is premature
  - [jan] we should have milestones at least and we also need to update the community / ecosystem on timing expectations
  - [saleh] bigger problem: we're trying to fit something into the wrong hole. The standard should recognize the trend that we need to separate between the browser and node. Afterwards we can solve building on top of that better standard. In the end our solution has to be biased against *someone* out in the community
  - [guy] agreed: we have to be goal oriented, have deliverables, have public status. E.g. What is the status of base-level implementation? W/o further information, community will make assumptions. Talk about phases (initial, loader support, …) and in what order they will come
  - [geoffrey] The docs & public info made it look like modules are imminent. Might want to communicate that it isn't.
  - [ljharb] does --experimental-* not help?
  - [geoffrey] not in practice, people are used to adding those kinds of flags
  - [ljharb] we need to have better education about flag meaning
  - [bradley] every time we press ahead with "minimal implementations", the more work we create in digging into them and picking them apart. Instead we should make choices about trade-offs

  - [myles] purpose was not to set explicit deadlines or rushing things. It was more about developing a mental model of how much time various phases take and what might happen when

* Pull request opened for import.meta.require on core [#130](https://github.com/nodejs/modules/issues/130)
  - 10 minute Timebox
  - alternative suggestion by myles for doing interop
  - [brad] it's premature to land the PR, there's an entire different thread (do early errors instead?). There is a moratorium on new features
  - [myles] purpose right now is to discuss it
  - [brad] first: loader ambiguity. Can a loader determine the format of a certain file? Risk of creating multiple instances of a singleton modules. One in cjs, one in esm.
  - [brad] up-babel - takes CJS, turns it into ESM. it has to guess which version of import to use
  - [myles] is this only for combination of transparent and import.meta.require
  - [brad] no, transparent interop alleviates it. Meta.require makes it look simpler but leads to weird behavior like duplicate singletons
  - [saleh] thinking about import.resolve - if there's import.meta.resolve, you don't need meta.require because you can always dynamic import. Do others think that would alleviate the concerns? If you can't resolve something, you won't require it. Meta.require is not something I'd encourage. Dynamic import using transparent interop to return exports object
  - [jan] where does the duplicate imports come from?
  - [brad] for example from import('x')/require('x') loading different copies of the source. There is a migration problem
  - [jan] isn't that already a problem with 'x.mjs' and 'x.js' and import('./x') / require('./x')
  - [brad] no, this adds new issues, beyond that

* Package-Name-Maps a proposal for bare imports in browsers [#51](https://github.com/nodejs/modules/issues/51)
  - 10 minute Timebox
  - [brad] no matter what we do - we'll never have 100% web compat. Forcing us into package-name-maps makes us highly dependent on what whatwg decides. We shouldn't be beholden to choices made for package-name-maps. Matching the web: good. But necessarily through p-n-m
  - [ljharb] didn't look too deeply into p-n-m proposal. Like the idea of having npm generate the map, being able to just use the files in a browser without mutating import paths. Not good for node to use a literal on-disk static mapping file. In favor of keeping directory searching etc.
  - [guy] clarify: when thinking about maps & compat. Relative imports w/ file extensions is where compat becomes a hard choice (requires removing the ability to insert extensions)
  - [kevin] we shouldn't try to fix things that aren't broken. Examples: Node's resolution mechanism, integration with npm for installing dependencies. No need to rewrite because there's a proposal to solve a similar problem in the browser

  - [geoffrey] if we want an incompatible module system, we already have CJS. understand concerns about being beholden to browser decisions. But users should be able to write code that runs in both big JS systems. Lot of value in consistent behavior, even if the spec doesn't *require* it and it might prevent some nice features
  - [brad] counterpoint to "we can't have the cake and eat it too". Node resolution algo has advantages. See: angular app that needed to change for minimal impl. Tools are already required for code the ecosystem is writing. There's a strange thing: why propose things that require tools and then require not using tools?
  - [ljharb] web compat is a big goal, but it's not the whole story. Even without web compat it's an important feature to be added to node. Might require tools for use in browser
  - [geoffrey] examples for maps are pretty short simple files. For things like jquery they can easily be written by hand. There are use cases where people won't need a build tool in all cases. You can create package.json by hand just like you can create a *map by hand
  - [gus (in chat)] that map isn't so simple, it has to hold **all** the bare specifiers an application uses, not just the ones your top-level code uses.
  - [saleh] haven't looked at details yet. Have used TS path mapping recently: relating a name to a path which is then used to resolve imports. We should all get more familiar with the challenges *maps might bring. Should talk about more details in the future - dedicated issue for discussion?
  - [myles] important: what kind of user experience do we want to have? Design to be forward-thinking. Might be CJS patterns - or new patterns. Re maps: no direct support necessarily but resolution algo compatible with them. When talking about browsers: lots of talk about browsers as leaders who not listen and node as follower. We should shift that relationship to be more of a partnership. Participation in those conversations is important
  - [jan] more features means more complexity. And those complexities tend to stack up. Removing features & complexities from the old system and not bringing it into the new one can be an opportunity.
  - [brad] already problems with things like redirects breaking import.meta.url which is an open issue. Using name maps still allows all kinds of complexity!
  - [guy] PR 3172 file extension - how does that enable package map compat? Should we have a vote? Should we add it to the agenda?
  - [myles] as a group we have a lot of conversation and are being very thorough. But - how are we moving forward? We need to reach consensus. Challenge: as a group we might want to become chartered. Not being able to reach consensus might prevent that. We wouldn't want to constantly escalate to voting and pushing to TSC. If it doesn't seem like this group can get consensus in the future - what does this mean? How can we get there?
  - [brad] BUT THERE'S HOPE! We're here and talking about it.

* transparent-or-not interop [#90](https://github.com/nodejs/modules/issues/90)
  - Skipped this week due to time concerns