Node.js Modules Team Meeting 2020-08-12

Links

* **Recording**: http://www.youtube.com/watch?v=PDFFQnbFEcM
* **GitHub Issue**: https://github.com/nodejs/modules/issues/543
* **Minutes Google Doc**: https://docs.google.com/document/d/1zjqjv8G9fDnpGG01yBS39-tRRZPQ5E8N5aset38FKKk/edit

Present

* Myles Borins (@MylesBorins)
* Bradley Farias (@bmeck)
* Guy Bedford (@guybedford)
* Doron Tsur (@qballer)
* Julian Londono (@julian-londono)
* Jordan Harband (@ljharb)
* Jan Krems (@jkrems)

Agenda

Announcements

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

We unflagged Top-Level Await in Node.js 14.8.0. There is an open pull-request to unflag Top-Level Await in the REPL, but it needs to be reviewed on it's own merit as it uses a completely different implementation from TLA in ESM. There is also an open discussion about support static import in the REPL, as it could share similar infrastructure to TLA.

nodejs/node

* Policy conditions [#34414](https://github.com/nodejs/node/pull/34414)

* Special treatment for package.json resolution and exports? [#33460](https://github.com/nodejs/node/issues/33460)

JHB: lots of tools need access to package.json, with "exports" there is no reliable way to get it in every situation. 3 paths forward:

1. do nothing, packages must explicitly expose package.json
2. automatically/implicitly expose package.json
3. Add API to resolve a package root

No PR yet, but it seems like 3 is a likely route (module.getPackageRoot()?). Seems sufficient to have the ecosystem to use that without relying on require/import.

MBS: We don't have a way of loading JSON in ESM right now. It is being worked on/standardized. I could see value in some API that loads the meta-data without all the boilerplate. This is separate from when "exports" cannot resolve a thing. I recognize people are used to having the ability to load `package.json` without opting into anything. Perhaps a opt-out via setting it to "undefined" in the JSON would work. Tools that need to get what is / is not exposed package.json is the static source of what is /is not available and they have to do this.

JKS: If we state that tools can use X to get the package.json, it would be surprising if there is a way to opt out. It would be surprising if you were allowed to opt-out.

MBS: A lot of these tools are expecting more than just package.json at the root and are relying on our loaders to get them. It can be a slippery slope for all the assets.

BFS: the semantics of any API are complex / the data may not exist

JKS: concern is around publishing a popular package, which then opts-out and then breaks the npm ecosystem.

BFS: yes, we saw this with is-promise, but the breakage was due to implicit not explicit choice to prevent access to package.json

JKS: Making an API to get the package root is useful one way or another. The only way for a coffeescript loader to determine the type of the package is that the loader needs to replicate behavior or make a fake filename and call the hook. Such an API would allow a loader to reuse a lot of the builtin machinery to do these tasks.

JHB: Will proceed making PR for API.

* module: CJS exports detection [#33416](https://github.com/nodejs/node/pull/33416)

GBD: Geoffrey did analysis/numbers based upon specific success criteria based upon name exports intent from readmes. The block on the PR landing is from Gus on criteria must be for exact output rather than the analysis. Unclear if we can get consensus on this. Suggestion to only have criteria to detect transpiled modules to get 90+% case needed. Are we still defining the criteria about what users expect / esModules check. 81% for esModule currently 90+% for readme.

MBS: Geoffreys last comment for esModule > 99% success rate for esModule compared to docs. Accuracy seems to be an acceptable level. Any small work we can do to improve this would be good.

GBD: *lack of clarity on if it is 99% or 81% accuracy*. If it is in the readme 99% accurate.

MBS: maybe we can ignore all modules lacking esModule.

JKS: numbers may be biased for a high percentage rather than package author expectation.

nodejs/modules

* Subpath extension patterns and wildcard expansions [#535](https://github.com/nodejs/modules/issues/535)

MBS: maybe we should look at stuff, dynamism is concerning. Perhaps things like an extension map in the package JSON as a separate feature.

JHB: prioritized static in loading, we should re-evaluate usability vs static.

* import assertions RFC [#427](https://github.com/nodejs/modules/issues/427)

* Import Maps and Node.js [#544](https://github.com/nodejs/modules/issues/544)

* Removing `getFormat` hook [#34114](https://github.com/nodejs/node/pull/34144/#issuecomment-666716738)

    - Issue for this created [here](https://github.com/nodejs/node/issues/34753)