SPEC STATUS

• Grammar and static semantics draft-complete. Allen has begun merging into ES6 drafts.

• Linking semantics written in pseudocode, transcribing to Word this week.

• Evaluation and loading semantics implemented in polyfill, next on my list to spec.

• @jorendorff hacked us up a tool to convert literate Markdown comments to Word.
OUR PLANS

• Won’t wait for November F2F to post updates.

• Will see more progress in Allen’s updates.

• @jorendorff working towards getting polyfill functional as self-hosted implementation for Firefox Nightly builds.
TRACK OUR PROGRESS

- https://github.com/jorendorff/js-loaders
TECHNICAL UPDATES

• Syntax is done. Community actively building tools. Absent any surprise ambiguities, further debate is unnecessary and unwise.

• Trickiest part of loading semantics involves concurrent loading scenarios, which @jorendorff did great work on. Now needs implementation testing.

• Event-loop semantics would ideally be in ES6 but it’s cleanly factored out so we can live without.
TECHNICAL UPDATES

• Almost all of the loader pipeline is async. Allows e.g. remote translation/compilation/analysis.

• Separated translation hook from Function/indirect eval hooks.

• Eliminated the complexity of loader “inheritance.” Nested virtualization can easily be implemented explicitly via composition.
TECHNICAL UPDATES

• Biggest simplification: eliminate inline modules.
• Does away with controversial and complex feature; door still open for lexical modules.
• Bundling belongs at browser layer, and was problematic for cross-origin loading anyway.
• Bundling formats still implementable...
USERLAND BUNDLING

• Loader logic with a custom cache
• Custom payload formats (e.g., JSON)
USERLAND BUNDLING

Script injection with dynamic definition:

```html
<script>
System.set("A", ...);
System.set("B", ...);
</script>
```
GENERIC BUNDLING

Better avenue, current web proposal:

```html
<script src="assets.zip$/lib/main.js">
</script>

<img src="assets.zip$/images/logo.jpg">
```
BROWSER LOADER

- Not part of ECMA-262.
- Will work with Yehuda, Anne, Alex, and others on a Web spec proposal.
- Not blocking ES6 deadline, but needs to start now so we can p(r)ol(l)yfill and experiment.
Need further discussion of sync vs. async entry points in HTML. Ecma-262 will simply specify two top-level non-terminals, one that allows imports and one that doesn’t.

- Need for module linking/registration with on-demand execution.
- Discuss integration with other Web standards orgs.