



# JavaScript Standard Library aka Built-in Modules

for stage 2

Michael Saboff  
Mattijs Hoitink  
Mark Miller

# Agenda

- Proposal Status
- Responses to June 2019 Advancement Objections
  - Adding ability to import from scripts
  - Unified Namespace
- Advance to Stage 2

# Current Status

Current Proposal:

<https://github.com/tc39/proposal-javascript-standard-library>

# Request to add import from Scripts

- The reason for the request is to allow the new features added via a built-in module are available to scripts via a synchronous import.
  - ▶ This request also requires the ability to polyfill similar to the import from module case.
- Two issues that make this problematic.
  - ▶ There is no current syntax or semantics to allow for this.
  - ▶ Modules are evaluated asynchronous (and the end of the microtask) from scripts.

# Should this be added to this proposal?

- This proposal builds upon the currently specified module semantics.
  - ▶ We didn't want to add any new surface area.
  - ▶ Imports are currently not allowed in Scripts.
  - ▶ Importing from scripts is orthogonal to JS Standard Libraries.
  - ▶ Champions have no desire to expand this proposal to importing from scripts as it opens up a whole new set of issues.
  - ▶ Believe that importing from scripts should be a separate proposal.
  - ▶ Have no desire to tie this proposal to a Import from Script proposal.

# Unified Standard Namespace

- Was an objection to move forward unless there was one unified namespace for browsers.
  - ▶ There will likely be several namespaces. (Hopefully not too many.)
  - ▶ Standards bodies will likely want their functionality in a dedicated namespace, e.g. TC-39, TC-53, W3C, ...
  - ▶ Other namespaces will likely have multiple stakeholders, e.g web components agreed upon by both W3C and Node Foundation

# Two Namespace Classes

## Standard Managed

Governed by one or more stakeholders

Openly communicated governance

Identifiers widely communicated and reserved in central repository

Name correspond to functional areas

Modules are standardized

Included modules for wide adoption

Intended for large development community

## Vendor / adhoc Managed

Informal or localized governance

Private / adhoc governance

Identifiers chosen from unused but preferably reserved in central repository

Name based on vendor criteria

Module inclusion based on adhoc criteria

Included module at vendors discretion

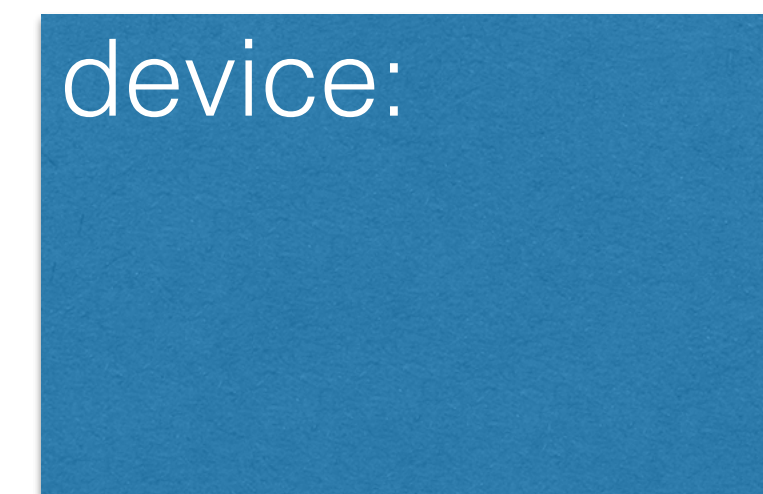
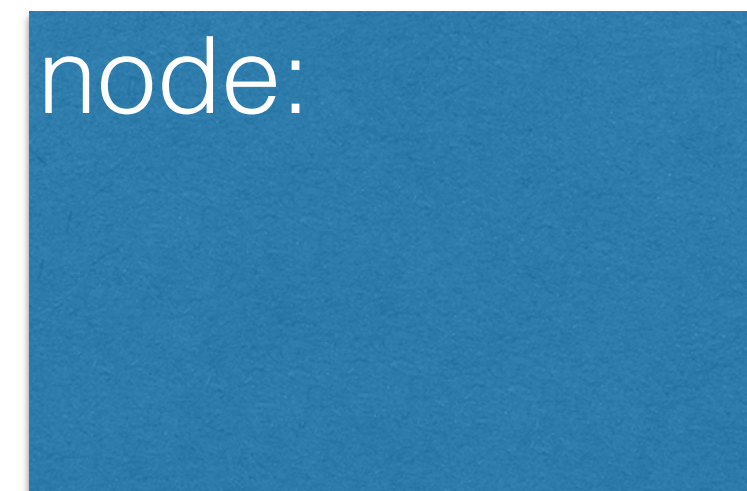
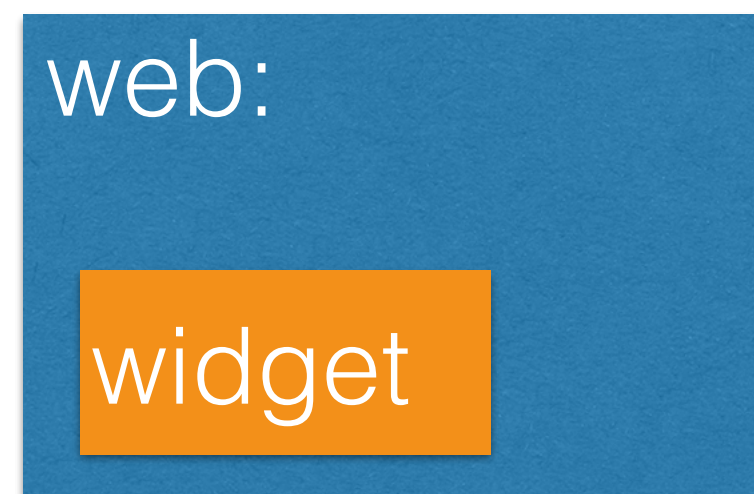
For vendor or product specific developers

# What About Modules in Multiple Namespaces

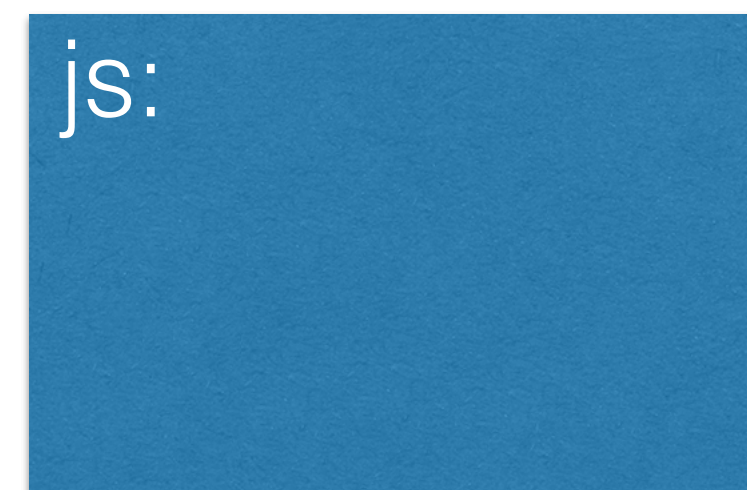
- There are two likely scenarios.
  1. Module useful in another namespace.  
e.g. node adopting a web module.
  2. Modules with the same name exists / proposed in two different namespaces. They have different functionality.



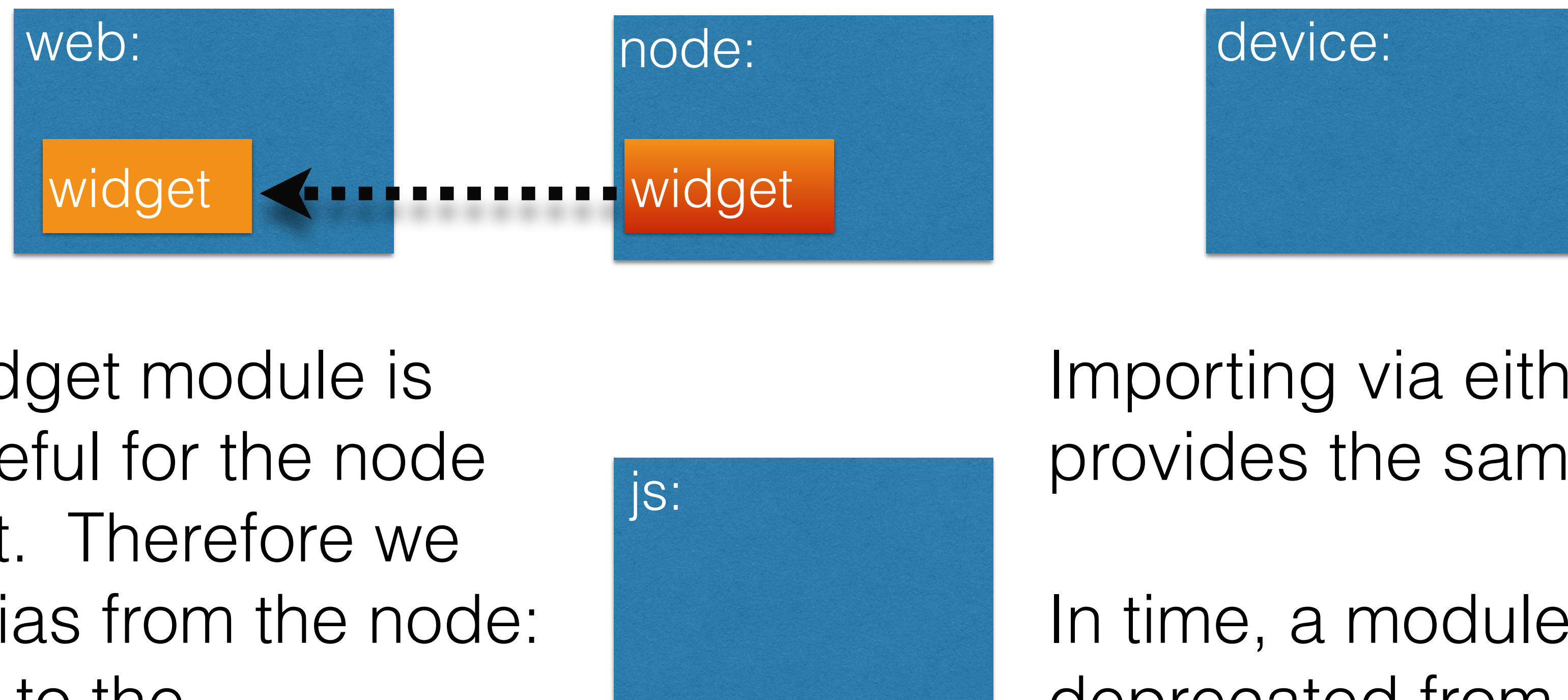
# Module useful in another namespace



The web:widget module is consider useful for the node environment.



# Module useful in another namespace



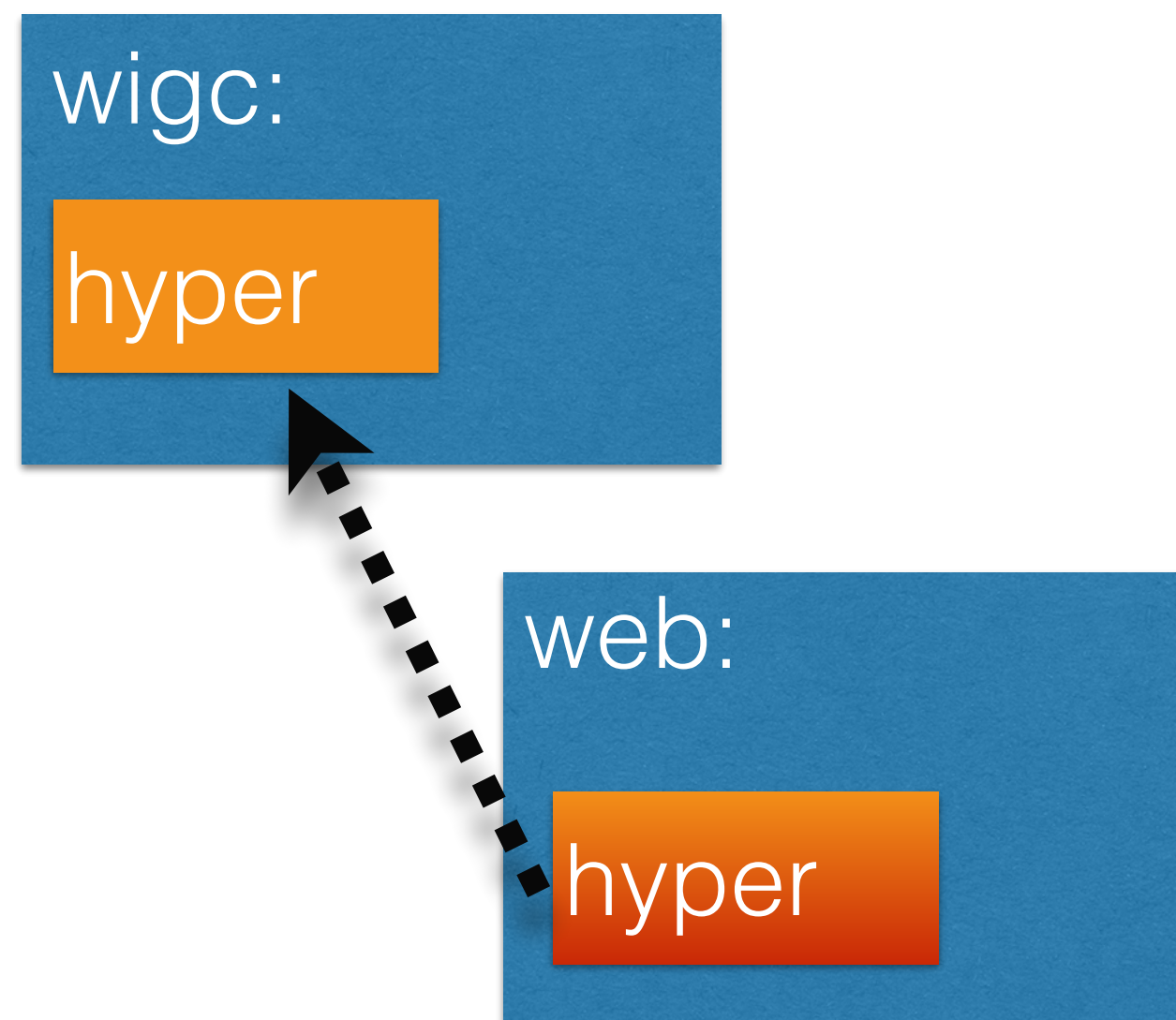
The `web:widget` module is considered useful for the `node` environment. Therefore we create an alias from the `node:` namespace to the `web:widget` module.

Importing via either namespace provides the same module.

In time, a module could be deprecated from one namespace to effect a “move”.

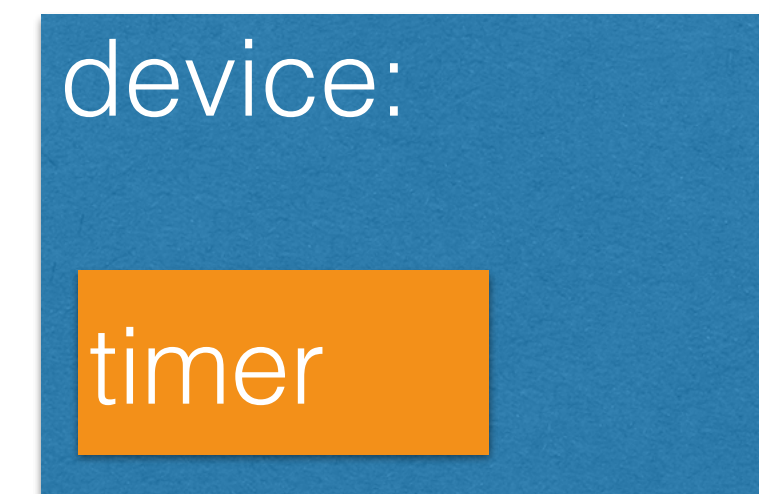
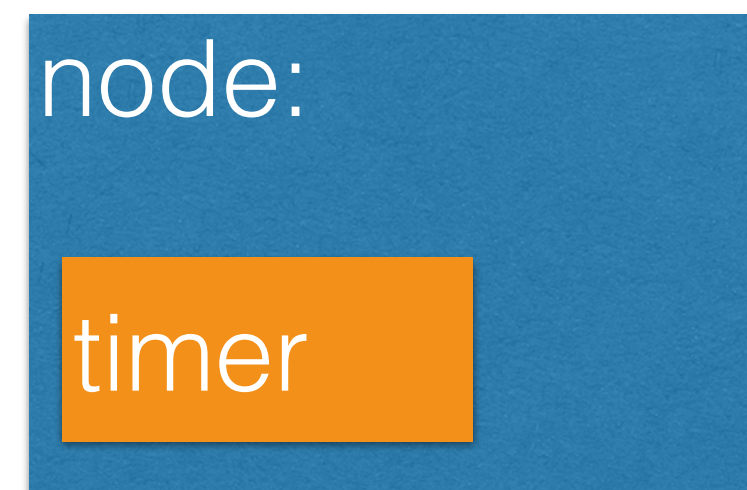
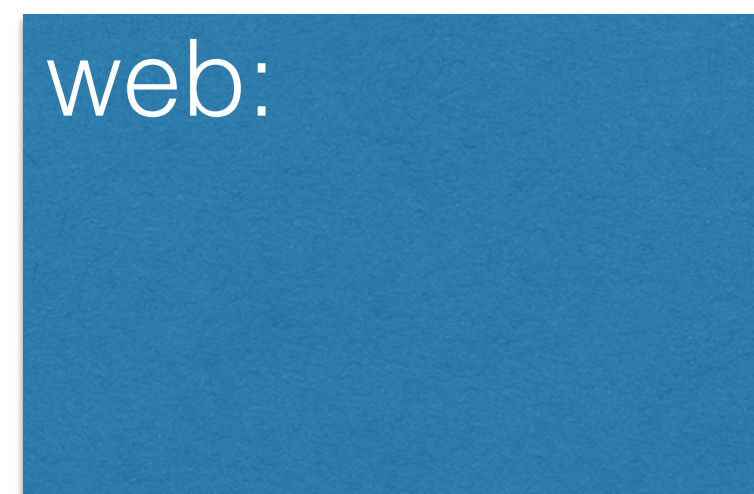
# Module useful in another namespace

- Consider a wigc: namespace where modules can incubate. When such a module is ratified by W3C, for example, an alias could be created from web:hyper to wigc:hyper.

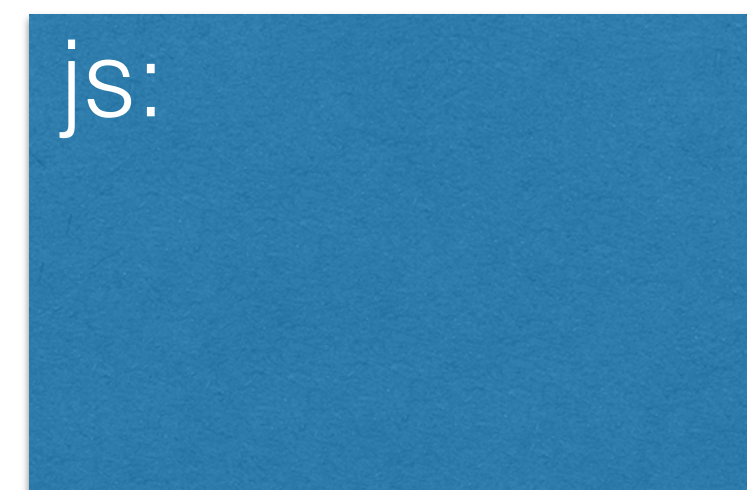




# Module name collisions



There exists a node:timer module as well as a device:timer module



They can easily exist in two distinct namespace.  
Any desire to alias would require reconciliation of differences.

# Our Namespace Proposal

- TC-39 is responsible for the content of the **js:** namespace (core language).
  - ▶ Outside parties are free to propose additions to the **js:** namespace.
  - ▶ TC-39 uses stage process to approve modules for the **js:** namespace
- We, TC-39, set up a repository for additional namespaces and share that repository with other stakeholders
  - ▶ We initially reserve specific namespaces, then open them up to other interested parties
  - ▶ The initial reserved set would include names like **web:**, **node:**, **device:**, etc

# Other Host Namespaces

- Hosts are free to use their own namespace (non-reserved) and include modules they choose.

Questions?

Stage 2?



Thank you!