# JavaScript Standard Library

## for stage 2

Mattijs Hoitink
Micheal Saboff

# Agenda

- Proposal Status

- Standard Library Namespace

- Immutable APIs

# Current Status

Current Proposal:
https://github.com/tc39/proposal-javascript-standard-library

# JSL Namespace

How do you import from the JSL?

# JSL Module Importing

- Following the same syntax for importing modules *(section 15.2)*

- This uses a *StringLiteral* to denote the *ModuleSpecifier*

- A JSL *ModuleSpecifier* will use the URI format:

    "**`<prefix>:`**`<modulename>`"

- The prefix is used to denote the functional domain of the module

# JSL Module Specifier

## Example

```
import { CivilDate } from "js:Temporal";
```

# Why Module Domains?

- Give developers clear guidance where a module can be used

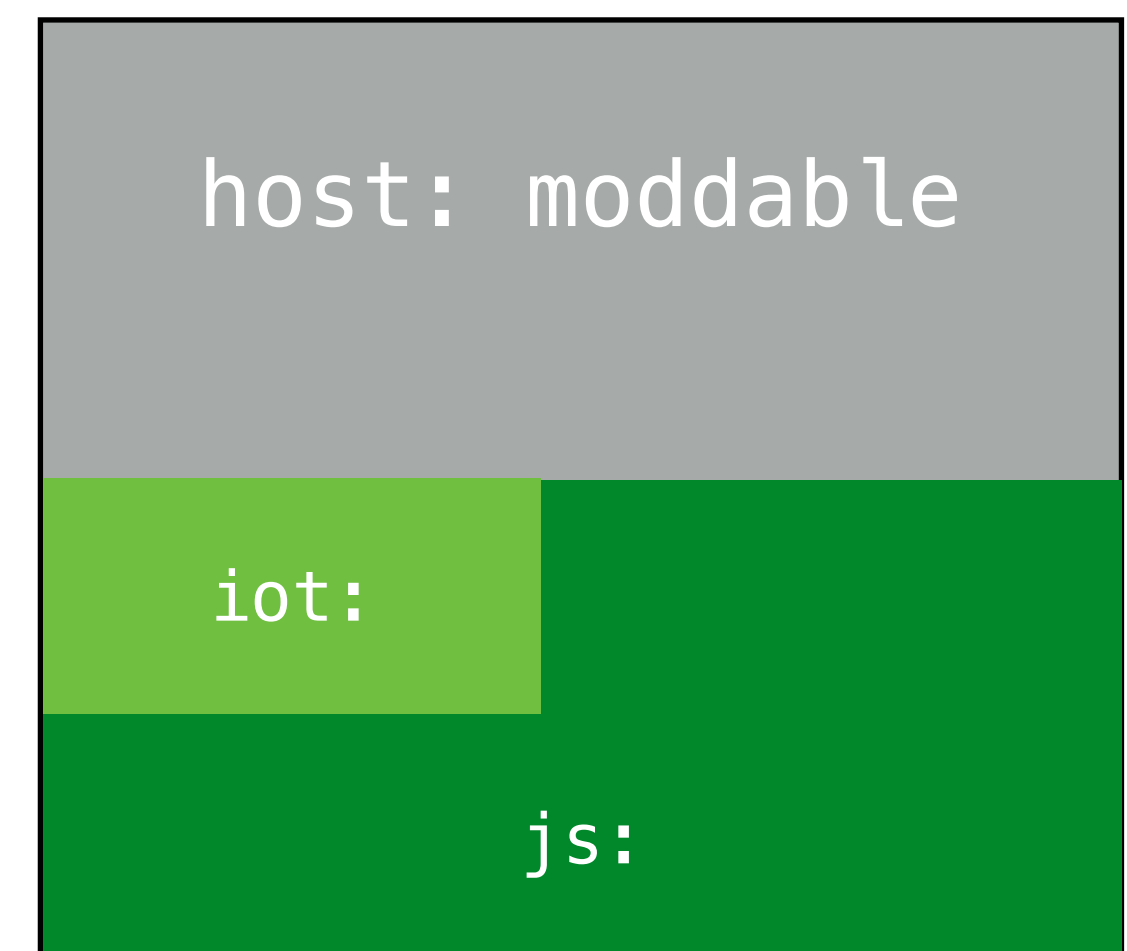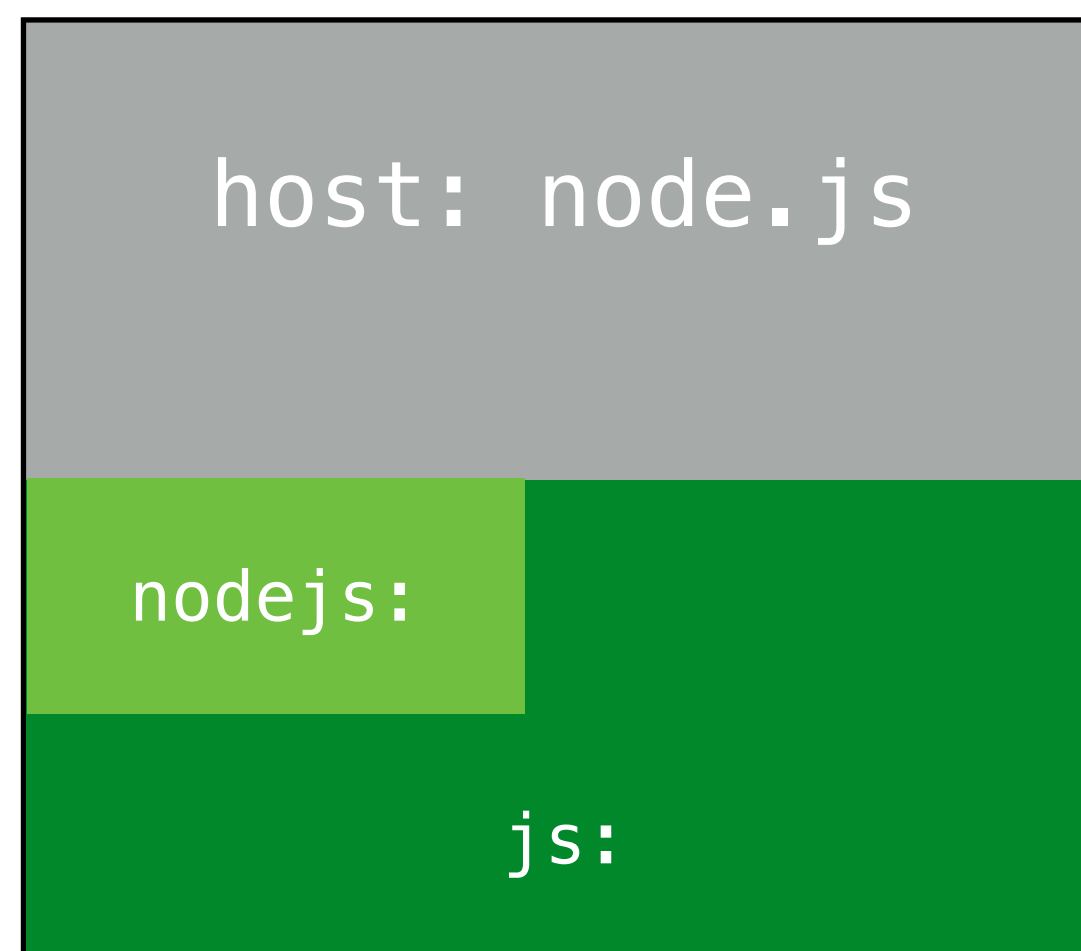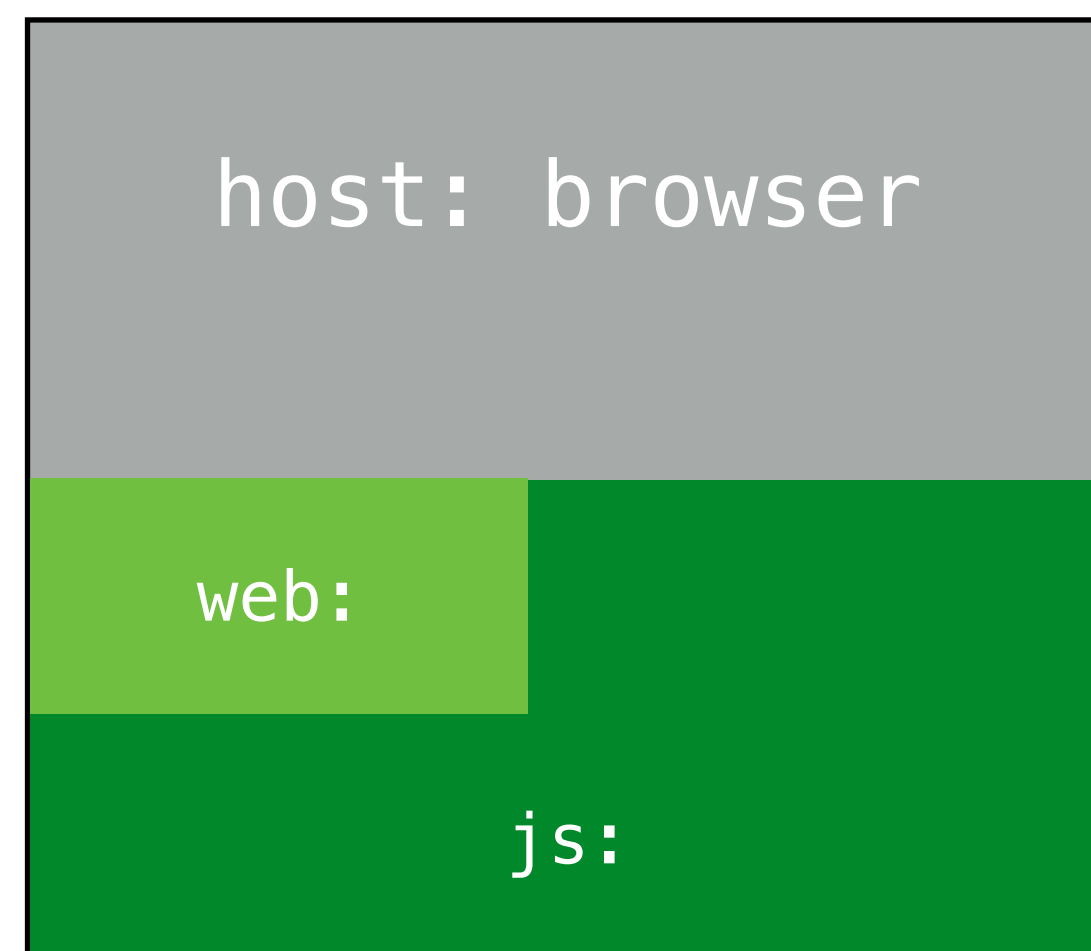- Facilitates writing portable JavaScript code

# KV Storage Built In Module

Example

```
import { storage } from "web:kv-storage";
```

# Host Specific Prefixes

- Namespace prefixes should split along host environments

# Namespaces
## To be registered with IANA

**Initial**

| | | **Future?** | |
|---:|:---|---:|:---|
| js: | JavaScript Language | iot: | Embedded Devices (TC53) |
| web: | Web Specific APIs | html: | HTML Apis |
| nodejs: | Node.js Standard Library | browser: | Browser Plugin API |

/

# Module Loading

Chained Loader

# Loading Chain

ModuleIdentifier
+
Referencing
Module

JSL

**1**

**resolve**

Host1

**2**

**resolve**

Host2

**3**

**load**

**5**

**4**

Module
NotFound
Error
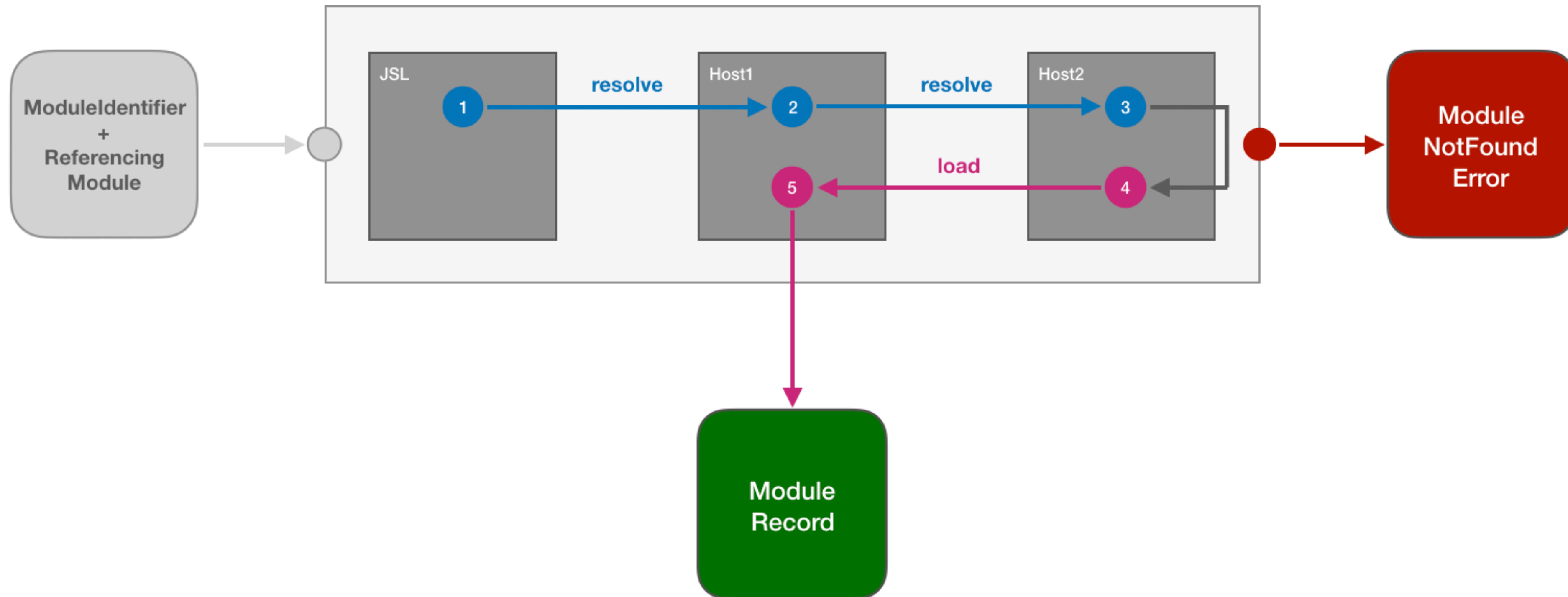
Module
Record

# Polyfilling Scenarios

- Add missing modules

- Update incomplete implementations

- Patch broken parts

*import-maps should handle these*

# Questions?

Thank you!