Ecma/TC39/2010/004

EMaker style modules for ECMAScript

Ihab A.B. Awad <ihab.awad@gmail.com>
Google, Inc.
Jan 2010

Goals

- Isolation
- Security
- No global namespace
- No constraints on simultaneous “versions”
- First class objects
- Non-blocking without boilerplate
- Zero-admin provisioning
- Uniform location and retrieval

Import syntax

```
import moduleName
```

Value: module function representing `moduleName`.

Keyword args: bindings for free variables of code of *`moduleName`*.

```
new (import 'util/point')({ x: 3, y: 4 })
    (import 'util/point')({ x: 3, y: 4 })
```

Module syntax

Any **FunctionBody** with free variables

May declare internal variables, return result (module instance)

```
var lastX;
return {
  getX: function() {
    return x;
  },
  setX: function(x_) {
    if (x_ > 0) { lastX = x; x = x_; }
  }
};
```

Module syntax

May use ctor style (for use with new)

```
var lastX;
this.getX = function() {
  return x;
};
this.setX = function(x_) {
  if (x_ > 0) { lastX = x; x = x_; }
};
```

Module syntax

Evaluated as `use lexical scope`

No aliasing of `this` (or anything else) to bottom scope

Primordials: discuss later

Implementation

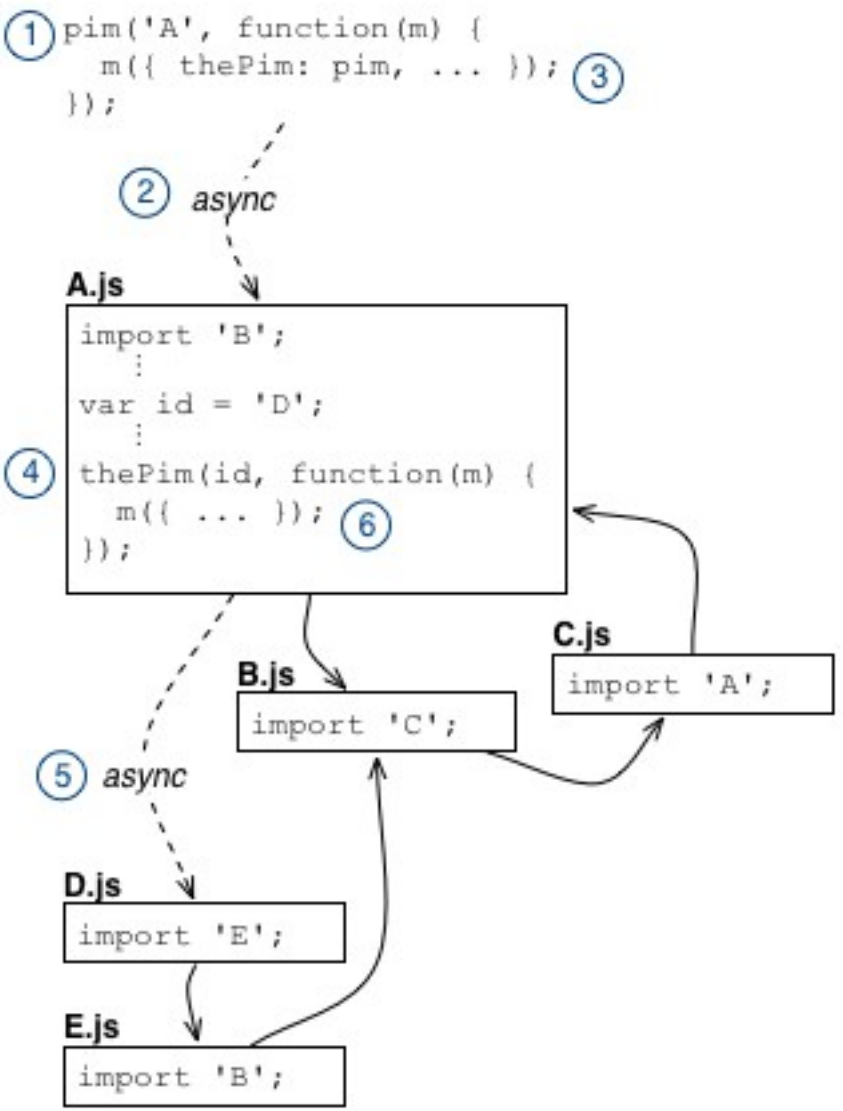
`import` is a special form

Value of `import` expr substituted at “compile” time

Entry point is some async host function

… *may* want to standardize

Implementation



Primordials

Optional parameter of module function

```
(import 'util/point')({
    x: 3, y: 4 },
    aContext)
```

Variant: Immutable module object

```
var table = [
  0.0, 0.0998334166, 0.198669331, /* ... */ ];

return {
  sin: function(x) { /* use 'table' */ },
  cos: function(x) { /* ... */ }
};
```

Variant: Alternate construction

Standardize the `pim` shown earlier?

… discussed in `modules_primordials`

Module construct in source?

```
var point = module pointey ({ x, y }) {
  return {
    getX: function() {
      /* point does not work */ },
    setX: function(x_) {
      /* pointey works */ }
  };
}
```

Packages

Simple idea

Local nickname $\rightarrow$ secure reference
secure: SHA-1 or signature