The state of the `load()`

A brief progress report on current ES module implementations

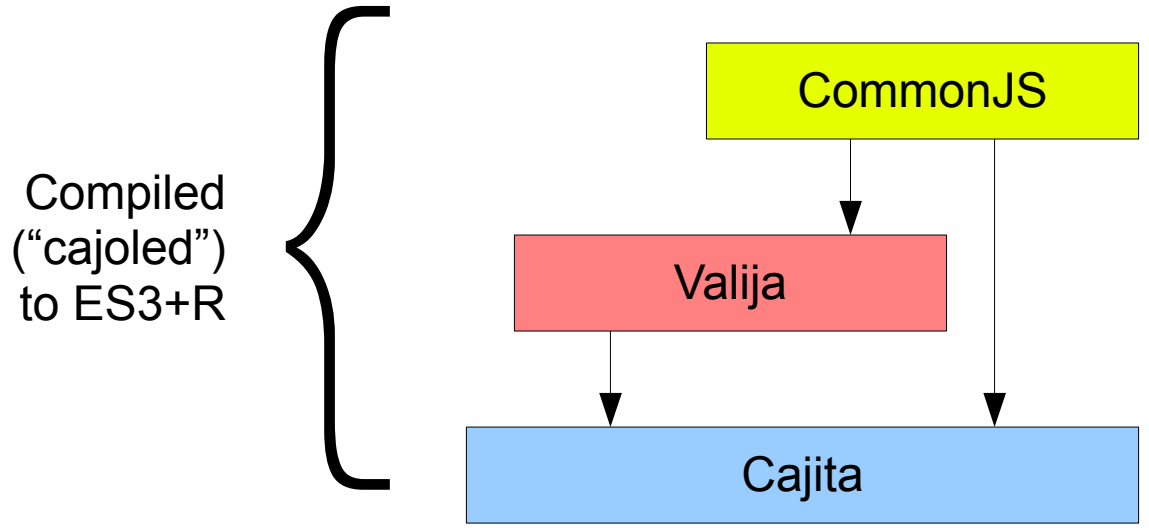
Ihab Awad <ihab.awad@gmail.com>, Google

November 6, 2009

Contents

- Quick overview of Caja
  - Cajita
  - Valija
  - CommonJS
- Modules in {Cajita, Valija, CommonJS}
  - How to use them
  - How they are implemented
- Modules in CommonJS / Narwhal

Caja



Cajita

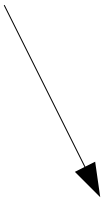
- ≅ ES3 - `this` - `prototype` + `freeze()`
  + *taming*

```
function makePoint(x, y) {
  return cajita.freeze({
    getX: function() {
      return x;
    },
    setX: function(_x) {
      if (_x > 0) { x = _x; }
    }
  });
}
```

Cajita

- Free variables captured in a “module function”

```
alert(x);
```

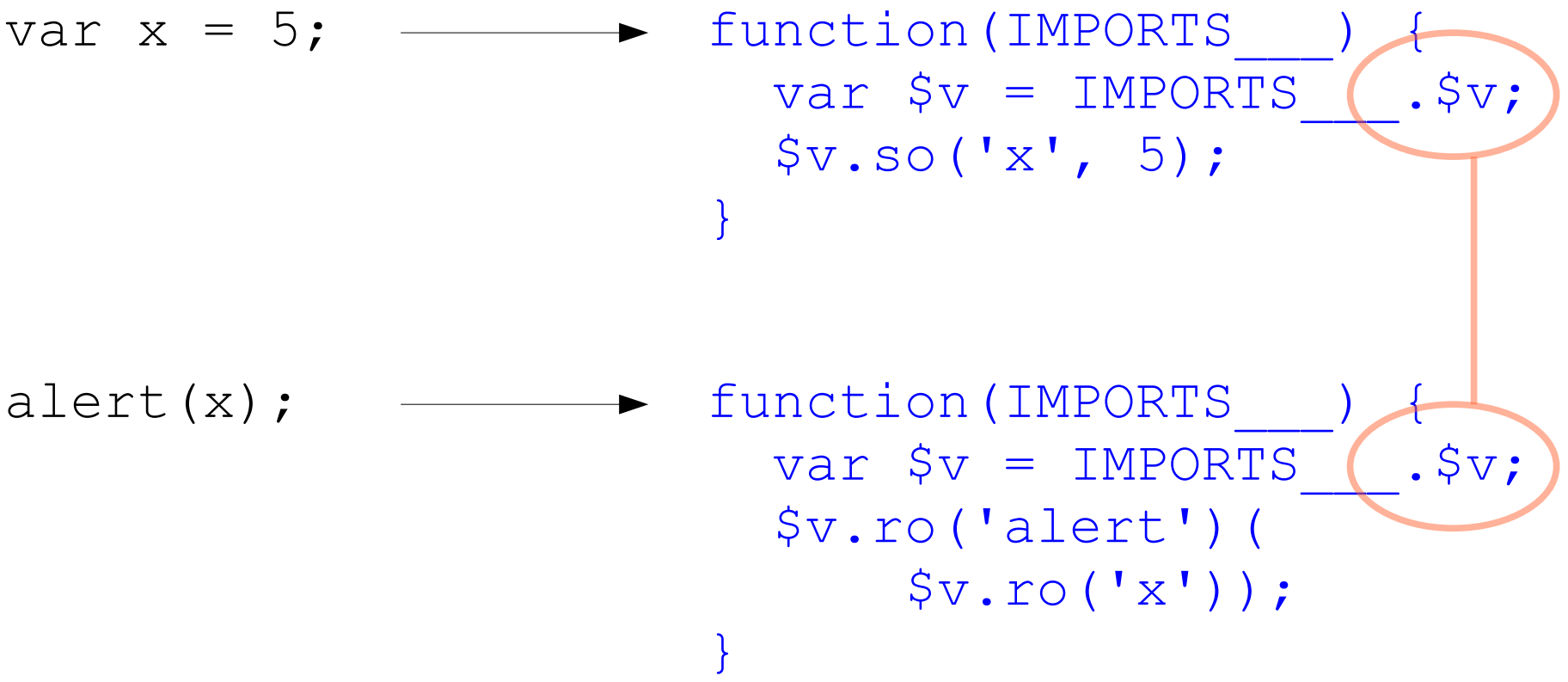


```
function(IMPORTS___) {
  IMPORTS___.alert(IMPORTS___.x);
}
```

Valija

- ≅ ES5 strict
- Compiles to Cajita code
  - Mutable prototypes, globals are emulated
- Modules may share top-level scope
  - As with independent scripts in same ES5 context
  - Emulated by the Valija sandbox, `$v`

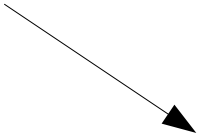
Valija


var x = 5;
function(IMPORTS___) {
var $v = IMPORTS___.$v;
$v.so('x', 5);
}
alert(x);
function(IMPORTS___) {
var $v = IMPORTS___.$v;
$v.ro('alert')(
$v.ro('x'));
}

CommonJS

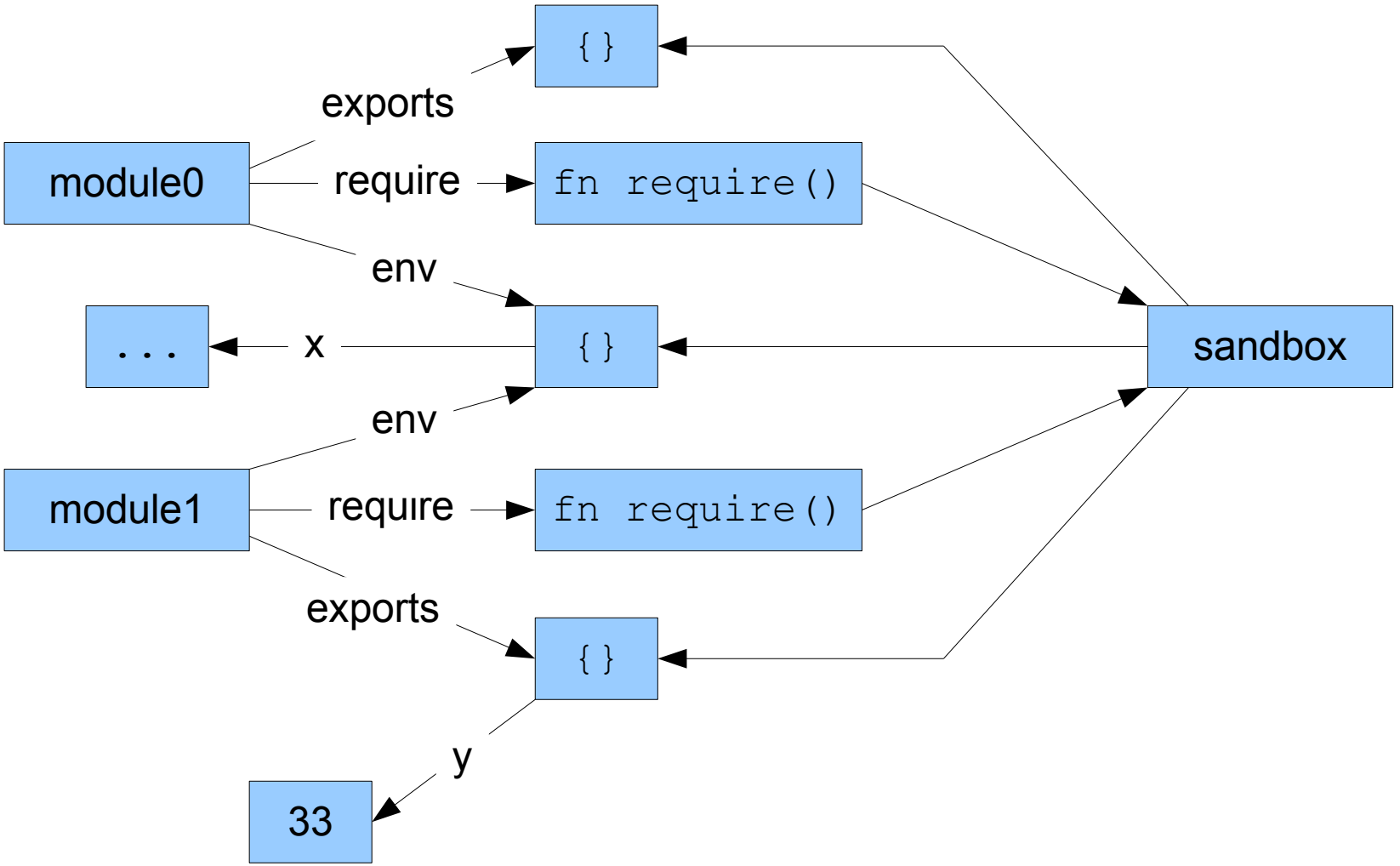
- Valija code with “sandbox” for modules

```
require('foo');
env.alert(env.x);
exports.y = 33;
```



```
function(IMPORTS___) {
  var $v = IMPORTS___.$v;
  var require = $v.ro('require');
  var env = $v.ro('env');
  var exports = $v.ro('exports');
  require('foo');
  env.alert(env.x);
  exports.y = 33;
}
```

CommonJS


{}
exports
module0
require
fn require()
env
...
x
{}
sandbox
env
module1
require
fn require()
exports
{}
y
33

Cajita module usage

Sync, arg must be a string literal

```
var mi = load('foo')({x: 3, y: 4})
```

Async, arg may be (string-valued) expression

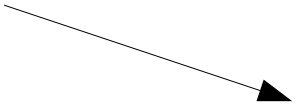
```
Q.when(load.async('f' + 'oo'), fn(m) {
 var mi = m({x: 3, y: 4});
});
```

Cajita module internals

Sync becomes annotation on module header

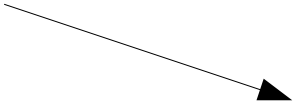
`load()` is chained so relative module IDs work

```
load('foo');
```



```
{
  instantiate: fn(IMPORTS___) {
    IMPORTS___.load('foo');
  },
  includedModules: ['foo']
}
```

```
load.async('f' + 'oo');
```



```
load.async('f' + 'oo');
```

Valija module usage

Semantics of `<script>` tag (shared globals)

In `foo.js`:

```
var x = 5;
```

In client:

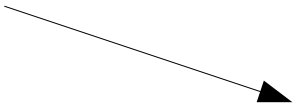
```
includeScript('foo');
alert(x);  // alerts 5

Q.when(includeScript.async('f' + 'oo'), fn(_) {
  alert(x);  // alerts 5
});
```

Valija module internals

```
includeScript('foo');
```

```
load('foo')({$v: $v});
```

```
includeScript.async('f' + 'oo');
```



```
includeScript.async('f' + 'oo');
```

CommonJS module usage

Sync, arg must be a string literal

```
var fooExports = require('foo');
```

Async, arg may be (string-valued) expression

```
Q.when(
  require.async('f' + 'oo'),
  function(fooExports) {
    // Use fooExports
  });
```

CommonJS module internals

```
require('foo');
```

```
$v.ro('require')(load('foo'));
```

```
require.async('f' + 'oo');
```

```
$v.ro('require).async(load.async('f' + 'oo'));
```

Modules in Narwhal

Core loader method

```
loader.evaluate = function (text, topId) {
  if (system.evaluate) {
    return system.evaluate(text, fileName, 1);
  } else {
    return new Function(
        "require", "exports",
        "module", "system", "print",
        text);
  }
};
```

Modules in Narwhal

Core loader method, contd.

```
var context = new
   Packages.org.mozilla.javascript.Context();

var evaluate = function (text, name, lineNo) {
  var scope = new Object();
  return context.compileFunction(
    scope,
    "function(require,exports,module,system,print)" +
    "{" +
      text +
    "}",
    ...);
};
```