Ecma/TC39/2010/048

Module scoping and linking

Sam and Dave

July 28, 2010

Internal modules

```
module MyLib {
    function log(s) { ... }
    module Util {
        export function frob(x) {
            log("frobbing " + x); // why not?
            ...
        }
    }
}
```

Internal modules, ctd.

- Within a file, lexical scope is lexical scope.
- Restricting scopes within a file is a refactoring hazard.

External modules

```
module MyLib {
    function log(x) { ... }
    module Util {
        module Even = load “even.js”;
        module Odd = load “odd.js”;
        ...
    }
    ...
}
```

External modules, ctd.

- External modules should still be linkable.
- Should avoid explicit linking sub-languages.
- External files shouldn’t be sensitive to global scope.

  ⇒ Externally **load**ed modules get *local* module graph.

External modules, ctd.

```
module ~~MyLib~~ {
    function ~~log~~(x) { ... }
    module ~~Util~~ {
        module Even = load "even.js";
        module Odd = load "odd.js";
        ...
    }
    ...
}
```

```
import Odd.odd;
...
```

```
import Even.even;
...
```

Summary

- Internal modules are lexically scoped — no harm, no foul.
- External modules can still be linked, even cyclically.
- External modules sensitive only to local module graph.

P.S.: web MRL's

- **load** "http://jquery.org/modules/jquery.js"
- **load** "./jquery.js" (relative to "__file__", so to speak)
- **load** "@harmony"