Desugaring을 이용한 JavaScript 모듈 시스템

강 성훈

Kang Seonghoon

PLRG @ KAIST

June 26, 2011

Module System for JavaScript

Traditionally there was no module system for JavaScript, and every code runs in a global scope (which one cannot fully control).

Consequently...

- JavaScript *module pattern* was emerged.
- Libraries frantically avoided frequently-used names like $.
- Subsets of JavaScript, like ADSafe, FBJS and Caja, were used to secure JavaScript from third parties.

Module System for JavaScript

Traditionally there was no module system for JavaScript, and every code runs in a global scope (which one cannot fully control). Consequently...

- JavaScript *module pattern* was emerged.
- Libraries frantically avoided frequently-used names like $.
- Subsets of JavaScript, like ADSafe, FBJS and Caja, were used to secure JavaScript from third parties.

The result: EPIC FAIL.

“Official” Module System for JavaScript

Due to much demand, JavaScript is finally going to get a module system in the next major revision.

```
module Math {
    export function sum(x, y) {
        return x + y;
    }
    export var pi = 3.141593;
}

module MathEx = require('http://example.com/mathex.js');

import Math.*;
import MathEx.gcd;
alert(sum(gcd(32, 48), pi));
```

Roads to a Formal Specification

JavaScript has been well known for its unusual semantics:

- There are strange scoping rules which allow the use of functions before their definitions.
- Modules retain this strange semantics, too.

In order to fully understand this module system we need a formal specification.

Our Work

We recently formalized the module system for JavaScript, in terms of desugaring rules from JavaScript with modules to the λ_{JS} core language:

$$
\begin{array}{l}
\mathit{desugar}[\![\mathit{stmt}\cdots]\!] = \\
\quad \textbf{let}\ (\$\text{global} = \textbf{ref}\ \{\text{“Object”}\colon \cdots, \text{“Array”}\colon \cdots, \cdots\}) \\
\quad\quad \textbf{let}\ (\$\text{Object} = (\textbf{deref}\ \$\text{global})[\text{“Object”}], \\
\quad\quad\quad \$\text{Array} = (\textbf{deref}\ \$\text{global})[\text{“Array”}], \cdots) \\
\quad\quad\quad \textbf{let}\ (\text{this} = \$\text{global}) \\
\quad\quad\quad\quad \mathit{desugar}_I[\![\mathit{stmt}\cdots]\!](\mathrm{Env}[\![\mathit{stmt}\cdots]\!]; \\
\quad\quad\quad\quad\quad \mathit{desugar}_S[\![\mathit{stmt}\cdots]\!](\mathrm{Env}[\![\mathit{stmt}\cdots]\!]; \\
\quad\quad\quad\quad\quad\quad \mathit{desugar}_R[\![\mathit{stmt}\cdots]\!](\epsilon * \mathrm{Env}[\![\mathit{stmt}\cdots]\!])))
\end{array}
$$

Future Work

- Implement and test our formalization in order to show its faithfulness; and
- Derive a practical desugaring of modules to plain JavaScript, not only to λ_{JS}