An Empirical Study on the Rewritability of the `with` Statement in JavaScript

ERC Workshop (16 January to 19 January, 2012)

Changhee Park (KAIST) Hongki Lee (KAIST) Sukyoung Ryu (KAIST)

Motivation

- Provide empirical data for `with` statements used in real-world JavaScript applications.
  - Amount
  - Usage patterns
- Check rewritability of `with` statements in real world.
  - Rewriting rules to remove `with` statements by replacing them with other statements

Introduction of the `with` statement

- Syntax and semantics in ECMAScript

Syntax

`with(`*exp*`)` *stmt* → body of the `with` statement

`with(`*obj*`)` *stmt* → `with` object

Semantics

1. Evaluate *exp* to a JavaScript object
2. Add *obj* to to the front of the scope chain
3. Evaluate *stmt*
4. Remove *obj* from the scope chain

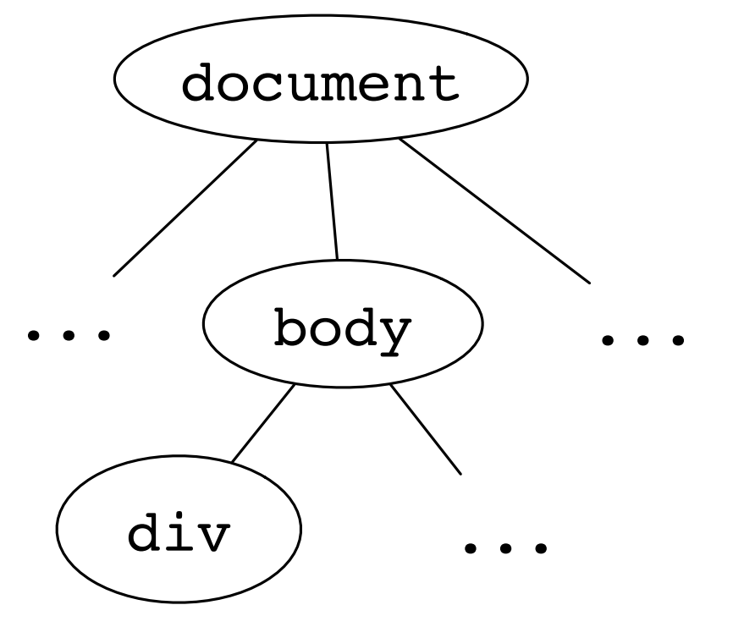
The `with` statement introduces a new scope at run time.

Good Parts

- Provide a convenient way to develop dynamically changing web contents

```
<html>
  …
  <body>
     <div> … </div>
     …
  </body>
  …
</html>
```

HTML document



DOM (Document Object Model) tree

Accessing Contents in the `div` Tag

```
document.body.children[0].style.textAlign="center";
document.body.children[0].style.fontSize=50;
```

```
with(document.body.children[0].style){
  textAlign="center";
  fontSize=50;
}
```

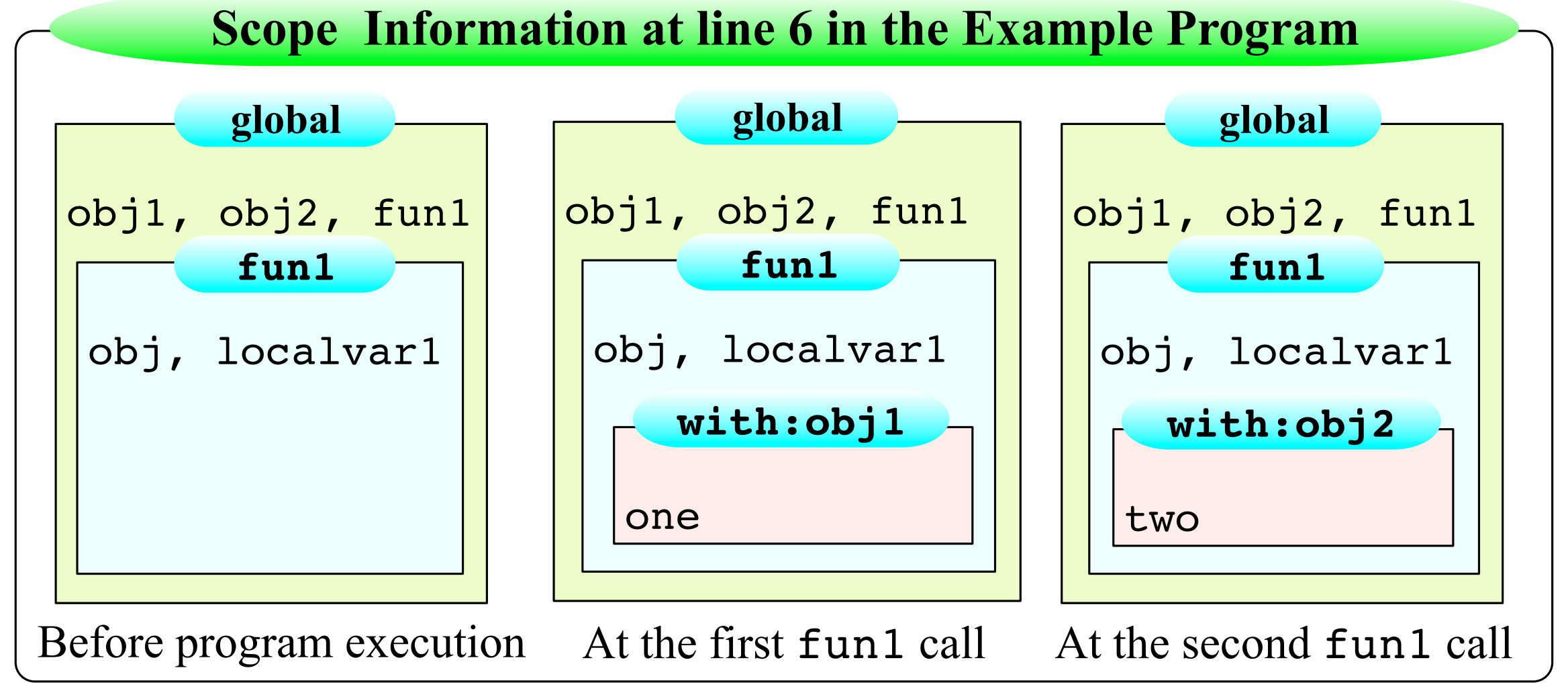
Instead of using long object accesses multiple times, the `with` statement enables users to use just field names in the body of the `with` statement.

Bad Parts

- Incur performance overheads
- Make static analysis infeasible

```
1: var obj1={one:1};
2: var obj2={two:2};
3: function fun1(obj){
4:  var localvar1=0;
5:  with(obj)
6:    localvar1=one;
7: }
8: fun1(obj1);
9: fun1(obj2);
```

Scope Information at line 6 in the Example Program



Dynamic scope introduction by the `with` statement makes identifier resolution unclear at static time.

Real-World Usage Patterns

Methodology

Tool: TracingSafari, Purdue univerity

Target: 944 world-wide most popular web sites by *alexa.com*

Data set:
- `LOADING` set
  - JavaScript code collected for 1 minute at loading time of each site
- `INTERACTION` set
  - JavaScript code collected for 2 minutes with user interactions (mouse click events)

Tool: `with` extractor using the Rhino JavaScript parser

`with` identification:
- Static `with` statements
  - Static `with` keyword detected by the Rhino parser
- Executed `with` statements
  - Actually evaluated `with` statements by the JavaScript interpreter
- Dynamic `with` statements
  - Dynamically instrumented and executed `with` statements

Amount

`LOADING` Set

`with` type	Web sites (944)	`with` counts	Unique `with`s
Static	136 (14.4%)	674	645
Executed	66 (6.9%)	855	163
Dynamic	29 (3.0%)	440	75

`INTERACTION` Set

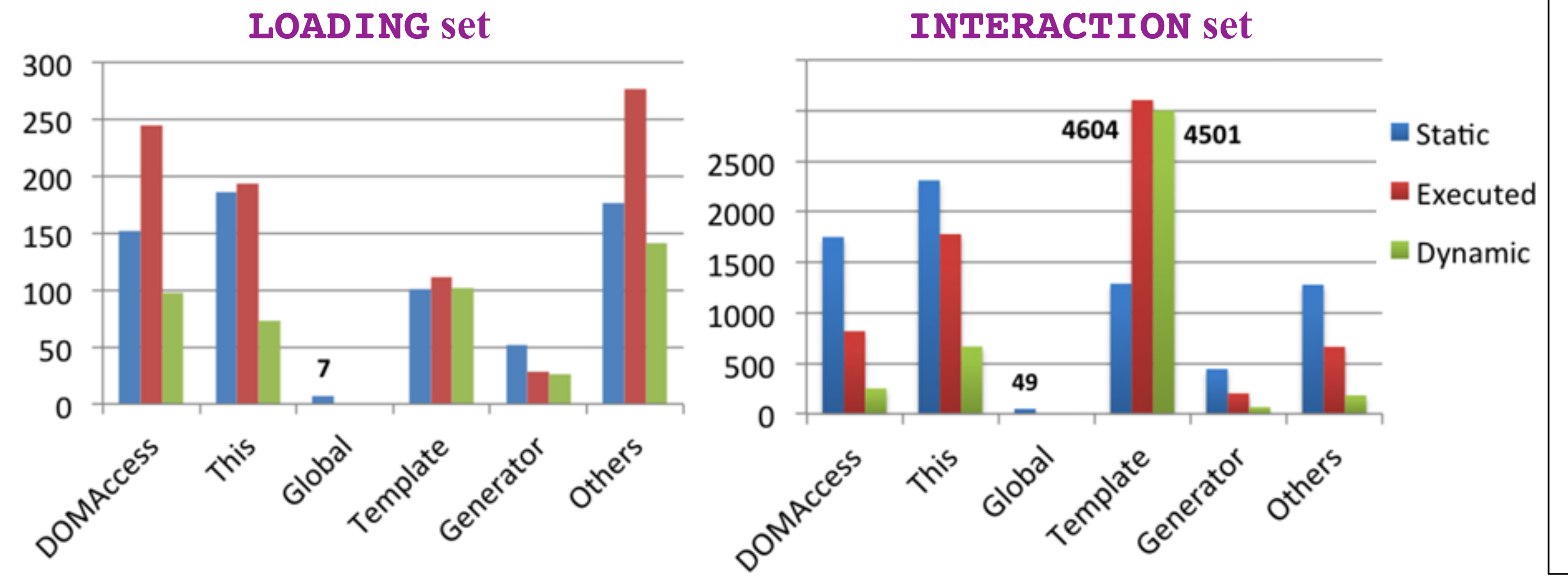
`with` type	Web sites (944)	`with` counts	Unique `with`s
Static	264 (27.9%)	7,115	1,611
Executed	147 (15.5%)	8,074	1,122
Dynamic	58 (6.1%)	5,665	696

27.9% of the top 944 sites have static occurrences of `with` statements with simple user interactions.

=> The `with` statement is being used unneglectably.

Usage Patterns

Pattern	`with` object	Description
DOMaccess	DOM elements	Access or change the values of DOM element attributes
This	`this`	Use the same naming convention between private and public properties
Global	`window`	Access the global scope with the `eval` function
Template	Template data	Process HTML templates
Generator	Any objects	Contain dynamic code generating functions
Others	Any objects	Not categorized into the above 5 patterns but used to avoid repeatedly accessing the `with` object



Examples

- This pattern

```
function simpleCons(x){
  var privateF;
  this.publicF=x;
  this.publicM=function(){
    with(this)
      privateF=publicF;
  }}             this.publicF
```

- Global pattern (ebay.com)

```
with(window)
  try { eval(_lf);
        return true;
  } catch(e) {}
```

- The Template pattern (163.com)

```
with(obj){url:"a.com",text:"b"}
  _.push('<a href="',url,
         '">',text,'</a>');
}
```

`<a href="a.com">b</a>`

Rewritability

Goal

- Defining the **`rewrite`** function

JavaScript program → `rewrite()` → Program without `with` statements

Main Idea

```
with(obj)
  … id …
```

Can we answer this question at static time??

`obj` has the `id` field?

`Yes` → `… obj.id …`

`No` → `… id …`

Rewriting the Identifier

```
with(obj)
  … id …
```

```
var $f=toObject(obj);
… ("id" in $f ? $f.id : id) …
```

Identifiers are rewritten to ternary conditional expressions.

Rewriting Rules

```
$rewrite_s[\![$with (e) s$\mid \Gamma]\!]$
  = If $\Gamma \equiv \bullet$
    Then {var $\alpha$ = toObject($rewrite_e[\![e \mid \Gamma]\!]$); $rewrite_s[\![s \mid \langle \alpha, \bullet, \text{False}\rangle]\!]$}
    Else Let $\Gamma = \langle \phi, \varphi, \beta \rangle$   where $\alpha \notin \phi$
         {var $\alpha$ = toObject($rewrite_e[\![e \mid \Gamma]\!]$); $rewrite_s[\![s \mid \langle \phi\, \alpha, \varphi, \beta\rangle]\!]$}
$rewrite_i[\![id \mid \Gamma]\!]$
  = If $\Gamma \equiv \bullet$ Then $id$
    Else Let $\Gamma = \langle \phi\, \alpha, \varphi, \beta \rangle$
         If $id \in \varphi$ Then $id$
         Else If $\phi \equiv \bullet$ Then ("$id$" in $\alpha$ ? $\alpha.id$ : $id$)
              Else ("$id$" in $\alpha$ ? $\alpha.id$ : $rewrite_i[\![id \mid \langle \phi, \varphi, \beta\rangle]\!]$)
$rewrite_e[\![lhs \odot e \mid \Gamma]\!]$
  = If $\Gamma \equiv \bullet$ Then $rewrite_l[\![lhs \mid \Gamma]\!] \odot rewrite_e[\![e \mid \Gamma]\!]$
    Else Let $\Gamma = \langle \phi\, \alpha, \varphi, \beta \rangle$
         If $lhs \equiv id$
         Then If $lhs \in \varphi$ Then $lhs \odot rewrite_e[\![e \mid \Gamma]\!]$
              Else If $\phi \equiv \bullet$
                   Then If $\beta \equiv$ True
                        Then ("$lhs$" in $\alpha$ ? $\alpha.lhs \odot e$ : $lhs \odot e$)
                        Else ("$lhs$" in $\alpha$ ? $\alpha.lhs \odot rewrite_e[\![e \mid \Gamma]\!]$
                                                  : $lhs \odot rewrite_e[\![e \mid \Gamma]\!]$)
                   Else If $\beta \equiv$ True
                        Then ("$lhs$" in $\alpha$
                                 ? $\alpha.lhs \odot e$
                                 : $rewrite_e[\![lhs \odot e \mid \langle \phi, \varphi, \beta\rangle]\!]$)
                        Else ("$lhs$" in $\alpha$
                                 ? $\alpha.lhs \odot rewrite_e[\![e \mid \Gamma]\!]$
                                 : $rewrite_e[\![lhs \odot rewrite_e[\![e \mid \Gamma]\!] \mid$
                                       $\langle \phi, \varphi, \text{True}\rangle]\!]$)
         Else $rewrite_l[\![lhs \mid \Gamma]\!] \odot rewrite_e[\![e \mid \Gamma]\!]$
```

- Implemented in Java : http://plrg.kaist.ac.kr/research/software

Rewriting the Assignment

```
with(obj)
  x=3;
```

Incorrect rewriting →

```
var $f=toObject(obj);
("x" in $f ? $f.x : x)=3;
```

ReferenceError **exception!!**

Correct rewriting ↓

```
var $f=toObject(obj);
("x" in $f ? $f.x=3 : x=3);
```

By placing the assignment in each branch of the conditional, we can preserve the original semantics of the `with` statement.

Rewritability Check

Pattern	Rewritability	Rewriting
DOMaccess	Yes	By the `rewrite` function
This	Yes	By the `rewrite` function
Global	Yes	By ECMAScript 5
Template	Yes	By the `rewrite` function
Generator	No	Generally not possible due to the dynamic code generating functions in the `with` statement
Others	Yes	By the `rewrite` function

We can rewrite all static `with` statements in all patterns except for Generator pattern.

=> 92% of static `with` statements in the `LOADING` set and 93% in the `INTERACTION` set are rewritable.