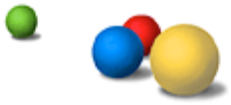
Securing EcmaScript 5: Adventures in Strategic Compromise

Mark S. Miller and the Cajadores

Thanks to many on EcmaScript committee



Overview

My "Expression Security Constraints" talk

- The *What* and *Why* of object-capabilities (ocaps)

This talk

- The *How* of doing ocaps in JavaScript

Why JavaScript?

- Dynamics of Language Adoption

Improving JavaScript in Stages

Improving JavaScript in Stages

EcmaScript 3:

One of the hardest oo languages to secure.

Complex server-side translator. Runtime overhead.

EcmaScript 5:

One of the easiest oo languages to secure.

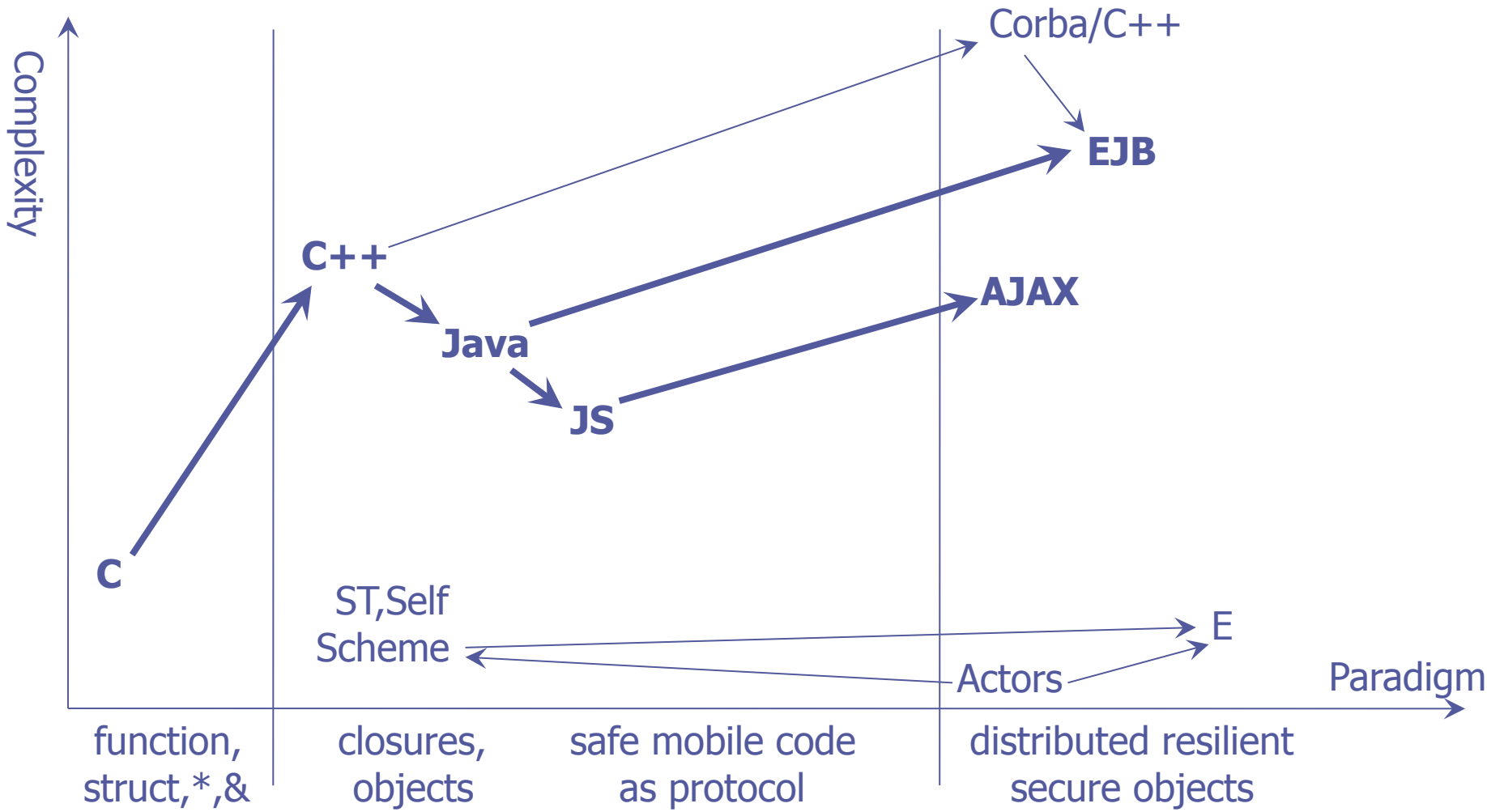
Simple client-side init and verifier. No runtime overhead.

Approx 3K download compressed.

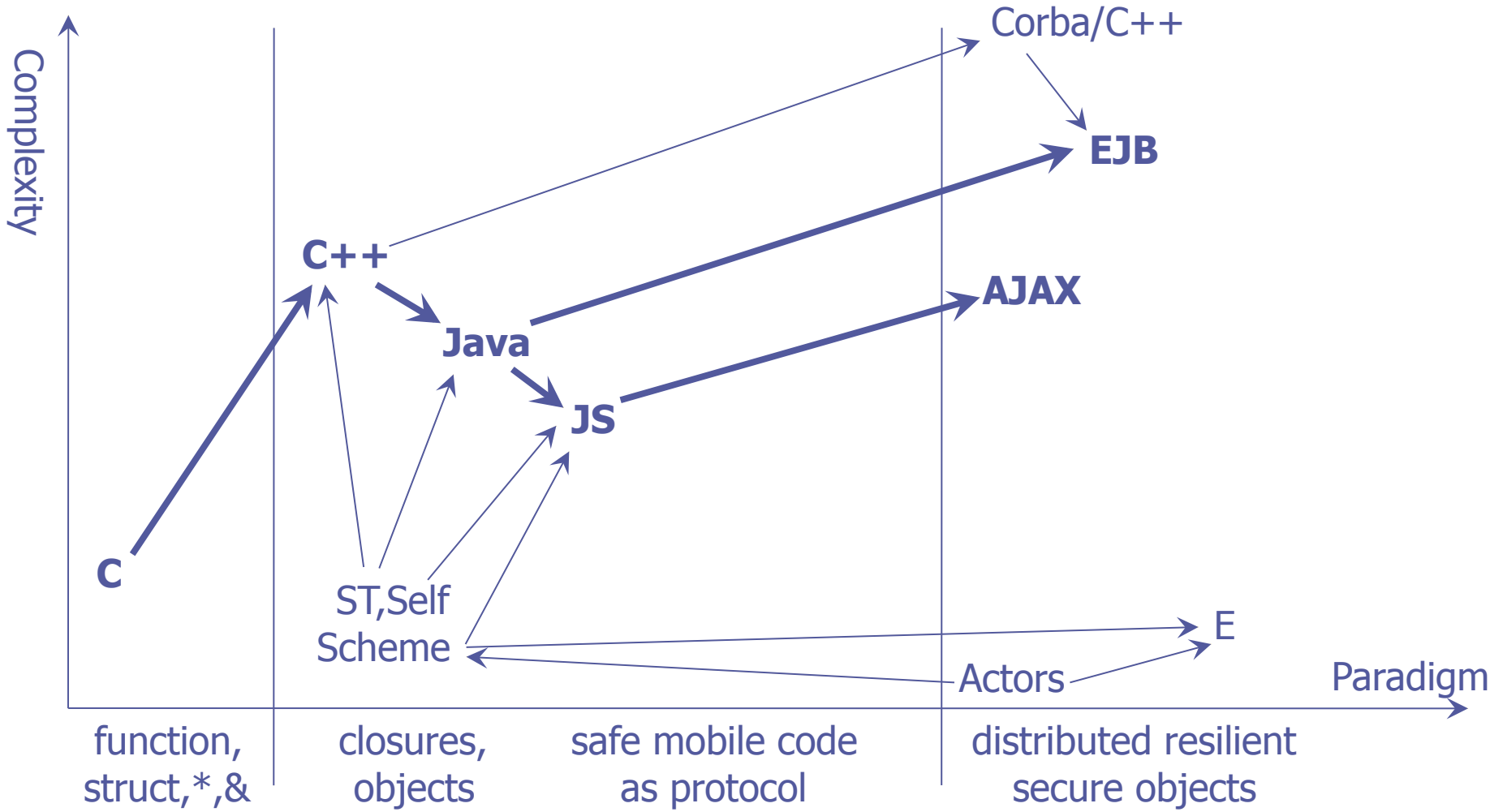
Improving JavaScript in Stages

Encapsulated objects	EcmaScript 3 (ES3)
Tamper proof objects	EcmaScript 5 (ES5)
Defensive objects	ES5 strict mode (ES5/strict)
Future objects on old browsers	ES5/3, SES5/3 (Caja)
Confining **offensive** objects	Secure EcmaScript (SES) on ES5
Distributed objects	Distributed Resilient SES (Dr. SES)
Robust objects	SESLint?

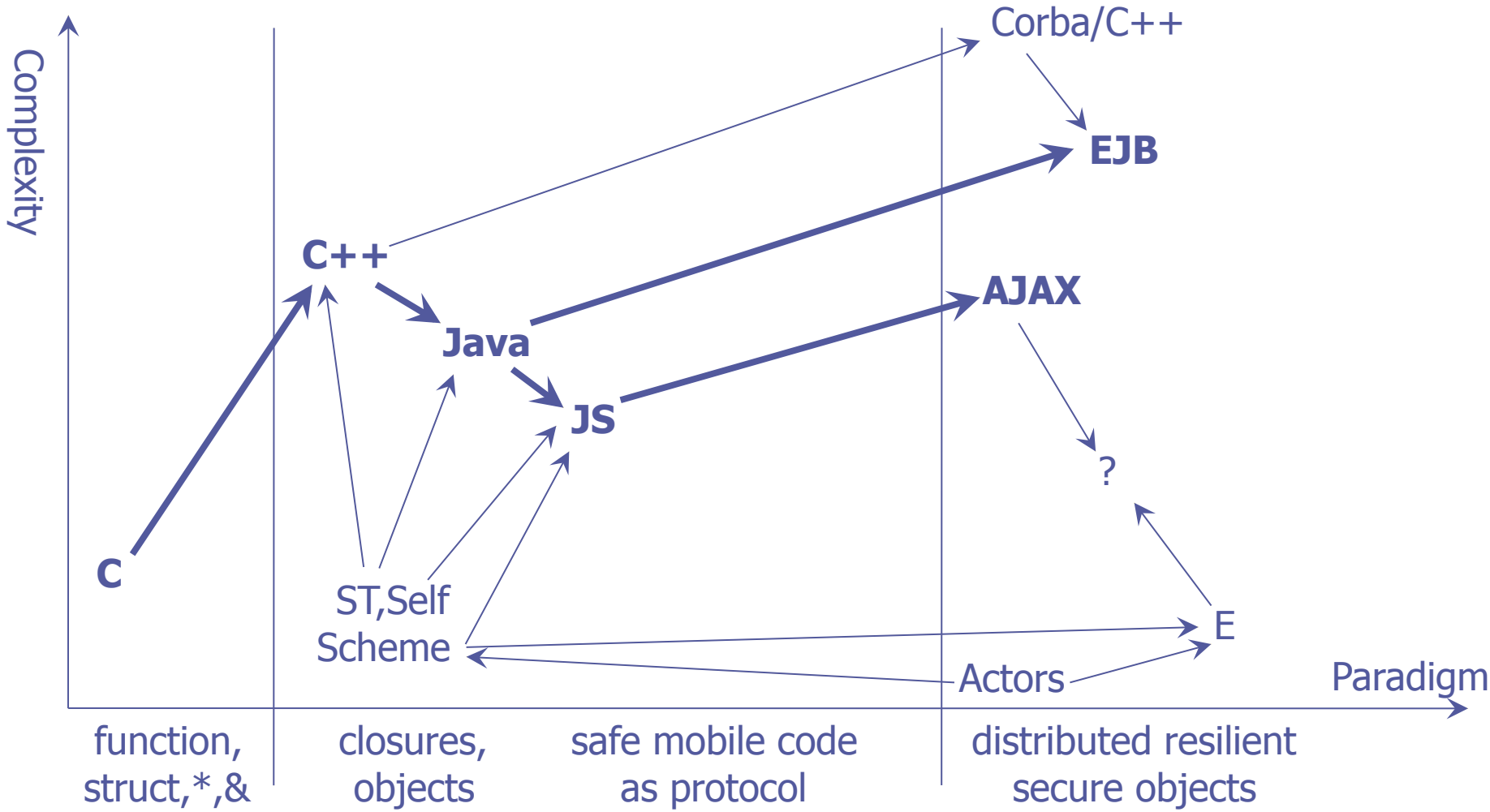
Adoption paths & progress



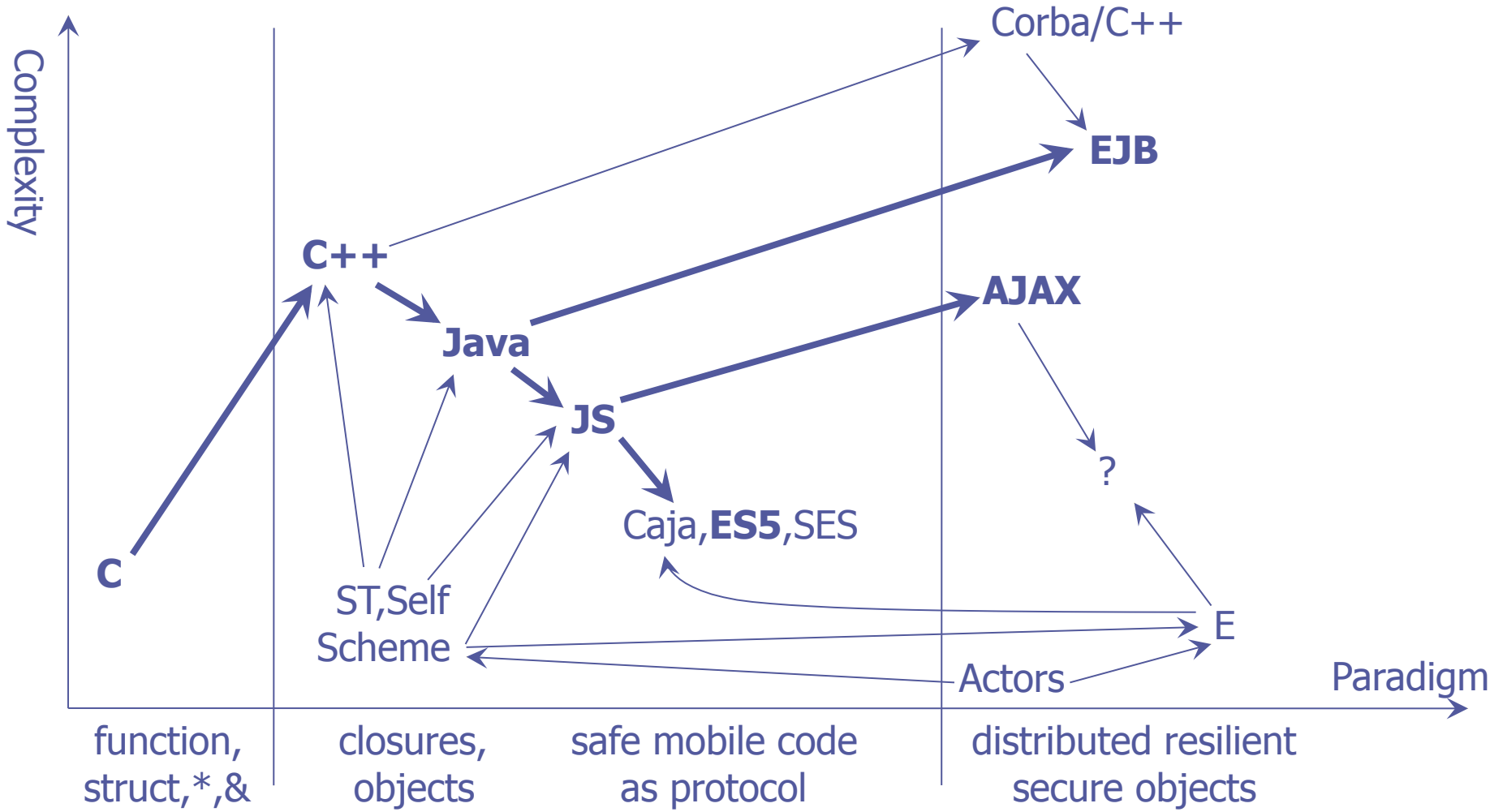
Legacy-free scouts lead way



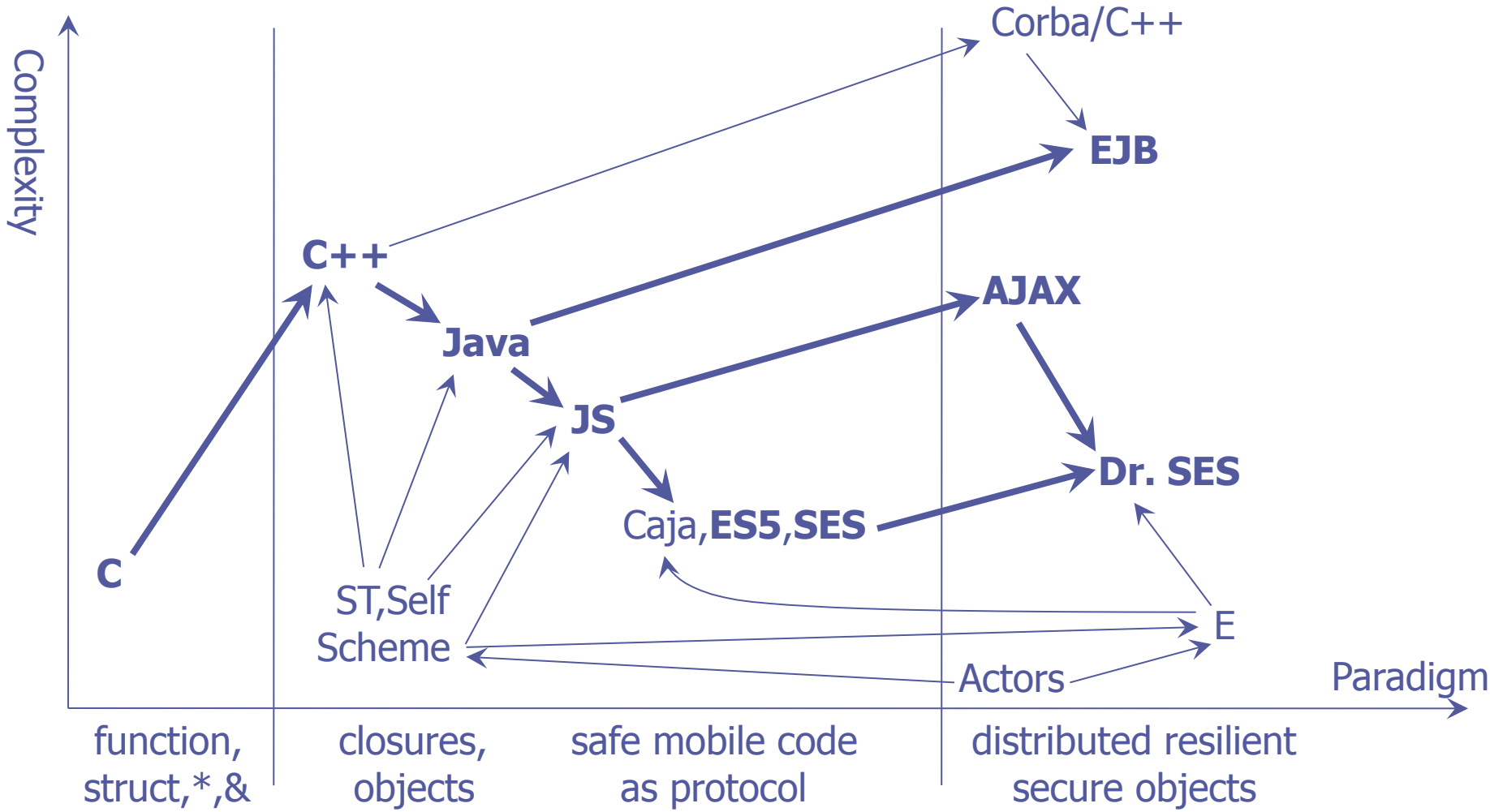
Too big a jump



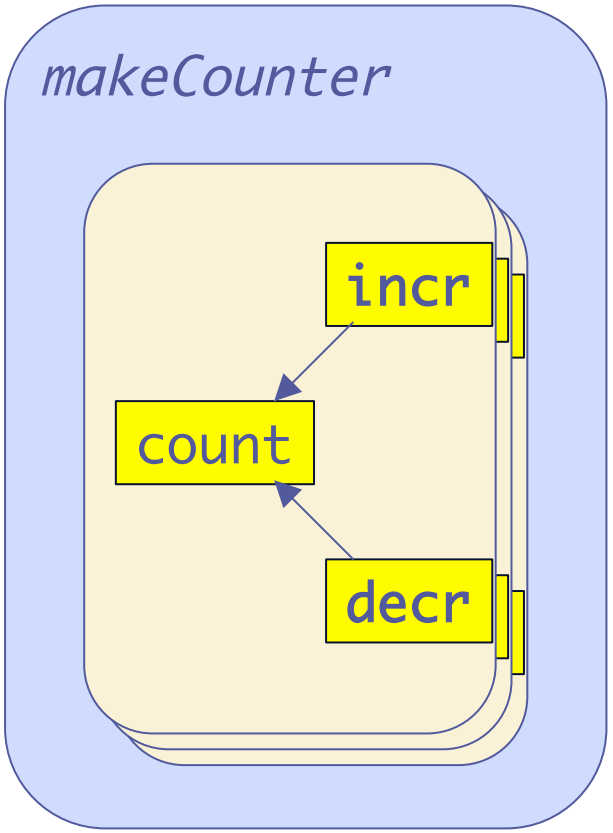
Today


Complexity
Corba/C++
EJB
C++
AJAX
Java
JS
?
Caja,**ES5**,SES
C
ST,Self
Scheme
E
Actors
Paradigm
function,
struct,*,&
closures,
objects
safe mobile code
as protocol
distributed resilient
secure objects

Tomorrow?

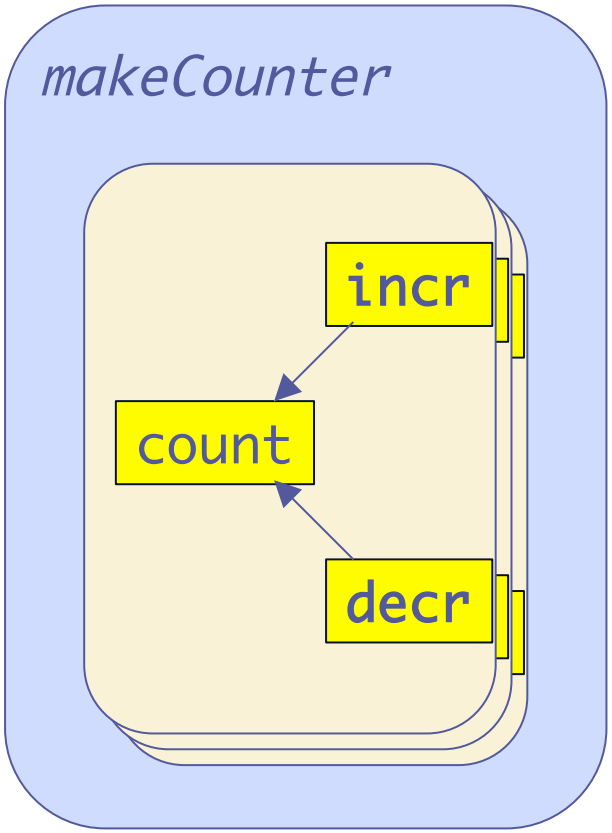

Complexity
Corba/C++
EJB
C++
AJAX
Java
JS
Dr. SES
Caja,ES5,SES
C
ST,Self
Scheme
E
Actors
Paradigm
function,
struct,*,&
closures,
objects
safe mobile code
as protocol
distributed resilient
secure objects

Encapsulated objects in ES3




```
function makeCounter() {
    var count = 0;
    return {
        incr: function() { return ++count; },
        decr: function() { return --count; }
    };
}
```

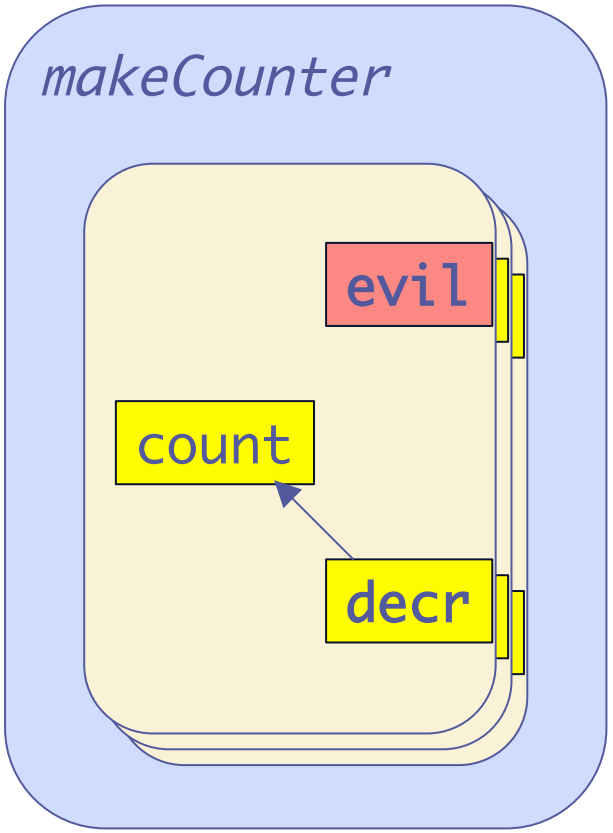
Encapsulated objects in ES3




```
function makeCounter() {
    var count = 0;
    return {
        incr: function() { return ++count; },
        decr: function() { return --count; }
    };
}
```

A *record* of *closures* hiding *state*
is a fine representation of an
object of *methods* hiding *instance vars*

Encapsulated objects in ES3




```
function makeCounter() {
    var count = 0;
    return {
        incr: function() { return ++count; },
        decr: function() { return --count; }
    };
}
```

Indefensible:

```
var counter = makeCounter();
bob(counter); carol(counter);
```

Bob says:

```
counter.incr = function evil(){…};
```

Robustness impossible in ES3

Objects necessarily mutable (monkey patching)

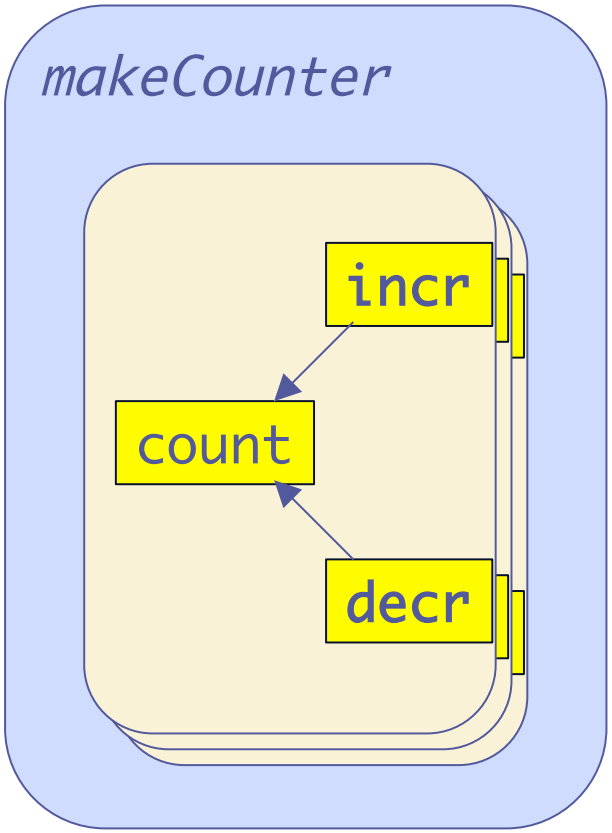
Not statically scoped – repaired by ES5

```
(function n() {…x…})                  // named function exprs
try{throw fn;}catch(f){f();…x…}       // thrown function
Object = Date; …{}…                   // "as if by"
```

Not statically scoped – repaired by ES5/strict

```
with (o) {…x…}                        // attractive but botched
delete x;                             // dynamic deletion
eval(str); …x…                        // eval exports binding
```

Tamper-proof objects in ES5




```
function makeCounter() {
    var count = 0;
    return Object.freeze({
        incr: function() { return ++count; },
        decr: function() { return --count; }
    });
}
```

Bob's attack fails:

```
counter.incr = function evil(){…};
```

Encapsulation Leaks in non-strict ES5

```
function doSomething(ifBobKnows, passwd) {
    if (ifBobKnows() === passwd) {
        //… do something with passwd
    }
}
```

Encapsulation Leaks in non-strict ES5

```
function doSomething(ifBobKnows, passwd) {
    if (ifBobKnows() === passwd) {
        //… do something with passwd
    }
}
```

Bob says:

```
var stash;
function ifBobKnows() {
    stash = arguments.caller.arguments[1];
    return arguments.caller.arguments[1] = badPasswd;
}
```

Encapsulation & Scope in ES5/strict

```
"use strict";
function doSomething(ifBobKnows, passwd) {
    if (ifBobKnows() === passwd) {
        //… do something with passwd
    }
}
```

Bob's attack fails:

```
return arguments.caller.arguments[1] = badPasswd;
```

Parameters not joined to arguments.

Encapsulation & Scope in ES5/strict

```
"use strict";
function doSomething(ifBobKnows, passwd) {
    if (ifBobKnows() === passwd) {
        //… do something with passwd
    }
}
```

Bob's attack fails:

```
return arguments.caller.arguments[1] = badPasswd;
```

Poison pills.

Encapsulation & Scope in ES5/strict

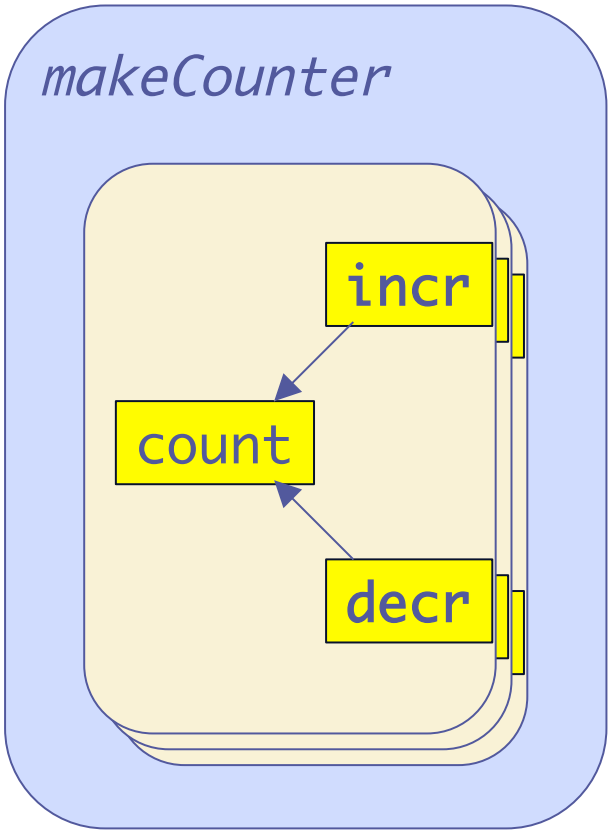
```
"use strict";
function doSomething(ifBobKnows, passwd) {
    if (ifBobKnows() === passwd) {
        //… do something with passwd
    }
}
```

Bob's attack fails:

```
return arguments.caller.arguments[1] = badPasswd;
```

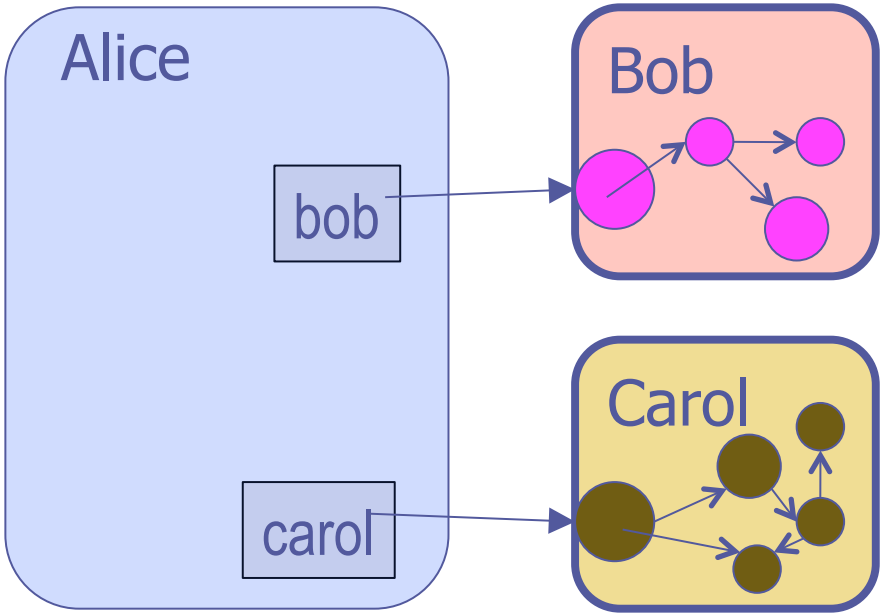
Even non-strict "**.caller**" can't reveal a strict caller.

Defensive objects in ES5/strict




```
"use strict";
function makeCounter() {
    var count = 0;
    return Object.freeze({
        incr: function() { return ++count; },
        decr: function() { return --count; }
    });
}
```

Confining **offensive** objects in SES



Alice says:

```
var bobSrc = //site B
var carolSrc = //site C
var bob = safeEval(bobSrc);
var carol = safeEval(carolSrc);
```

Bob and Carol are ***confined***.

Only Alice controls how they can interact or get more connected.

The Mashup problem: Code as Media

```
<html> <head> <title>Basic Mashup</title> <script>
   function animate(id) {
      var element = document.getElementById(id);
      var textNode = element.childNodes[0];
      var text = textNode.data;
      var reverse = false;
      element.onclick = function() { reverse = !reverse; };
      setInterval(function() {
         textNode.data = text = reverse ? text.substring(1) + text[0]
                  : text[text.length-1] + text.substring(0, text.length-1);
      }, 100);
   }
</script> </head> <body onload="animate('target')">
   <pre id="target">Hello Programmable World!  </pre>
</body> </html>
```

caja-corkboard.appspot.com

Caja Corkboard Demo

grammable World! Hello Pro

— *erights@google.com, 2010-10-04 13:30:40.185506*

Edit Delete

(Error contacting Caja service)

— *kpreid.switchb.org, 2010-08-22 12:26:41.953037*

View Source

Greetings from Rosetta Code!

Not just a <marquee>:

World! Hello

— *kpreid.switchb.org, 2010-08-13 19:06:55.712467*

View Source

Cajoling-of-URLs test: you should see 2 links to google.com and 2 images.

Static	Dynamic
Link	Link
	

— *kpreid.switchb.org, 2010-08-13 00:27:22.459179*

View Source

Testing 123.

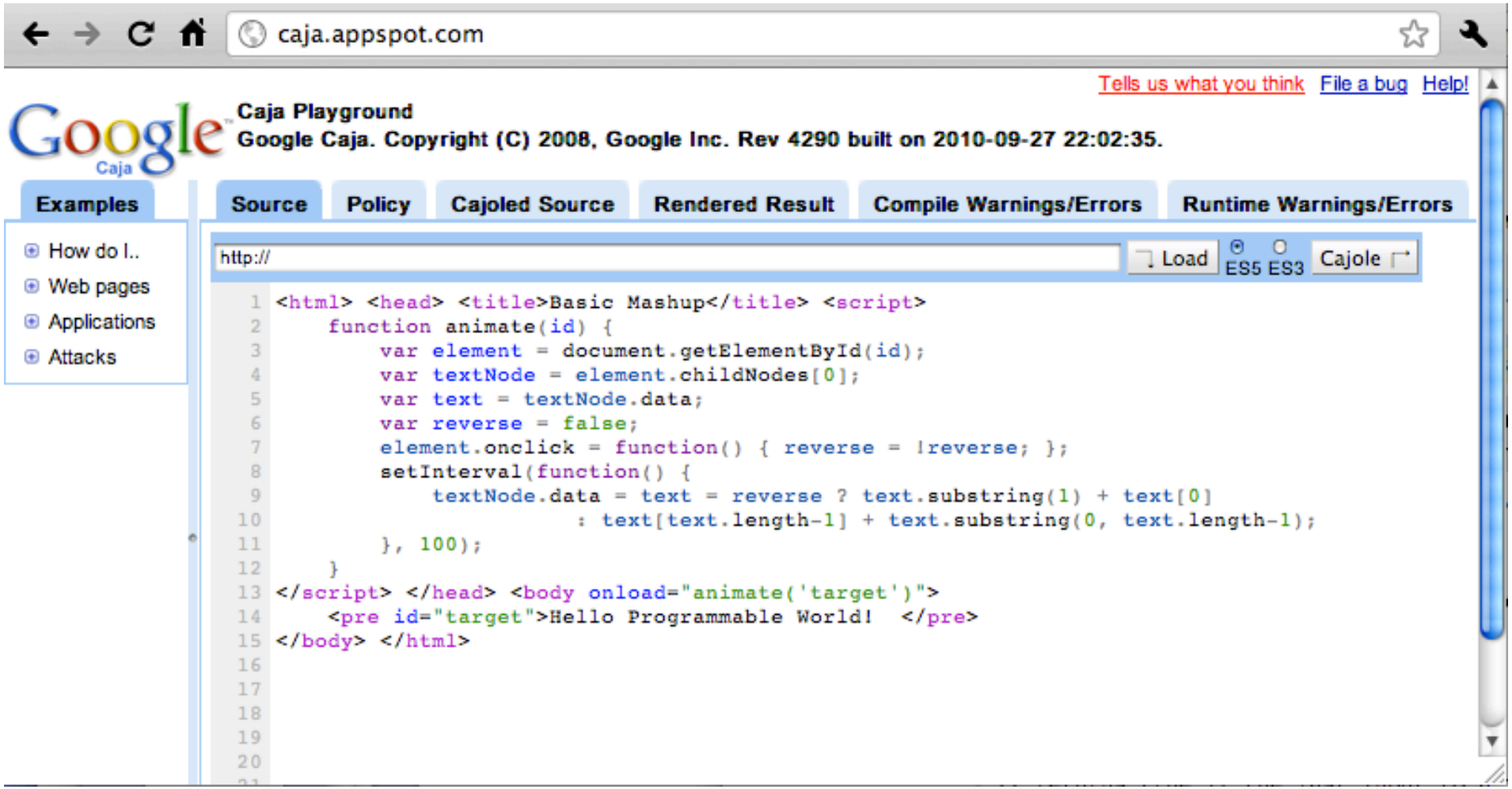
— *kpreid.switchb.org, 2010-08-10 22:21:44.542621*

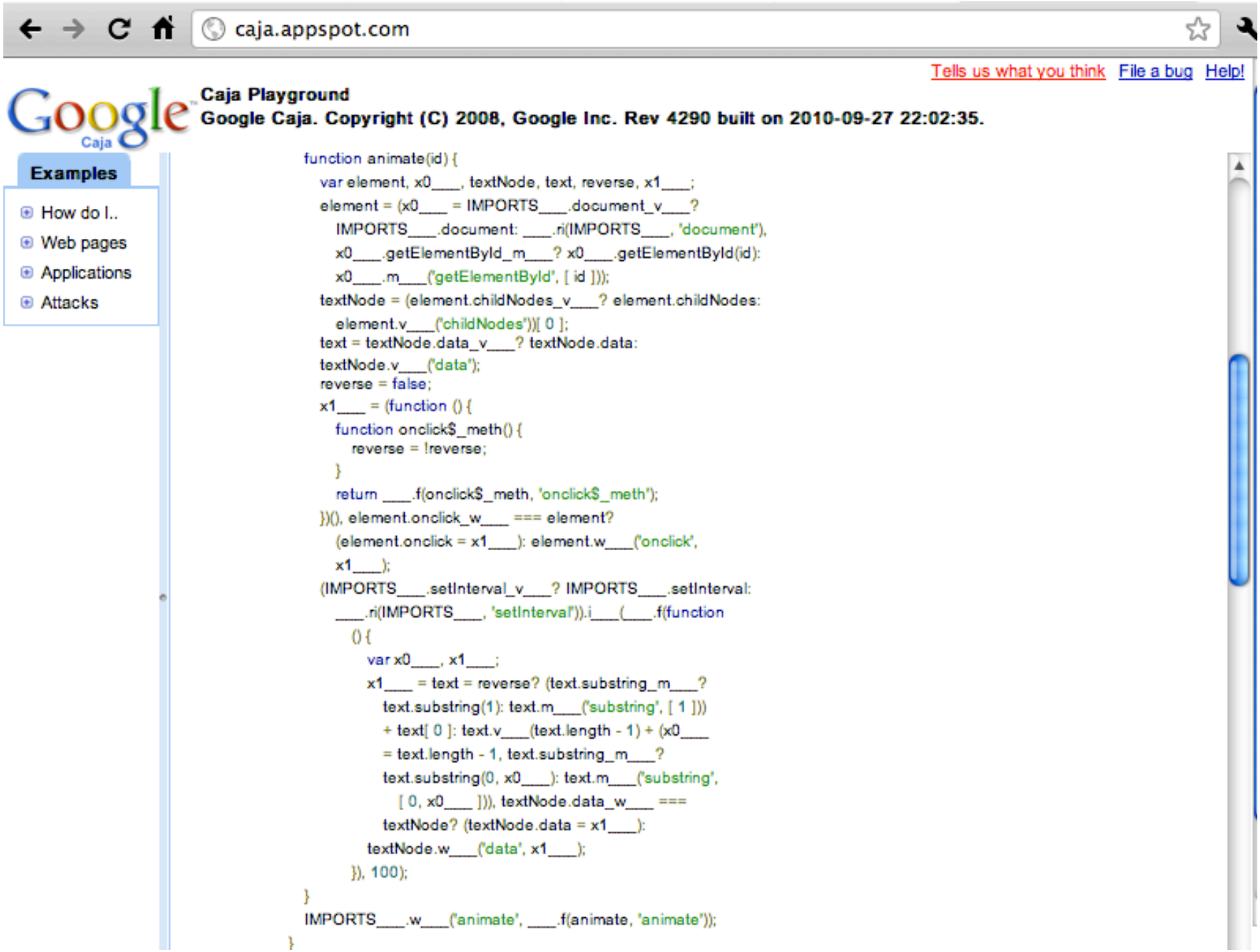
View Source

— *kpreid.switchb.org, 2010-07-24 00:43:10.844801*

View Source

Future objects on old browsers




caja.appspot.com
Tells us what you think
File a bug
Help!
Google Caja
Caja Playground
Google Caja. Copyright (C) 2008, Google Inc. Rev 4290 built on 2010-09-27 22:02:35.
Examples
How do I..
Web pages
Applications
Attacks
function animate(id) {
var element, x0___, textNode, text, reverse, x1___;
element = (x0___ = IMPORTS___.document_v___?
IMPORTS___.document: ___.ri(IMPORTS___, 'document'),
x0___.getElementById_m___? x0___.getElementById(id):
x0___.m___('getElementById', [id]));
textNode = (element.childNodes_v___? element.childNodes:
element.v___('childNodes'))[0];
text = textNode.data_v___? textNode.data:
textNode.v___('data');
reverse = false;
x1___ = (function () {
function onclick$_meth() {
reverse = !reverse;
}
return ___.f(onclick$_meth, 'onclick$_meth');
})(), element.onclick_w___ === element?
(element.onclick = x1___): element.w___('onclick',
x1___);
(IMPORTS___.setInterval_v___? IMPORTS___.setInterval:
___.ri(IMPORTS___, 'setInterval')).i___(___.f(function
() {
var x0___, x1___;
x1___ = text = reverse? (text.substring_m___?
text.substring(1): text.m___('substring', [1]))
+ text[0]: text.v___(text.length - 1) + (x0___
= text.length - 1, text.substring_m___?
text.substring(0, x0___): text.m___('substring',
[0, x0___])), textNode.data_w___ ===
textNode? (textNode.data = x1___):
textNode.w___('data', x1___);
}), 100);
}
IMPORTS___.w___('animate', ___.f(animate, 'animate'));
}

Turning ES5/strict into SES

- Monkey patch away bad non-std behaviors
- Remove non-whitelisted primordials
- Install leaky **WeakMap** emulation
- Make virtual global **root**
- Freeze whitelisted global variables
- Replace **eval** & **Function** with safe alternatives
- Freeze accessible primordials

Monkey patch away bad non-std behaviors

Secure ES5 reality, not just spec.

Ch16 exemptions destroy security reasoning.

- /x/.test('axb');

true

- RegExp.leftContext;

a

- new RegExp('(.|\r|\n)*','').exec();

axb,b

Monkey patch away bad non-std behaviors

Specified primordials replaceable, whew!

```
var unsafeExec = RegExp.prototype.exec;
RegExp.prototype.exec = function(specimen) {
    return unsafeExec.call(this, String(specimen));
}
var unsafeTest = …
```

Monkey patch away bad non-std behaviors

Unspecified primordials unspecified, darn!

- delete RegExp.leftContext;

false

- 'leftContext' in RegExp;

true

Allowed by Ch16 exemptions

Monkey patch away bad non-std behaviors

Most specified globals replaceable, whew!

```
var UnsafeRegExp = RegExp;
RegExp = function(pattern, flags) {
    return UnsafeRegExp(pattern, flags);
}
RegExp.prototype = UnsafeRegExp.prototype;
RegExp.prototype.constructor = RegExp;
```

Remove non-whitelisted primordials

Object.getOwnPropertyNames(o) → string[]
Object.getPrototypeOf(o) → o | null

```
var whitelist = {
  "escape": true,                    // ES5 Appendix B de-facto
  "unescape": true,                  // ES5 Appendix B de-facto
  "Object": {
    "prototype": {
      "constructor": "*",
      "toString": "*",
      "toLocaleString": "*",
      "valueOf": true,
      "hasOwnProperty": true,
      "isPrototypeOf": true,
      "propertyIsEnumerable": true
    },
    "getPropertyDescriptor": true,   // ES-Harmony proposal
    "getPropertyNames": true,        // ES-Harmony proposal
    "identical": true,               // ES-Harmony strawman
    "getPrototypeOf": true,
    "getOwnPropertyDescriptor": true,
    "getOwnPropertyNames": true,
```

Remove non-whitelisted primordials…

… or die trying.

- delete Object.caller;

false

- 'caller' in Object;

true

The trivial secureability of ES5 is easily lost.

A suggestion for ES5 implementers:

Please make all non-std properties configurable (deletable).

Install leaky **WeakMap** emulation

Too kludgy to explain in this talk.

A suggestion for ES5 implementers:

Please implement the ES-Harmony ***WeakMap*** *proposal.*

Make virtual global **root**

```
var root = Object.create(null, {
    Object: {value: Object},
    Array: {value: Array},
    NaN: {value: NaN},
    //…
});
```

Freeze whitelisted global variables

```
Object.defineProperties(global, {
    Object: {value: Object},
    //…
});
```

Global object as a whole remains unfrozen.

If I only had a parser

```
var unsafeEval = eval;
eval = function(src) {
    src = ‘ “use strict”; ’ + String(src);
    var ast = parse(src);
    freevars(ast).forEach(function(name) {
        if (!(name in root)) { throw …; }
    });
    return unsafeEval(
        ‘(function() { return (’ + src + ‘\n); })’
    ).call(root);
}
```

Replace **eval** with confining **eval**

```
var unsafeEval = eval;
eval = function(src) {
    src = verifyStrictExpression(src);
    return unsafeEval(
        ‘(function() {’ +
        ‘    with (this) {’ +              // poison forbidden globals
        ‘        return function() {’ + // root as “top level” this
        ‘            “use strict”;’ +       // enforces lexical scoping
        ‘            return (’ + src + ‘\n);’ +
        ‘};}})').call(makeScope(src)).call(root);
}
```

Replace **eval** with confining **eval**

```
var unsafeEval = eval;
eval = function(src) {
    src = verifyStrictExpression(src);
    return unsafeEval(
        '(function() {' +
        '    with (this) {' +              // poison forbidden globals
        '        return function() {' + // root as "top level" this
        '            "use strict";' +       // enforces lexical scoping
        '            return (' + src + '\n);' +
        '};}})').call(makeScope(src)).call(root);
}
```

Syntax verification “without” parsing

```
function verifyStrictExpression(src) {
    src = String(src);
    UnsafeFunction(‘ “use strict”;’ + src);
    try {
        UnsafeFunction(‘ “use strict”; (’ + src + ‘\n);’);
    } catch (ex) {
        return ‘(function() {’ + src + ‘\n}).call(this)’;
    }
    return src;
}
```

Poison All Possible Globals

```
function makeScope(src) {
    var scope = Object.create(root), a, ID = /???/gm;
    while (a = ID.exec(src)) {
        if (!(a[0] in scope)) {
            Object.defineProperty(scope, a[0], { get: function() { throw …; });
        }
    }
    return Object.freeze(scope);
}
```

~~Poison All Possible Globals~~

A suggestion for ES5 implementers:

Please expose a parser API.

(SES only)
Freeze accessible primordials

def(root); // **def**ine **def**ensive objects

Recursively freeze by property and prototype traversal.

Object.getOwnPropertyNames(o) → string[]
Object.getPrototypeOf(o) → o | null
Object.freeze(o) → o

Script code *vs.* **eval** code

```
<script src="ses.js"></script>
<script src="domado.js"></script>
<script>
   var bobSrc = … // untrusted code from somewhere
   eval(bobSrc)({document: attenuate(document), …});
</script>
```

eval code is confined. Script code is privileged.

- global vars in scope.

Script code *vs.* **eval** code

```
<script src="ses.js"></script>
<script src="domado.js"></script>
<script>
    var bobSrc = … // untrusted code from somewhere
    eval(bobSrc)({document: attenuate(document), …});
</script>
```

eval code is confined. Script code is privileged.

global vars in scope.

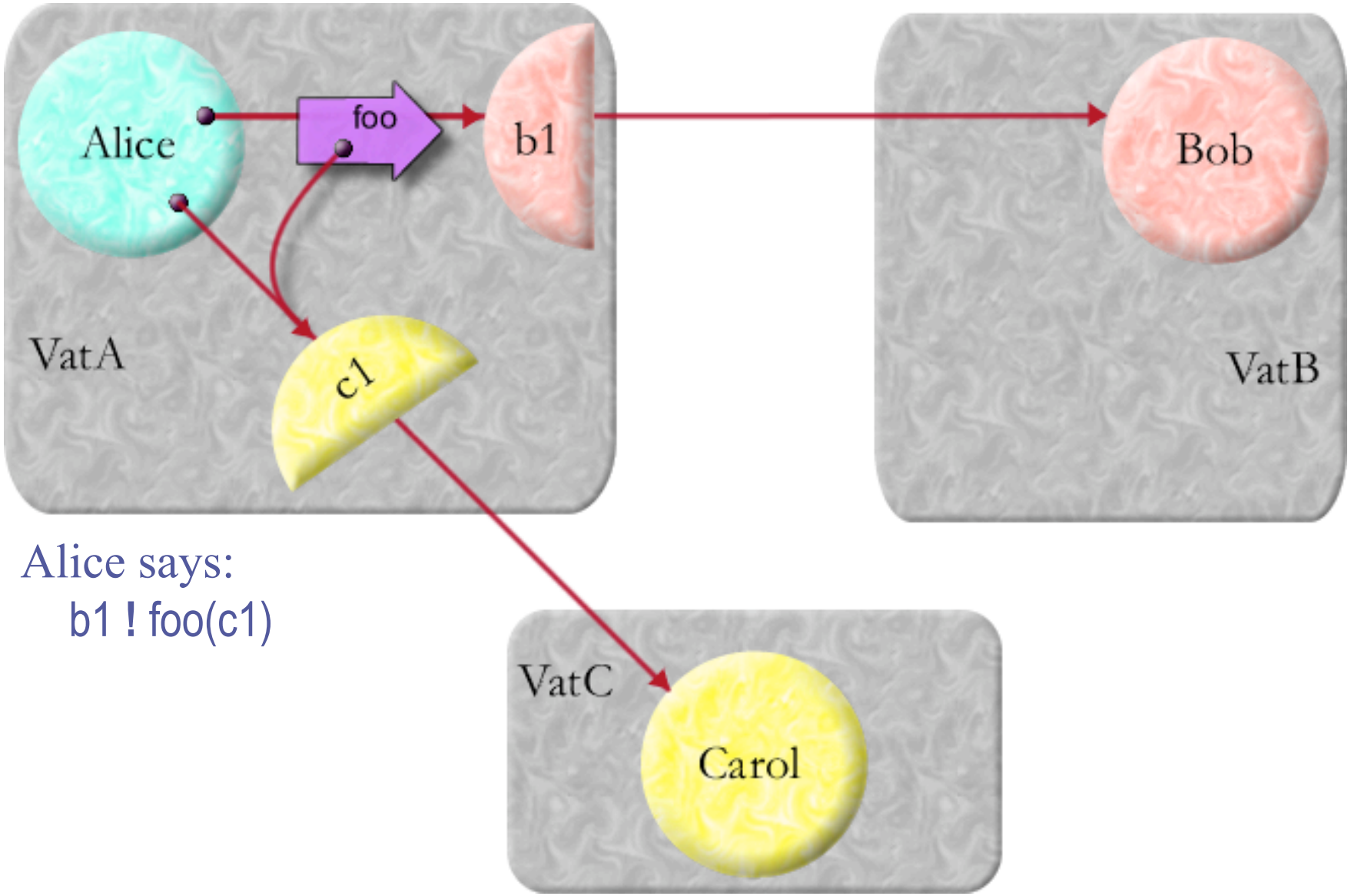
Script code *vs.* **eval** code

<script src="domado.js"></script>

A suggestion for ES5 implementers:

Please implement the ES-Harmony ***Proxy*** *proposal.*

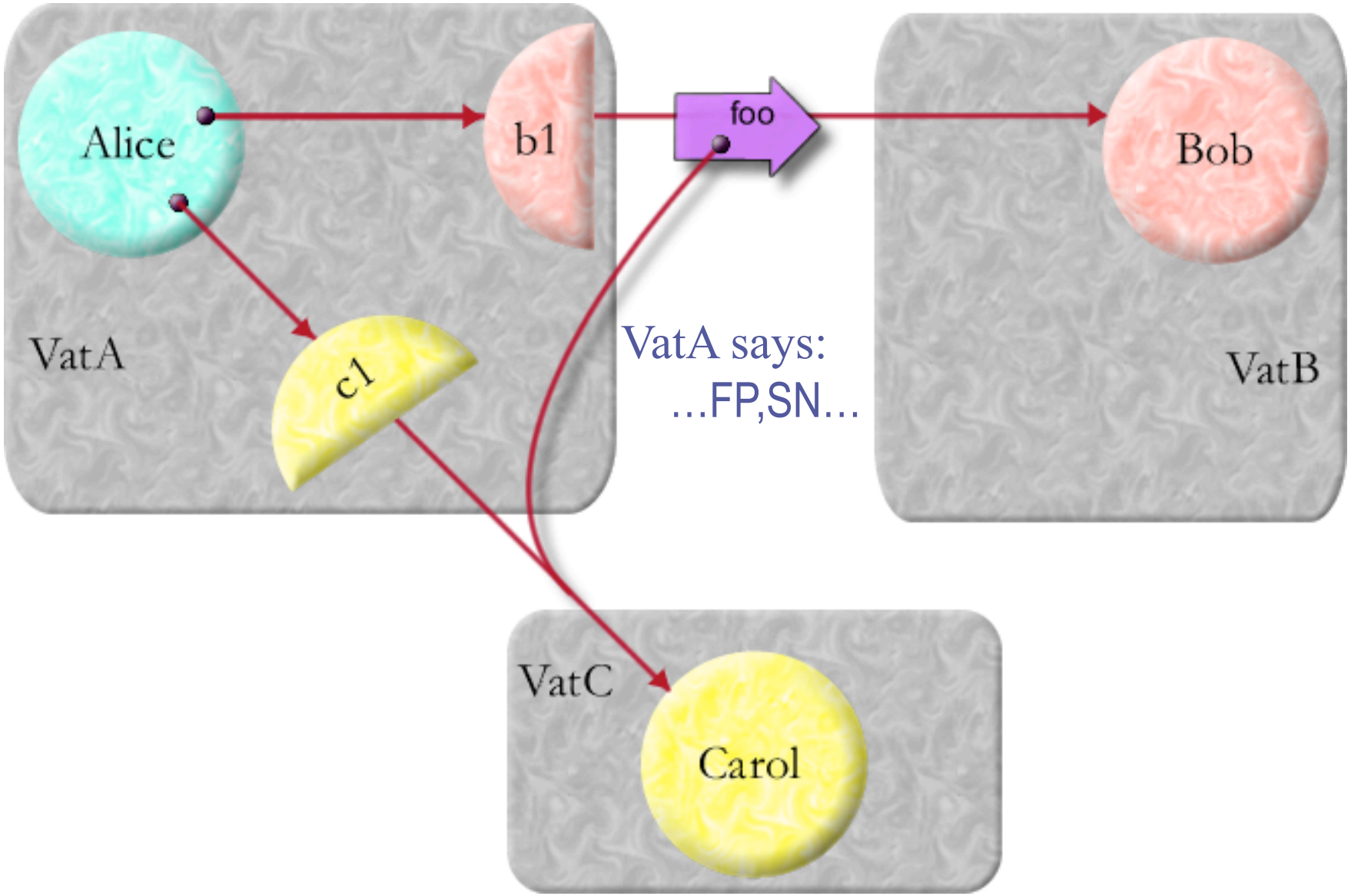
Distributed objects in Dr. SES



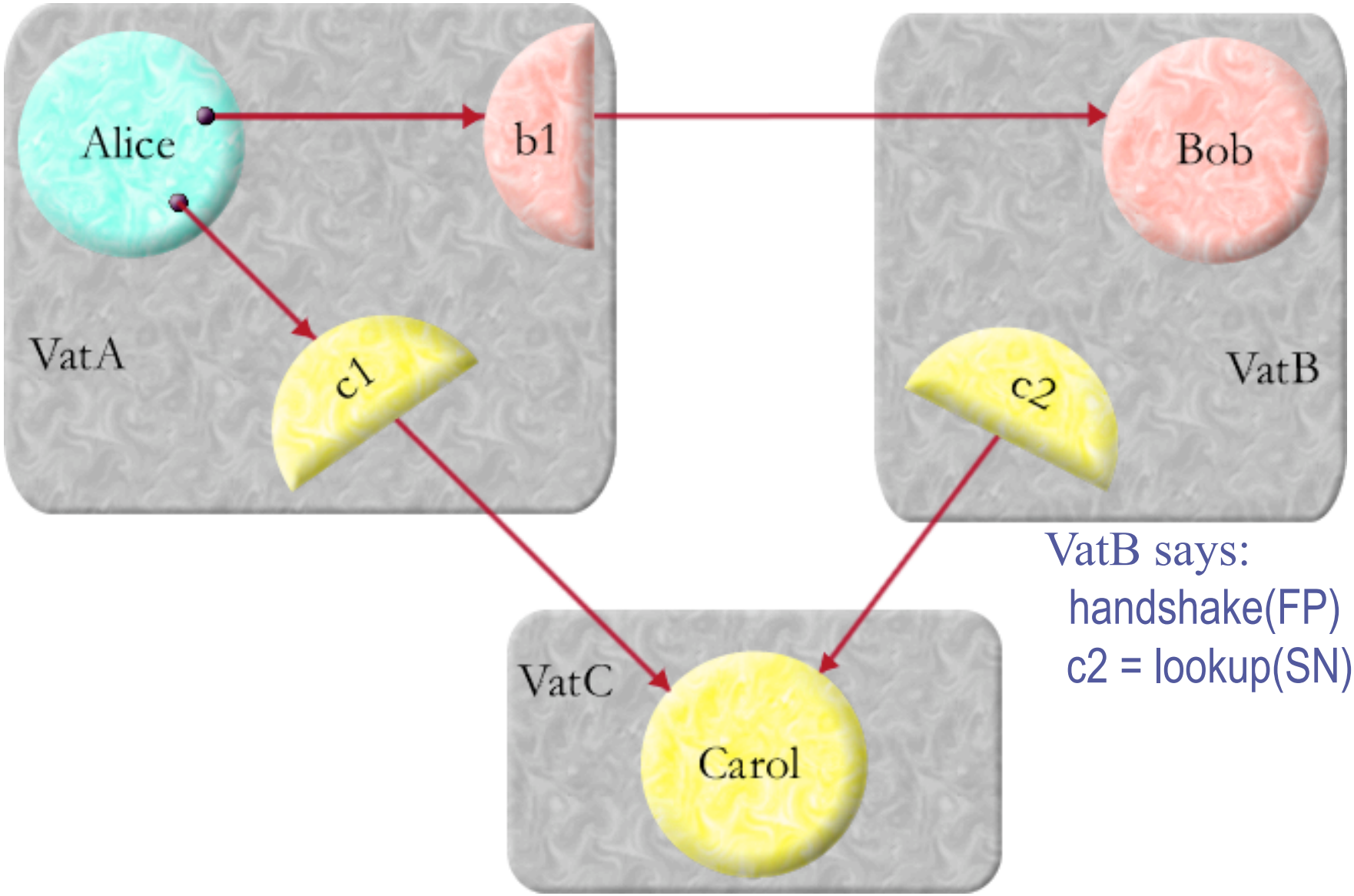
Alice says:

b1 ! foo(c1)

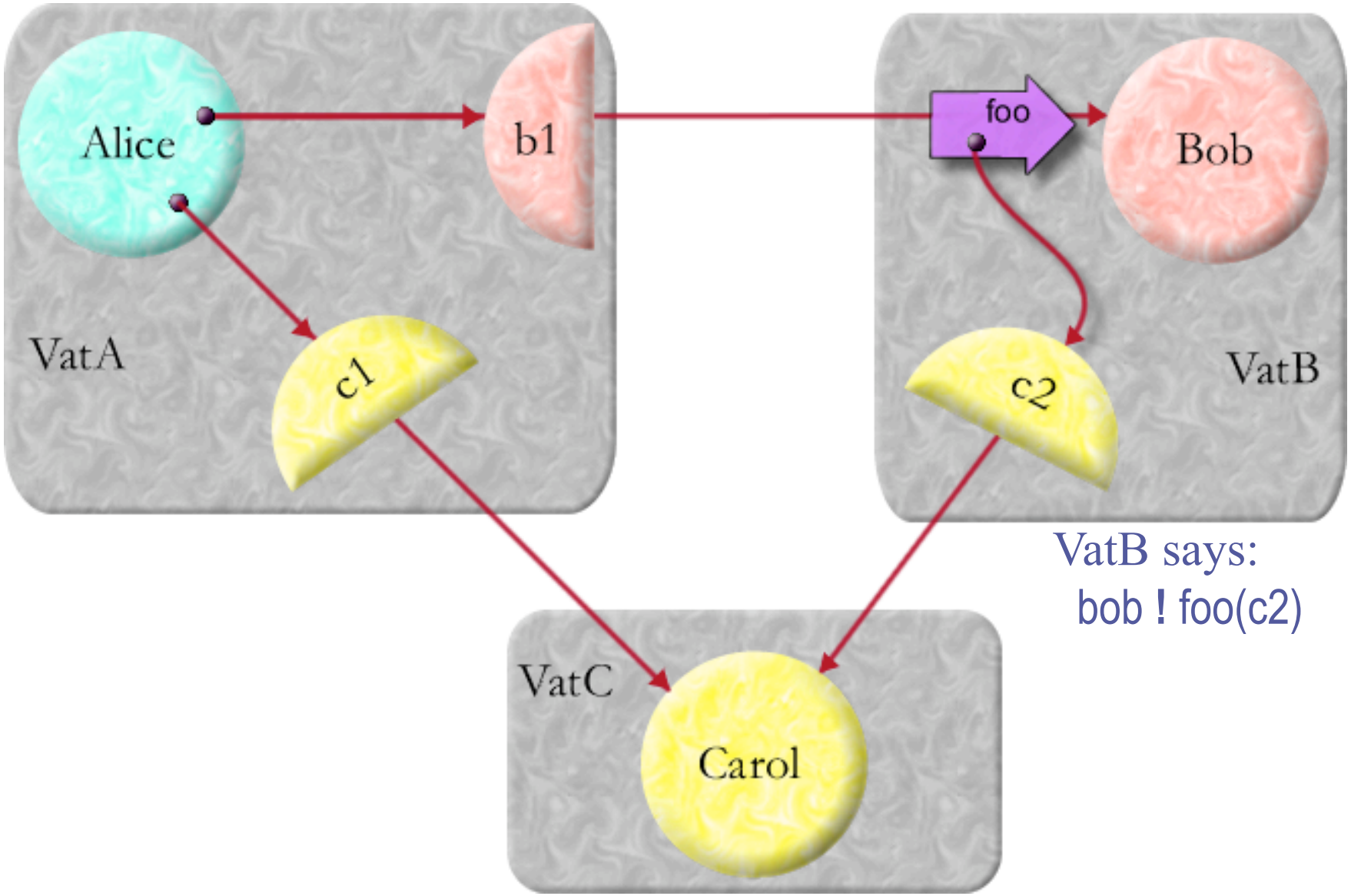
Distributed objects in Dr. SES



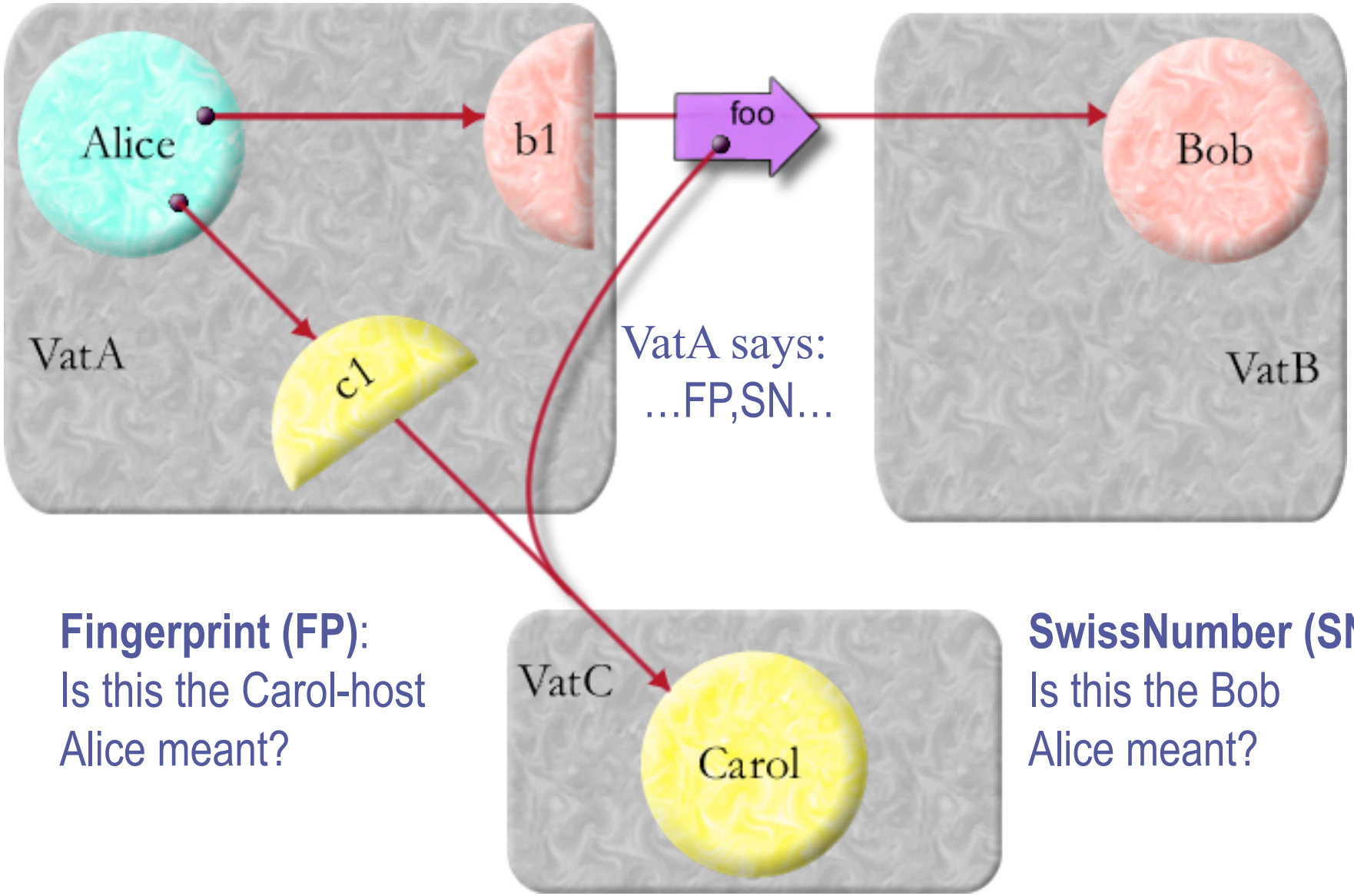
Distributed objects in Dr. SES



Distributed objects in Dr. SES



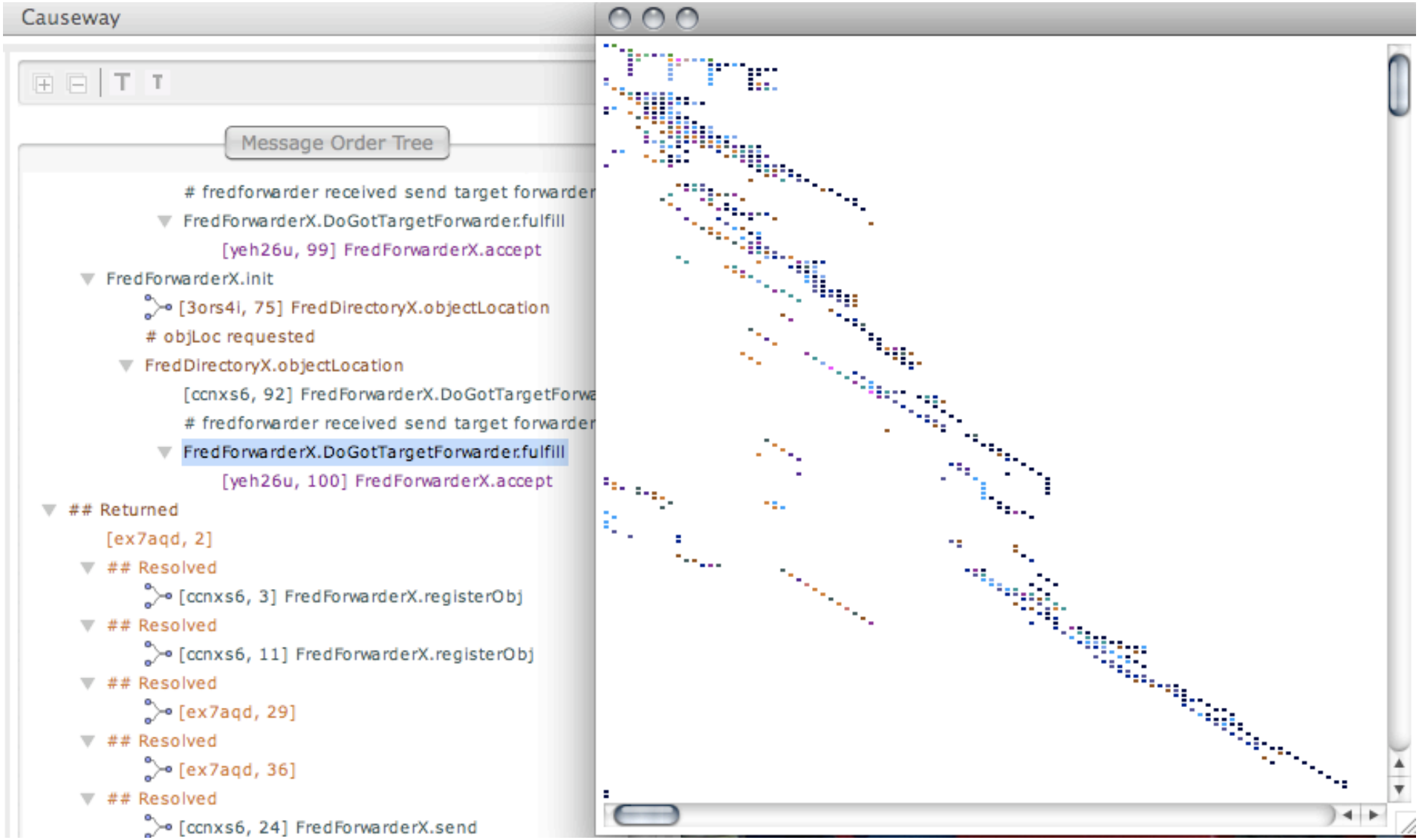
The Two Impostor Problems



Fingerprint (FP):
Is this the Carol-host Alice meant?

SwissNumber (SN):
Is this the Bob Alice meant?

Causeway distributed debugger



Robust objects with SESLint?

```
function makeTable() {
    var array = [];
    return def({
        store: function(i, v) { array[i] = v; },
        queue: function(v)  { array.push(v); }
    });
}
```

Robust objects with SESLint?

```
function makeTable() {
    var array = [];
    return def({
        store: function(i, v) { array[i] = v; },
        queue: function(v)  { array.push(v); }
    });
}
```

Bob says:

```
var stash;
table.store('push', function(v) { stash = this; });
table.queue(88); // stash stole array!
```

Robust objects with SESLint?

```
function makeTable() {
    /*@encapsulate*/ var array = [];
    return def({
        store: function(i, v) { array[i] = v; },
        queue: function(v)  { array.push(v); }
    });
}
```

Robust objects with SESLint?

```
function makeTable() {
    /*@encapsulate*/ var array = [];
    return def({
        store: function(i, v) { array[i] = v; },
        queue: function(v)  { array.push(v); }
    });
}
```

Robust objects with SESLint?

```
function makeTable() {
    /*@encapsulate*/ var array = [];
    return def({
        store: function(i, v) { array[+i] = v; },
        queue: function(v)  { array.push(v); }
    });
}
```

Bob's attack fails

```
table.store('push', function(v) { stash = this; });
```

Caja Roadmap

	Cajita	SES5/3	SES/ES5-strict
+	Valija	ES5/3	Sandboxed ES5-strict
+	ref_send / server-proxy	→	ref_send / UMP
+		server-server captp	captp / web-sockets
+		"!" sending sugar →	
Subtotal:		**Dr. SES5/3**	**Dr. SES**
+	Sanitize HTML & CSS →		
+	Domita / uncajoled JS	Domado / SES →	
=	**Caja Yesterday**	**Caja Tomorrow**	**Caja on ES5,HTML5**

Questions?

Async object ops as JSON/REST ops

Object operations

```
var result = bob.foo;
var resultP = bob ! foo;
var result = bob.foo(carol);
var resultP = bob ! foo(carol);
bob ! foo = newFoo;
delete bob ! foo;
```

https: JSON/RESTful operations

local only get

GET https://…q=foo

local only call

POST https://…q=foo {…}

PUT https://…q=foo {…}

DELETE http://…q=foo

Async object ops as JSON/REST ops

Object operations	**https: JSON/RESTful operations**
~~var *result* = bob.foo;~~	~~*local only get*~~
var *resultP* = bob ! foo;	GET https://…q=foo
~~var *result* = bob.foo(carol);~~	~~*local only call*~~
var *resultP* = bob ! foo(carol);	POST https://…q=foo {…}
~~bob ! foo = newFoo;~~	~~PUT https://…q=foo {…}~~
~~delete bob ! foo;~~	~~DELETE http://…q=foo~~

Async object ops as JSON/REST ops

Object operations

var *resultP* = bob ! foo;

var *resultP* = bob ! foo(carol);

https: JSON/RESTful operations

GET https://…q=foo

POST https://…q=foo {…}

Async object ops as JSON/REST ops

Object operations

```
var resultP = bob ! foo;
var resultP = bob ! foo(carol);

Q.when(resultP, function(result) {
  …result…
}, function (ex) {
  …ex…
});
```

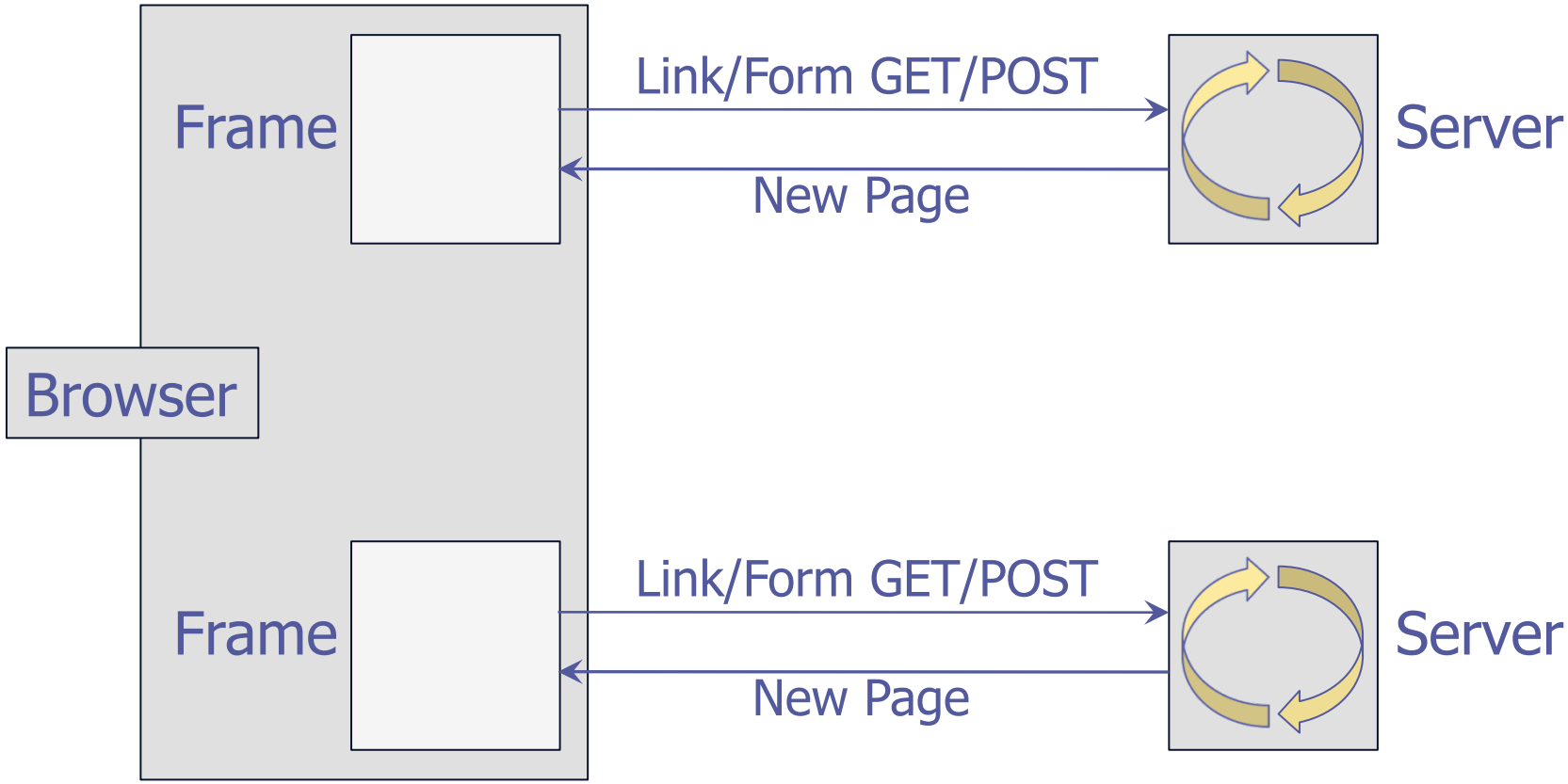
https: JSON/RESTful operations

```
GET https://…q=foo
POST https://…q=foo {…}
```

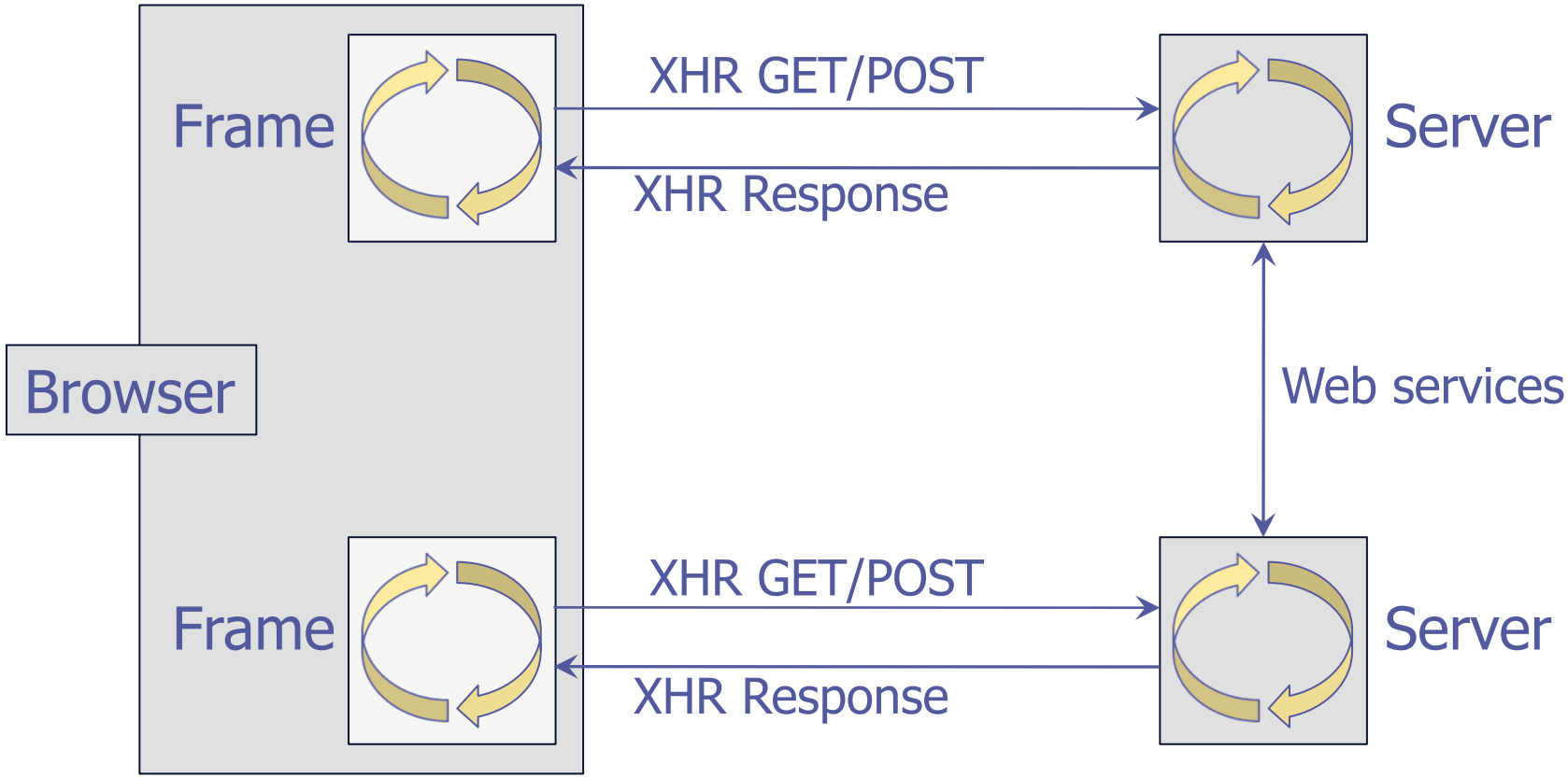
Register for notification using

```
xhr.onreadystatechange = …
```

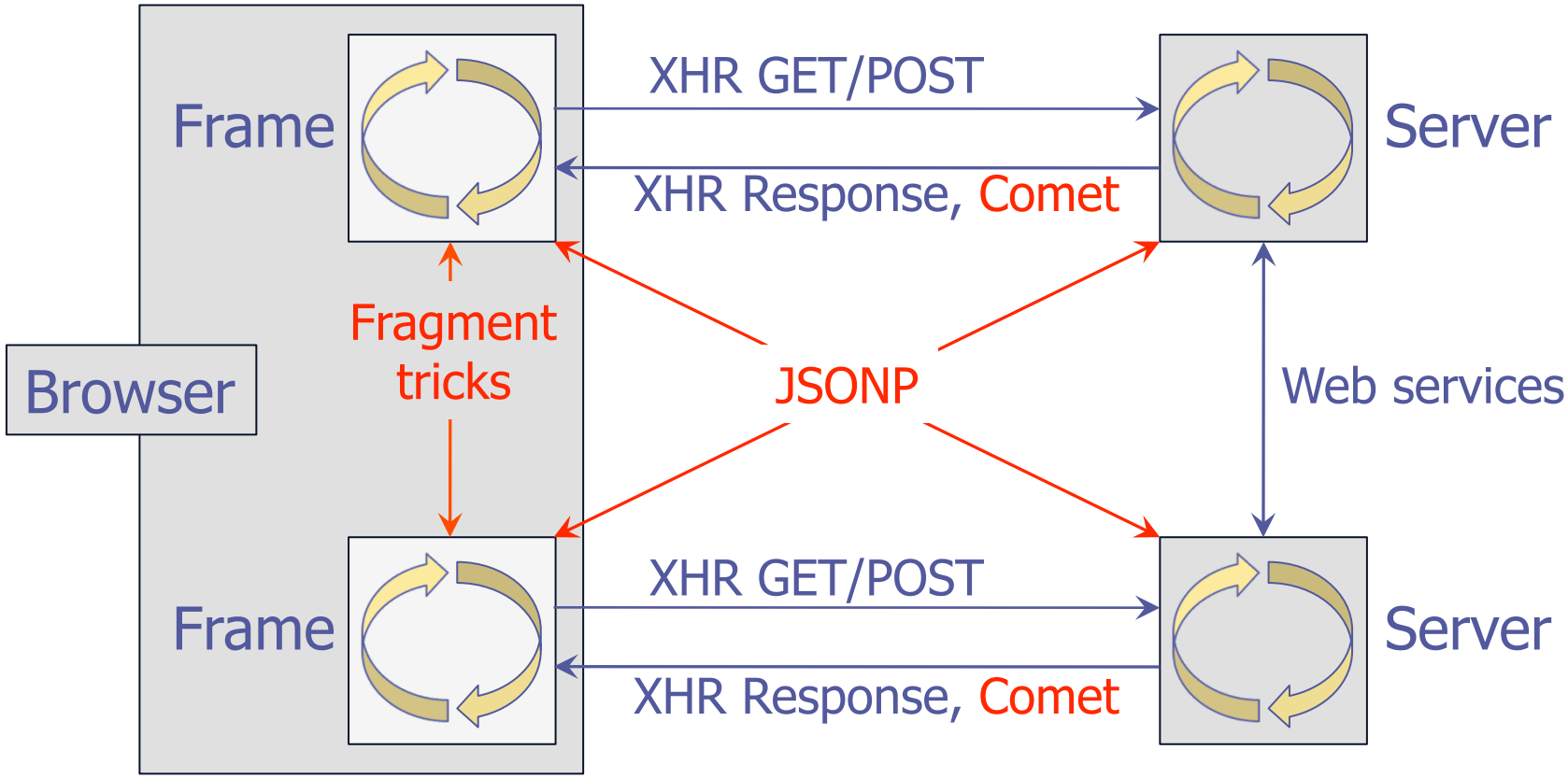
Original Web



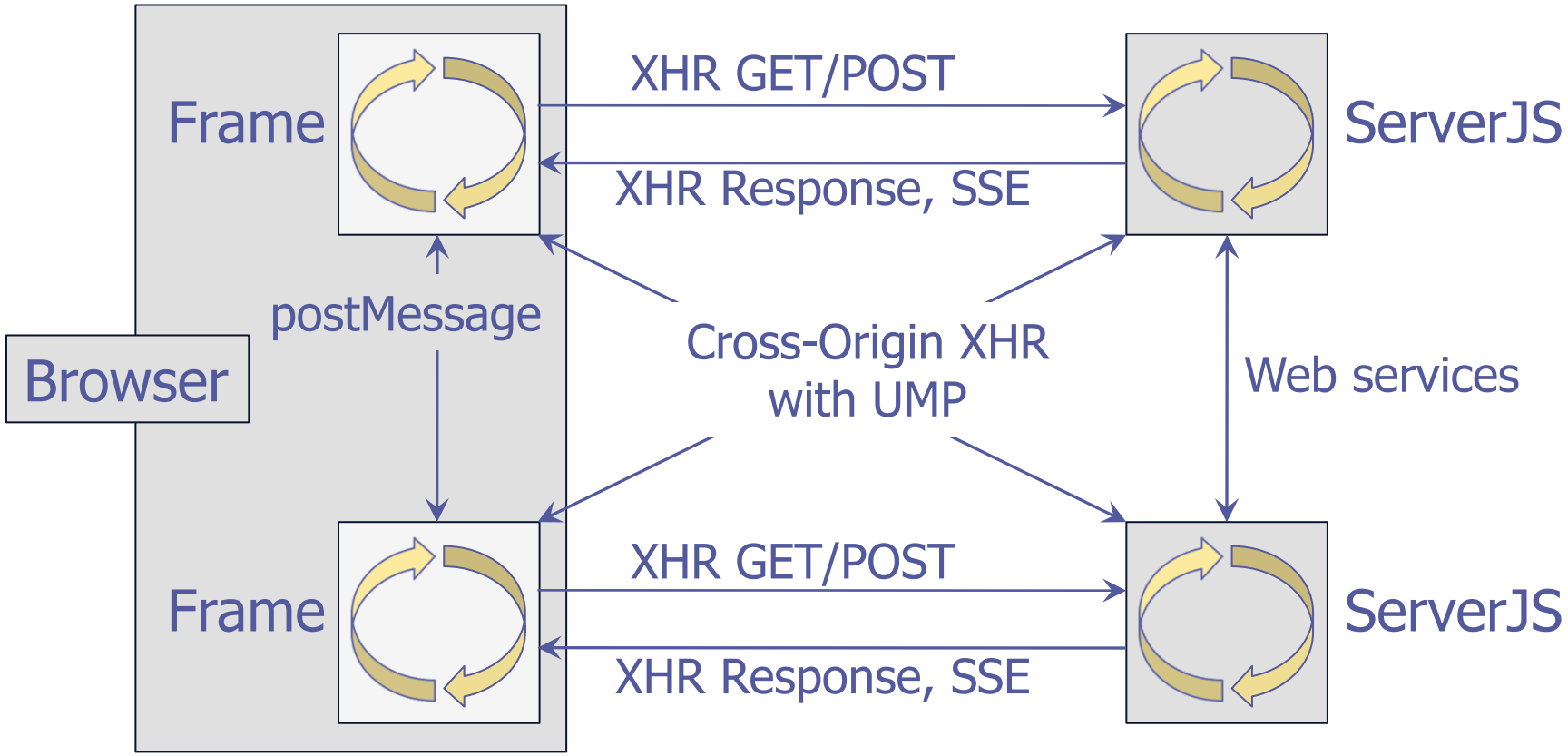
Ajax = Mobile code + async msgs



Kludging Towards Distributed Objects



A Web of Distributed Objects



"def" is for defining defended objects

```
var defended = WeakMap();
function def(root) {
    var defending = WeakMap(), defendingList = [];
    function recur(val) {
        if (val !== Object(val) || defended.get(val) || defending.get(val)) { return; }
        defending.set(val, true); defendingList.push(val);
        Object.freeze(val);
        recur(Object.getPrototypeOf(val));
        Object.getOwnPropertyNames(val).forEach(function(p) {
            var desc = Object.getOwnPropertyDescriptor(val, p);
            recur(desc.value); recur(desc.get); recur(desc.set);
        });
    }
    recur(root);
    defendingList.forEach(function(obj) {
        defended.set(obj, true);
    });
    return root;
}
```

"Nat" validates its arg is a UInt32

```
function Nat(arg) {
    if (arg === arg >>> 0) { return arg; }
    throw new TypeError('Not a UInt32: ' + arg);
}
```

"makeCaretaker" for defended targets

```
function makeCaretaker(target) {
    var wrapper = (typeof target !== 'function') ? {} : function(var_args) {
        return target.apply(this, arguments);
    };
    Object.getOwnPropertyNames(target).forEach(function(p) {
        var desc = Object.getOwnPropertyDescriptor(target, p);
        Object.defineProperty(wrapper, p, desc);
    });
    return def({
        wrapper: wrapper,
        revoke: function() { target = null; }
    });
}
```

"makeMembrane" for defended targets

```
function makeMembrane(target) {
    var enabled = true;
    function wrap(wrapped) {
        if (wrapped !== Object(wrapped)) { return wrapped; }
        var wrapper = (typeof wrapped !== 'function') ? {} : function(var_args) {
            return wrap(wrapped.apply(wrap(this), Array.slice(arguments, 0).map(wrap)));
        };
        Object.getOwnPropertyNames(wrapped).forEach(function(p) {
            var desc = Object.getOwnPropertyDescriptor(wrapped, p);
            Object.defineProperty(wrapper, p, desc);
        });
        return wrapper;
    }
    return def({
        wrapper: wrap(target),
        revoke: function() { enabled = false; }
    });
}
```

Future objects on old browsers in ES5/3

Reserve the *___ namespace.

a[i] → a.v___(i)

a[+i] → a[+i] // implicitly whitelist numbers

Encoding attributes in hidden properties

a.x → a.x_v___ ? a.x : a.v___('x')

Whitelist and fastpath

a.x = 88 → a.x_w___ === a ? a.x = 88 : a.w___('x', 88)

Enables catchall proxies

Override generic operation to trap to handle