Secure Distributed Programming
with Object-capabilities in JavaScript

Mark S. Miller and the Cajadores

Google 

Overview

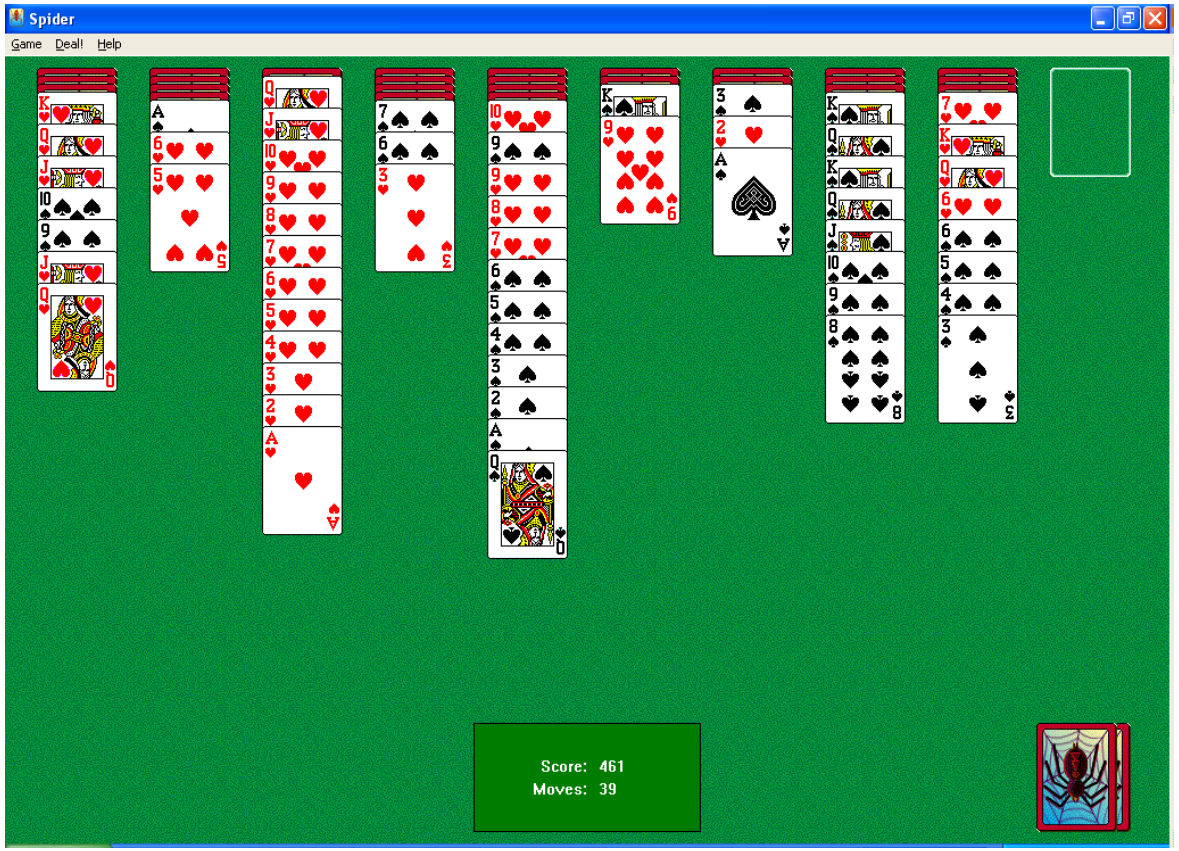
- Why object-capability (ocap) security?
- Local ocap security in JavaScript
- Flexible secure mobile code
- Distributed crypto-caps in JavaScript
- Secure distributed object programming

Early Choice. Late Despair

- ACLs and OCaps start in mid ‘60s.
  - ACLs: “Who is making this request?”
  - OCaps: “Is this request authorized?”
- ‘70s: Industry took ACL fork in road.
- ‘90s to present: Rise of Malware

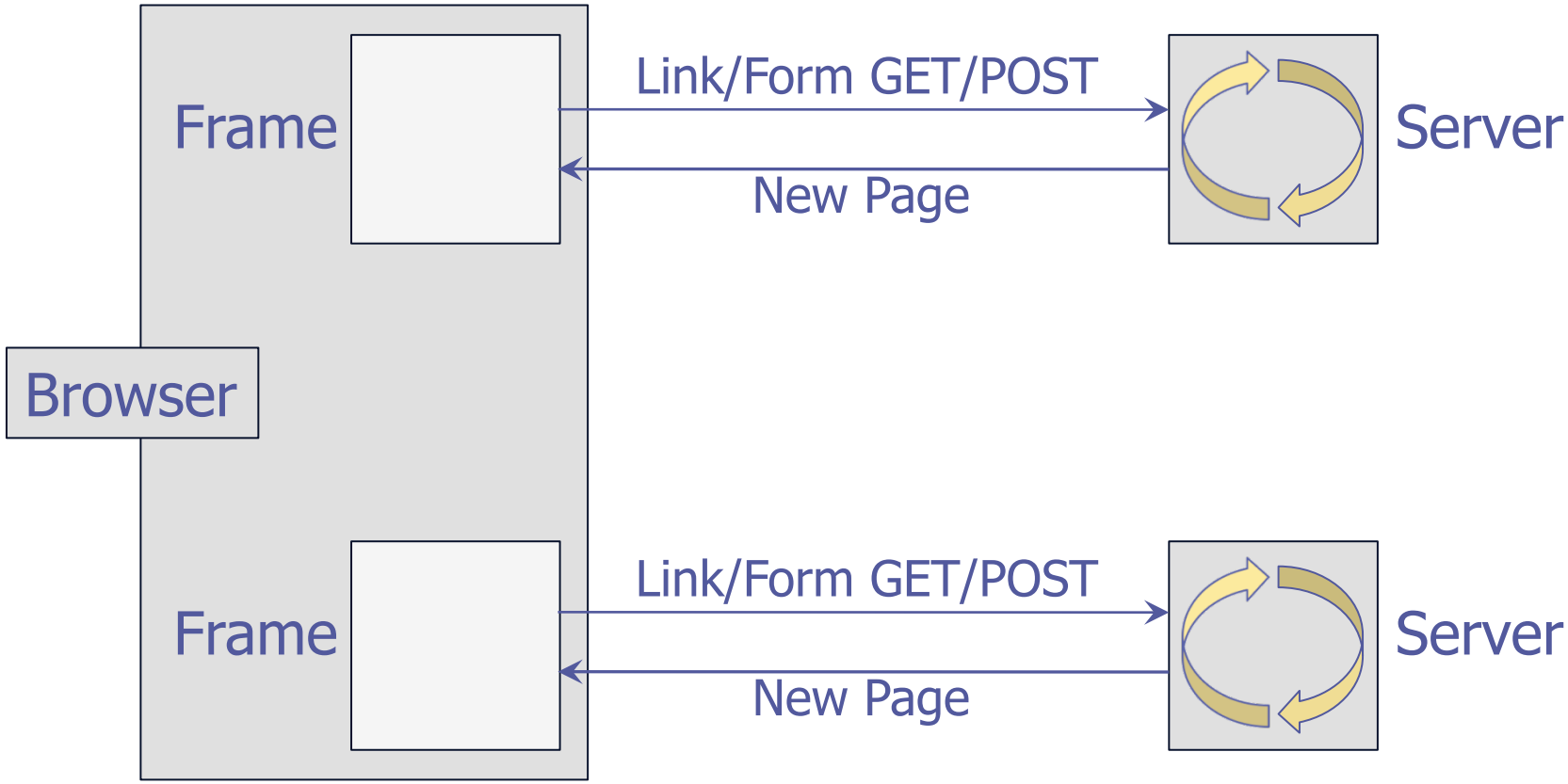
A Very Powerful Program

A Very Powerful Program

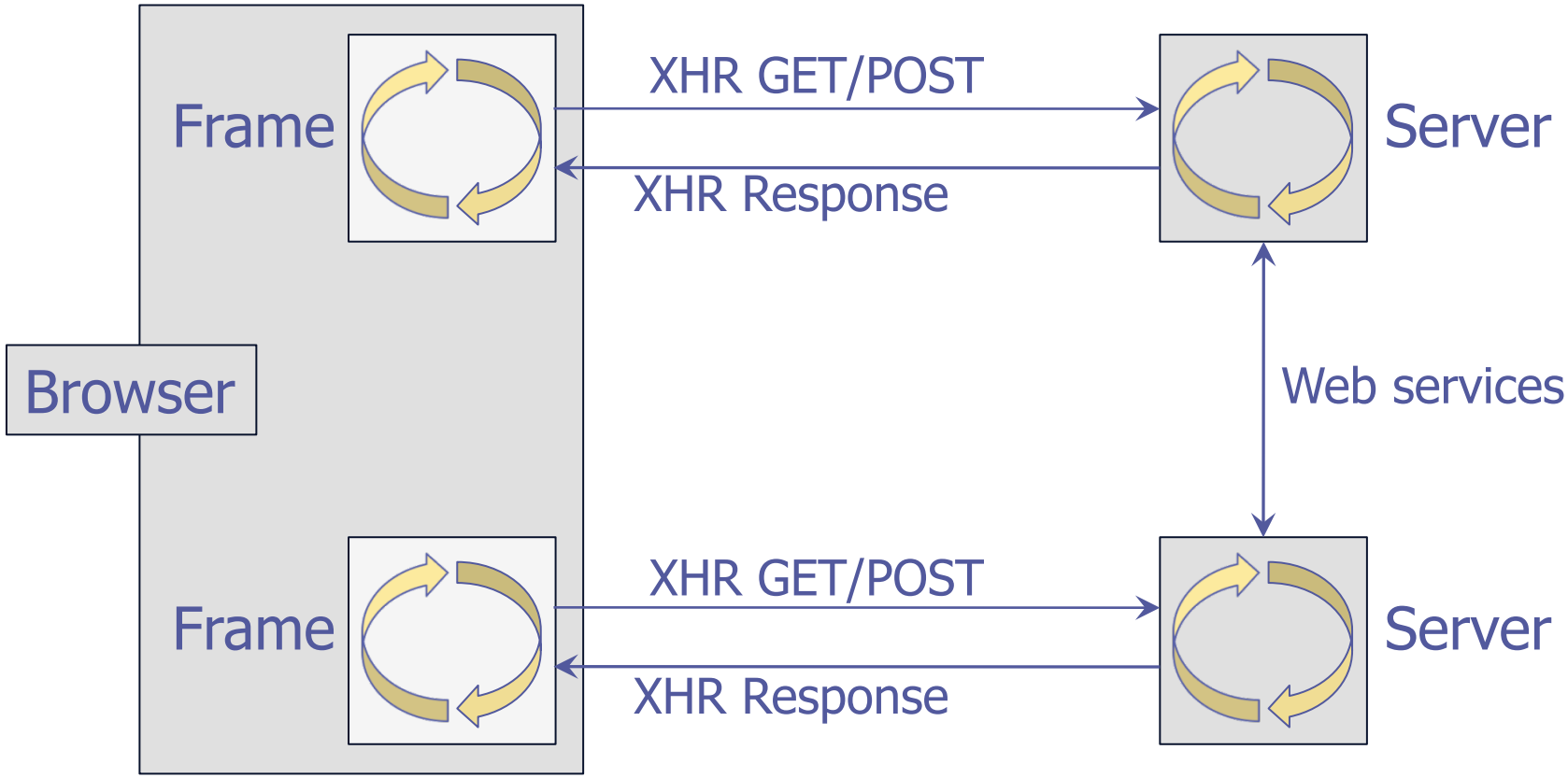


This program can delete any file you can.

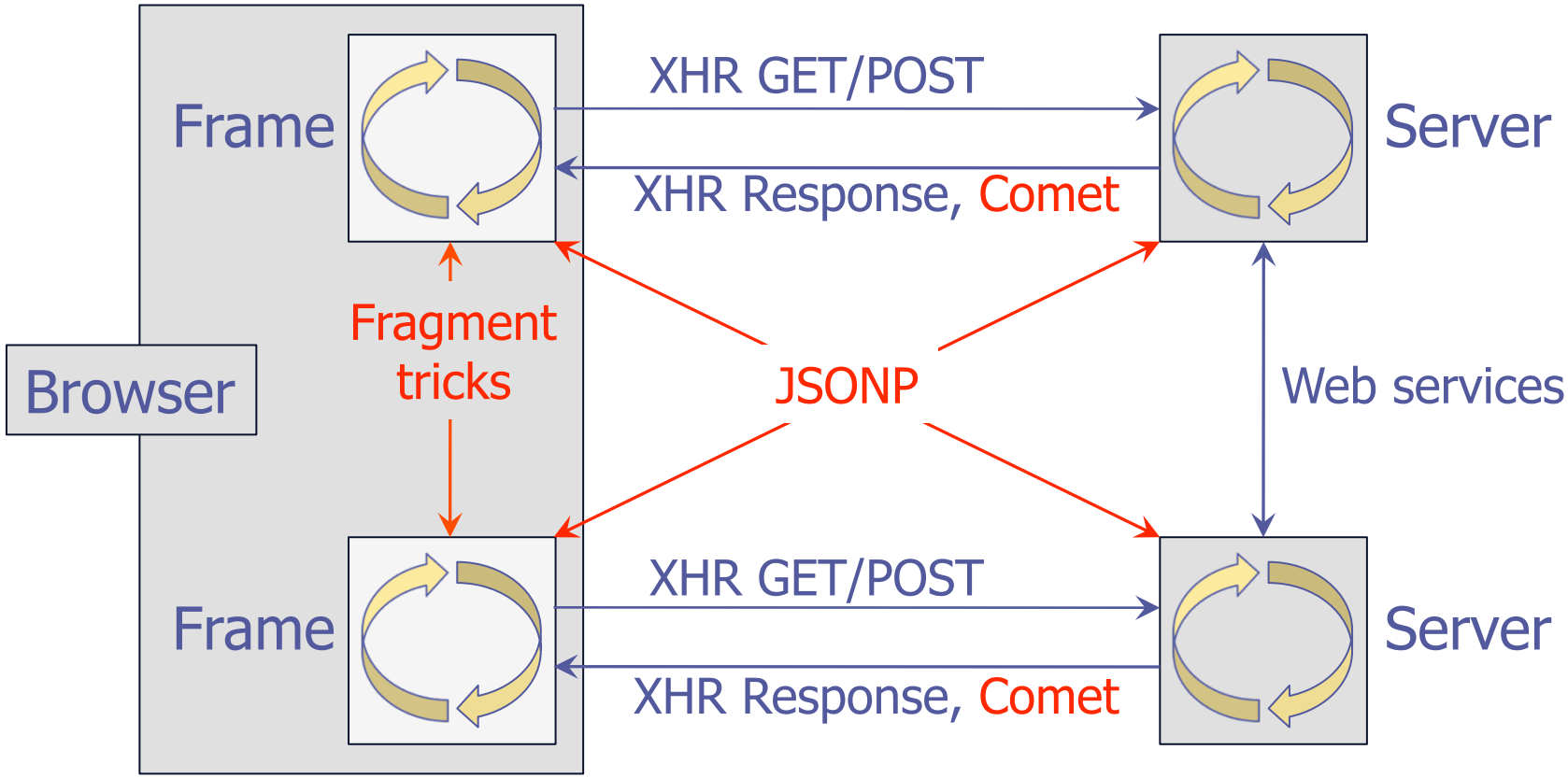
Original Web



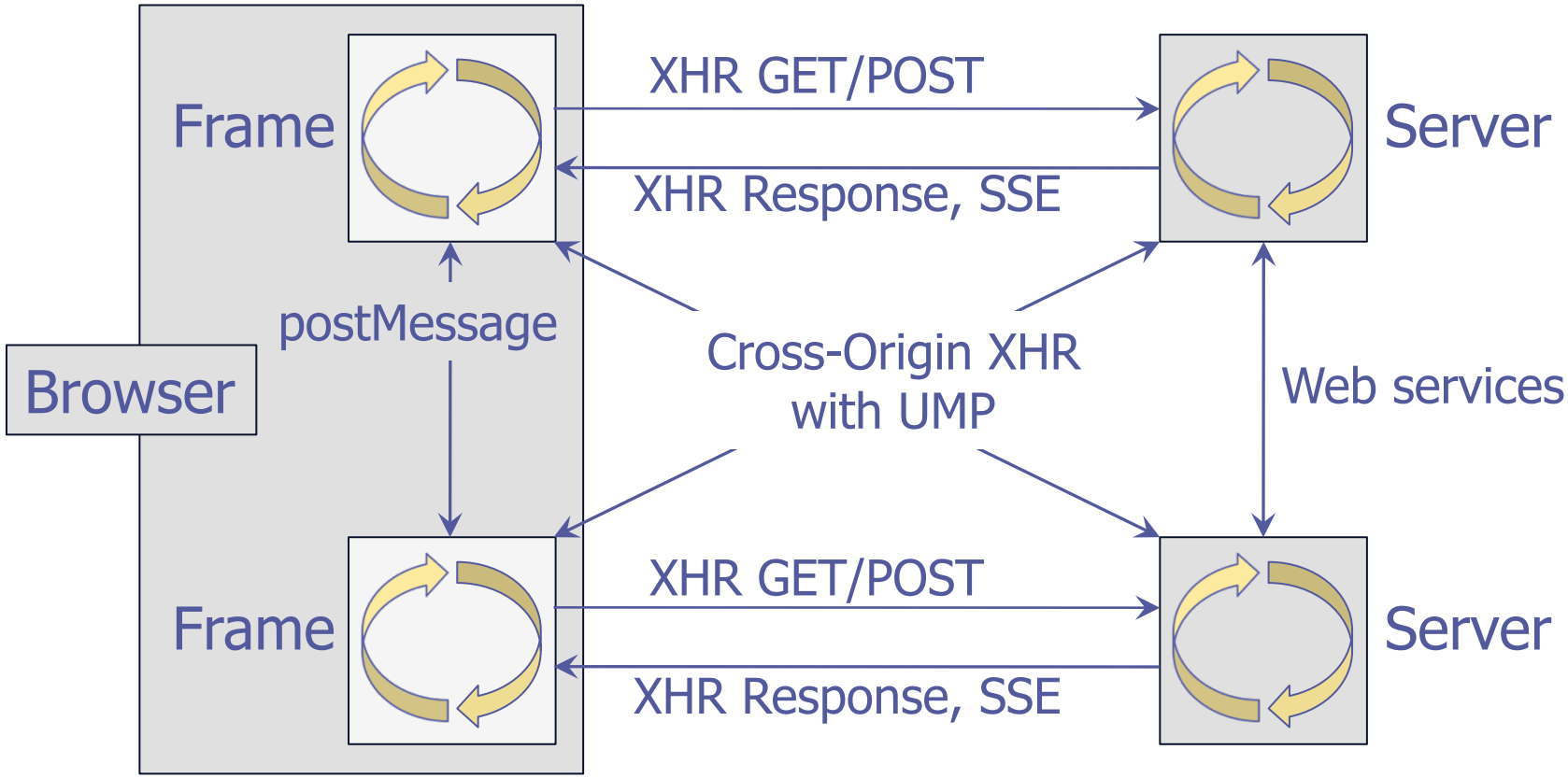
Ajax = Mobile code + async msgs



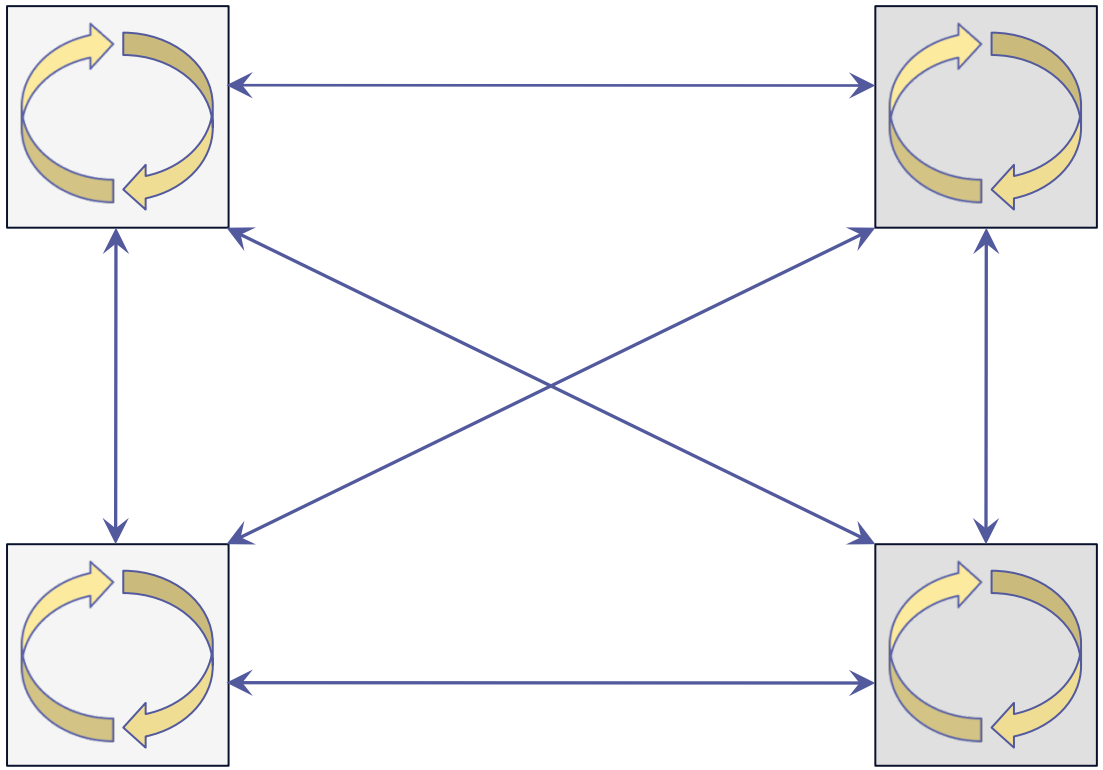
Kludging Towards Distributed Objects



A Web of Distributed Objects

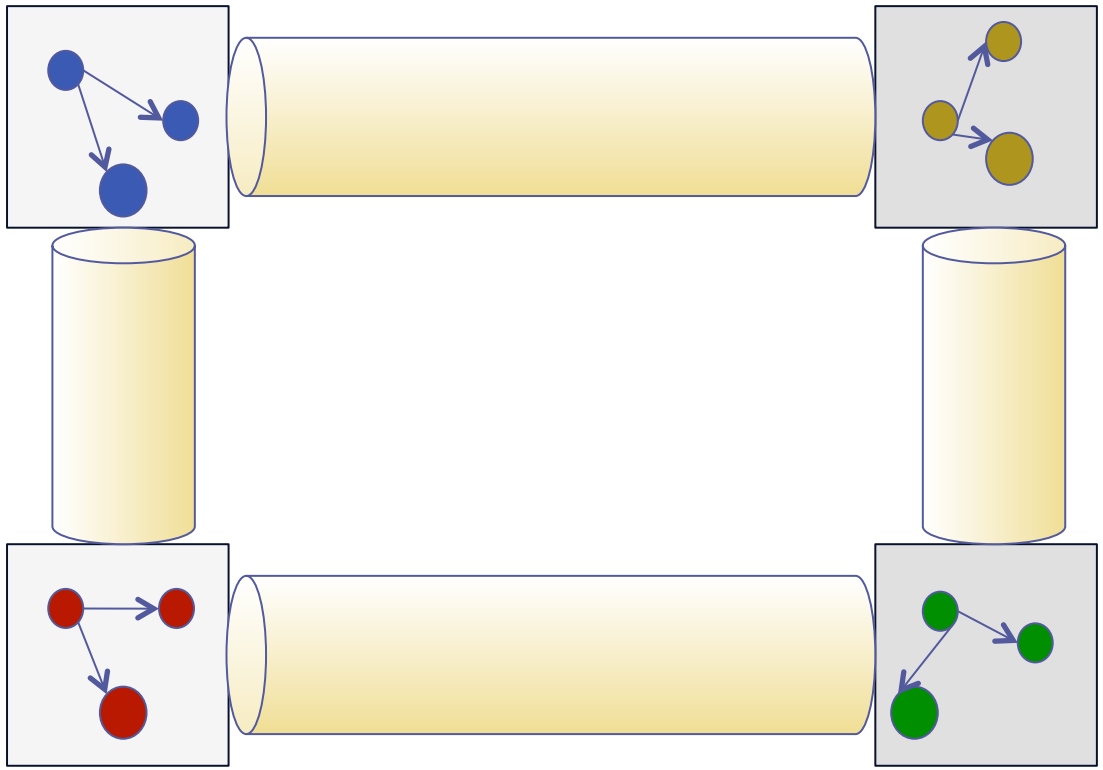


A Web of Distributed Objects

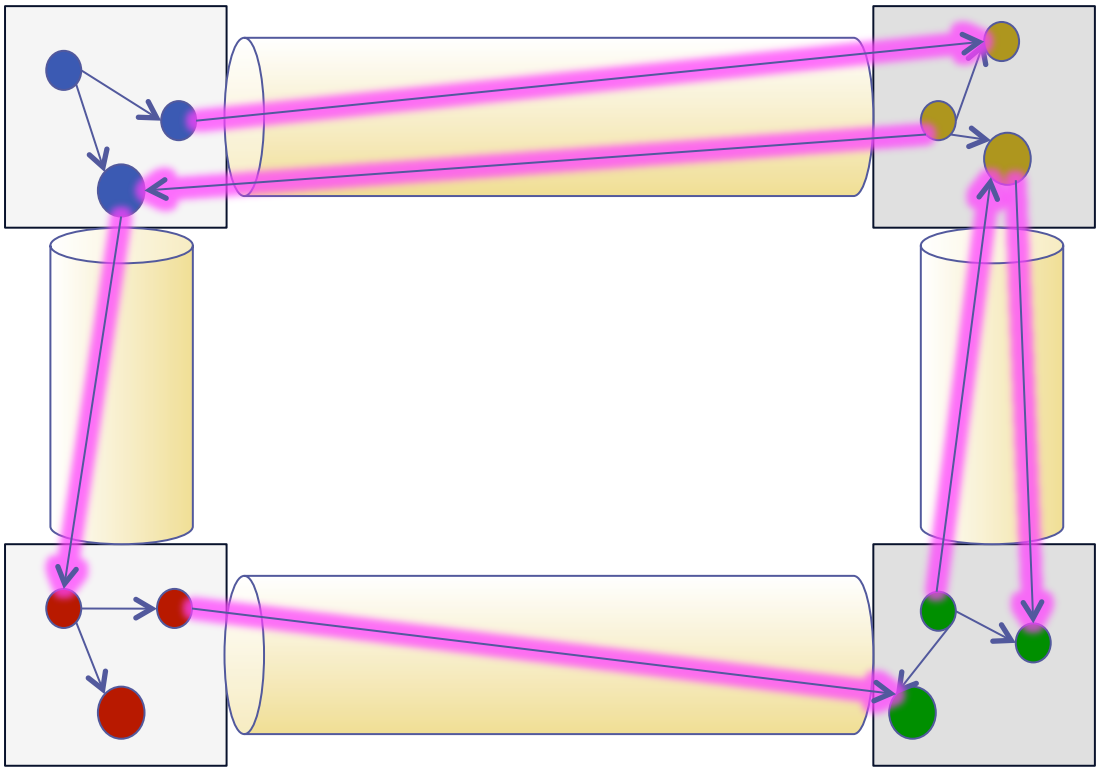


Mobile messages, code, objects, references

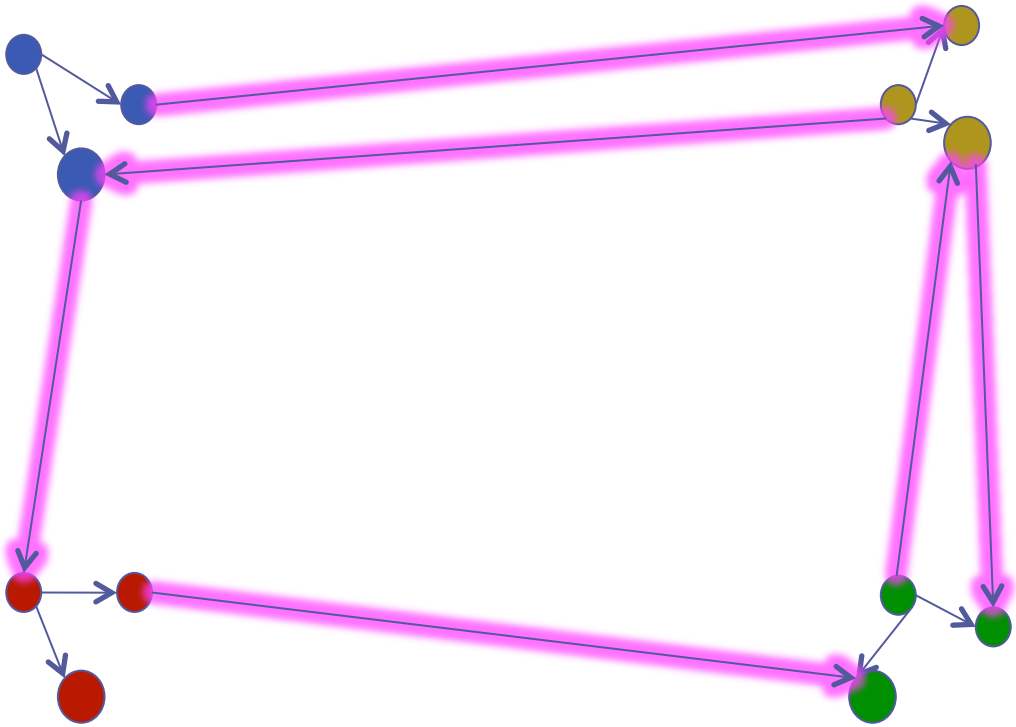
A Web of Distributed Objects



A Web of Distributed Objects



A Web of Distributed Objects



A Very Powerful Email Message

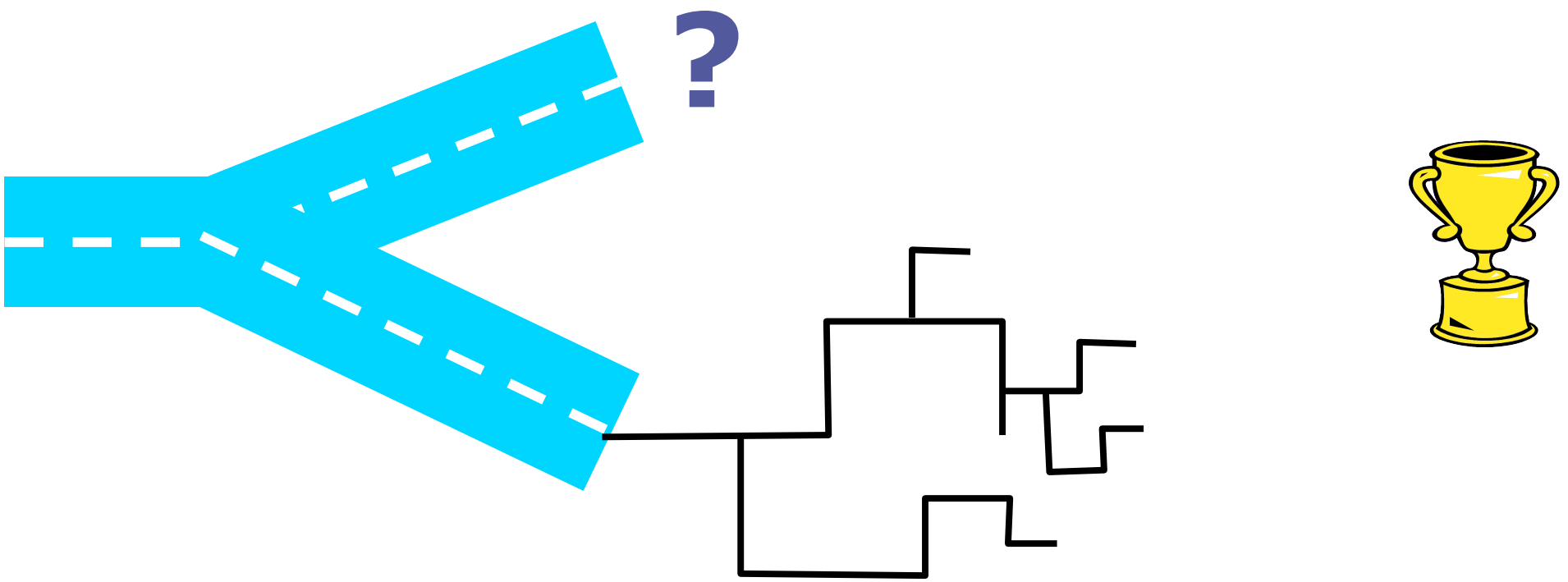
A Very Powerful Email Message

```
<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>
```

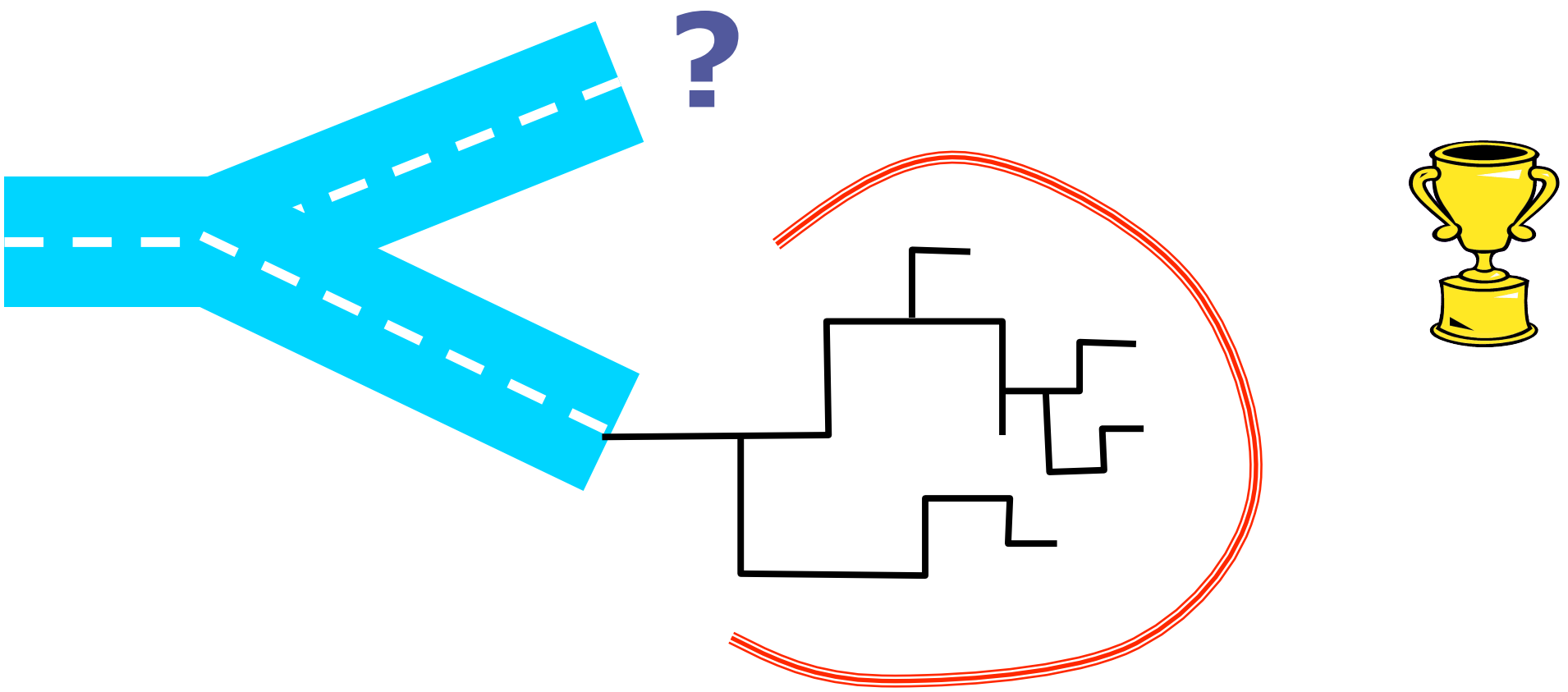
Active Content: Mobile Code as Media

```
<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>
```

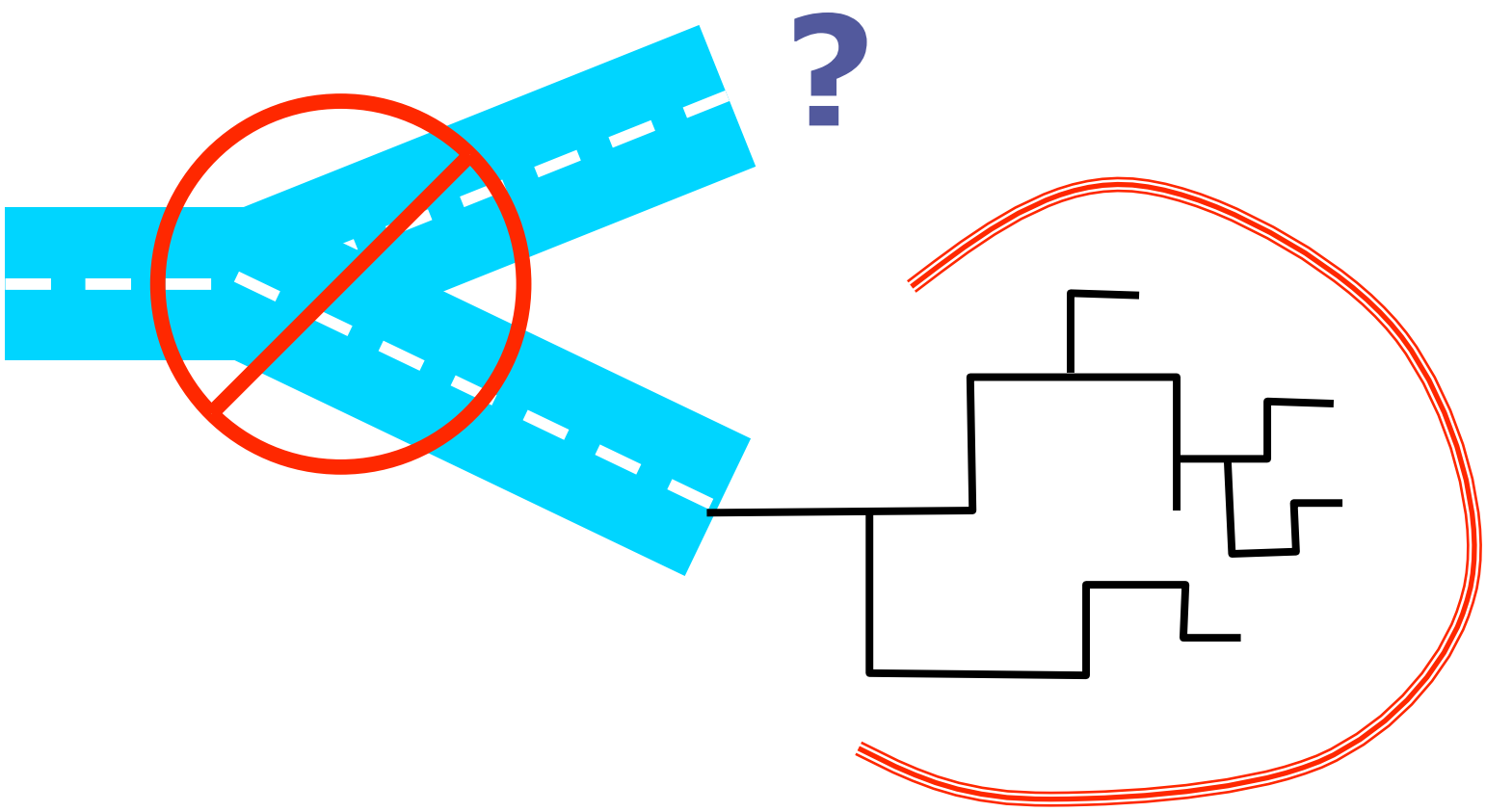
The Road Not Taken



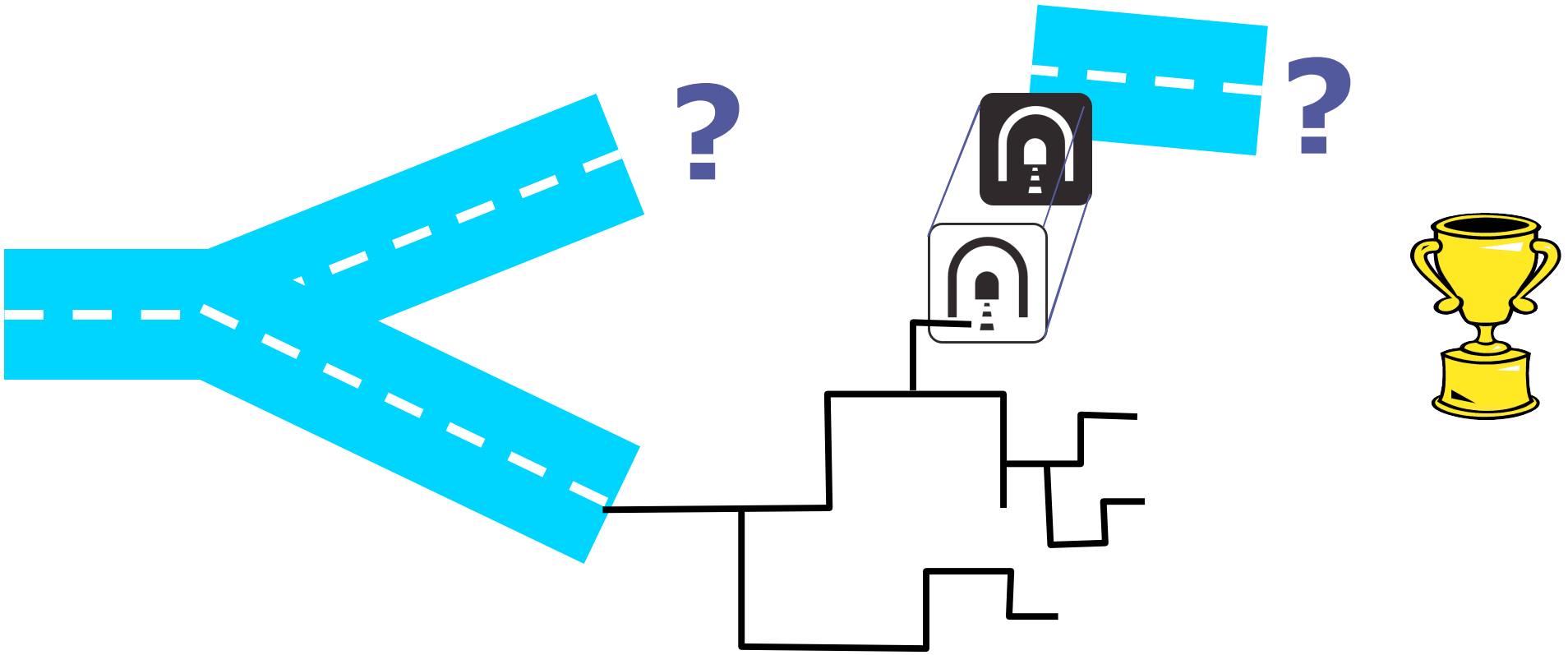
The Road Not Taken



The Road Not Taken



The Road Not Taken



Security as Extreme Modularity

Modularity: Avoid needless dependencies

Security: Avoid needless vulnerabilities

Vulnerability is a form of dependency

Modularity:

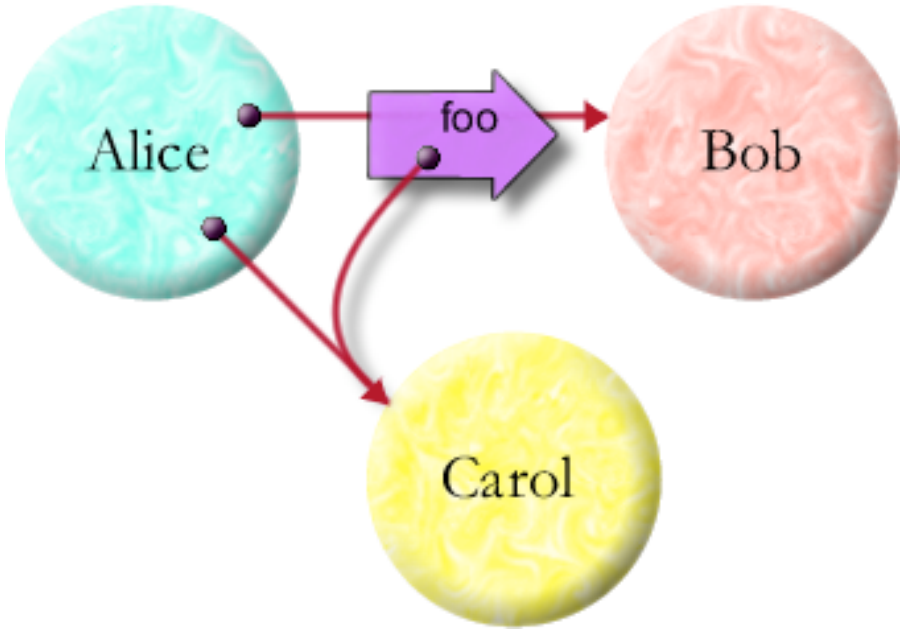
- Principle of info hiding - need to know

Security:

- Principle of least authority - need to do

Connectivity by…

Alice says: bob.foo(carol)



… *Introduction*

ref to Carol

ref to Bob

decides to share

… Parenthood

… Endowment

… Initial Conditions

How might object Bob come to know object Carol?

OCaps: Small step from pure objects

Memory safety and encapsulation

\+ Effects ***only*** by using held references

\+ No powerful references by default

OCaps: Small step from pure objects

Memory safety and encapsulation

\+ Effects ***only*** by using held references

\+ No powerful references by default

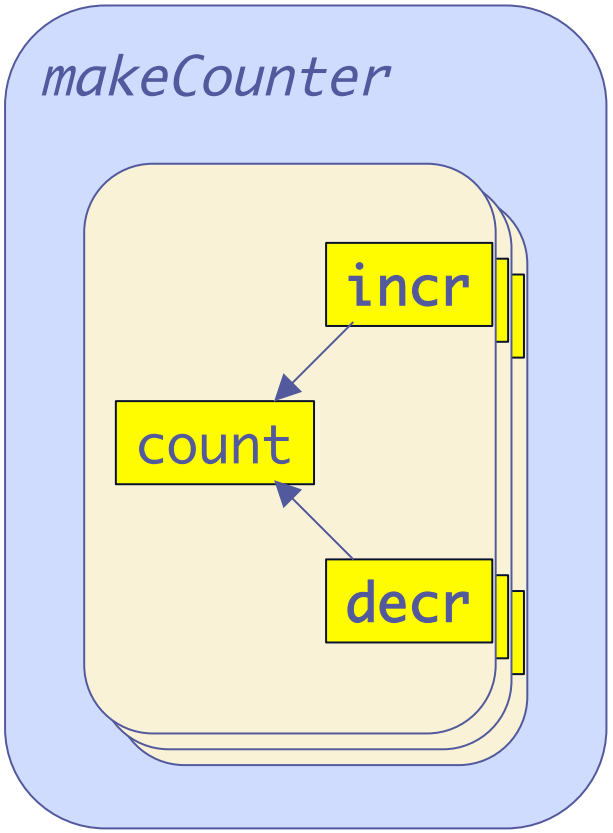
Reference graph ≡ Access graph

Only connectivity begets connectivity

Natural *Least Authority*

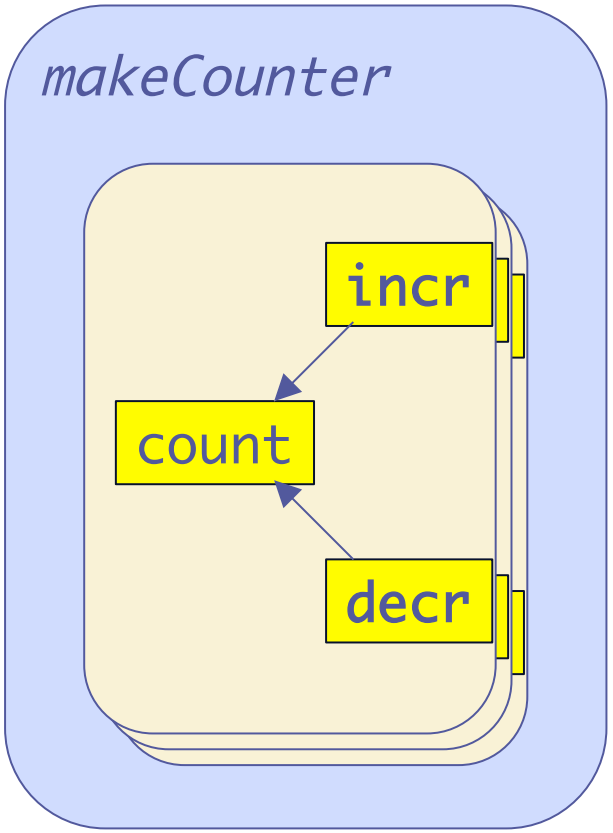
OO expressiveness for security patterns

Objects as Closures in JavaScript




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

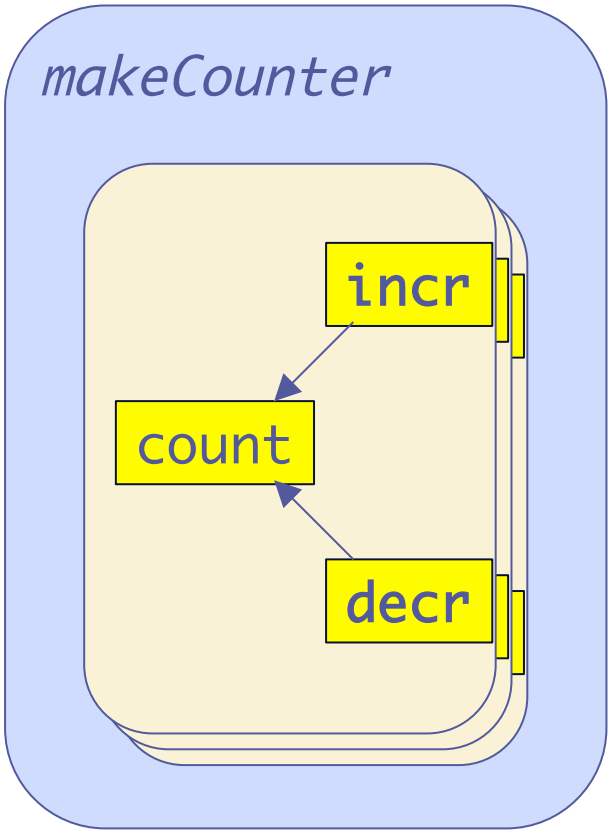
Objects as Closures in JavaScript




```
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*

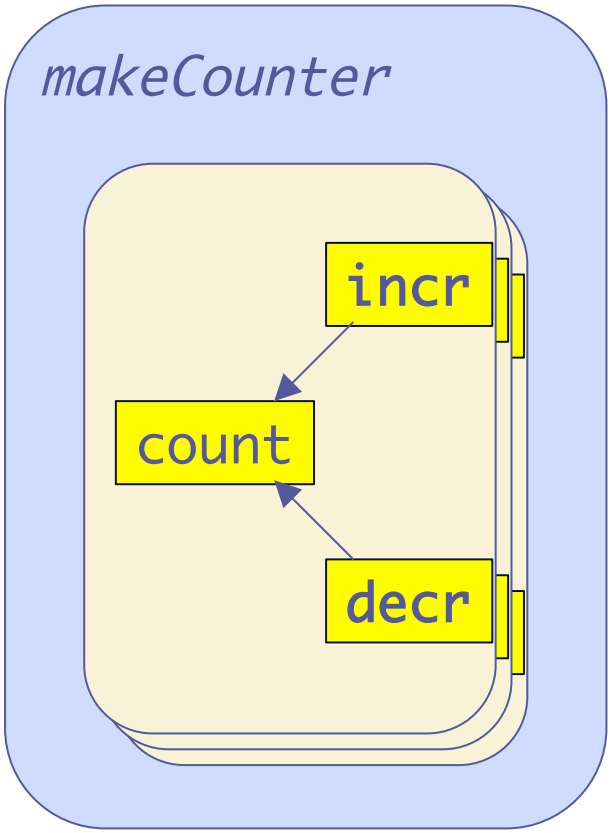
Objects as Closures in EcmaScript 3




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

- Scoping confusions
- Encapsulation leaks
- Pervasive mutability

Defensive Objects in SES on ES5




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

A *tamper-proof* record of *lexical* closures *encapsulating* state is a *defensive* object

caja-corkboard.appspot.com

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

View Source

...ggle about. -- David-Sarah

What version and OS? Can't reproduce on either machine I have handy. -- kpreid

— Anon, 2010-07-24 00:41:44.706661

Edit Delete

Unicode test. You should see two bullets and two (if you've got the font for it) U+1040E DESERET CAPITAL LETTER WU (interleaved).
•𐐎•𐐎

— kpreid.switchb.org, 2010-07-23 00:29:17.917977

View Source

Sean B. Palmer

— kpreid.switchb.org, 2010-07-22 17:05:53.107415

View Source

— erights@google.com (Logout), just now

Post This

This is a Caja demo. You can enter any HTML you like, and it will display as well as we currently support and yet not allow you to take over anyone else's postings or otherwise disrupt the application (other than by making the page load slower or hang).

This site is intended to demonstrate how to straightforwardly use Caja in a web application as a "better HTML sanitizer"; see CorkboardDemo on the Caja wiki for a tutorial.

Background image by Parée Erica (used under Creative Commons Attribution license).

caja-corkboard.appspot.com

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

View Source

...ggle about. -- David-Sarah

What version and OS? Can't reproduce on either machine I have handy. -- kpreid

— Anon, 2010-07-24 00:41:44.706661

Edit Delete

Unicode test. You should see two bullets and two (if you've got the font for it) U+1040E DESERET CAPITAL LETTER WU (interleaved).
•𐐎•𐐎

— kpreid.switchb.org, 2010-07-23 00:29:17.917977

View Source

Sean B. Palmer

— kpreid.switchb.org, 2010-07-22 17:05:53.107415

View Source

```
<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>
```

— erights@google.com (Logout), just now

Post This

This is a Caja demo. You can enter any HTML you like, and it will display as well as we currently support and yet not allow you to take over anyone else's postings or otherwise disrupt the application (other than by making the page load slower or hang).

This site is intended to demonstrate how to straightforwardly use Caja in a web application as a "better HTML sanitizer"; see CorkboardDemo on the Caja wiki for a tutorial.

Background image by Parée Erica (used under Creative Commons Attribution license).

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

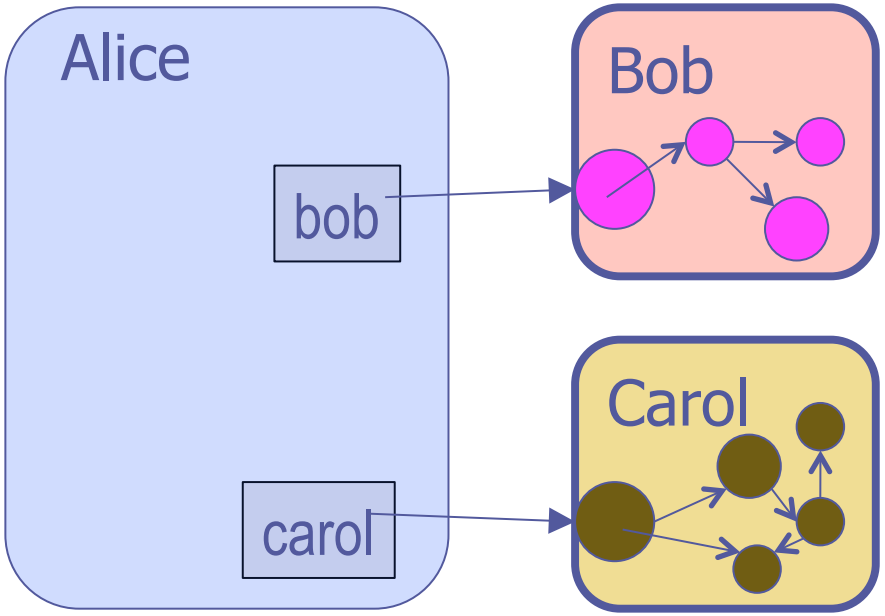
Turning EcmaScript 5 into SES

```
<script src="initSES.js"></script>
```

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

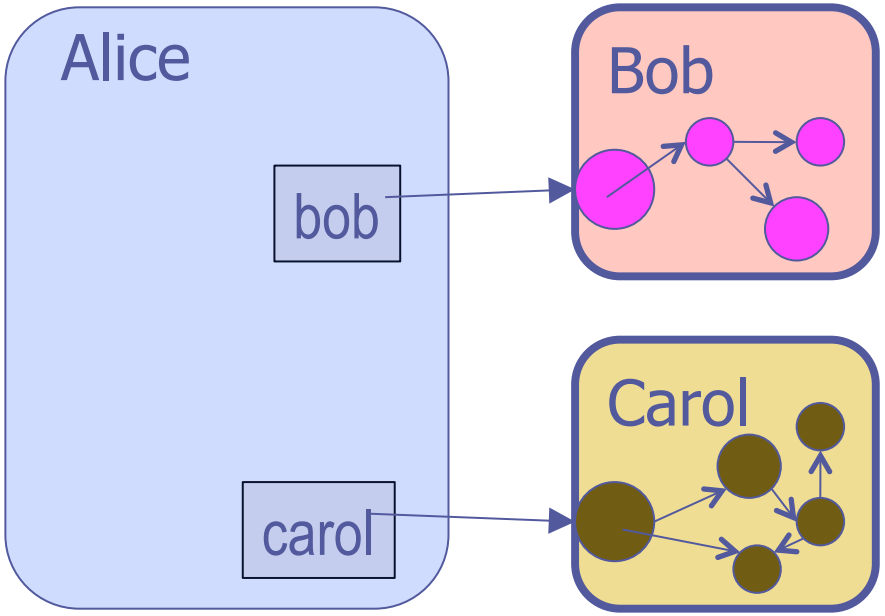
No powerful references by default



Alice says:

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

No powerful references by default



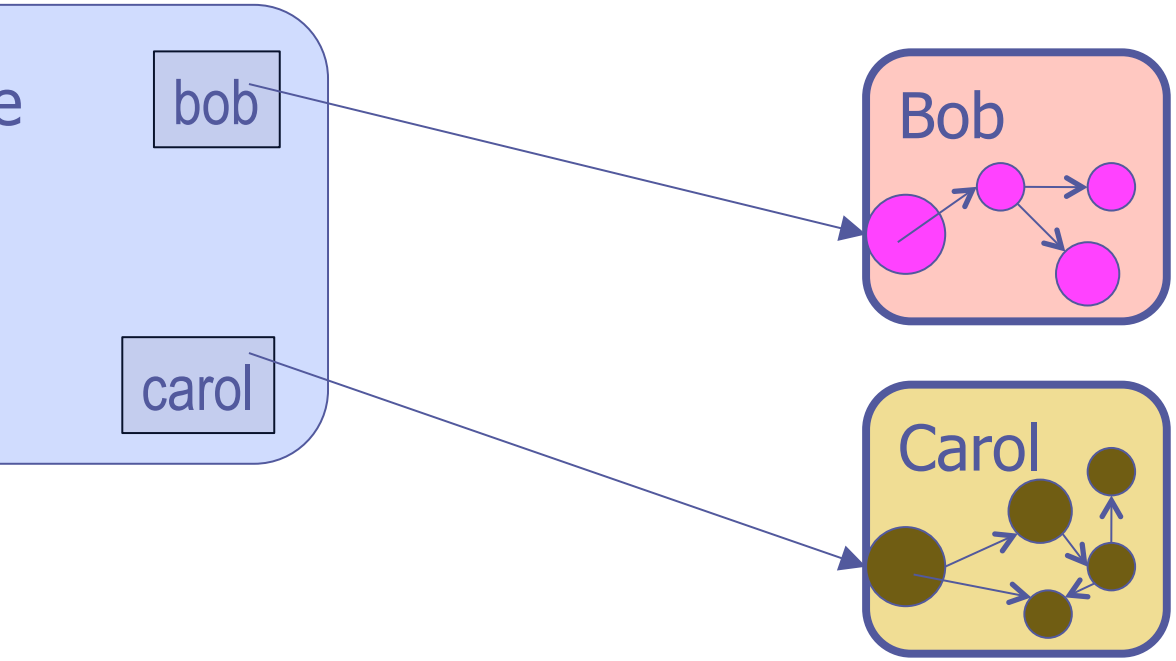
Alice says:

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

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

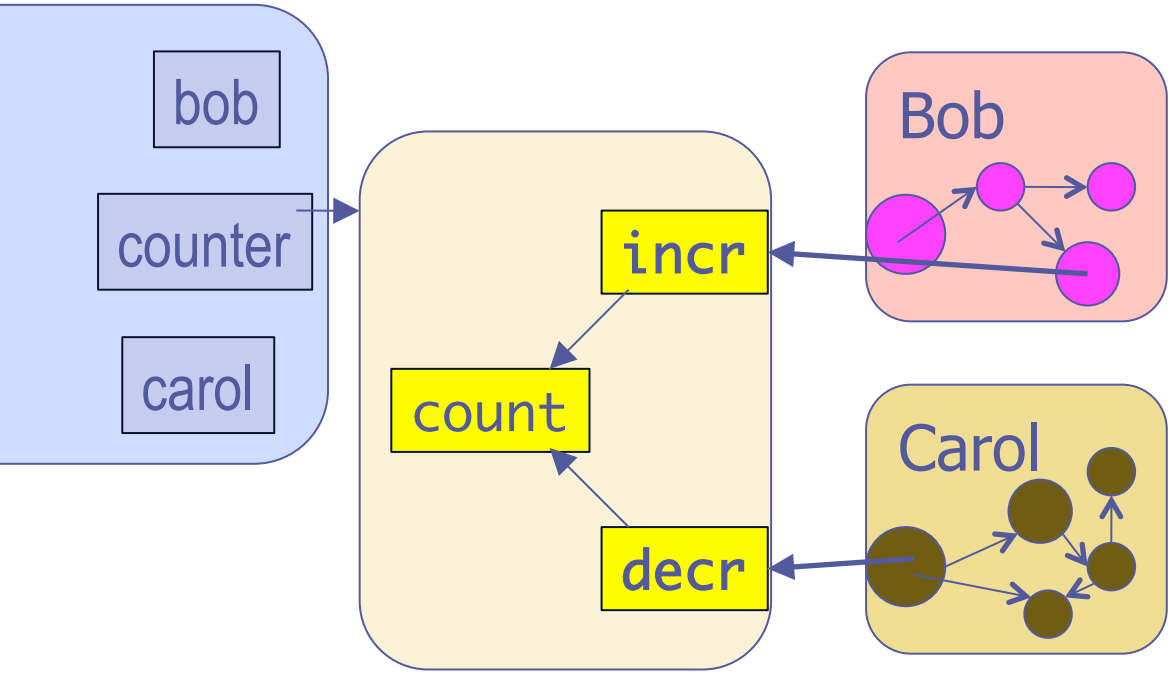
Only Alice controls how they can interact or get more connected.

No powerful references by default



Alice says:

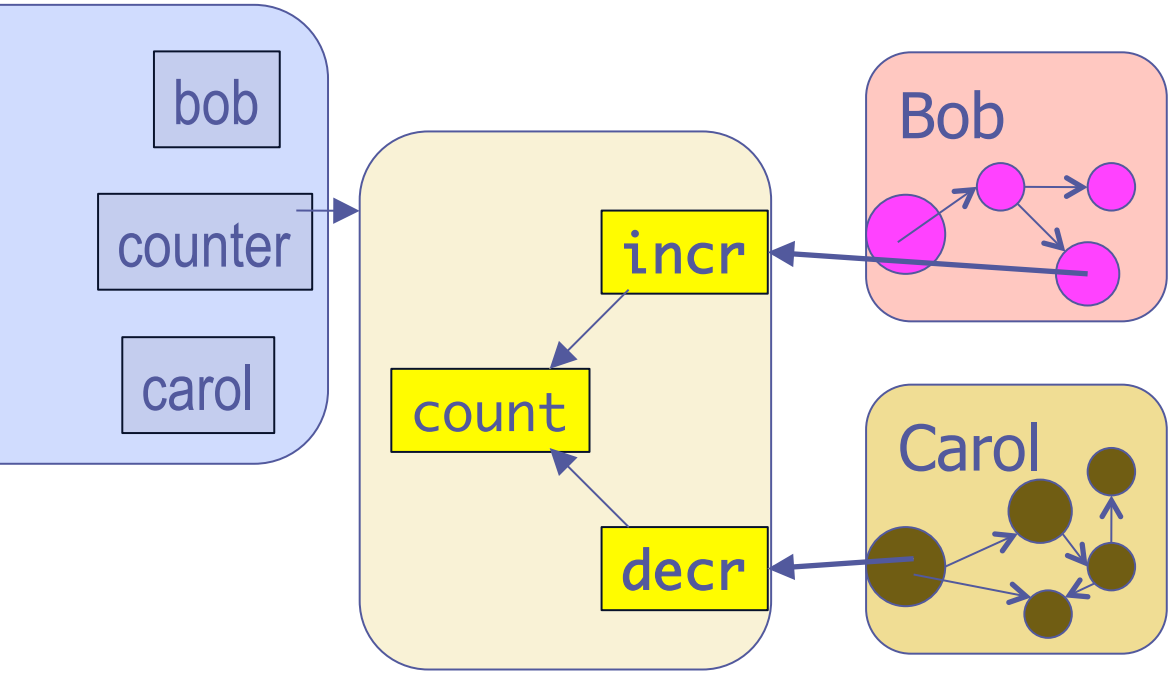
Only connectivity begets connectivity



Alice says:

```
var counter = makeCounter();
bob(counter.incr);
carol(counter.decr);
bob = carol = null;
```

Only connectivity begets connectivity



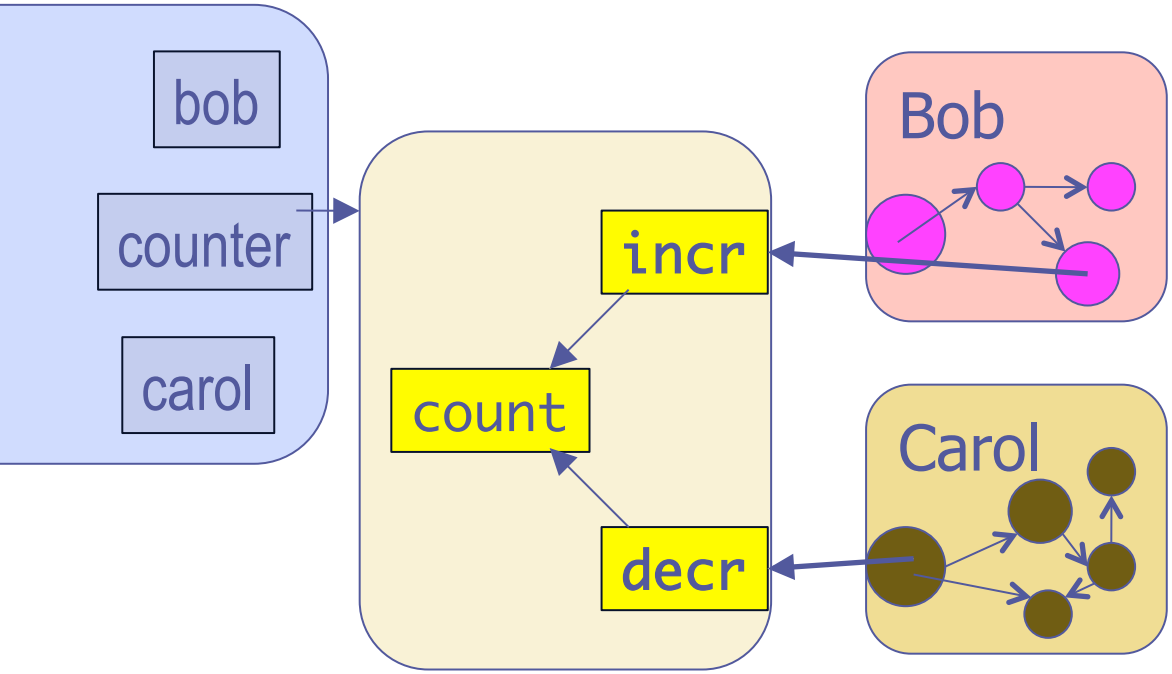
Alice says:

```
var counter = makeCounter();
bob(counter.incr);
carol(counter.decr);
bob = carol = null;
```

Bob can only count up and see result. Carol only down.

Alice can only do both.

Only connectivity begets connectivity



Alice says:

```
var counter = makeCounter();
bob(counter.incr);
carol(counter.decr);
bob = carol = null;
```

Express policy by the behavior of the objects you provide.

Dr. SES
Distributed Resilient Secure EcmaScript

Linguistic abstraction for safe messaging

- Stretch reference graph between event loops & machines
- Crypto analog of memory safety

SES + Promise library + infix "!" syntax

("Q" Library usable today without "!" syntax)

Unguessable URLs as Crypto-Caps

https://www.example.com/app/#mhbqcmmva5ja3

How are secrets like object references?

Dr. SES
Distributed Resilient Secure EcmaScript

var *result* = bob.foo(carol);	Local-only immediate call
var *resultP* = bobP ! foo(carol);	Eventual send

Dr. SES
Distributed Resilient Secure EcmaScript

var *result* = bob.foo(carol);	Local-only immediate call
var *resultP* = bobP ! foo(carol);	Eventual send
var *result* = bob.foo;	Local-only immediate get
var *resultP* = bobP ! foo;	Eventual get

Dr. SES
Distributed Resilient Secure EcmaScript

```
var resultP = bobP ! foo(carol);
```
Eventual send

```
var resultP = bobP ! foo;
```
Eventual get

Dr. SES
Distributed Resilient Secure EcmaScript

var *resultP* = bobP ! foo(carol);	Eventual send
var *resultP* = bobP ! foo;	Eventual get

Dr. SES
Distributed Resilient Secure EcmaScript

```
var resultP = bobP ! foo(carol);
```
Eventual send

```
var resultP = bobP ! foo;
```
Eventual get

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

Async object ops as JSON/REST ops

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

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

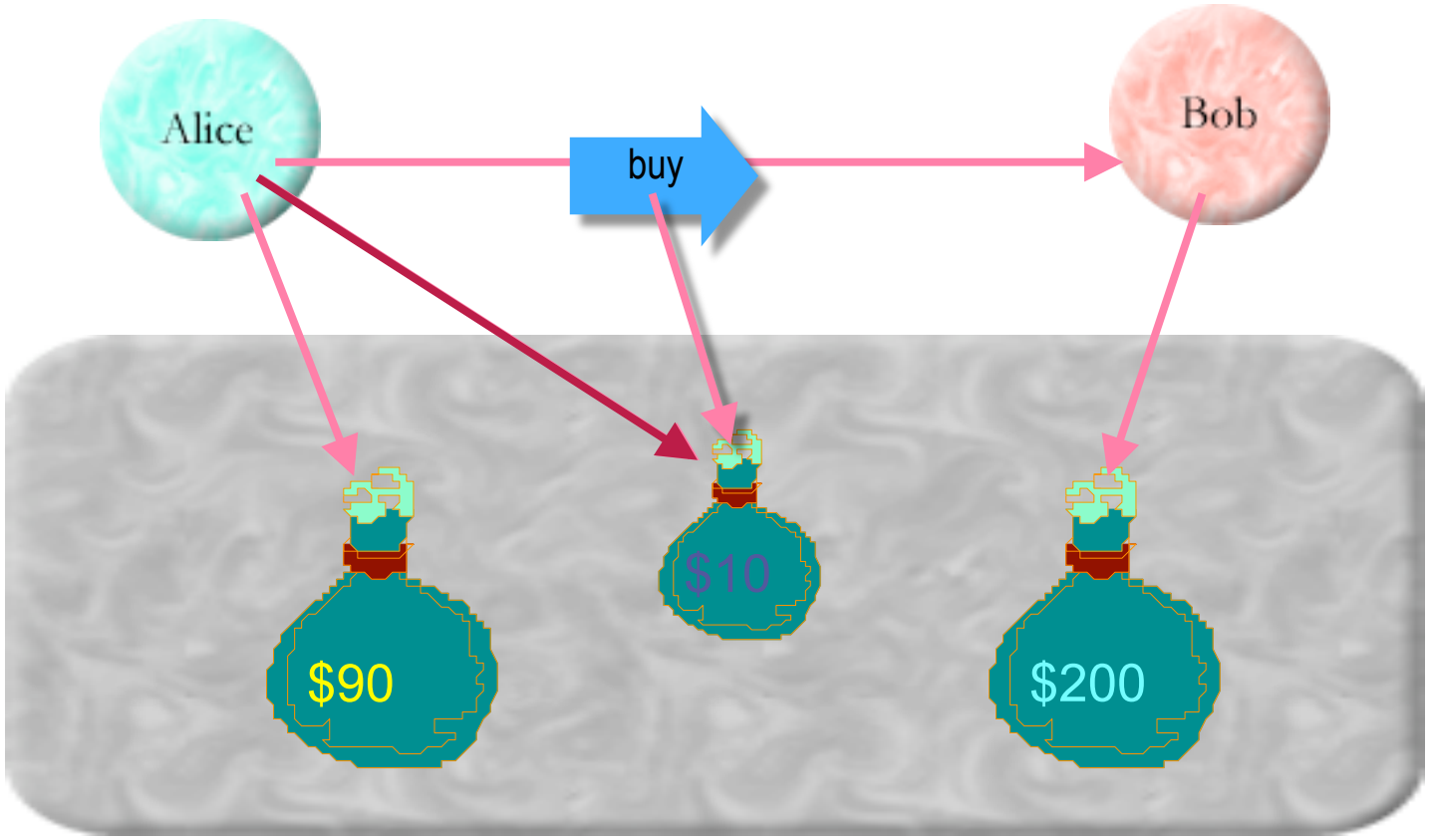
```
POST https://…q=foo {…}
GET https://…q=foo

xhr.onreadystatechange = …
```

Distributed Secure Currency

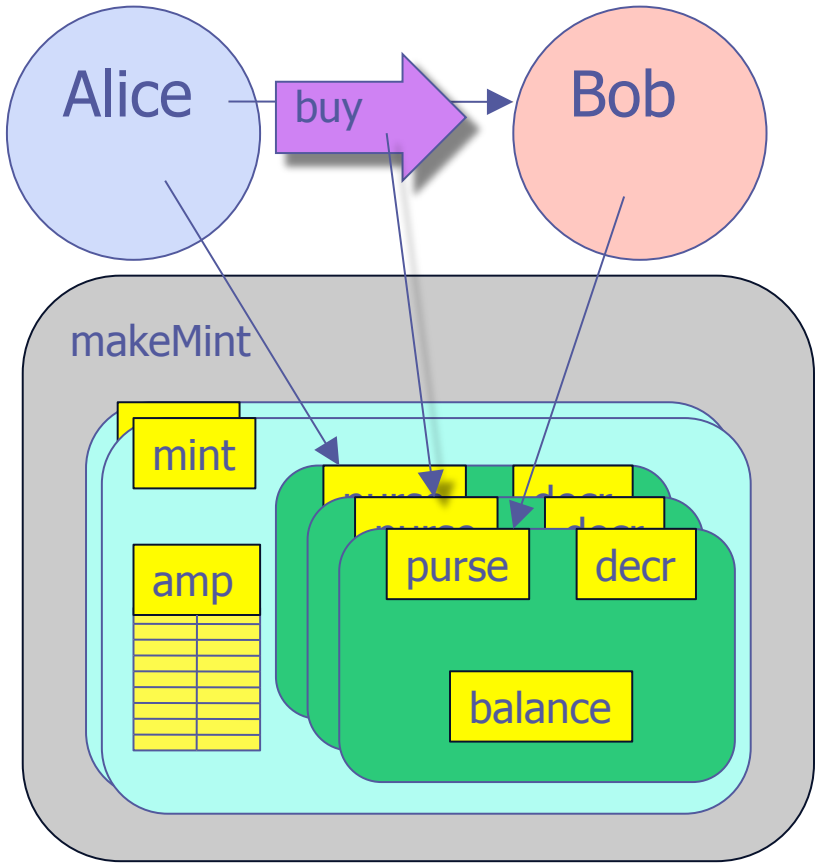
```
var paymentP = myPurse ! makePurse();
paymentP ! deposit(10, myPurse);
var goodP = bobP ! buy(desc, paymentP);
```

```
return Q(paymentP).when(function(p) {
    return Q(myPurse ! deposit(10, p)).when(function(_) {
        return good; }, …
```



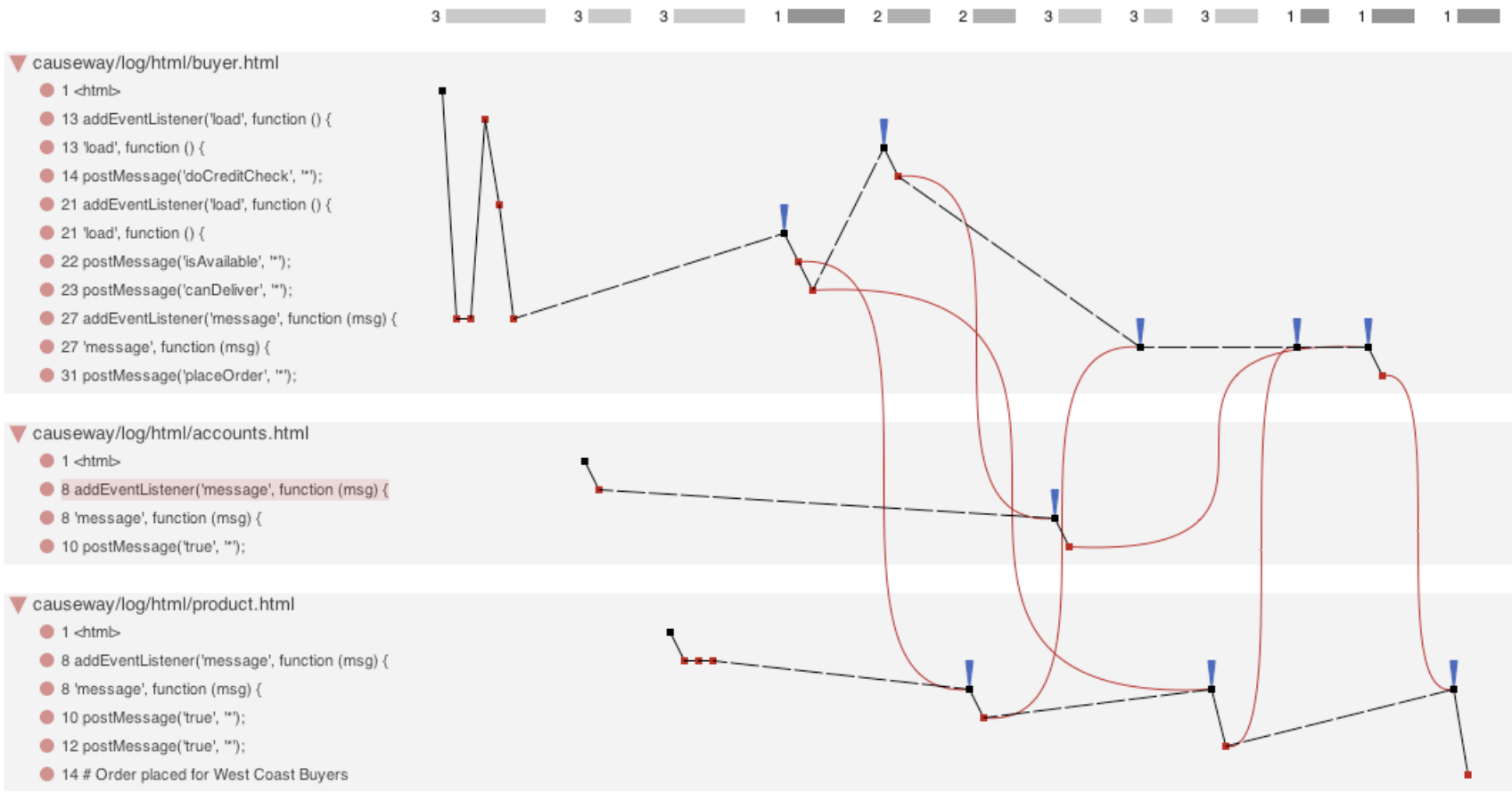
Money as "factorial" of secure coding

No explicit crypto



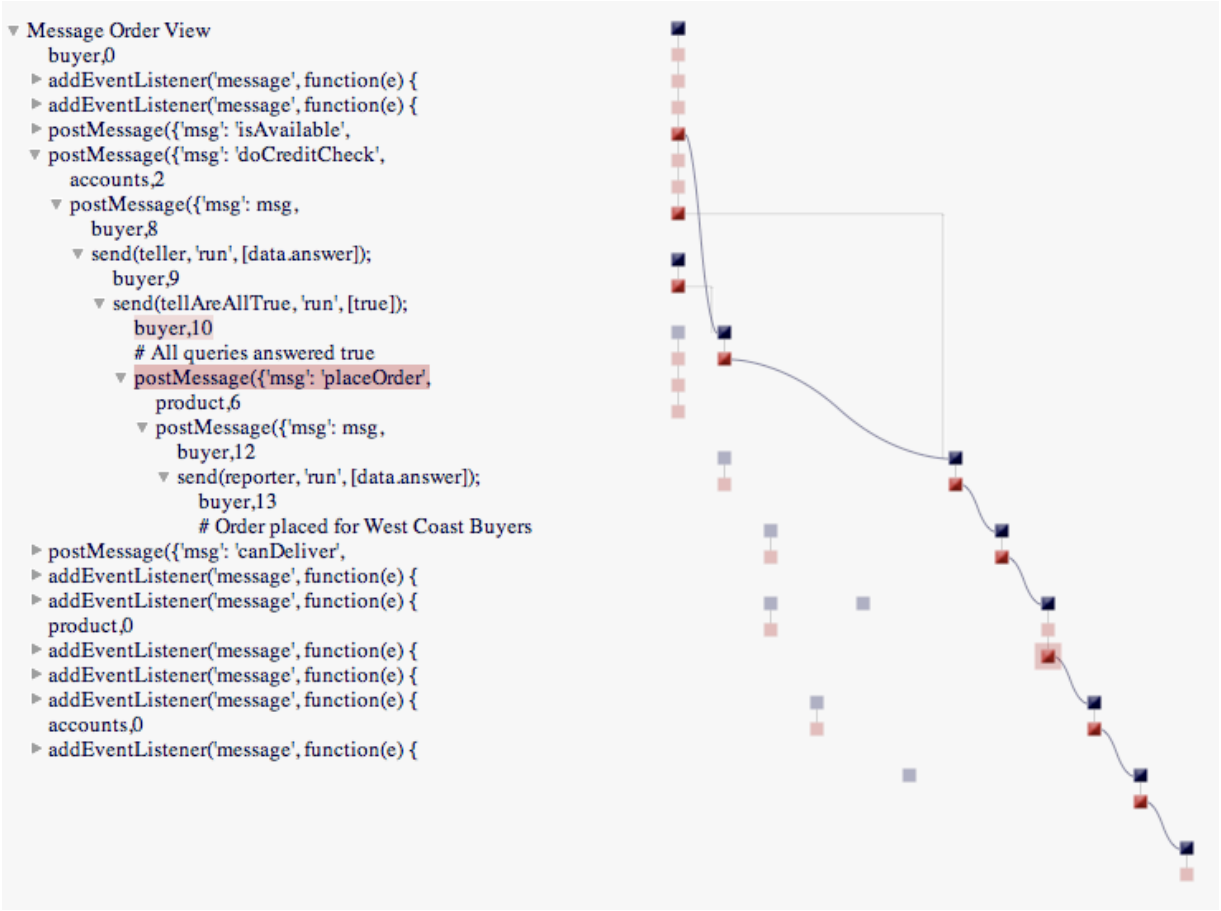

```
function makeMint() {
    var amp = WeakMap();
    return function mint(balance) {
        var purse = def({
            getBalance: function() { return balance; },
            makePurse: function() { return mint(0); },
            deposit: function(amount, src) {
                Nat(balance + amount);
                amp.get(src)(Nat(amount));
                balance += amount;
        } });
        function decr(amount) {
            balance = Nat(balance – amount);
        }
        amp.set(purse, decr);
        return purse;
    }
}
```

Causeway Distributed Debugger



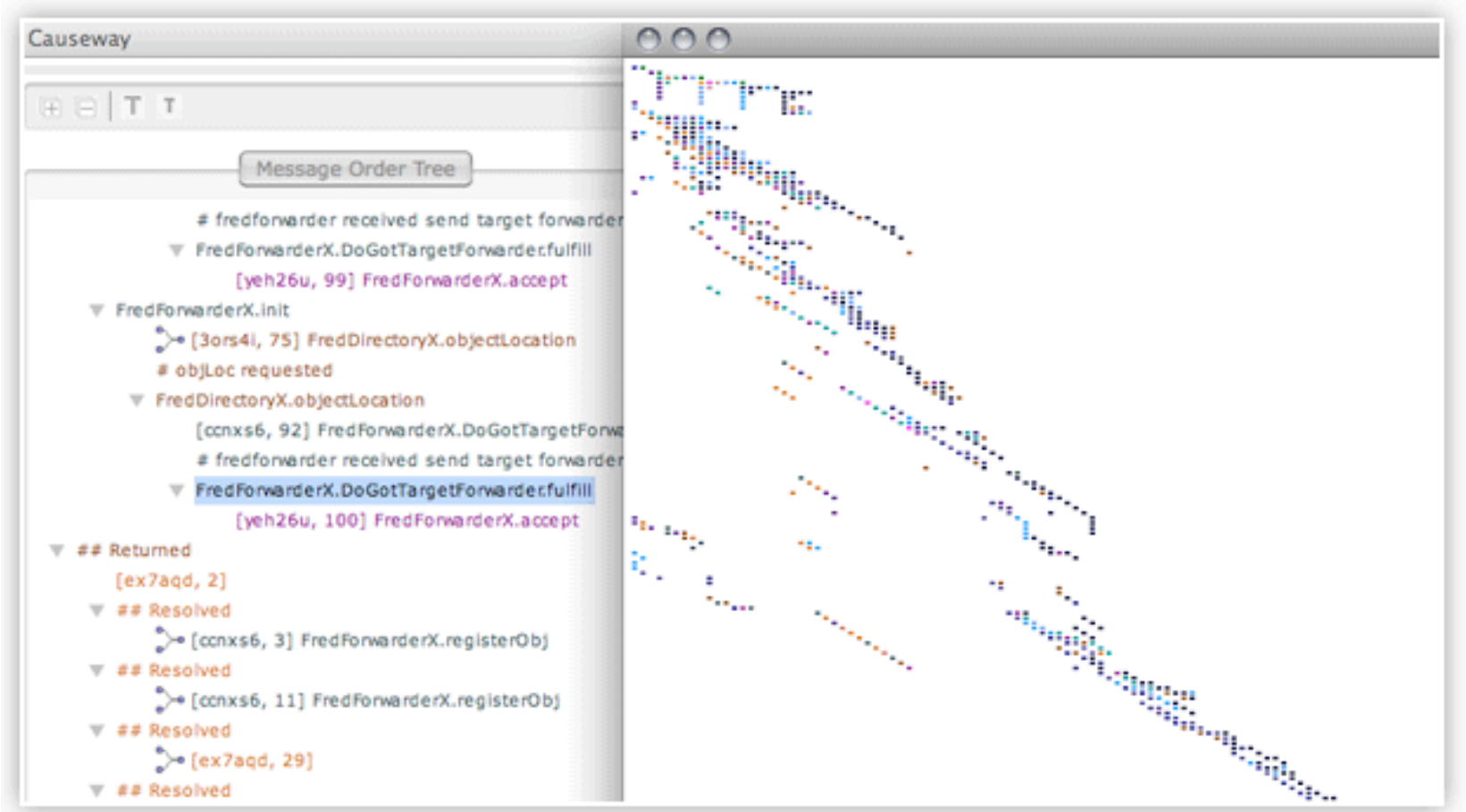
Sourcilloscope

Causeway Distributed Debugger



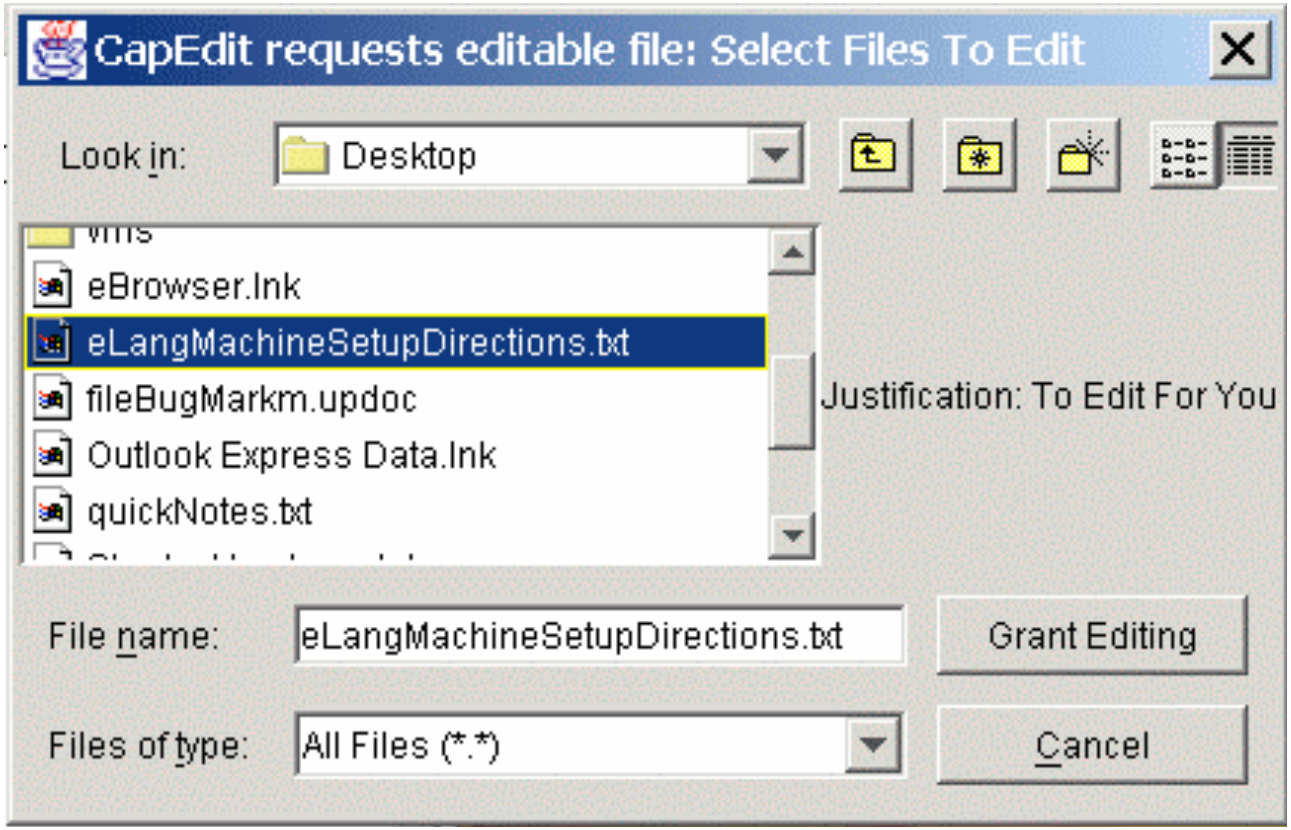
Causality Grid

Causeway Distributed Debugger



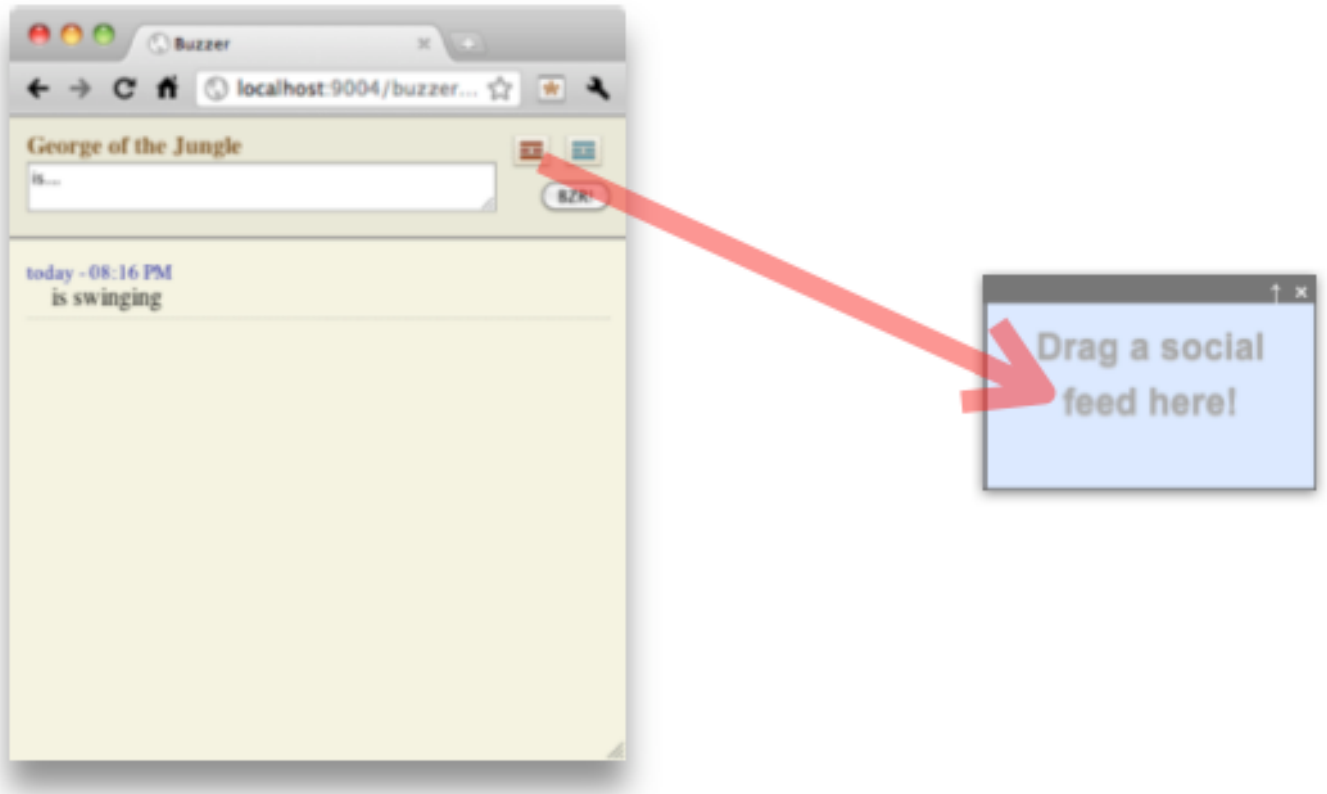
Causality Grid at Scale

Authorization by Designation



CapDesk

Authorization by Designation



Belay

Questions?

- Why object-capability (ocap) security?
- Local ocap security in JavaScript
- Flexible secure mobile code
- Distributed crypto-caps in JavaScript
- Secure distributed programming