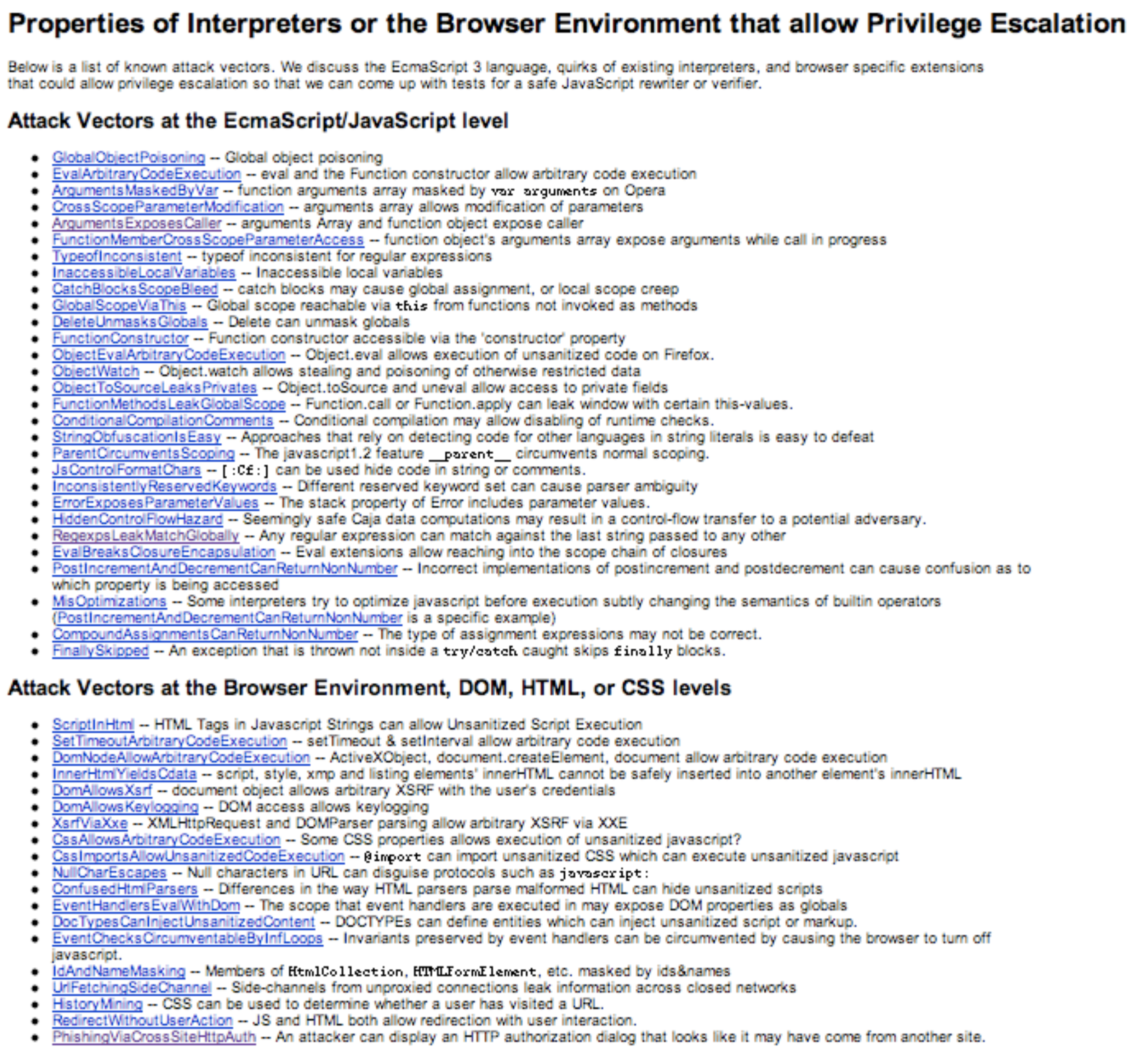
Secure Distributed Programming on EcmaScript 5 + HTML5 platforms

Mark S. Miller and the Cajadores

with thanks to Tyler Close

How to lose an arms race



Properties of Interpreters or the Browser Environment that allow Privilege Escalation

Below is a list of known attack vectors. We discuss the EcmaScript 3 language, quirks of existing interpreters, and browser specific extensions that could allow privilege escalation so that we can come up with tests for a safe JavaScript rewriter or verifier.

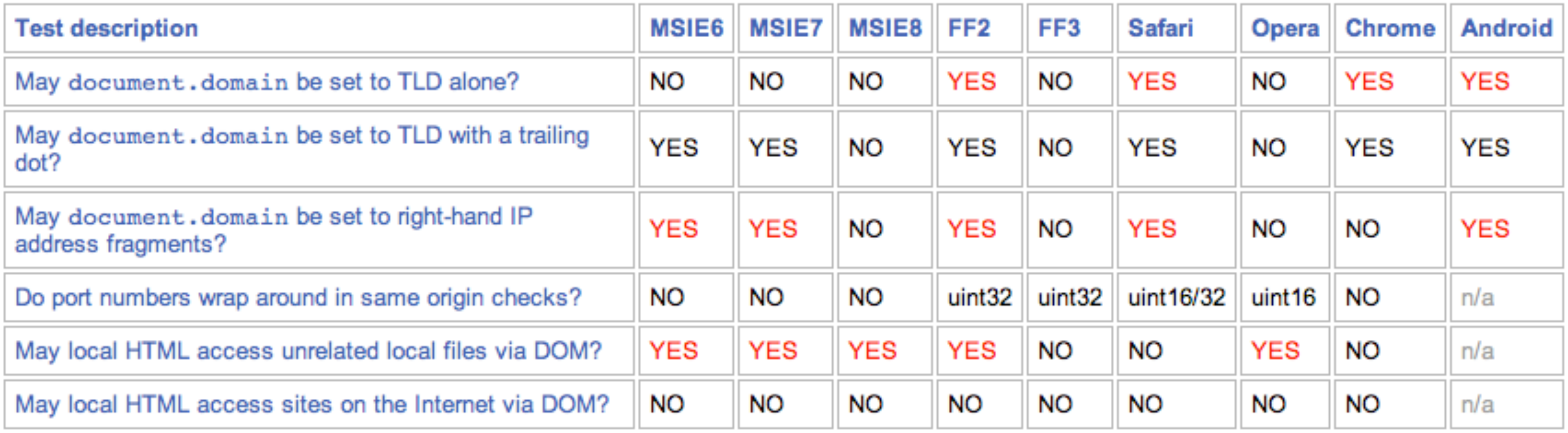
Attack Vectors at the EcmaScript/JavaScript level

- GlobalObjectPoisoning -- Global object poisoning
- EvalArbitraryCodeExecution -- eval and the Function constructor allow arbitrary code execution
- ArgumentsMaskedByVar -- function arguments array masked by `var arguments` on Opera
- CrossScopeParameterModification -- arguments array allows modification of parameters
- ArgumentsExposesCaller -- arguments Array and function object expose caller
- FunctionMemberCrossScopeParameterAccess -- function object's arguments array expose arguments while call in progress
- TypeofInconsistent -- typeof inconsistent for regular expressions
- InaccessibleLocalVariables -- Inaccessible local variables
- CatchBlocksScopeBleed -- catch blocks may cause global assignment, or local scope creep
- GlobalScopeViaThis -- Global scope reachable via `this` from functions not invoked as methods
- DeleteUnmasksGlobals -- Delete can unmask globals
- FunctionConstructor -- Function constructor accessible via the 'constructor' property
- ObjectEvalArbitraryCodeExecution -- Object.eval allows execution of unsanitized code on Firefox.
- ObjectWatch -- Object.watch allows stealing and poisoning of otherwise restricted data
- ObjectToSourceLeaksPrivates -- Object.toSource and uneval allow access to private fields
- FunctionMethodsLeakGlobalScope -- Function.call or Function.apply can leak window with certain this-values.
- ConditionalCompilationComments -- Conditional compilation may allow disabling of runtime checks.
- StringObfuscationIsEasy -- Approaches that rely on detecting code for other languages in string literals is easy to defeat
- ParentCircumventsScoping -- The javascript1.2 feature `__parent__` circumvents normal scoping.
- JsControlFormatChars -- `[:Cf:]` can be used hide code in string or comments.
- InconsistentlyReservedKeywords -- Different reserved keyword set can cause parser ambiguity
- ErrorExposesParameterValues -- The stack property of Error includes parameter values.
- HiddenControlFlowHazard -- Seemingly safe Caja data computations may result in a control-flow transfer to a potential adversary.
- RegexpsLeakMatchGlobally -- Any regular expression can match against the last string passed to any other
- EvalBreaksClosureEncapsulation -- Eval extensions allow reaching into the scope chain of closures
- PostIncrementAndDecrementCanReturnNonNumber -- Incorrect implementations of postincrement and postdecrement can cause confusion as to which property is being accessed
- MisOptimizations -- Some interpreters try to optimize javascript before execution subtly changing the semantics of builtin operators (PostIncrementAndDecrementCanReturnNonNumber is a specific example)
- CompoundAssignmentsCanReturnNonNumber -- The type of assignment expressions may not be correct.
- FinallySkipped -- An exception that is thrown not inside a `try/catch` caught skips `finally` blocks.

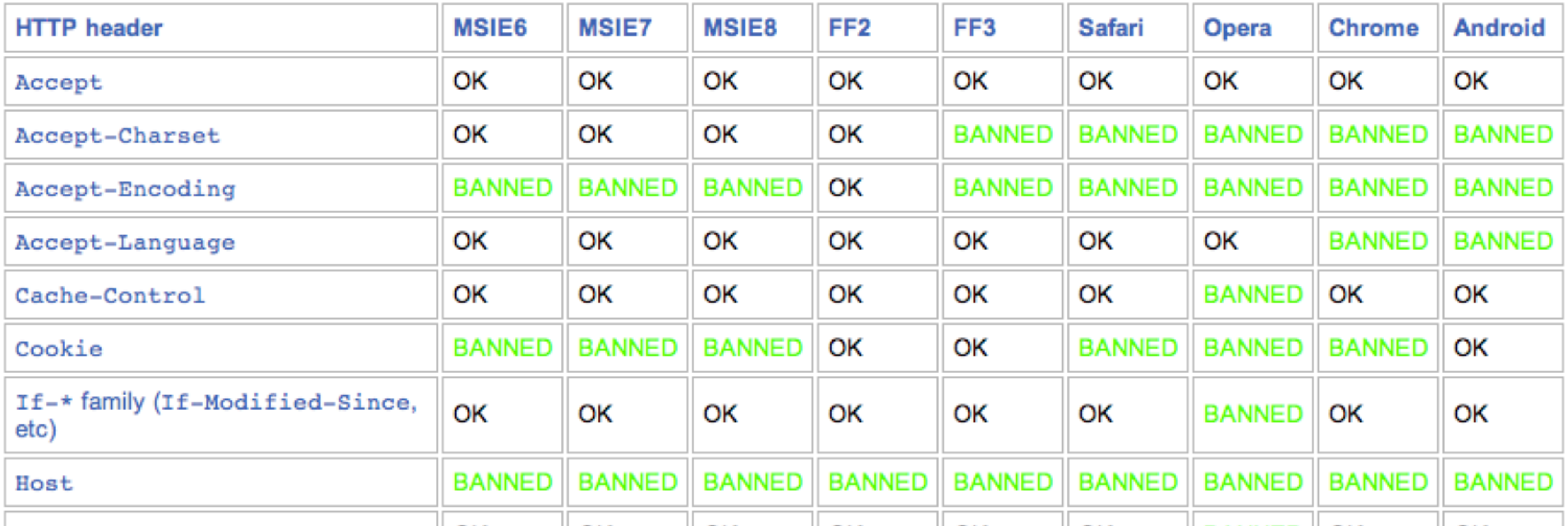
Attack Vectors at the Browser Environment, DOM, HTML, or CSS levels

- ScriptInHtml -- HTML Tags in Javascript Strings can allow Unsanitized Script Execution
- SetTimeoutArbitraryCodeExecution -- setTimeout & setInterval allow arbitrary code execution
- DomNodeAllowArbitraryCodeExecution -- ActiveXObject, document.createElement, document allow arbitrary code execution
- InnerHtmlYieldsCdata -- script, style, xmp and listing elements' innerHTML cannot be safely inserted into another element's innerHTML
- DomAllowsXsrf -- document object allows arbitrary XSRF with the user's credentials
- DomAllowsKeylogging -- DOM access allows keylogging
- XsrfViaXxe -- XMLHttpRequest and DOMParser parsing allow arbitrary XSRF via XXE
- CssAllowsArbitraryCodeExecution -- Some CSS properties allows execution of unsanitized javascript?
- CssImportsAllowUnsanitizedCodeExecution -- `@import` can import unsanitized CSS which can execute unsanitized javascript
- NullCharEscapes -- Null characters in URL can disguise protocols such as `javascript:`
- ConfusedHtmlParsers -- Differences in the way HTML parsers parse malformed HTML can hide unsanitized scripts
- EventHandlersEvalWithDom -- The scope that event handlers are executed in may expose DOM properties as globals
- DocTypesCanInjectUnsanitizedContent -- DOCTYPEs can define entities which can inject unsanitized script or markup.
- EventChecksCircumventableByInfLoops -- Invariants preserved by event handlers can be circumvented by causing the browser to turn off javascript.
- IdAndNameMasking -- Members of `HtmlCollection`, `HTMLFormElement`, etc. masked by ids&names
- UrlFetchingSideChannel -- Side-channels from unproxied connections leak information across closed networks
- HistoryMining -- CSS can be used to determine whether a user has visited a URL.
- RedirectWithoutUserAction -- JS and HTML both allow redirection with user interaction.
- PhishingViaCrossSiteHttpAuth -- An attacker can display an HTTP authorization dialog that looks like it may have come from another site.

How to lose an arms race

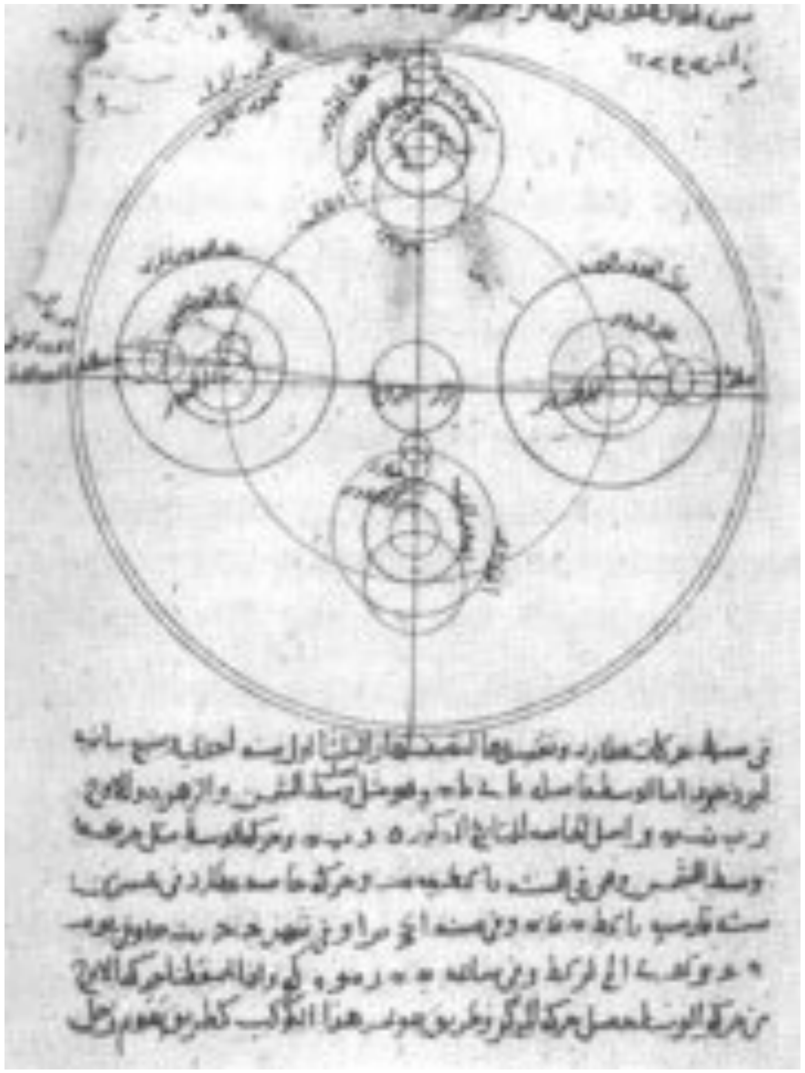


Test description	MSIE6	MSIE7	MSIE8	FF2	FF3	Safari	Opera	Chrome	Android
May `document.domain` be set to TLD alone?	NO	NO	NO	YES	NO	YES	NO	YES	YES
May `document.domain` be set to TLD with a trailing dot?	YES	YES	NO	YES	NO	YES	NO	YES	YES
May `document.domain` be set to right-hand IP address fragments?	YES	YES	NO	YES	NO	YES	NO	NO	YES
Do port numbers wrap around in same origin checks?	NO	NO	NO	uint32	uint32	uint16/32	uint16	NO	n/a
May local HTML access unrelated local files via DOM?	YES	YES	YES	YES	NO	NO	YES	NO	n/a
May local HTML access sites on the Internet via DOM?	NO	NO	NO	NO	NO	NO	NO	NO	n/a

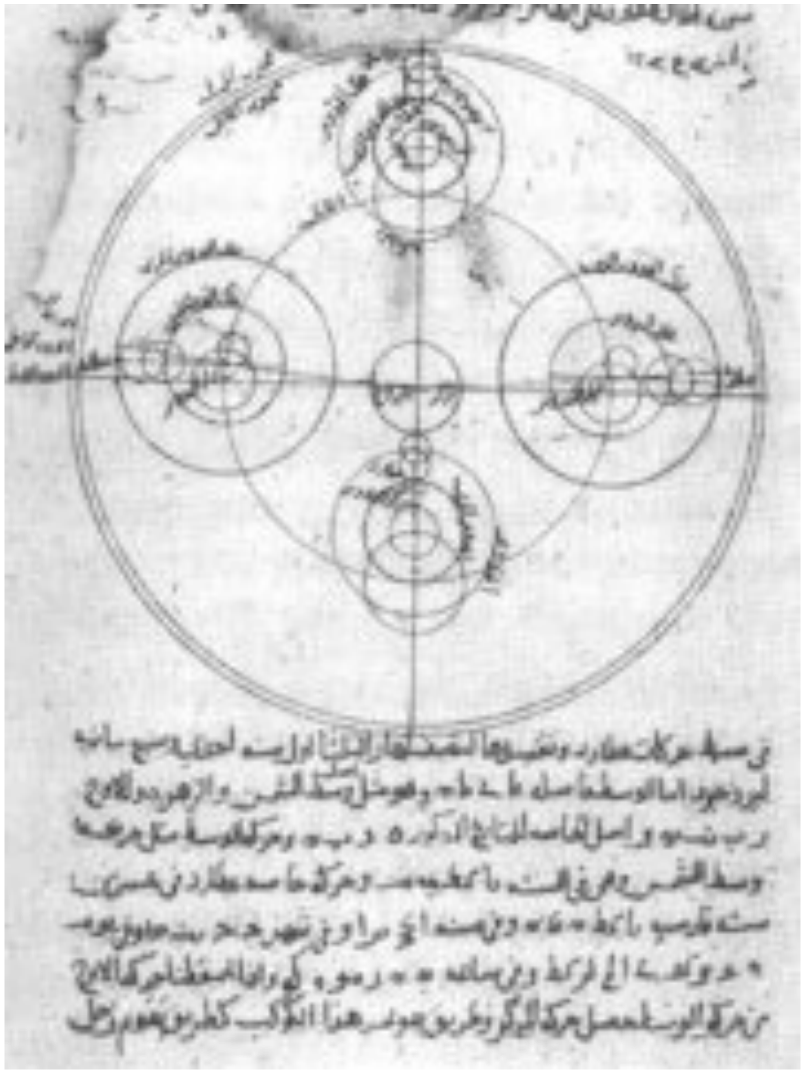


HTTP header	MSIE6	MSIE7	MSIE8	FF2	FF3	Safari	Opera	Chrome	Android
`Accept`	OK	OK	OK	OK	OK	OK	OK	OK	OK
`Accept-Charset`	OK	OK	OK	OK	BANNED	BANNED	BANNED	BANNED	BANNED
`Accept-Encoding`	BANNED	BANNED	BANNED	OK	BANNED	BANNED	BANNED	BANNED	BANNED
`Accept-Language`	OK	OK	OK	OK	OK	OK	OK	BANNED	BANNED
`Cache-Control`	OK	OK	OK	OK	OK	OK	BANNED	OK	OK
`Cookie`	BANNED	BANNED	BANNED	OK	OK	BANNED	BANNED	BANNED	OK
`If-*` family (`If-Modified-Since`, etc)	OK	OK	OK	OK	OK	OK	BANNED	OK	OK
`Host`	BANNED	BANNED	BANNED	BANNED	BANNED	BANNED	BANNED	BANNED	BANNED

Doomed to never ending tinkering?



Doomed to never ending tinkering?

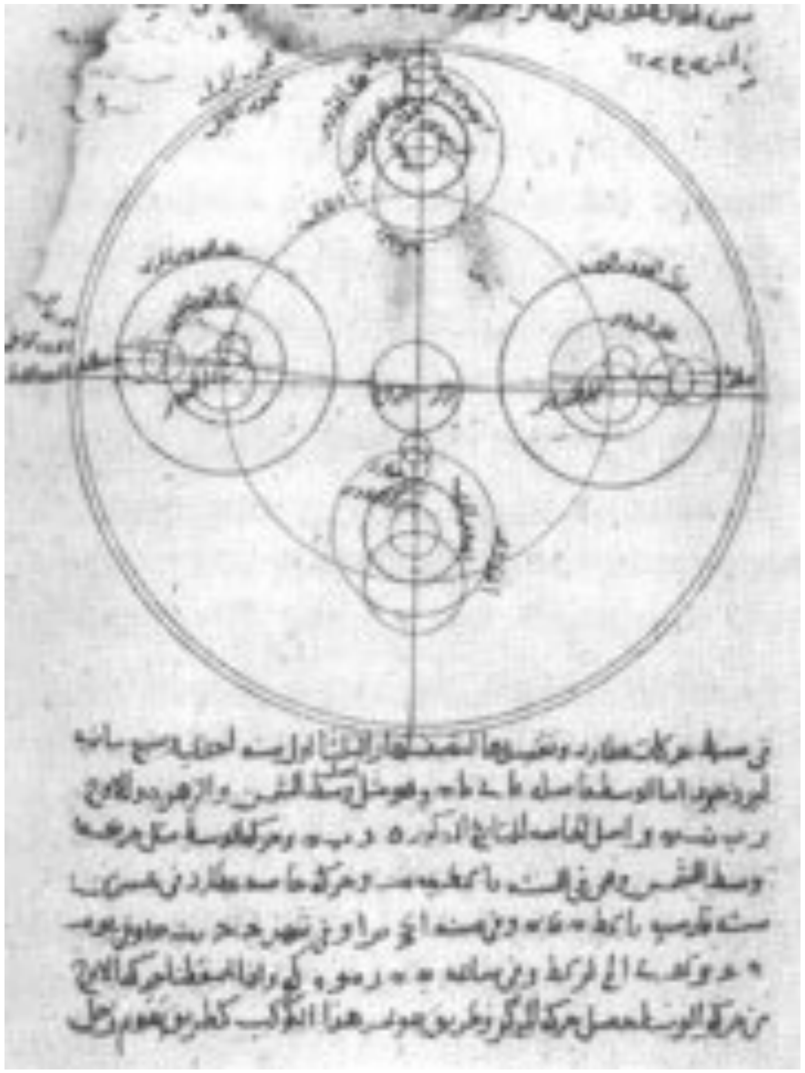


Identity-centric access
- HTTP auth info
- client side certs
- script, img, fragment holes

Cookies
- augments attacker's authority
- → confused deputies

Origin: header "fix"
- → subtler confused deputies

Doomed to never ending tinkering?



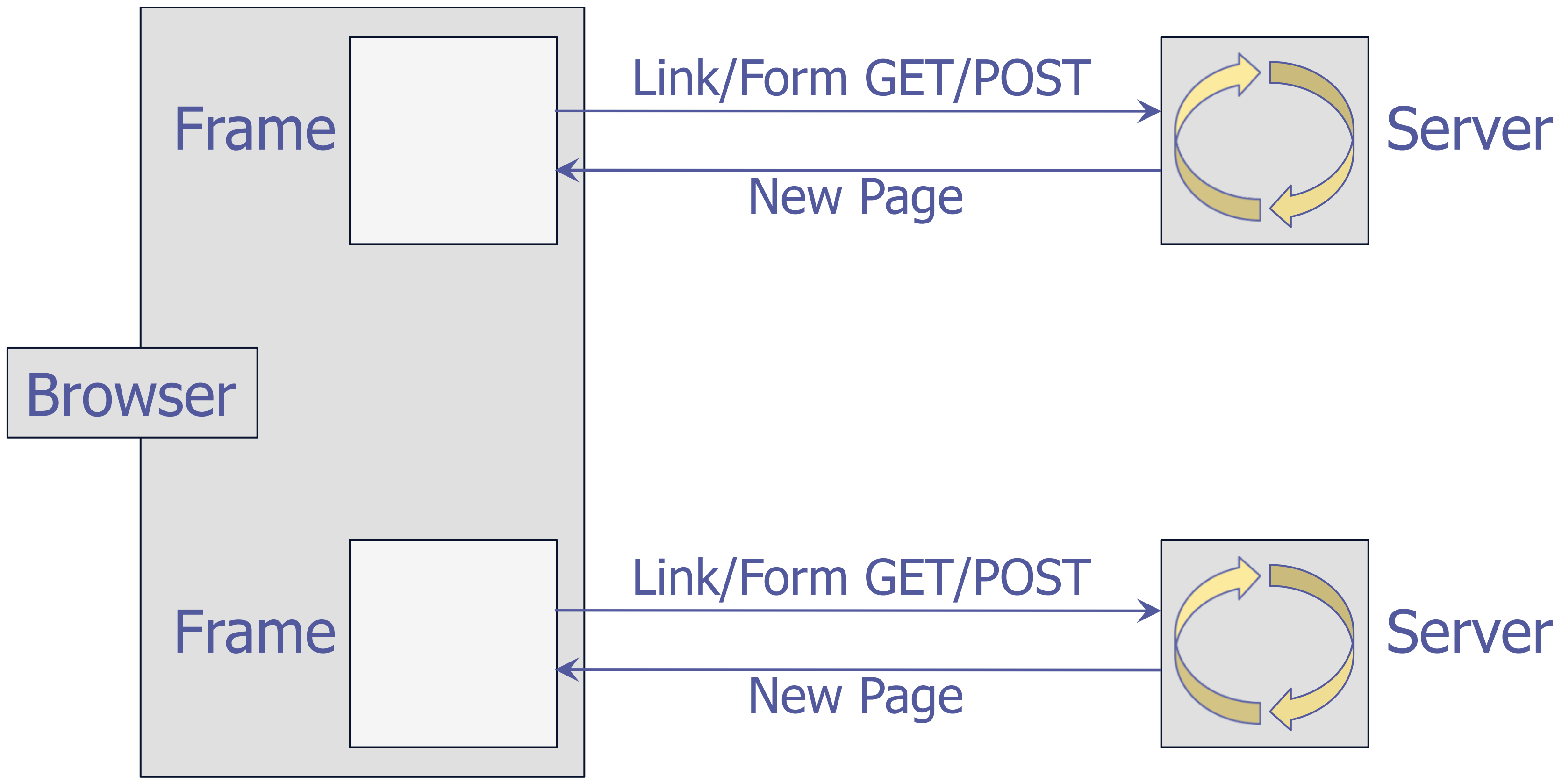
Identity-centric access
- HTTP auth info
- client side certs
- script, img, fragment holes

Cookies
- augments attacker's authority
- → confused deputies

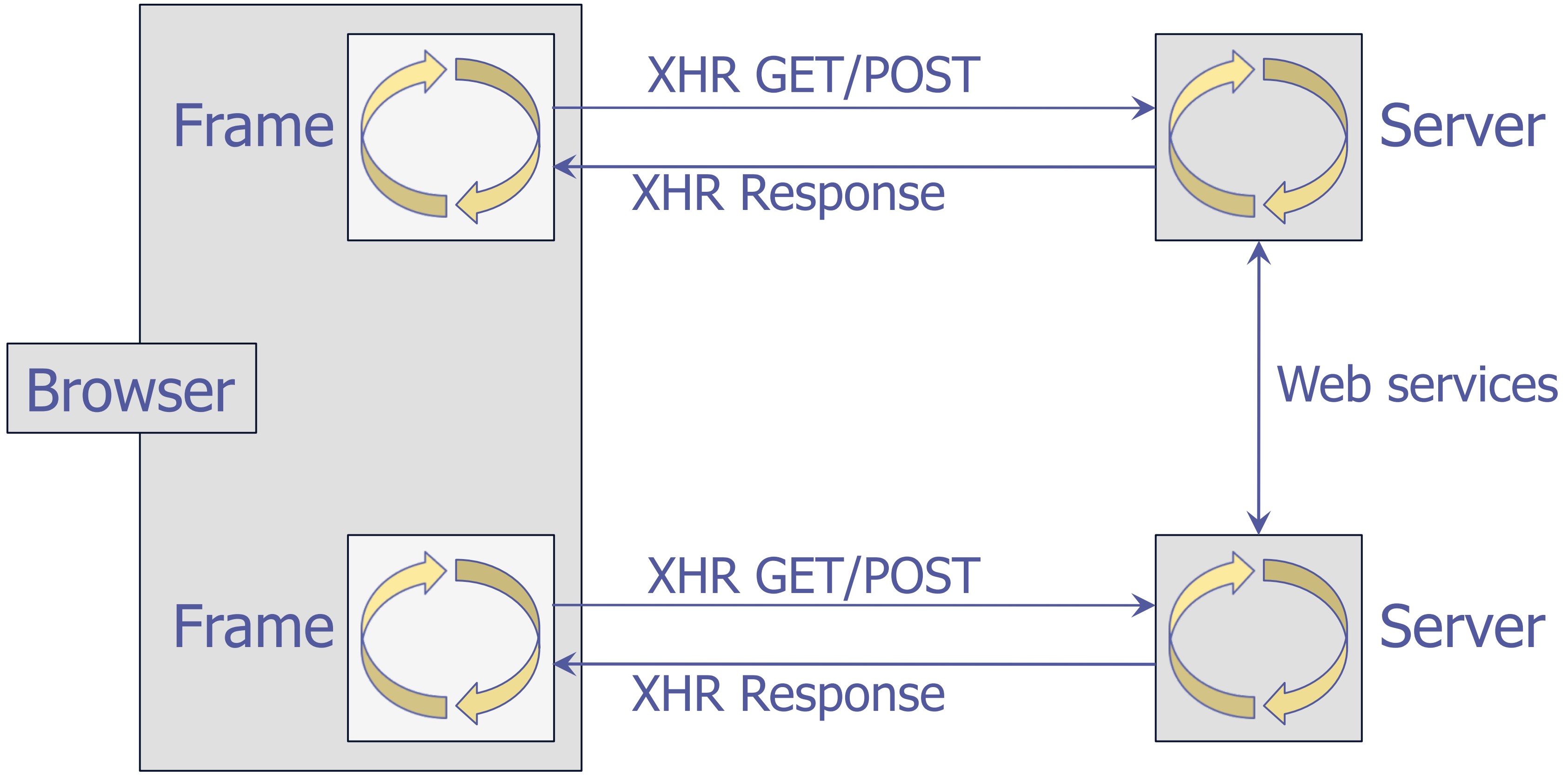
Origin: header "fix"
- → subtler confused deputies

Identity-centric *vs.* Authorization-centric

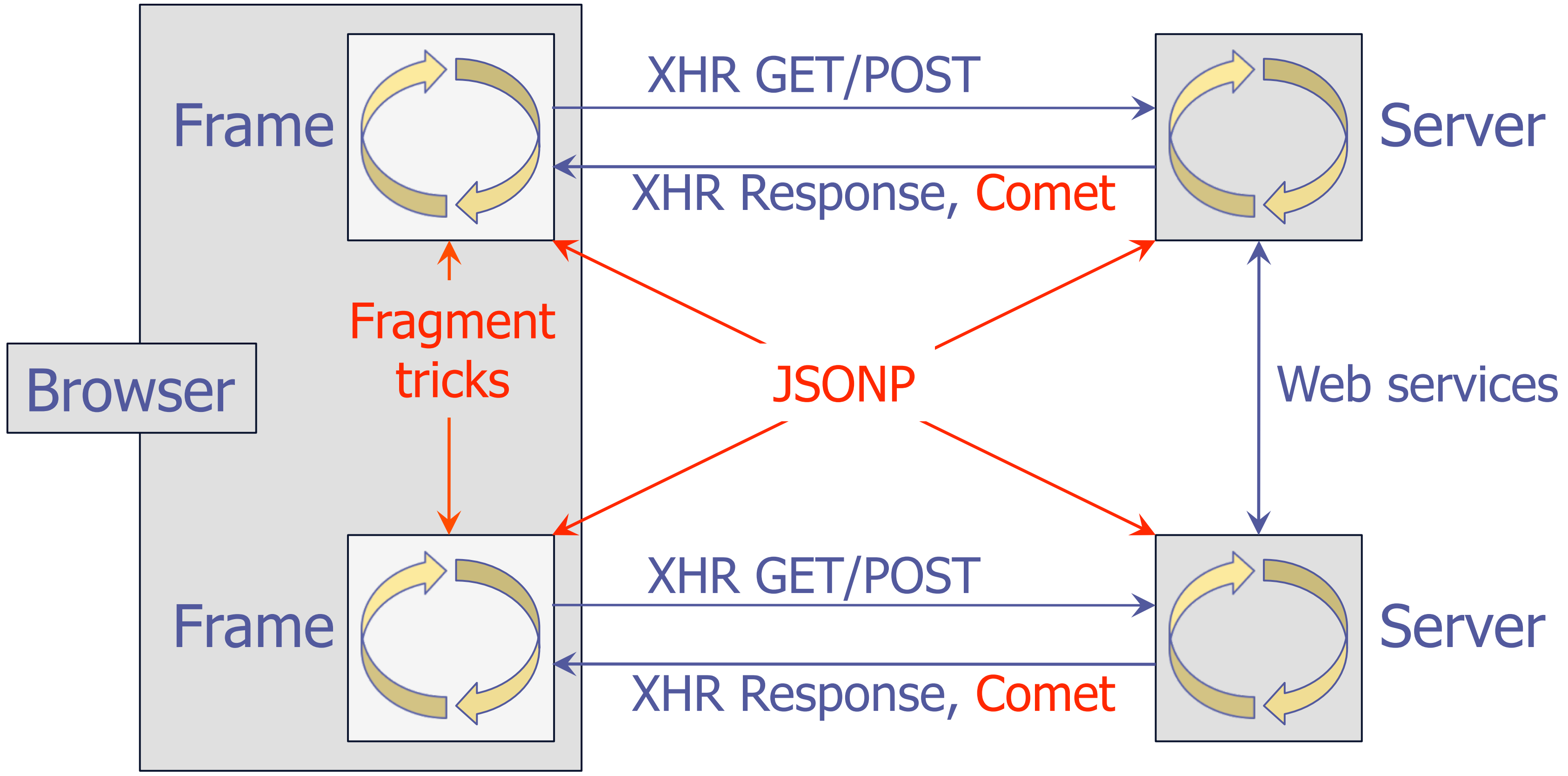
Original Web



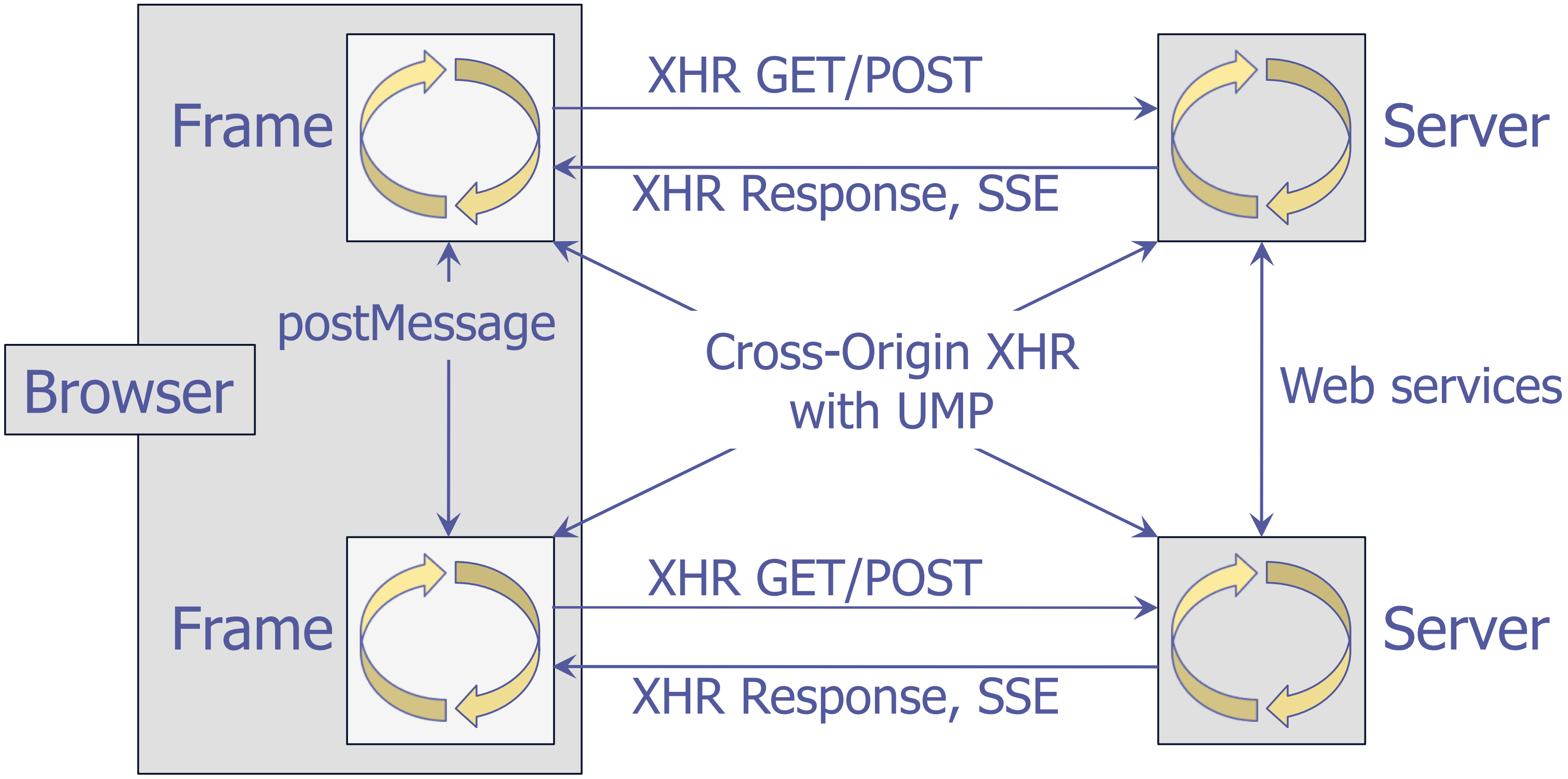
Ajax = Mobile code + async msgs



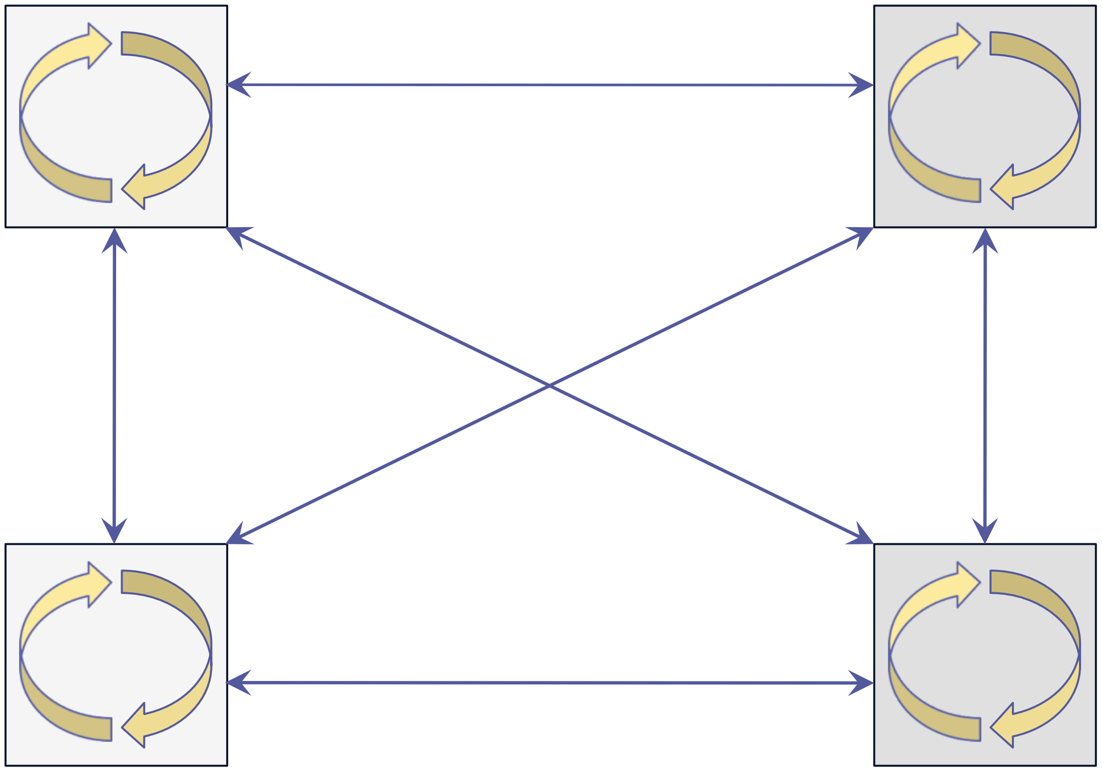
Kludging Towards Distributed Objects



A Web of Distributed Objects



A Web of Distributed Objects



Mobile messages, code, objects

Safe Mobile Messages: *Uniform* XHR

- As in "*Uniform* Resource Locator"
  - Designation (ideally) independent of requestor context
- Ignore browser's "helpful" extras
  - HTTP Auth info, client side certs, cookies, Origin: header,
  - Like IP address: use only for forensics & emergencies
- Authorize based only on payload
  - HTTPS URL or request body – info the requestor ***knows***
- Waive response "protection"
  - **`Access-Control-Allow-Origin: *`**

Safe Mobile Code: OCaps in JavaScript

EcmaScript 3:

One of the hardest oo languages to secure.

Caja: Complex server-side translator. Runtime overhead.

EcmaScript 5:

One of the easiest oo languages to secure.

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

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

Approx 5K download compressed.

Security as Extreme Modularity

Modularity: Avoid needless dependencies

Security: Avoid needless vulnerabilities

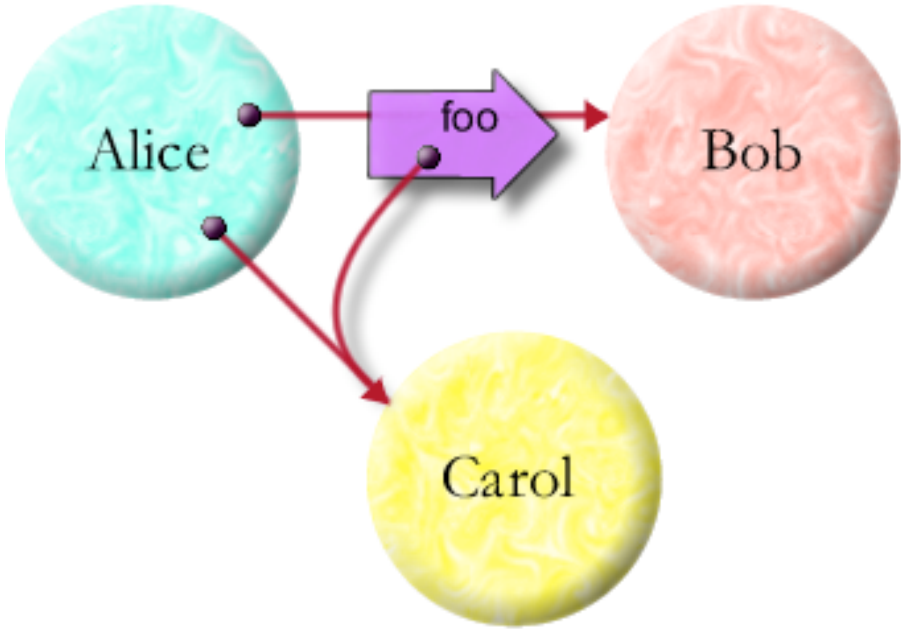
Vulnerability is a form of dependency

Mod: Principle of info hiding - need to know.

Sec: 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

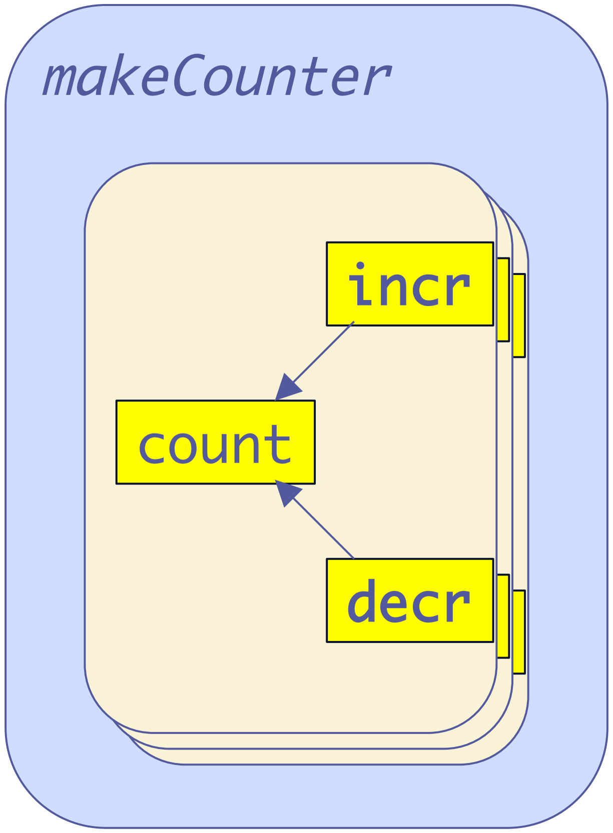
Reference graph ≡ Access graph

Only connectivity begets connectivity

Natural *Least Authority*

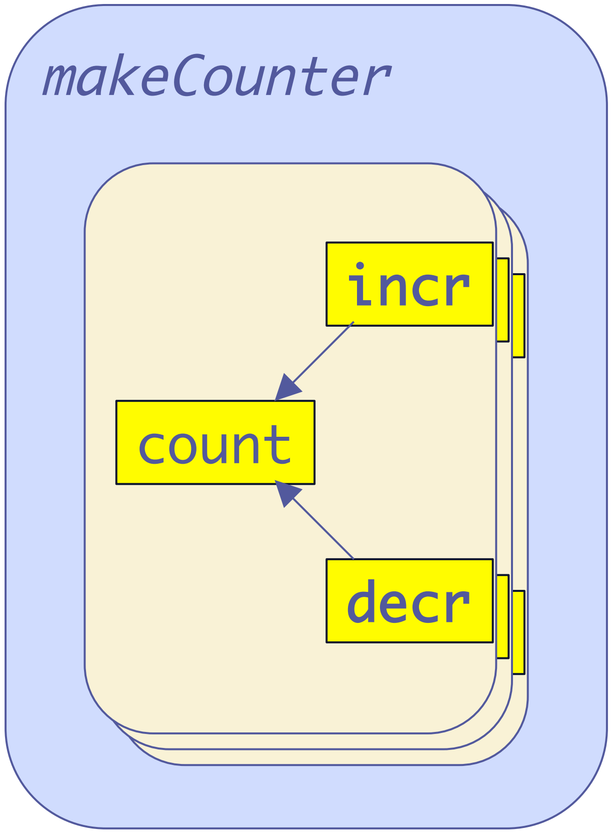
OO expressiveness for security patterns

Objects as Closures



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

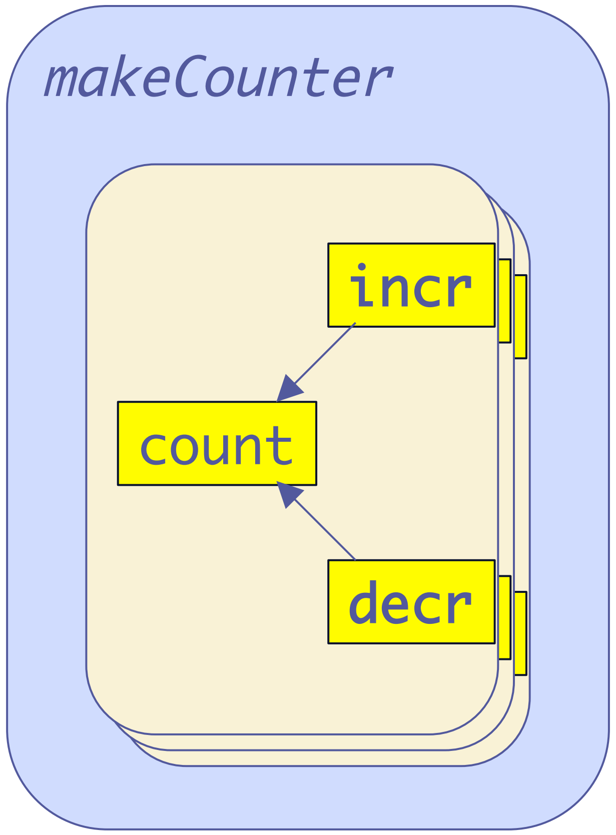
Objects as Closures



```
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 ES5/strict



```
“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

Turning ES5 into SES

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

Monkey patch away bad non-std behaviors

Remove non-whitelisted primordials

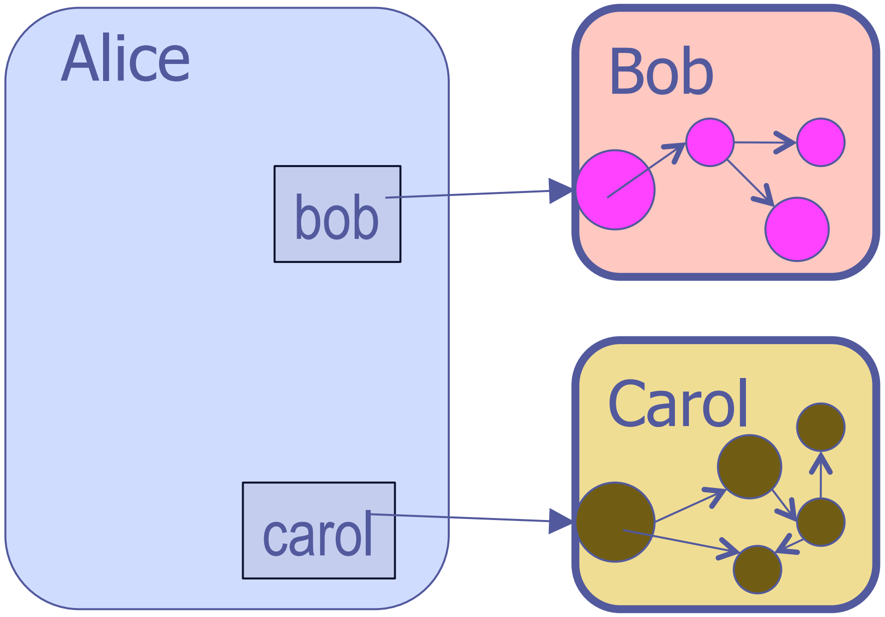
Install leaky **WeakMap** emulation

Make virtual global **root**

Freeze whitelisted global variables

- Replace **eval** & **Function** with safe alternatives
- Freeze accessible primordials

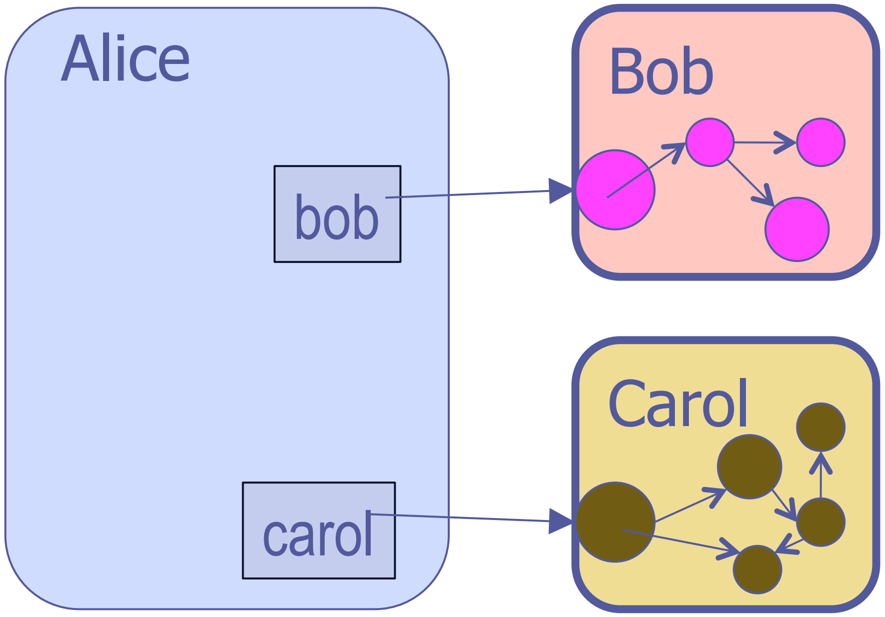
No powerful references by default



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

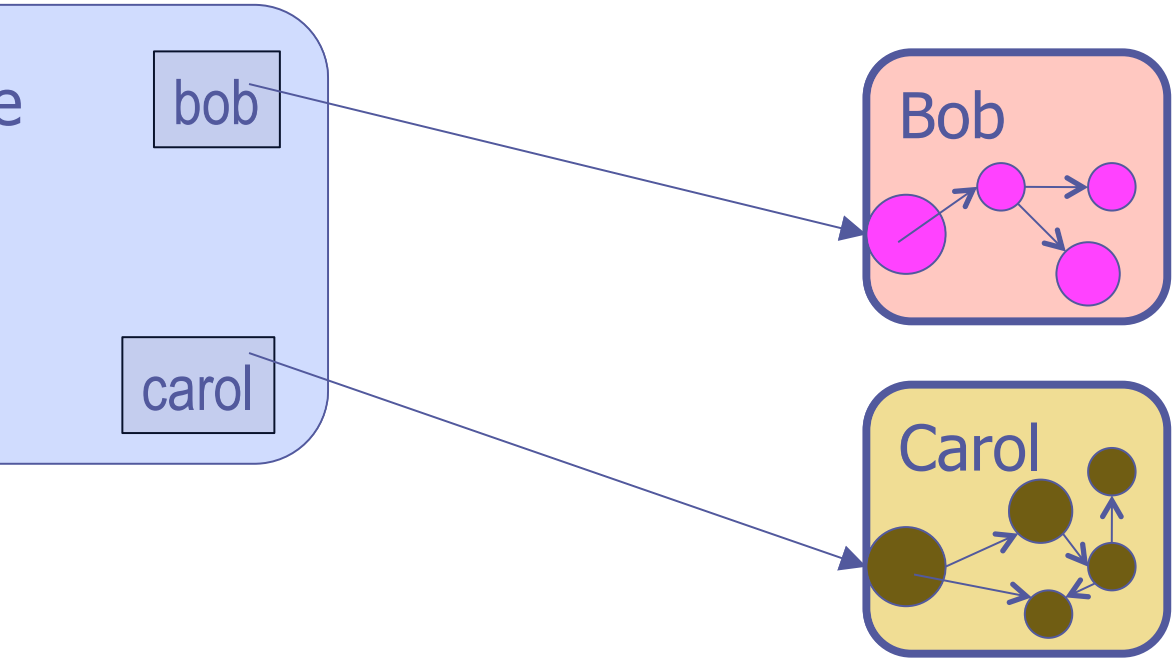
var *bob* = **eval**(bobSrc);

var *carol* = **eval**(carolSrc);

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

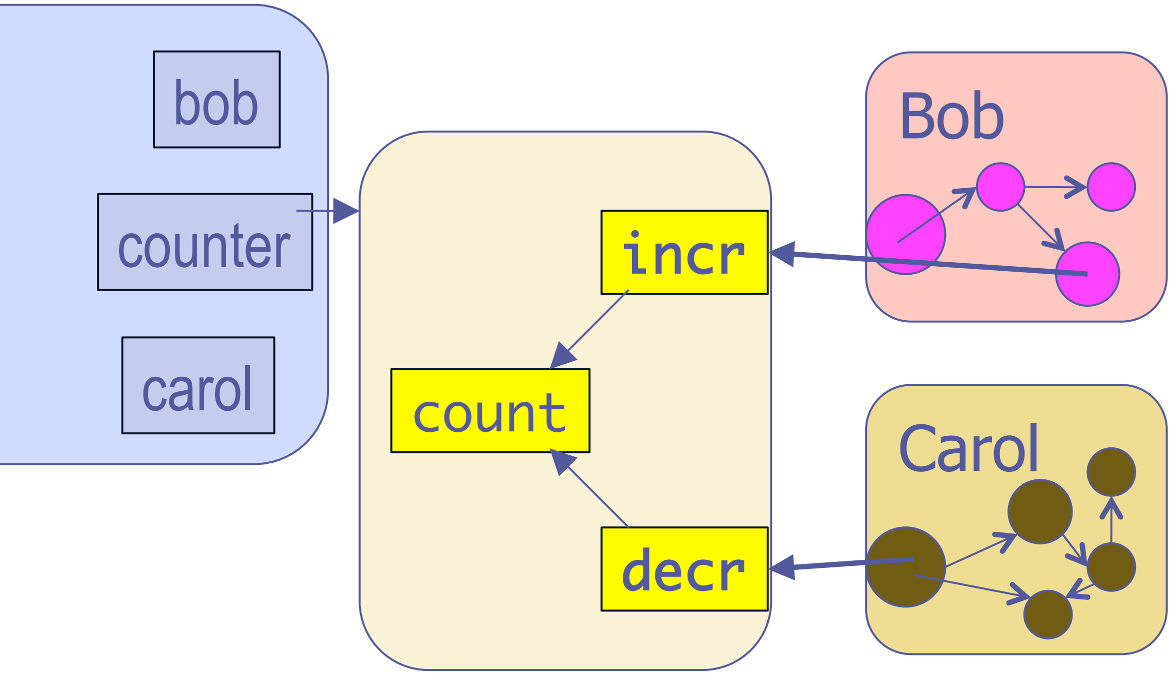
Only Alice controls how they can interact or get more connected.

No powerful references by default



Alice says:

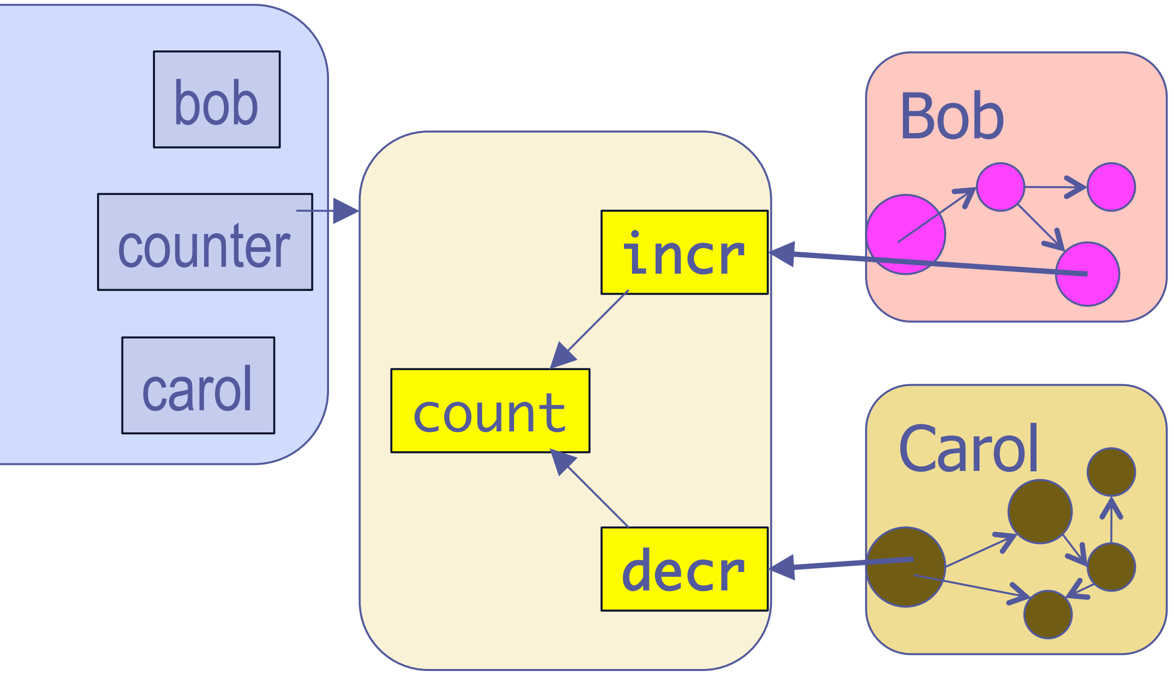
Only connectivity begets connectivity



Alice says:

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

Only connectivity begets connectivity



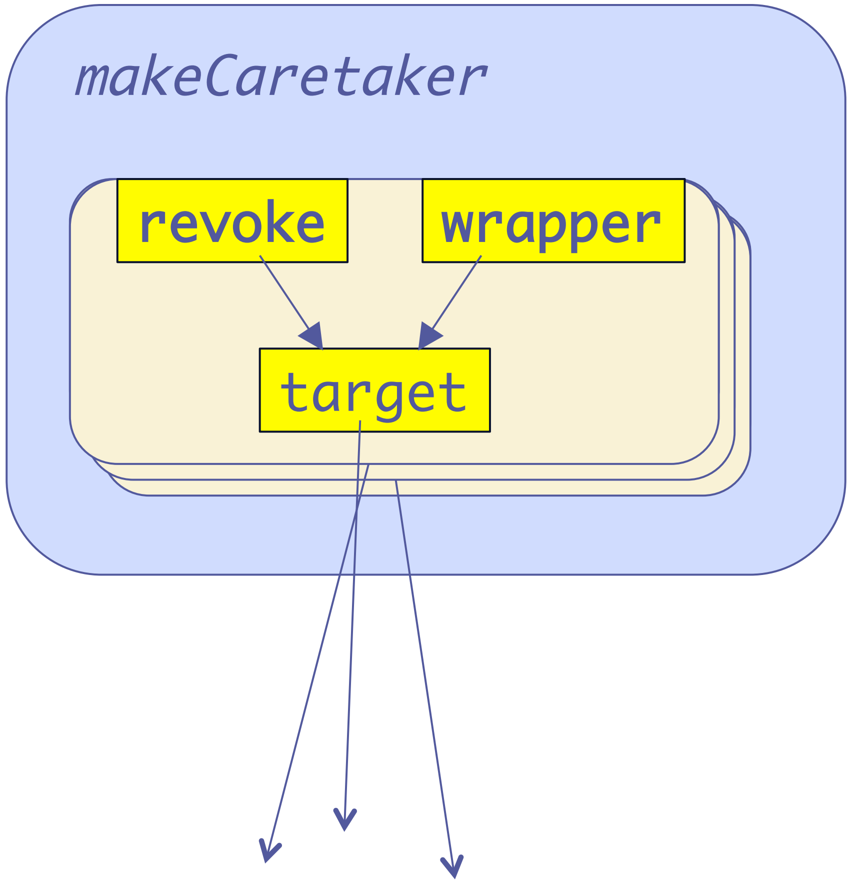
Alice says:

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

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

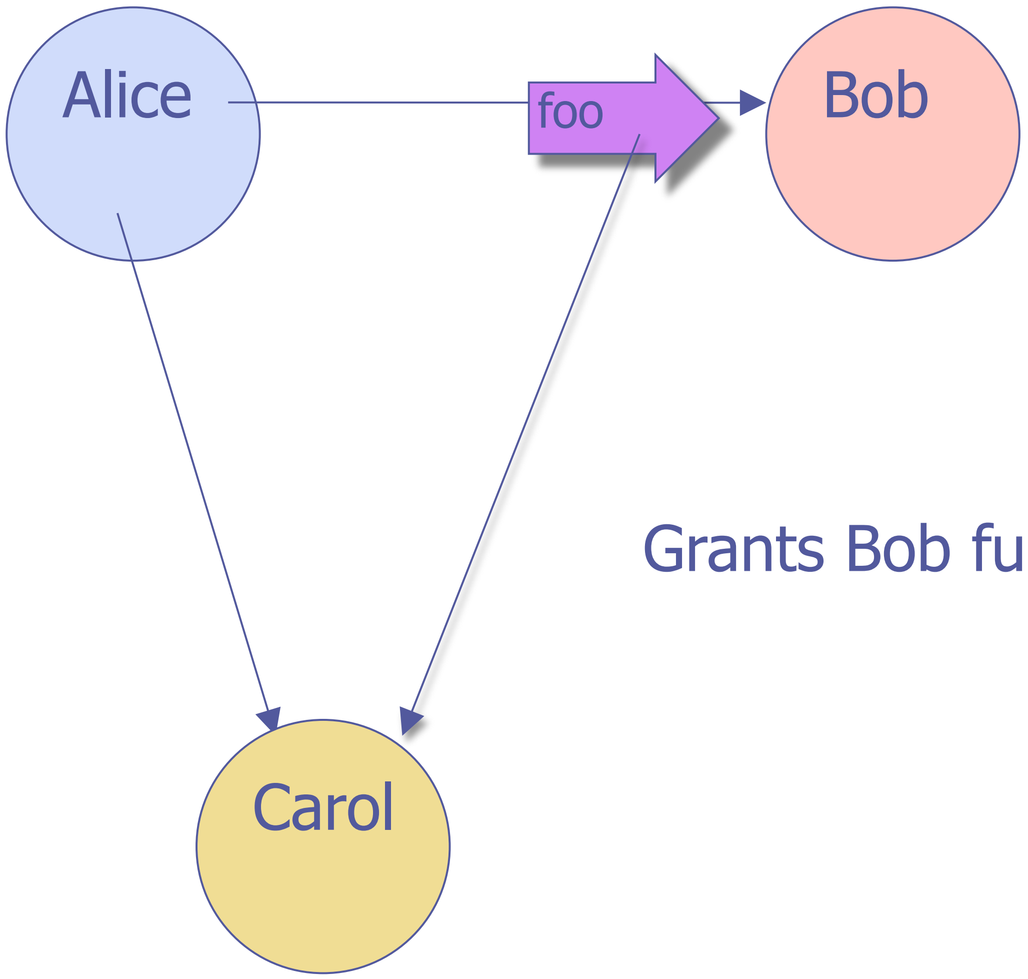
Alice can only do both.

Revocable Function Forwarder



```
function makeFnCaretaker(target) {
  return def({
    wrapper: function(…args) {
      return target(…args);
    },
    revoke: function() { target = null; }
  });
}
```

Unconditional Access

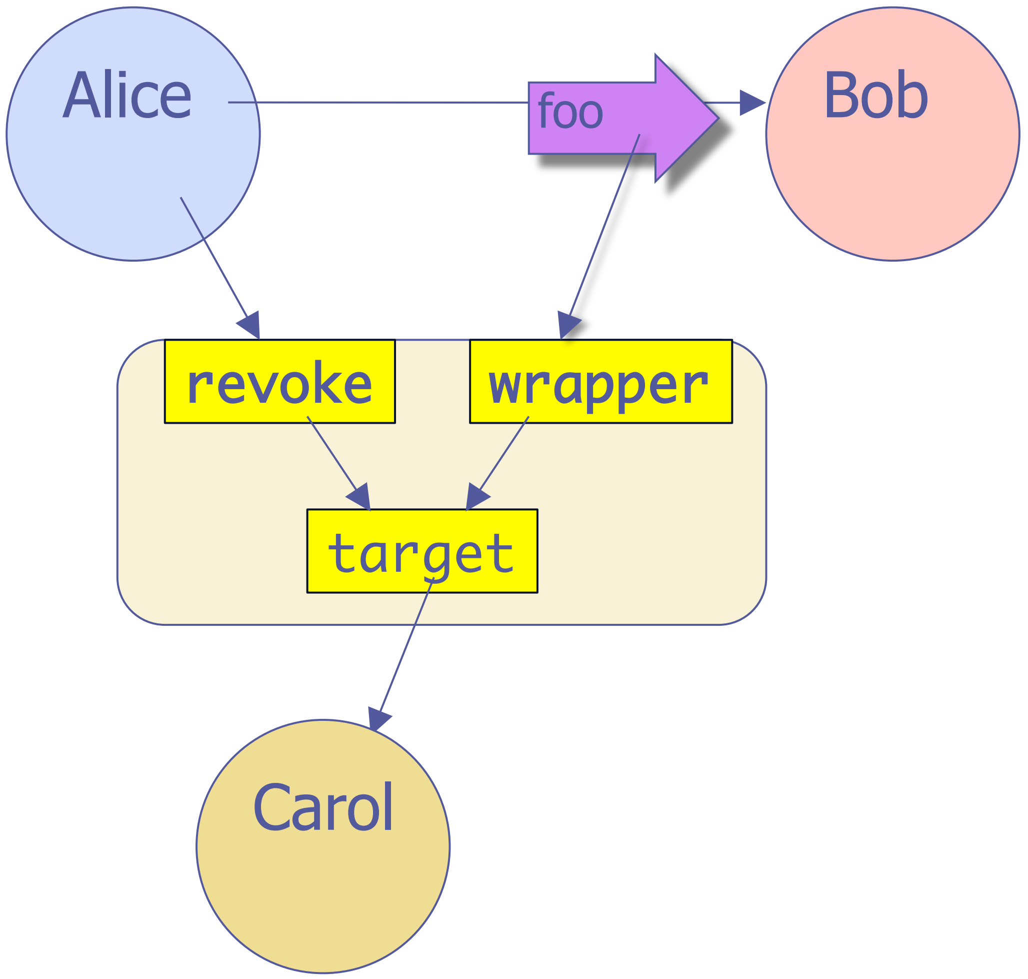


Alice says:

```
bob.foo(carol);
```

Grants Bob full access to Carol forever

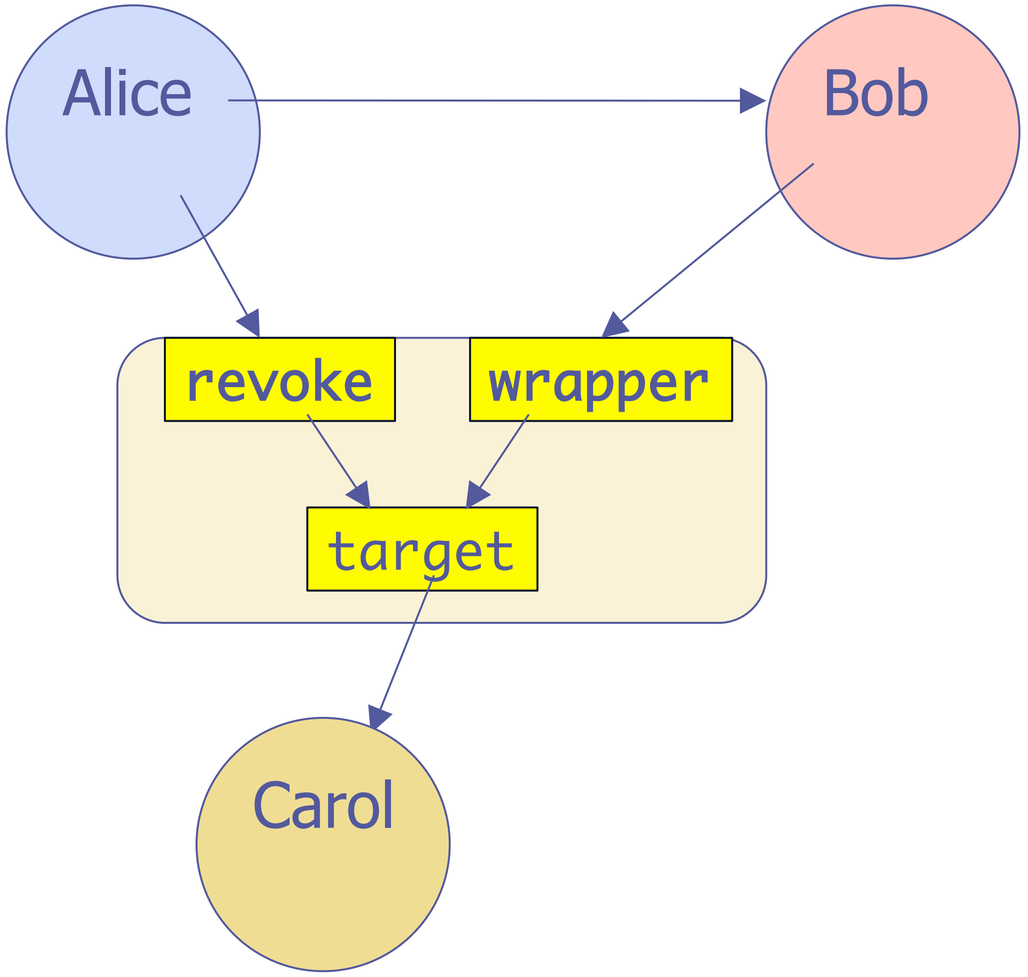
Revocability ≡ Temporal attenuation



Alice says:

```
var ct = makeCaretaker(carol);
bob.foo(ct.wrapper);
```

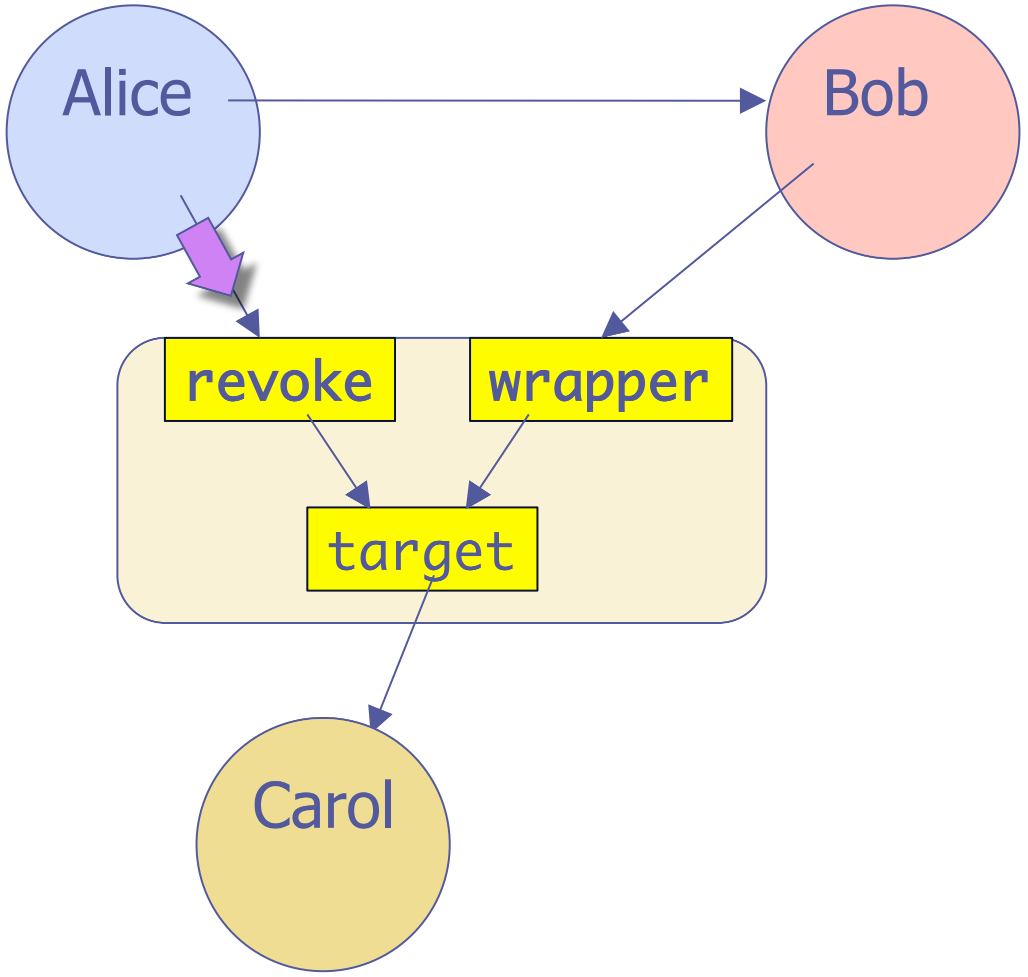
Revocability ≡ Temporal attenuation



Alice says:

```
var ct = makeCaretaker(carol);
bob.foo(ct.wrapper);
//…
```

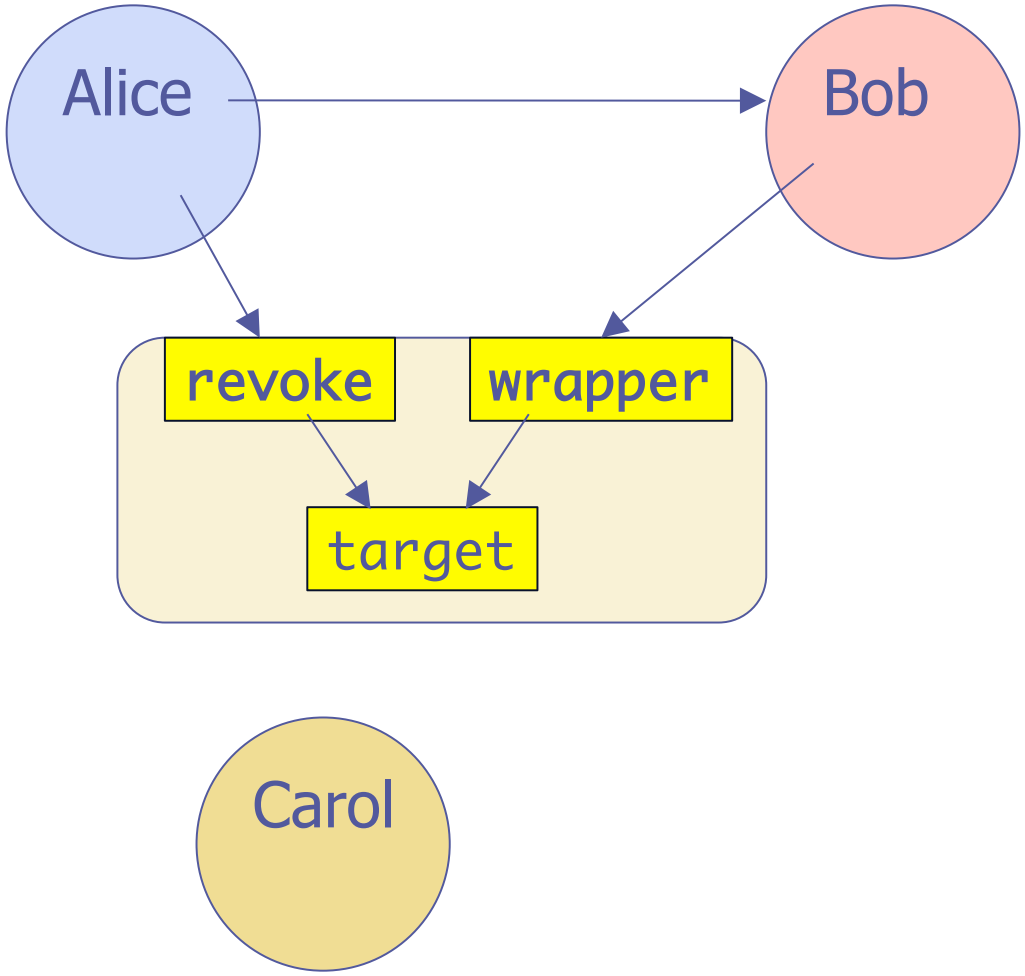
Revocability ≡ Temporal attenuation



Alice says:

```
var ct = makeCaretaker(carol);
bob.foo(ct.wrapper);
//…
ct.revoke();
```

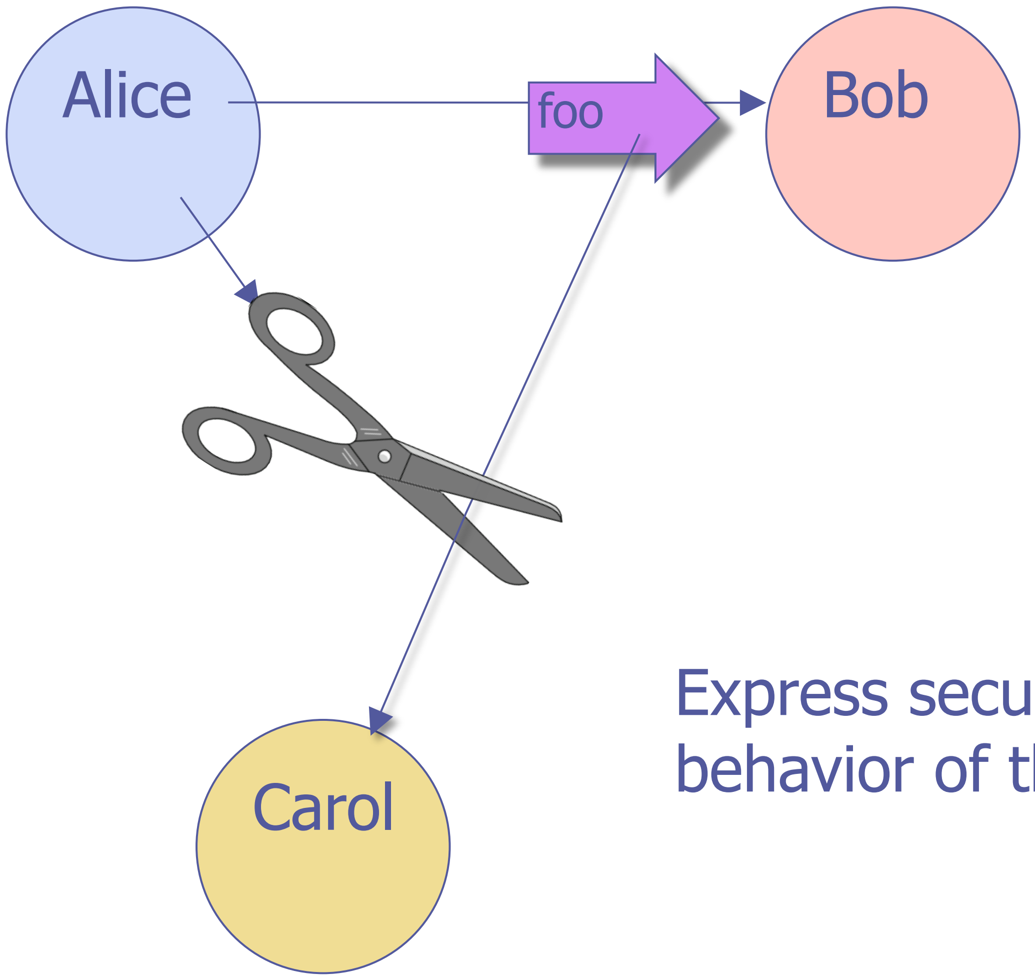
Revocability ≡ Temporal attenuation



Alice says:

```
var ct = makeCaretaker(carol);
bob.foo(ct.wrapper);
//…
ct.revoke();
```

Attenuators ≡ Access Abstractions

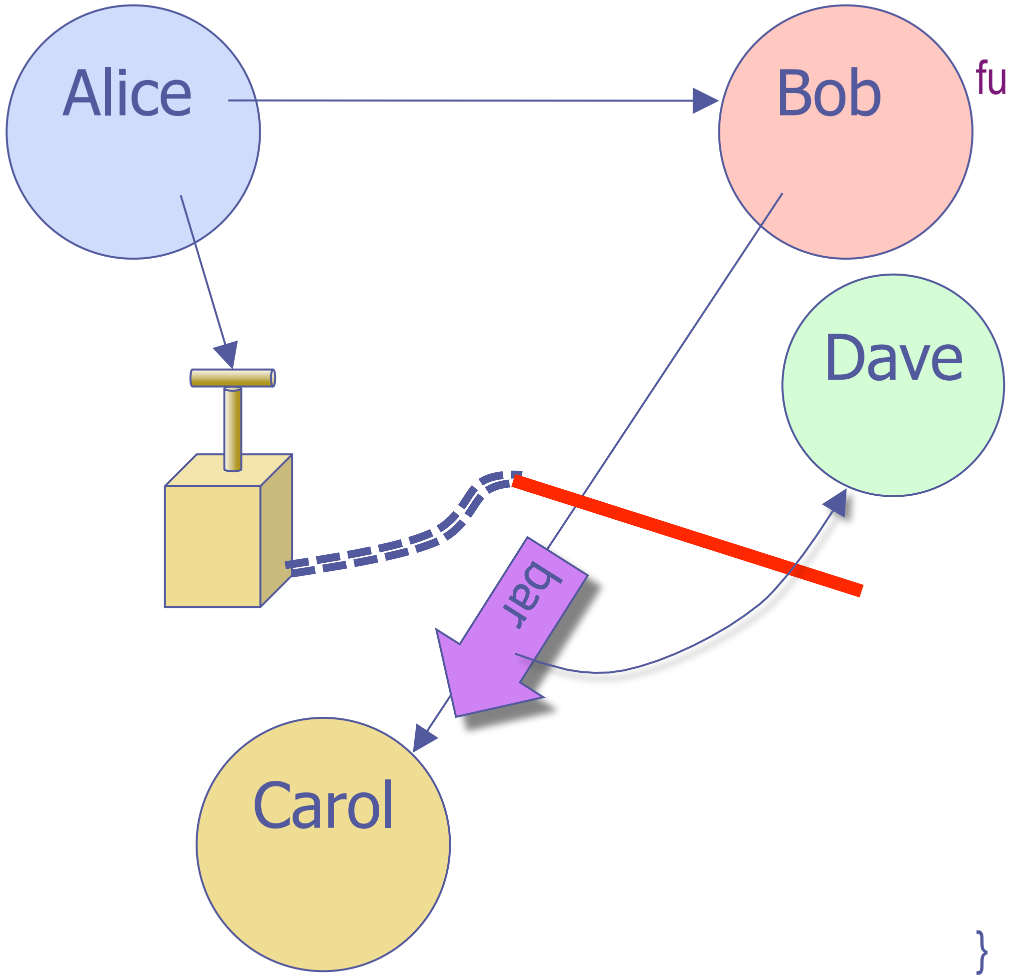


Alice says:

```
var ct = makeCaretaker(carol);
bob.foo(ct.wrapper);
```

Express security policy by the behavior of the objects you provide

Membranes: Transitive Interposition



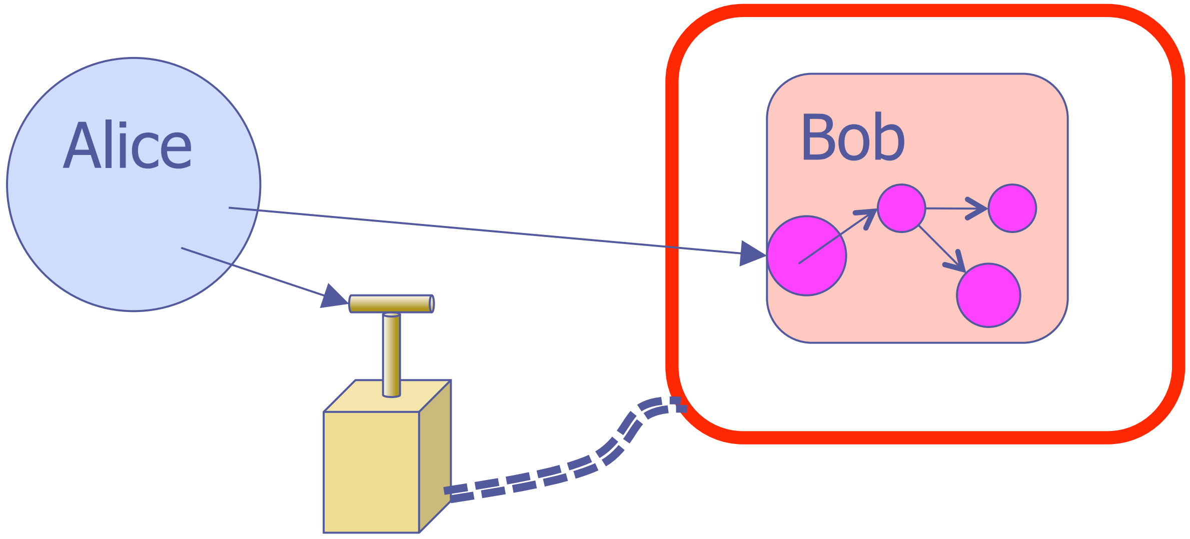
```
function makeFnMembrane(target) {
  var enabled = true;
  function wrap(wrapped) {
    if (wrapped !== Object(wrapped)) {
      return wrapped;
    }
    return function(…args) {
      if (!enabled) { throw new Error(“revoked”); }
      return wrap(wrapped(…args.map(wrap));
  }  }
  return def({
    wrapper: wrap(target),
    revoke: function() { enabled = false; }
  });
}
```

Attenuators Compose

```
function makeROFile(file) {
    return def({
        read: file.read,
        getLength: file.getLength
    });
}
var rorFile = makeROFile(revocableFile);
```

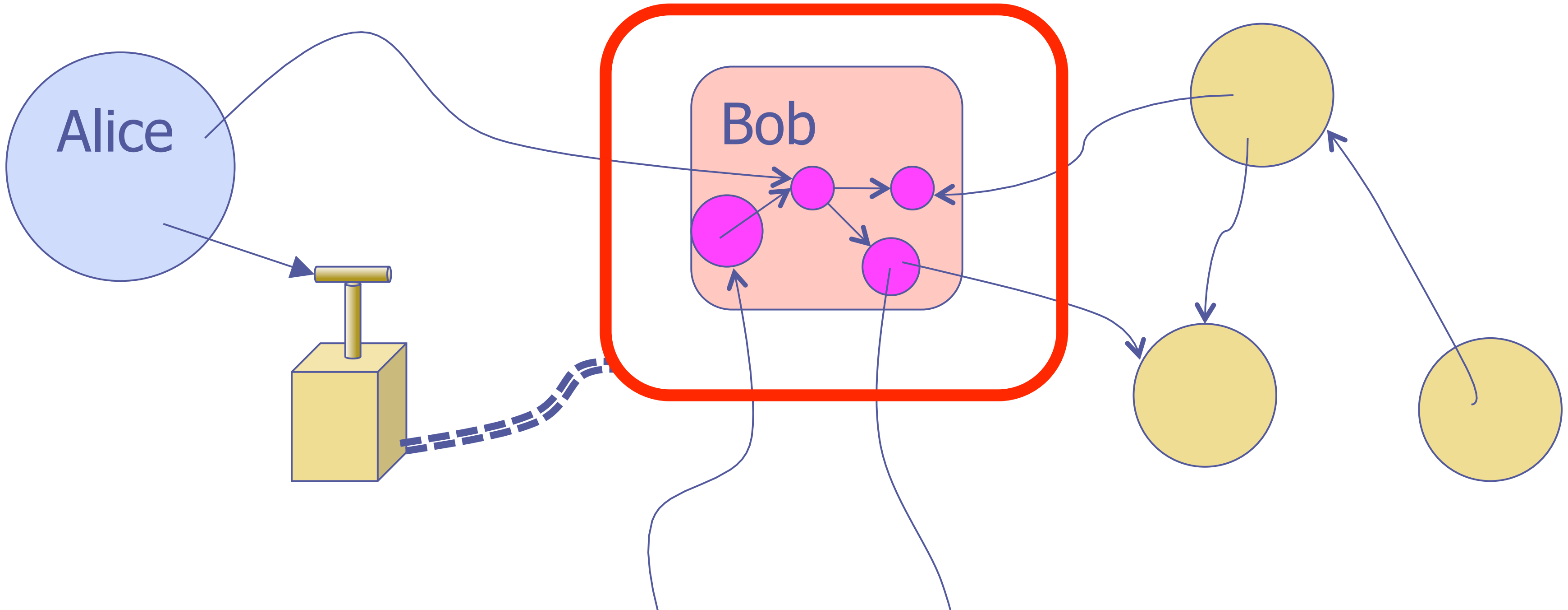
Membrane eval → compartment

var *compartment* = makeMembrane(eval);
var *vbob* = compartment.wrapper(bobSrc);



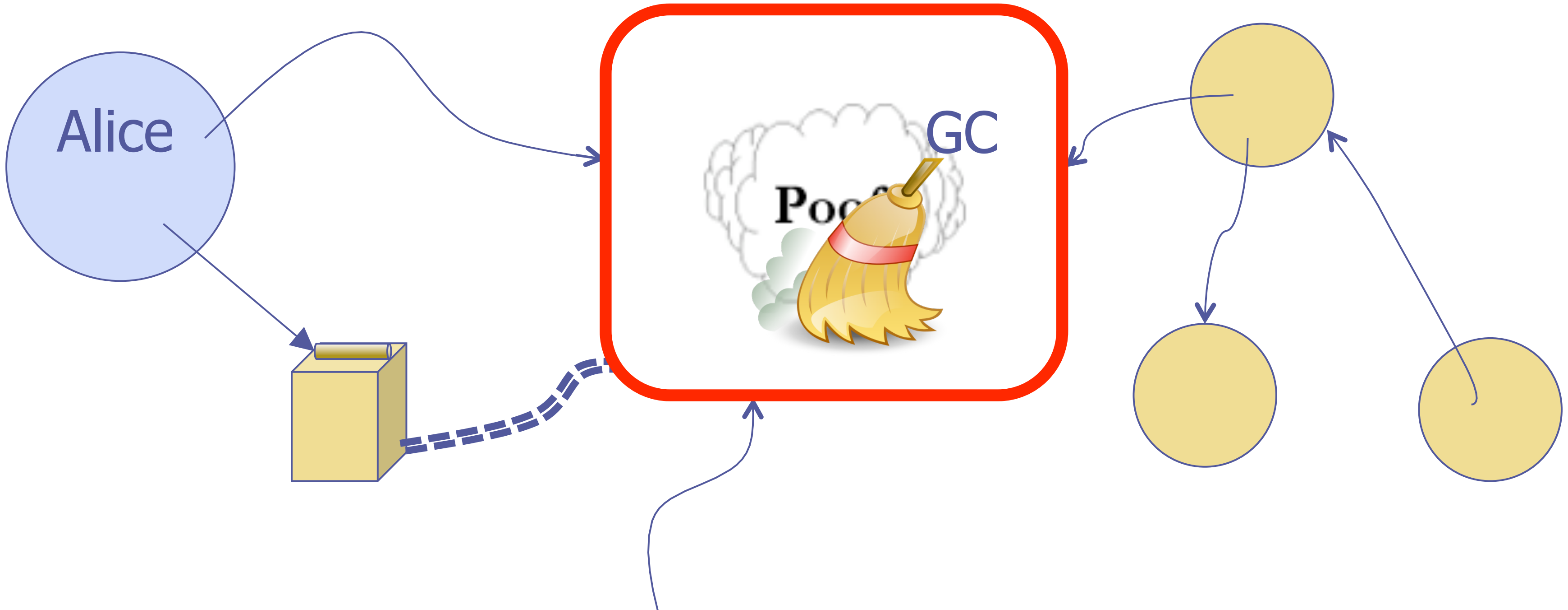
Membrane eval → compartment

```
var compartment = makeMembrane(eval);
var vbob = compartment.wrapper(bobSrc);
//…
```



Membrane eval → compartment

```
var compartment = makeMembrane(eval);
var vbob = compartment.wrapper(bobSrc);
//…
compartment.revoke();
```



Dr. SES
Distributed Resilient Secure EcmaScript

Linguistic abstraction for safe messaging

- Stretch reference graph between machines
- Preserve distributed "memory safety"

SES + Promise lib* + optional infix "!" syntax

- Current standards missing only syntactic convenience

*ref_send by Tyler Close, qcomm by Kris Kowal, and caja-captp by Kevin Reid

Dr. SES
Distributed Resilient Secure EcmaScript

Object operation syntax	**Library call**
var *result* = bob.foo(carol);	Local only call
var *resultP* = bobP ! foo(carol);	Q.post(bobP, ‘foo’, [carol])

Dr. SES
Distributed Resilient Secure EcmaScript

Object operation syntax	**Library call**
var *result* = bob.foo(carol);	
var *resultP* = bobP ! foo(carol);	Q.post(bobP, ‘foo’, [carol])
var *result* = bob.foo;	
var *resultP* = bobP ! foo;	Q.get(bobP, ‘foo’)
bob.foo = newFoo;	
bobP ! foo = newFoo;	Q.put(bobP, ‘foo’, newFoo)
delete bob.foo;	
delete bobP ! foo;	Q.delete(bobP, ‘foo’)

Dr. SES
Distributed Resilient Secure EcmaScript

Object operation syntax	**Library call**
~~var *result* = bob.foo(carol);~~	
var *resultP* = bobP ! foo(carol);	~~Q.post(bobP, ‘foo’, [carol])~~
~~var *result* = bob.foo;~~	
var *resultP* = bobP ! foo;	~~Q.get(bobP, ‘foo’)~~
~~bob.foo = newFoo;~~	
~~bobP ! foo = newFoo;~~	~~Q.put(bobP, ‘foo’, newFoo)~~
~~delete bob.foo;~~	
~~delete bobP ! foo;~~	~~Q.delete(bobP, ‘foo’)~~

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

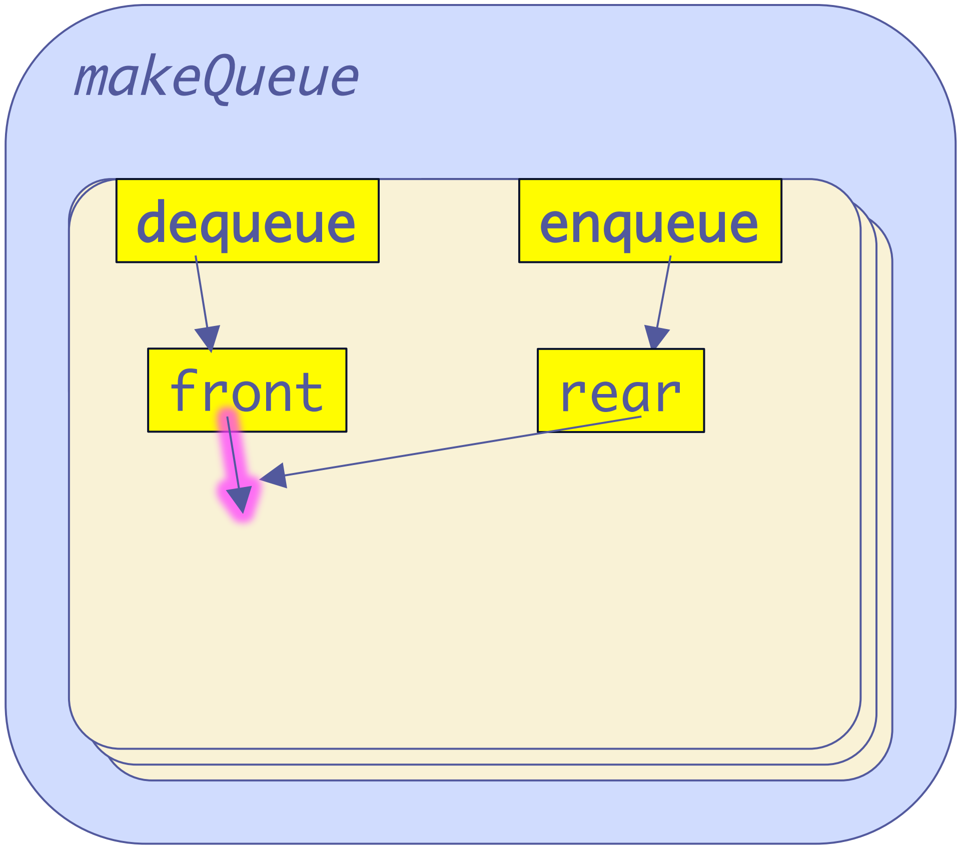
`var resultP = bobP ! foo(carol);`	Eventual send
`var resultP = bobP ! foo;`	Eventual get

Dr. SES
Distributed Resilient Secure EcmaScript

`var resultP = bobP ! foo(carol);`	Eventual send
`var resultP = bobP ! foo;`	Eventual get
`Q.defer();`	{promise: promise, resolve: resolve}
`Q.when(resultP, function(result) {`	Register callbacks

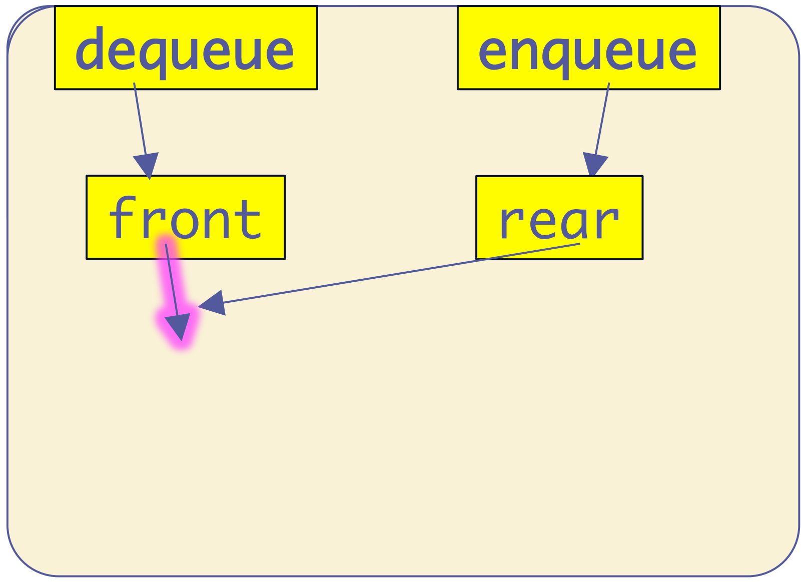
```
Q.when(resultP, function(result) {
  …result…
}, function (ex) {
  …ex…
});
```

Infinite Queue



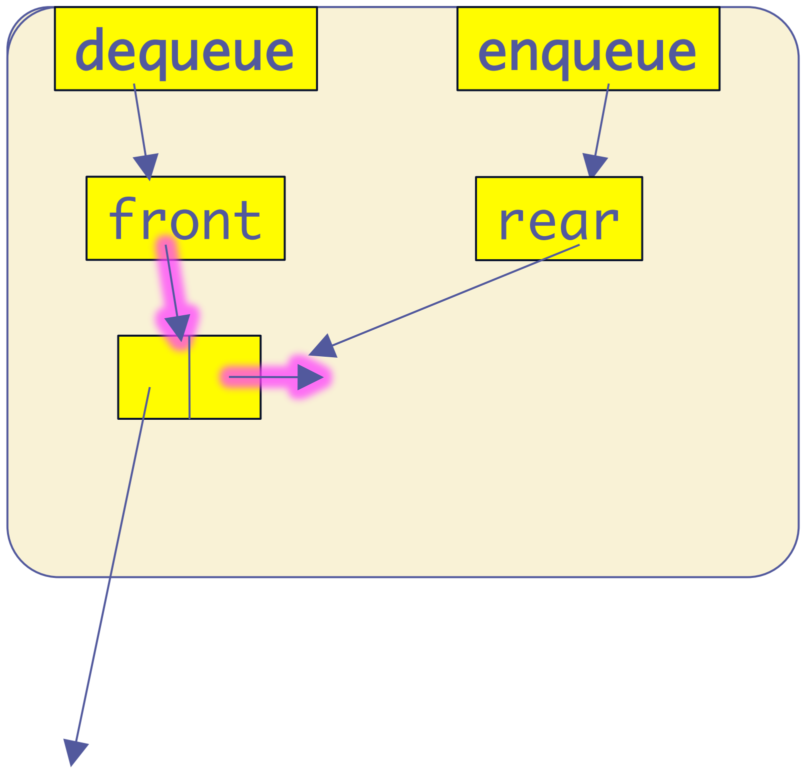
```
function makeQueue() {
  var ends = Q.defer();
  var front = ends.promise;
  var rear = ends.resolve;
  return def({
    enqueue: function(elem) {
      var next = Q.defer();
      rear({first: elem, rest: next.promise});
      rear = next.resolve;
    },
    dequeue: function() {
      var result = front ! first;
      front = front ! rest;
      return result;
} });}
```

Infinite Queue



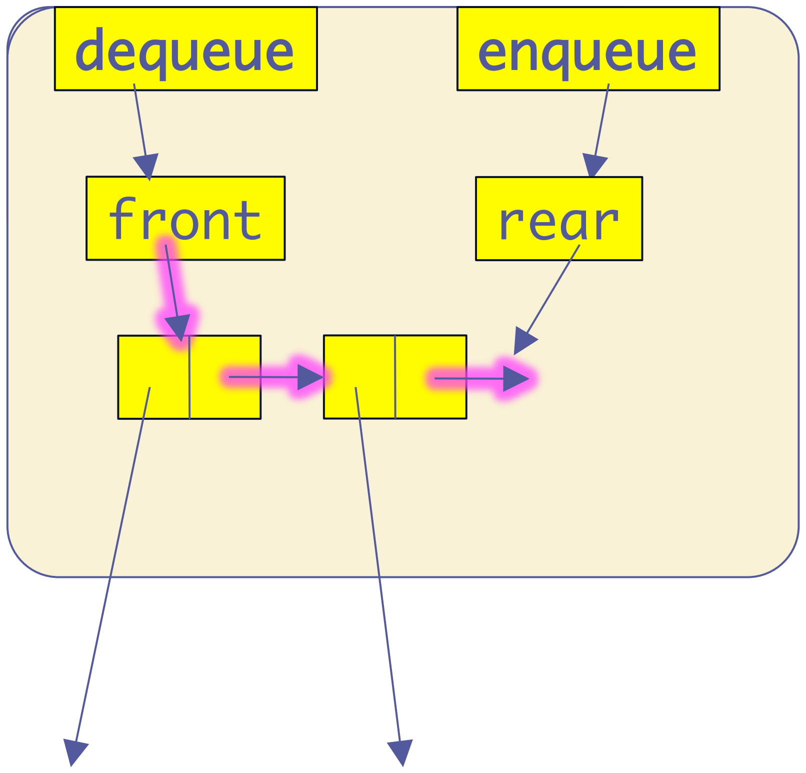
```
function makeQueue() {
  var ends = Q.defer();
  var front = ends.promise;
  var rear = ends.resolve;
  return def({
    enqueue: function(elem) {
      var next = Q.defer();
      rear({first: elem, rest: next.promise});
      rear = next.resolve;
    },
    dequeue: function() {
      var result = front ! first;
      front = front ! rest;
      return result;
} });}
```

Infinite Queue



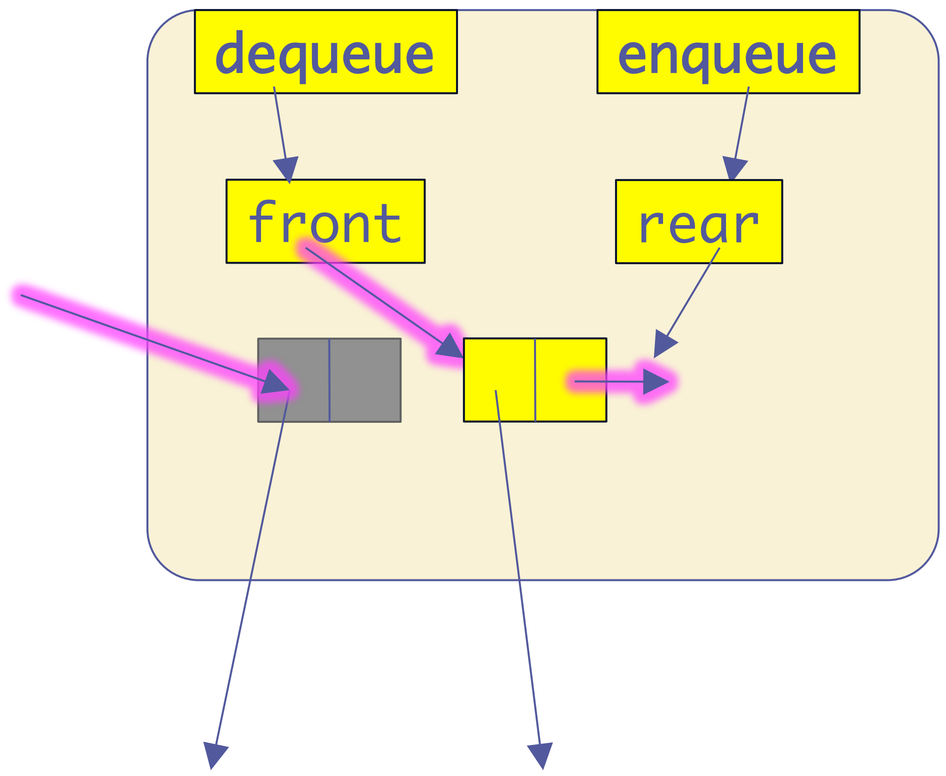
```
function makeQueue() {
  var ends = Q.defer();
  var front = ends.promise;
  var rear = ends.resolve;
  return def({
    enqueue: function(elem) {
      var next = Q.defer();
      rear({first: elem, rest: next.promise});
      rear = next.resolve;
    },
    dequeue: function() {
      var result = front ! first;
      front = front ! rest;
      return result;
} });}
```

Infinite Queue



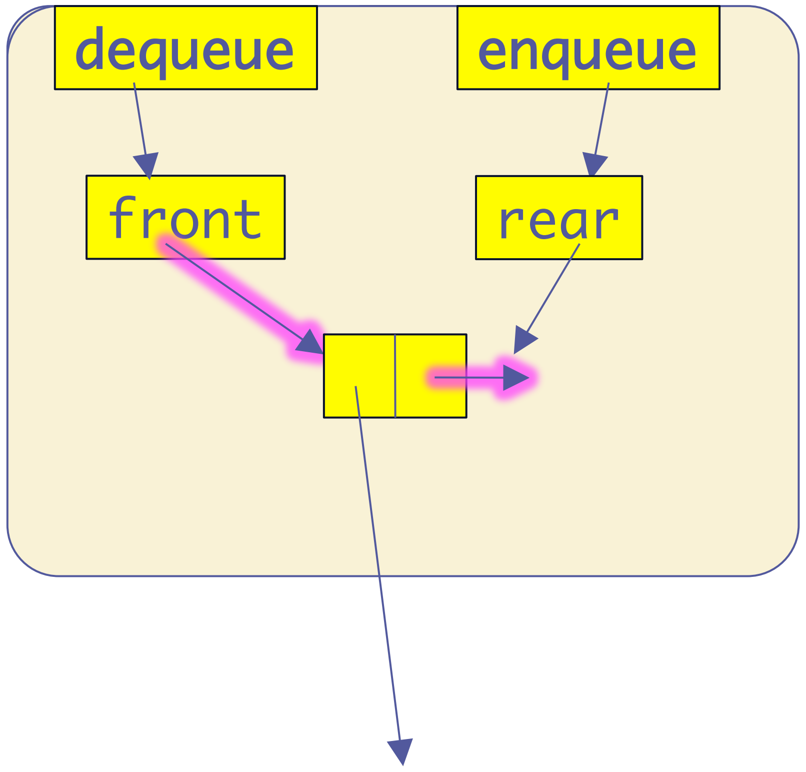
```
function makeQueue() {
 var ends = Q.defer();
 var front = ends.promise;
 var rear = ends.resolve;
 return def({
  enqueue: function(elem) {
   var next = Q.defer();
   rear({first: elem, rest: next.promise});
   rear = next.resolve;
  },
  dequeue: function() {
   var result = front ! first;
   front = front ! rest;
   return result;
} });}
```

Infinite Queue



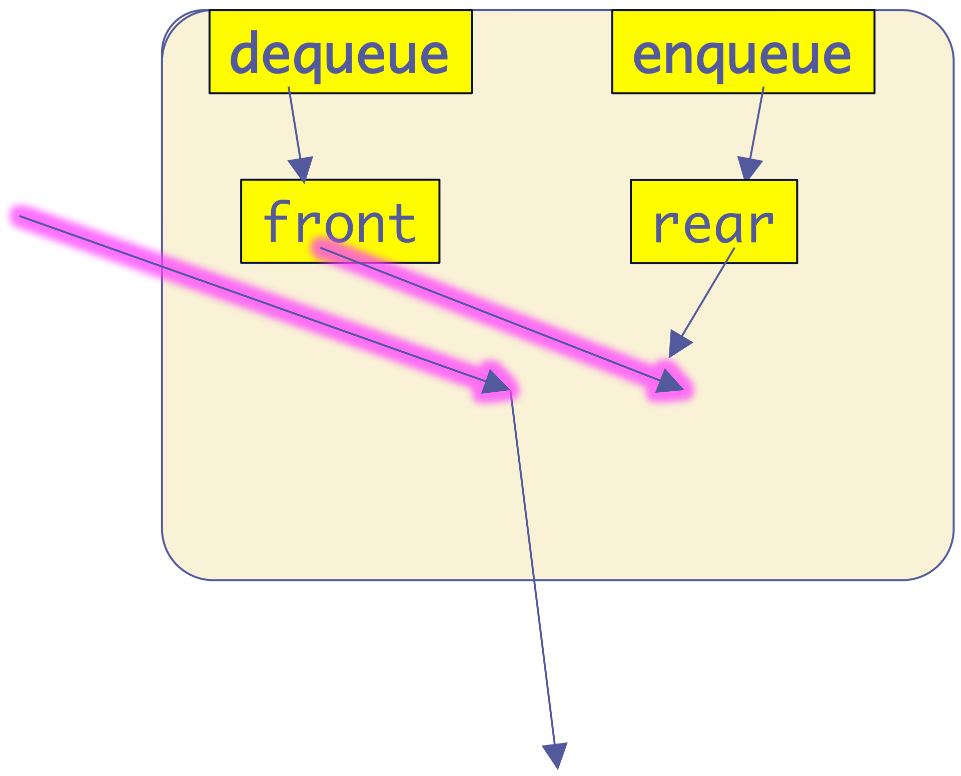
```
function makeQueue() {
  var ends = Q.defer();
  var front = ends.promise;
  var rear = ends.resolve;
  return def({
    enqueue: function(elem) {
      var next = Q.defer();
      rear({first: elem, rest: next.promise});
      rear = next.resolve;
    },
    dequeue: function() {
      var result = front ! first;
      front = front ! rest;
      return result;
} });}
```

Infinite Queue



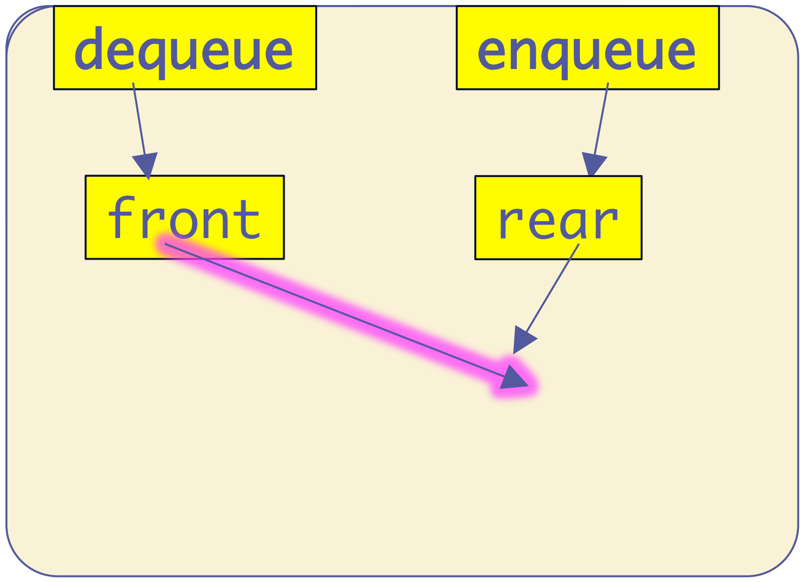
```
function makeQueue() {
  var ends = Q.defer();
  var front = ends.promise;
  var rear = ends.resolve;
  return def({
    enqueue: function(elem) {
      var next = Q.defer();
      rear({first: elem, rest: next.promise});
      rear = next.resolve;
    },
    dequeue: function() {
      var result = front ! first;
      front = front ! rest;
      return result;
} });}
```

Infinite Queue



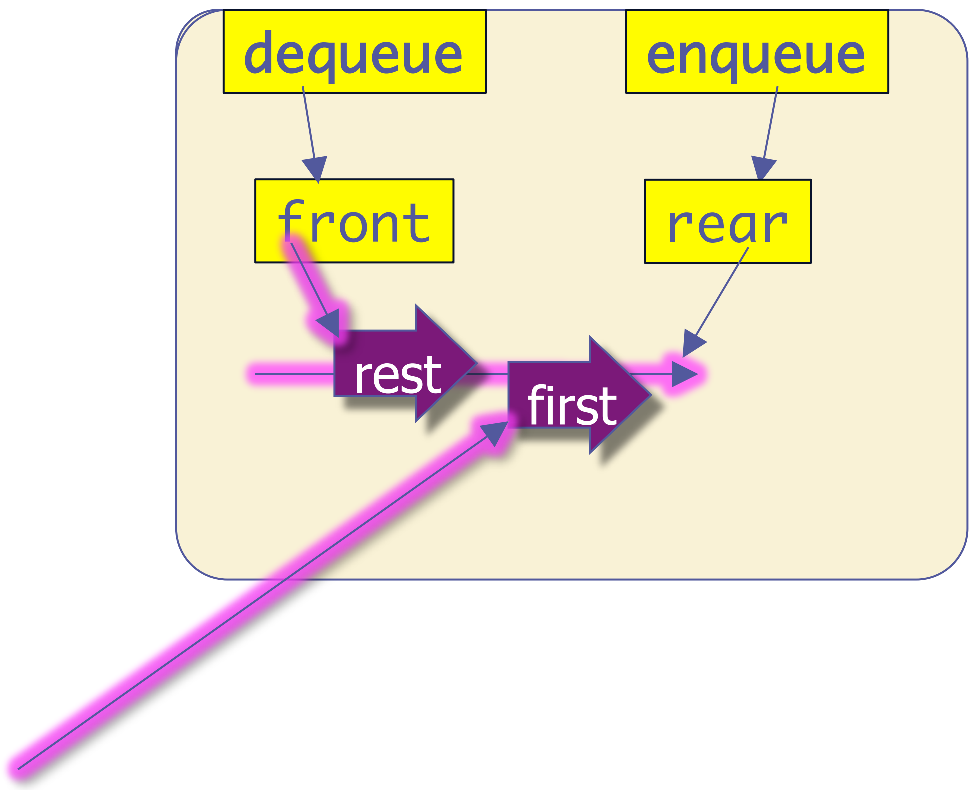
```
function makeQueue() {
  var ends = Q.defer();
  var front = ends.promise;
  var rear = ends.resolve;
  return def({
    enqueue: function(elem) {
      var next = Q.defer();
      rear({first: elem, rest: next.promise});
      rear = next.resolve;
    },
    dequeue: function() {
      var result = front ! first;
      front = front ! rest;
      return result;
} });}
```

Infinite Queue



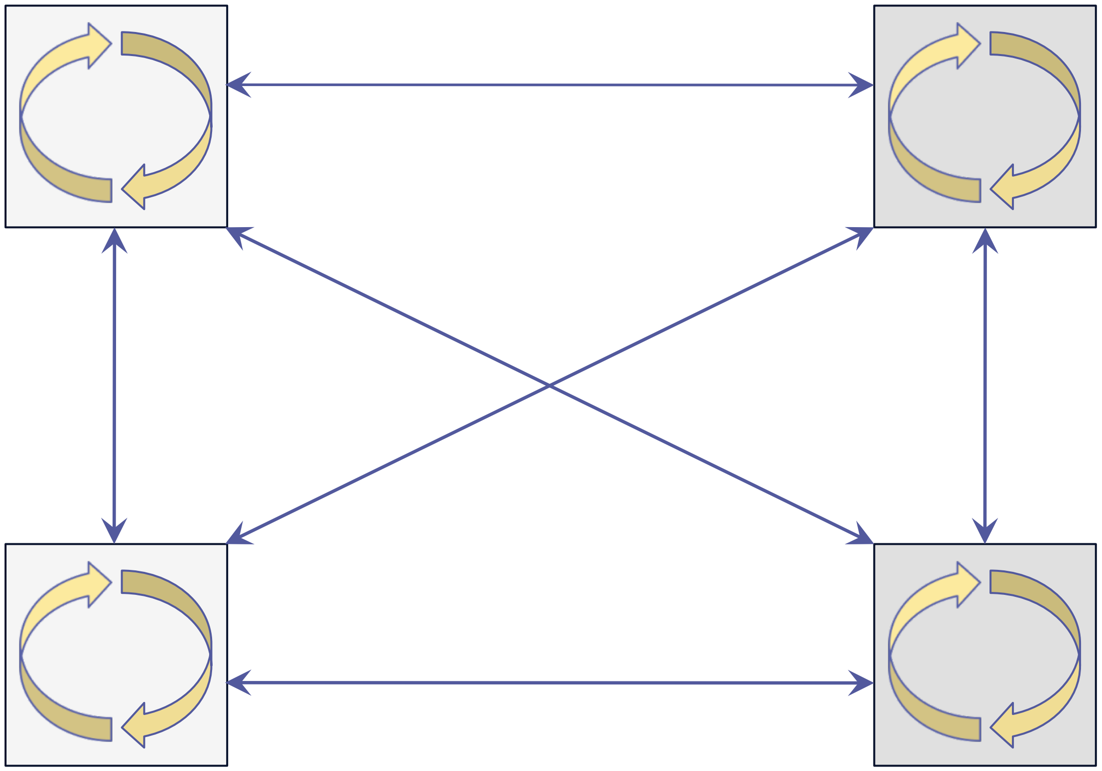
```
function makeQueue() {
  var ends = Q.defer();
  var front = ends.promise;
  var rear = ends.resolve;
  return def({
    enqueue: function(elem) {
      var next = Q.defer();
      rear({first: elem, rest: next.promise});
      rear = next.resolve;
    },
    dequeue: function() {
      var result = front ! first;
      front = front ! rest;
      return result;
} });}
```

Infinite Queue

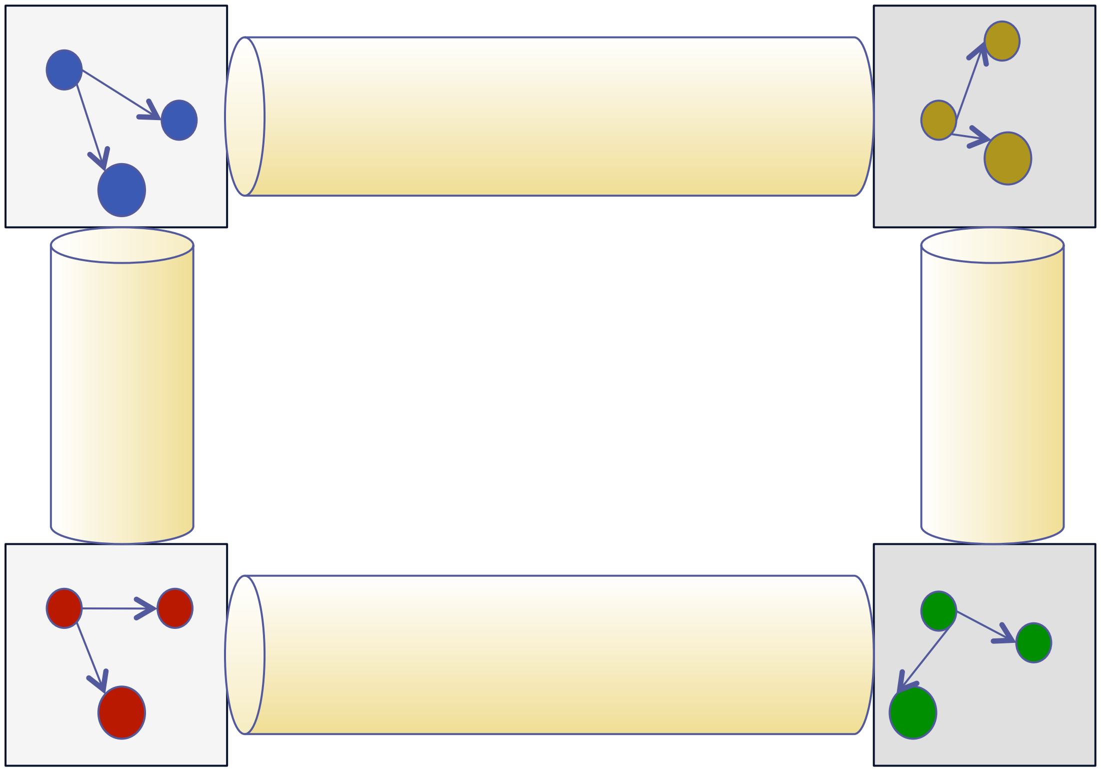


```
function makeQueue() {
  var ends = Q.defer();
  var front = ends.promise;
  var rear = ends.resolve;
  return def({
    enqueue: function(elem) {
      var next = Q.defer();
      rear({first: elem, rest: next.promise});
      rear = next.resolve;
    },
    dequeue: function() {
      var result = front ! first;
      front = front ! rest;
      return result;
} });}
```

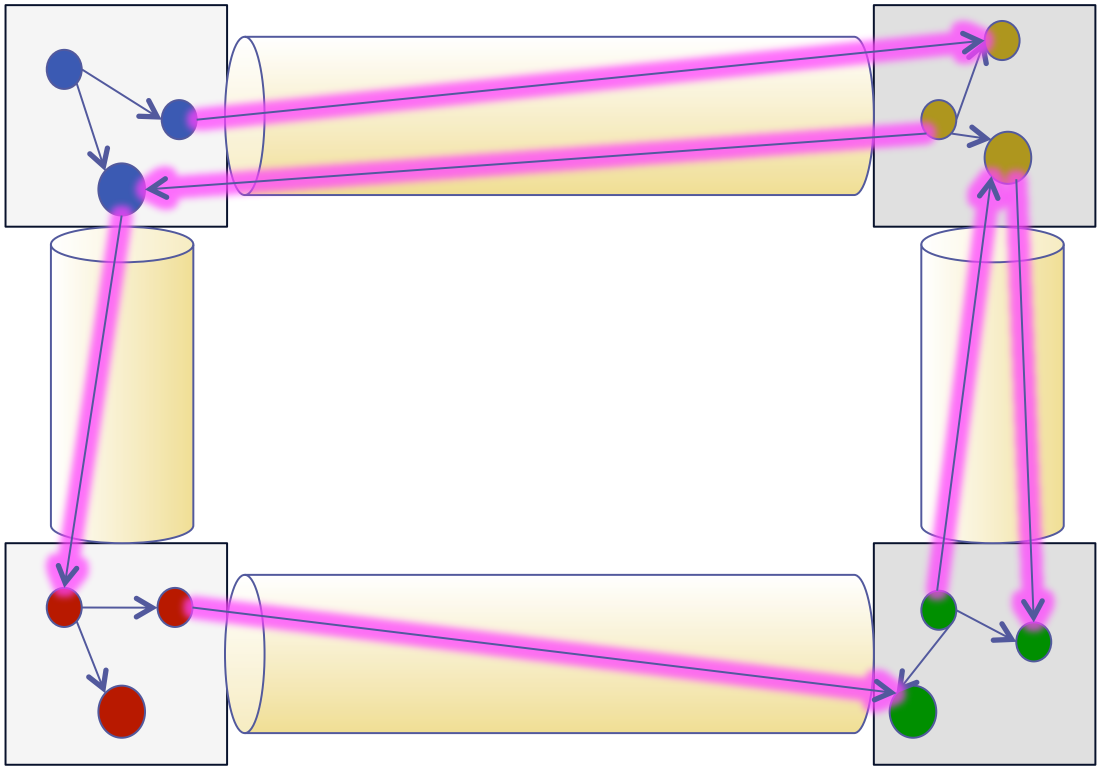
A Web of Distributed Objects



A Web of Distributed Objects



A Web of Distributed Objects



Async object ops as JSON/REST ops

Object operations

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

var resultP = bobP ! foo;
```

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

https: JSON/RESTful operations

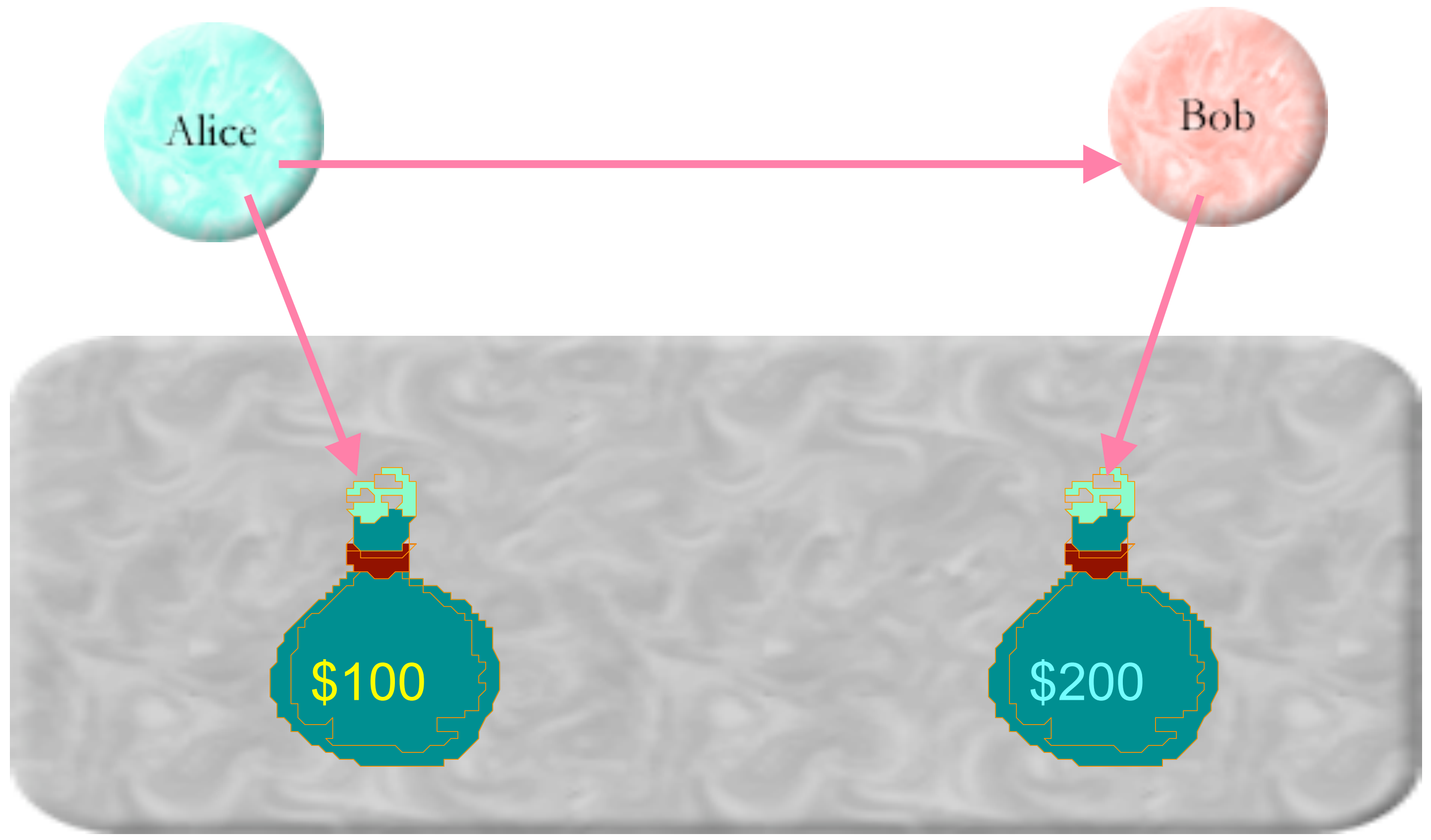
```
POST https://…q=foo {…}

GET https://…q=foo
```

Register for notification using

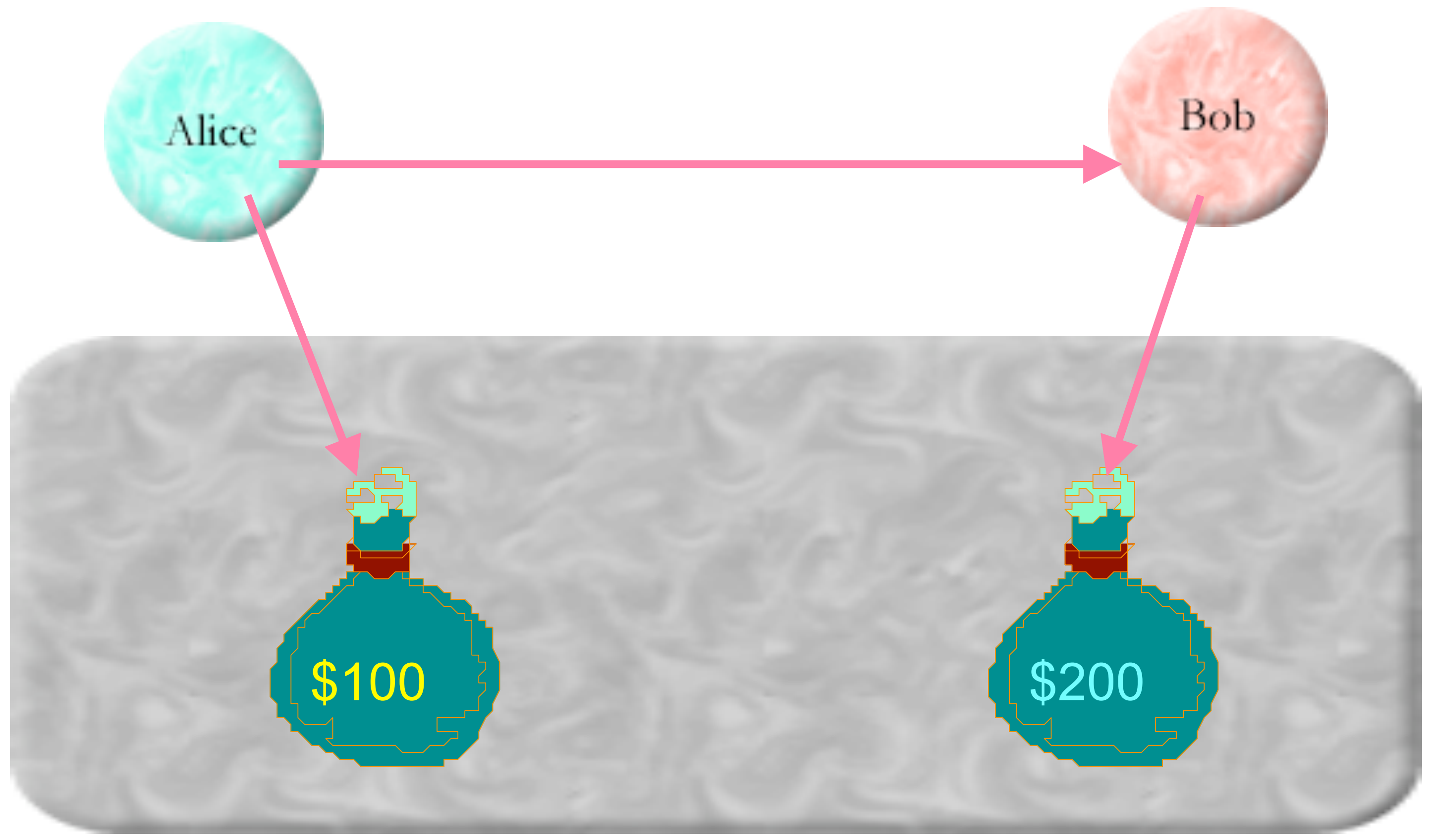
```
xhr.onreadystatechange = …
```

Distributed Secure Currency



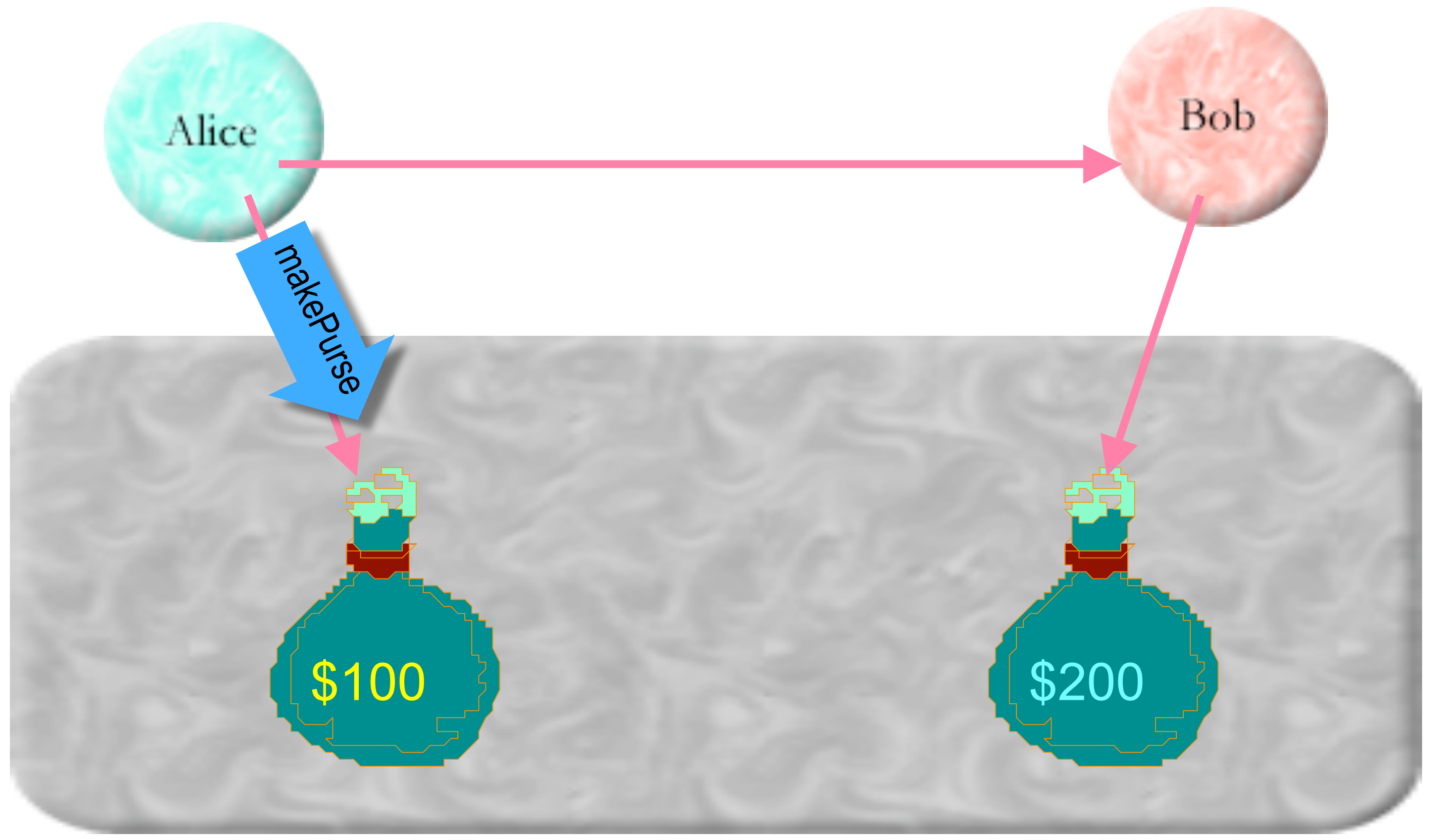
Distributed Secure Currency

```
var paymentP = myPurse ! makePurse();
```



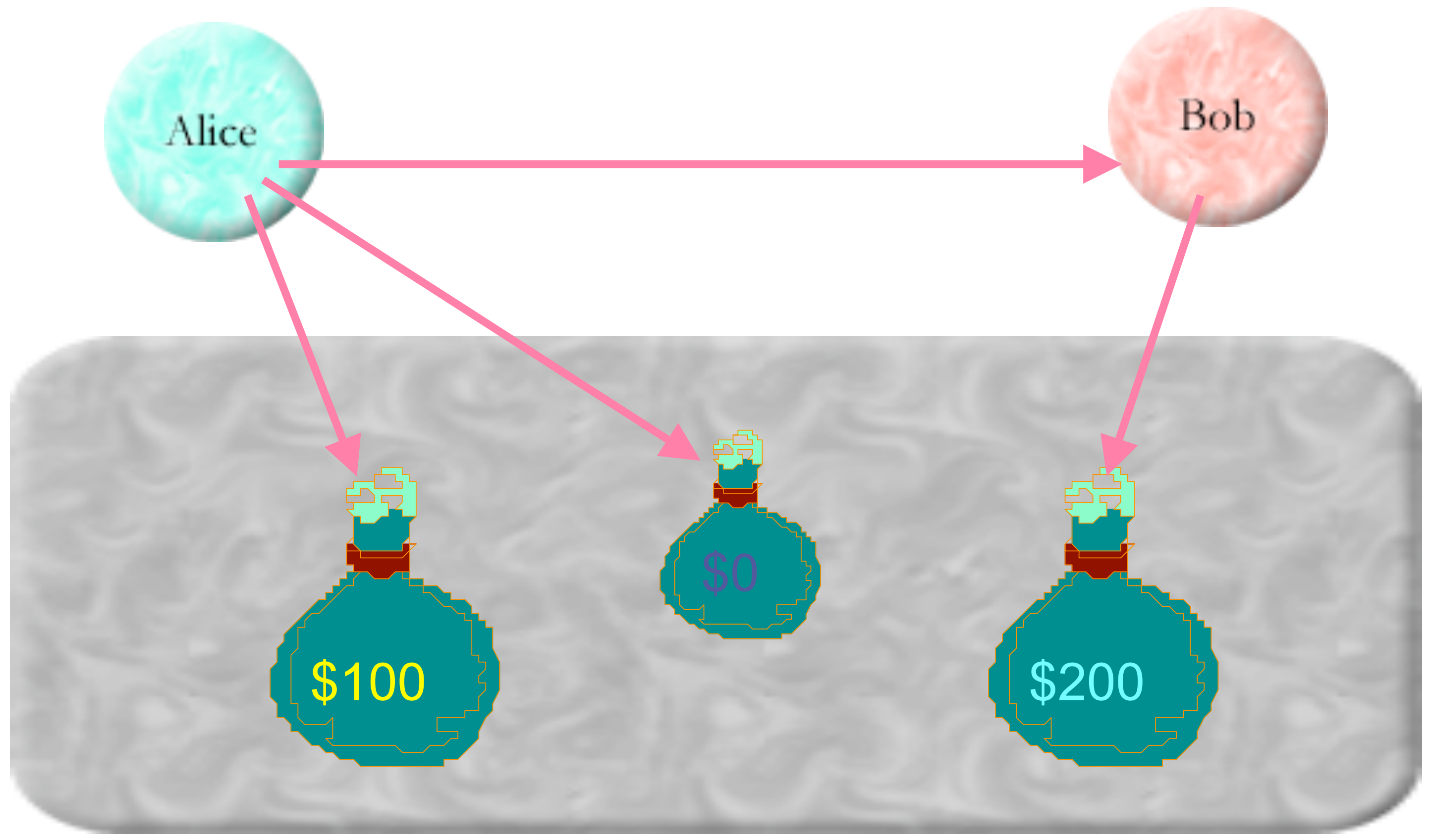
Distributed Secure Currency

```
var paymentP = myPurse ! makePurse();
```



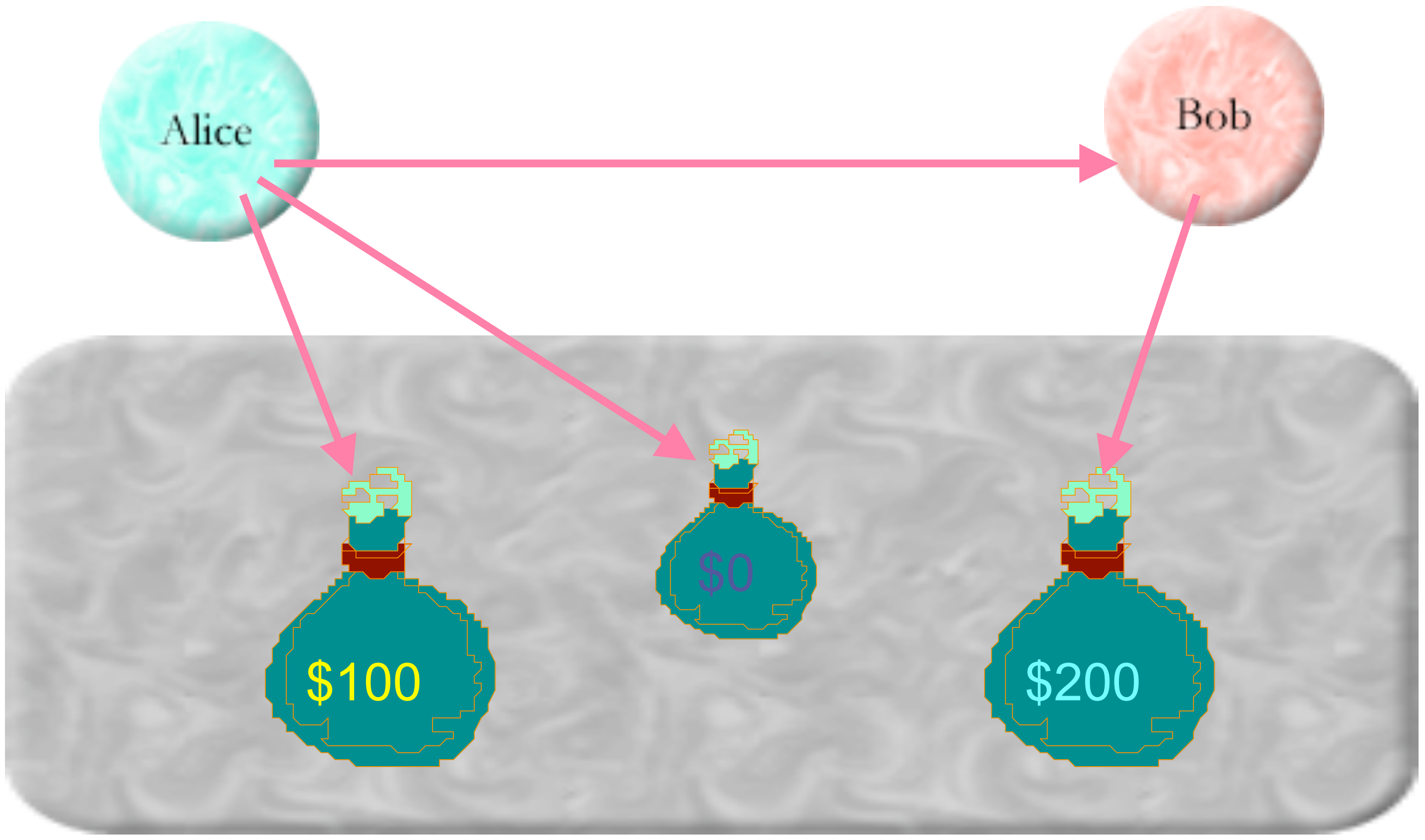
Distributed Secure Currency

```
var paymentP = myPurse ! makePurse();
```



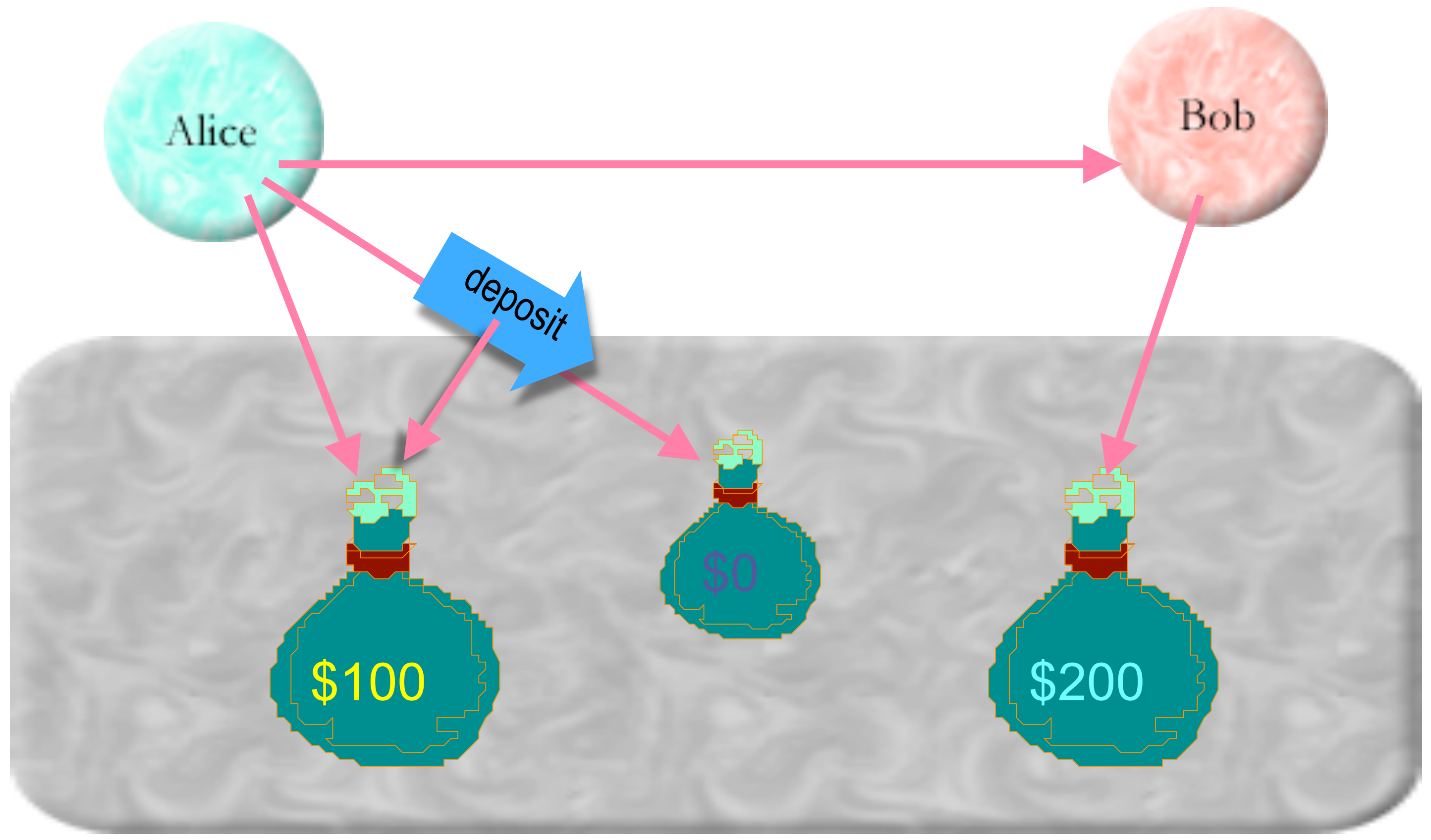
Distributed Secure Currency

```
var *paymentP* = myPurse ! makePurse();
paymentP ! deposit(10, myPurse);
```



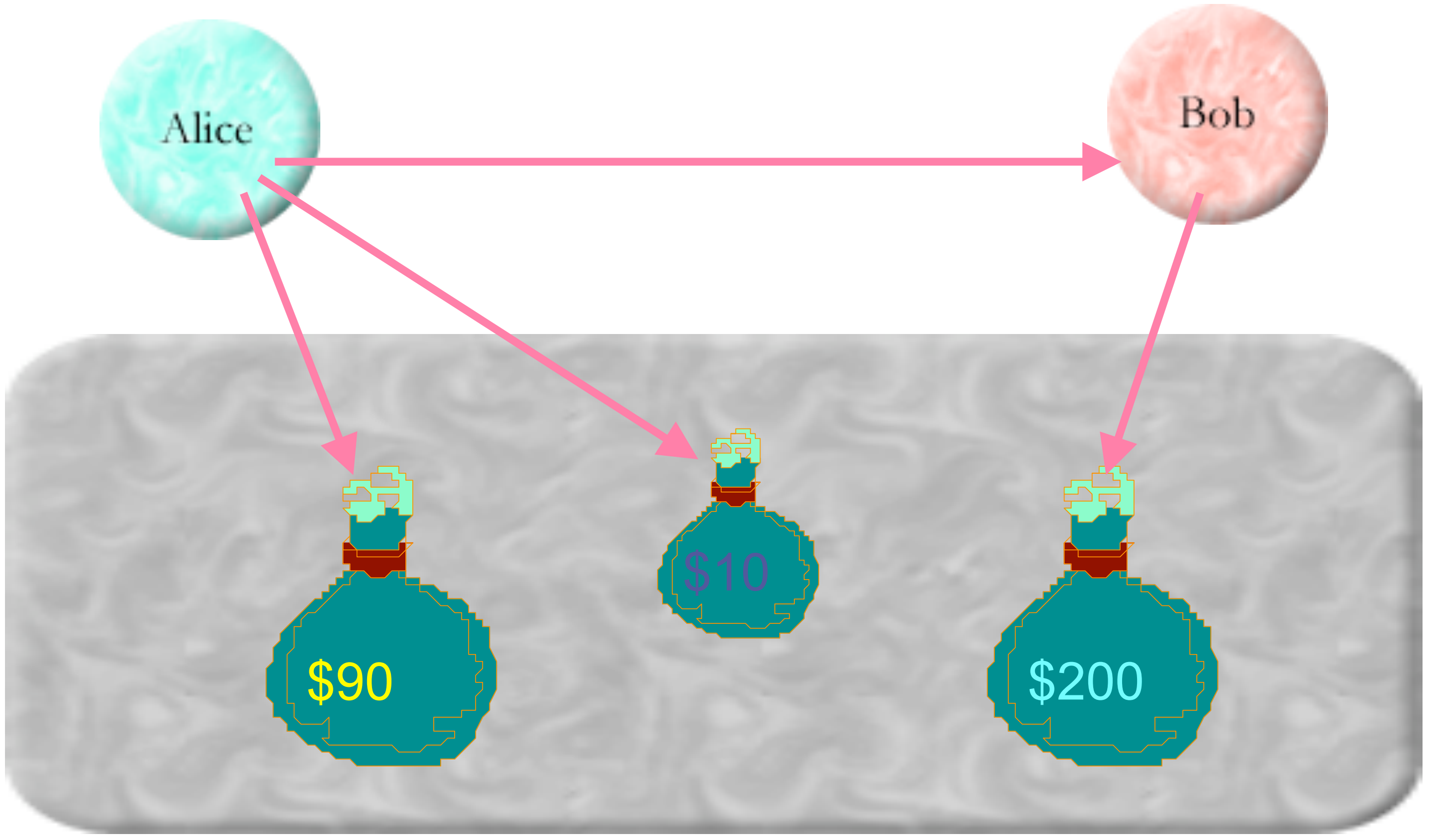
Distributed Secure Currency

```
var paymentP = myPurse ! makePurse();
paymentP ! deposit(10, myPurse);
```



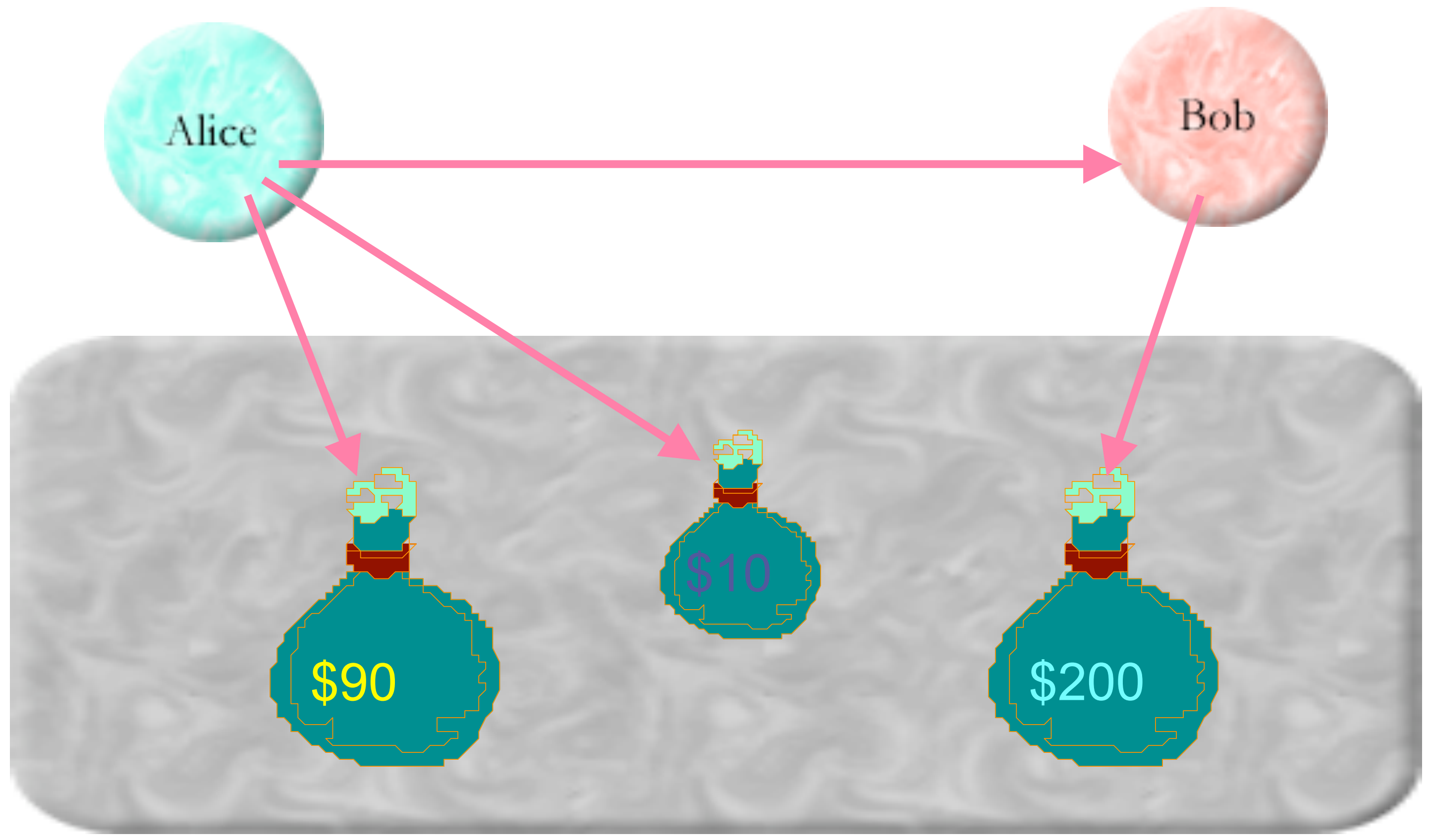
Distributed Secure Currency

```
var paymentP = myPurse ! makePurse();
paymentP ! deposit(10, myPurse);
```



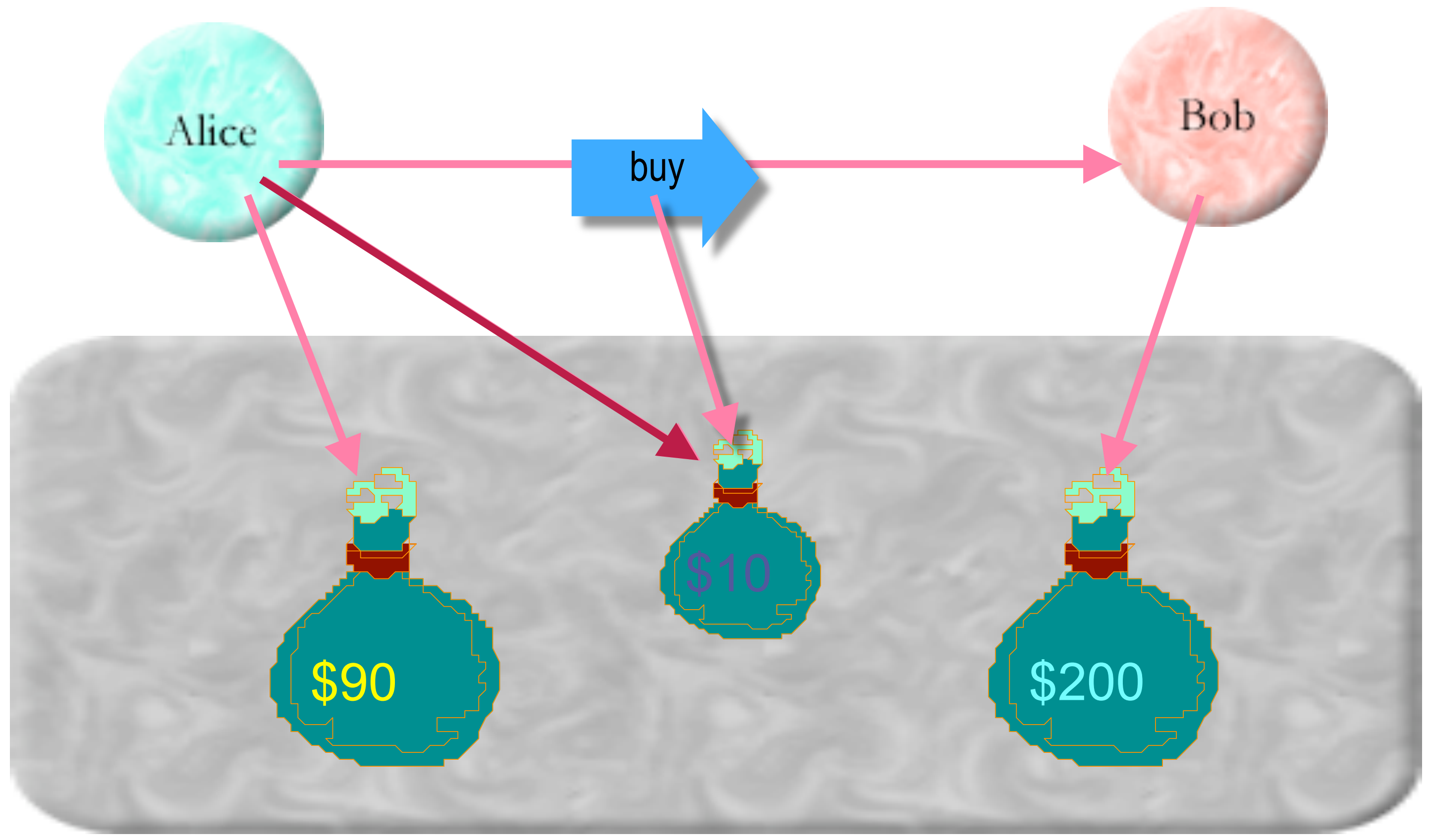
Distributed Secure Currency

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



Distributed Secure Currency

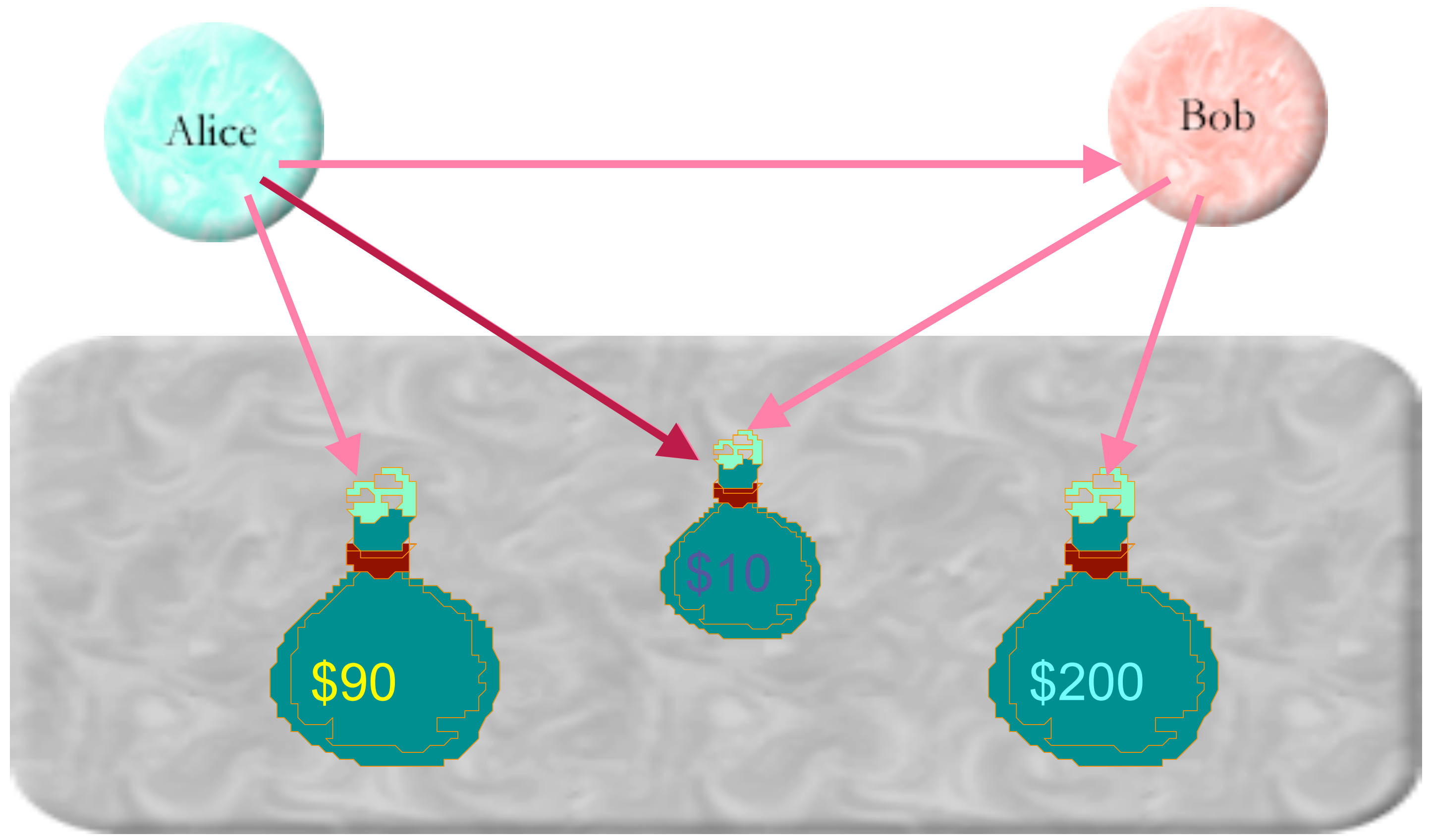
```
var paymentP = myPurse ! makePurse();
paymentP ! deposit(10, myPurse);
var goodP = bobP ! buy(desc, paymentP);
```



Distributed Secure Currency

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

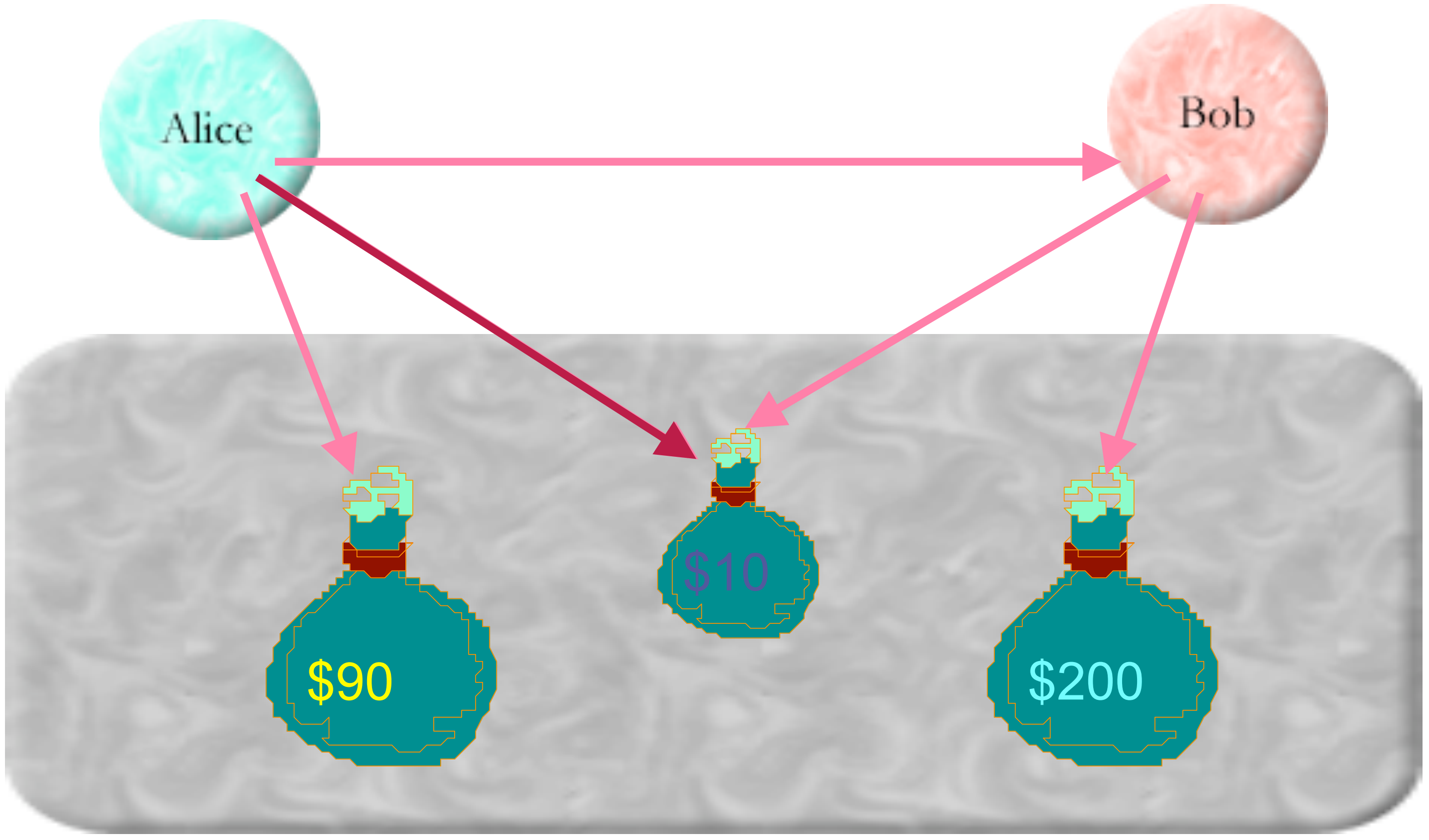
```
return Q.when(paymentP, function(p) {
```



Distributed Secure Currency

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

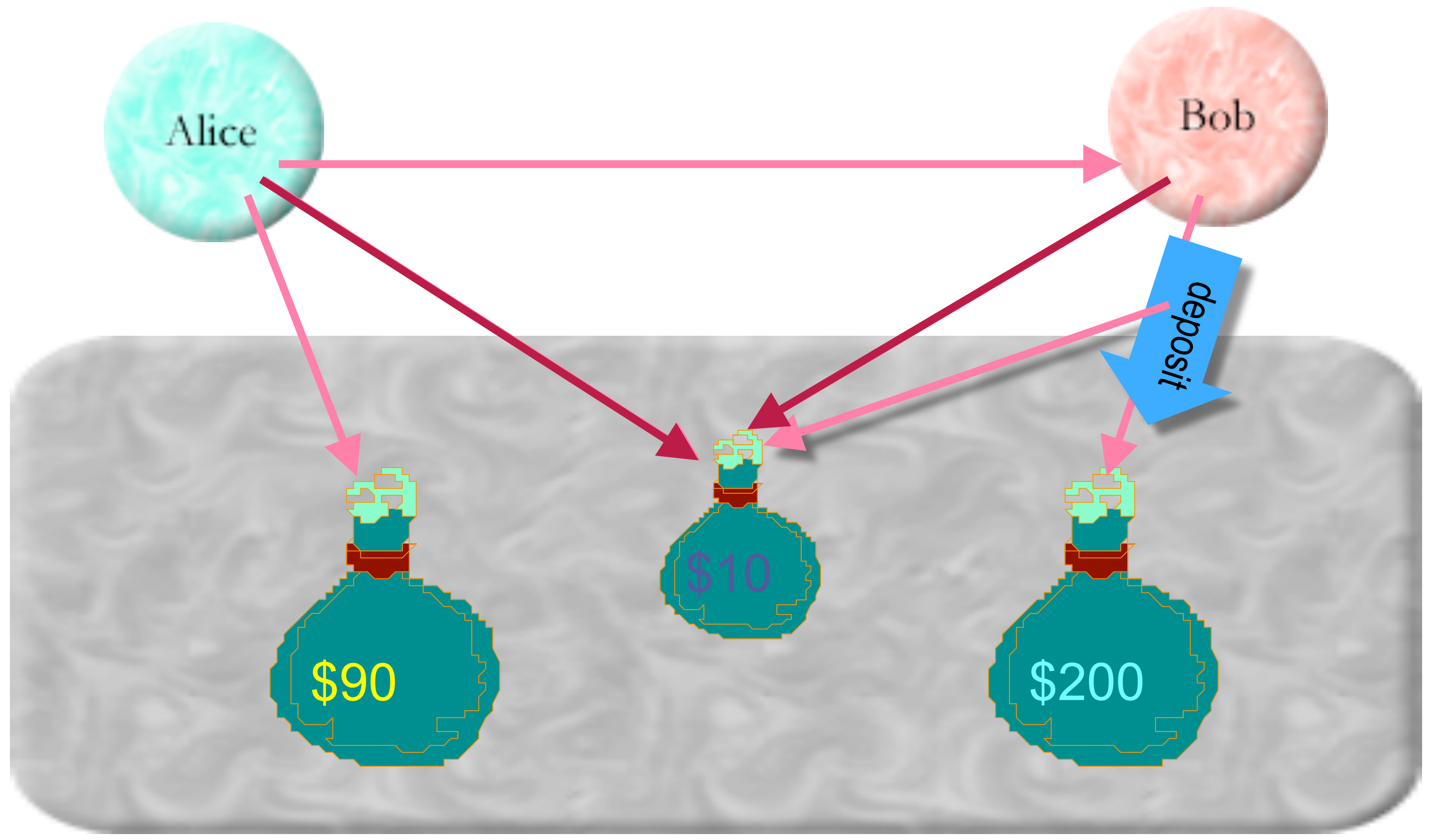
```
return Q.when(paymentP, function(p) {
   return Q.when(myPurse ! deposit(10, p), function(_) {
```



Distributed Secure Currency

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

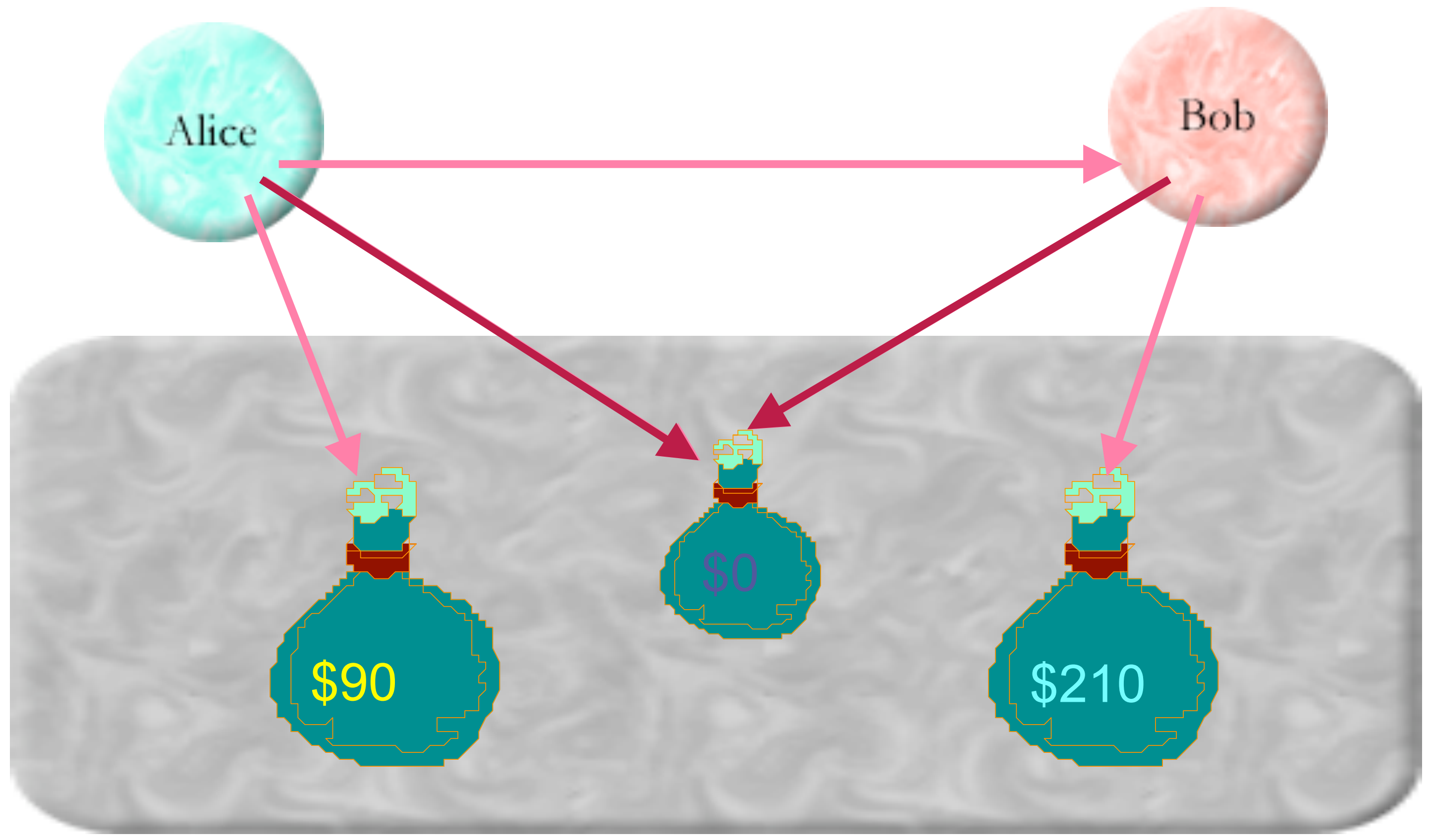
```
return Q.when(paymentP, function(p) {
   return Q.when(myPurse ! deposit(10, p), function(_) {
```



Distributed Secure Currency

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

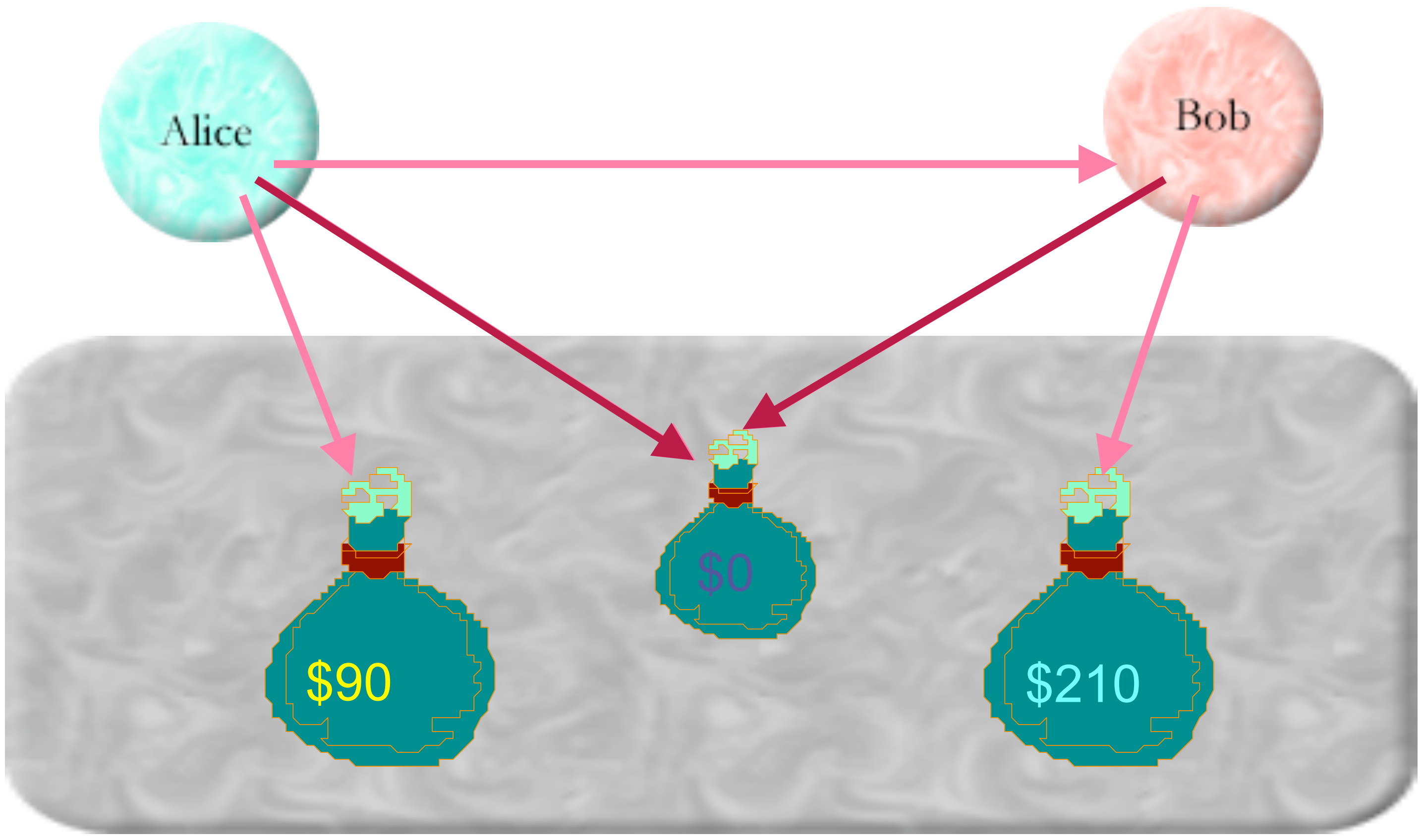
```
return Q.when(paymentP, function(p) {
   return Q.when(myPurse ! deposit(10, p), function(_) {
```



Distributed Secure Currency

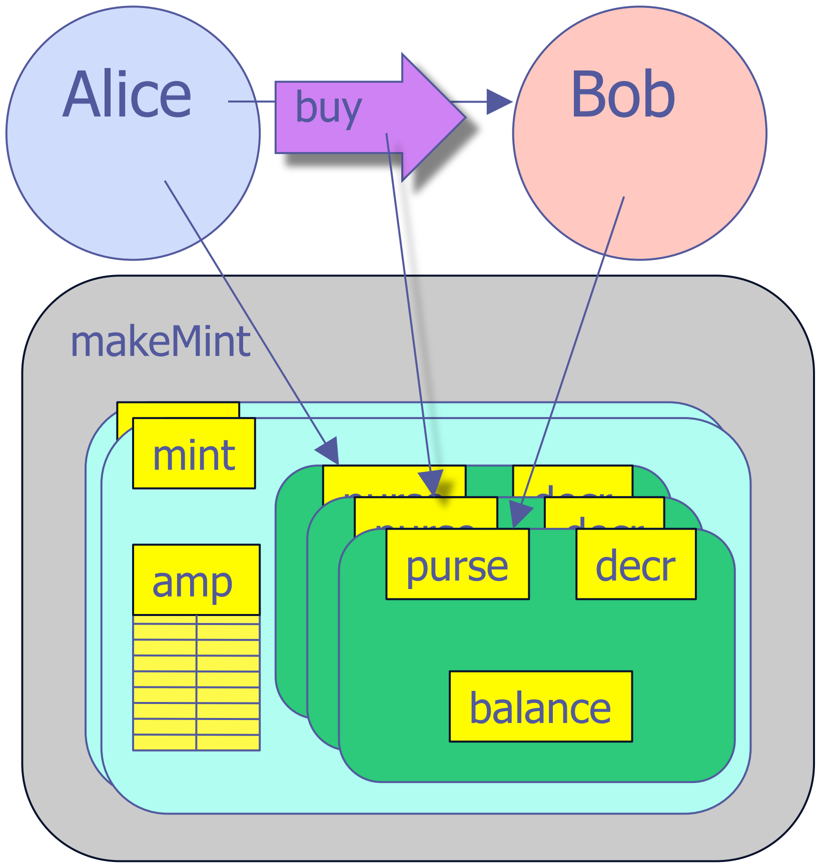
```
var paymentP = myPurse ! makePurse();
paymentP ! deposit(10, myPurse);
var goodP = bobP ! buy(desc, paymentP);
```

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



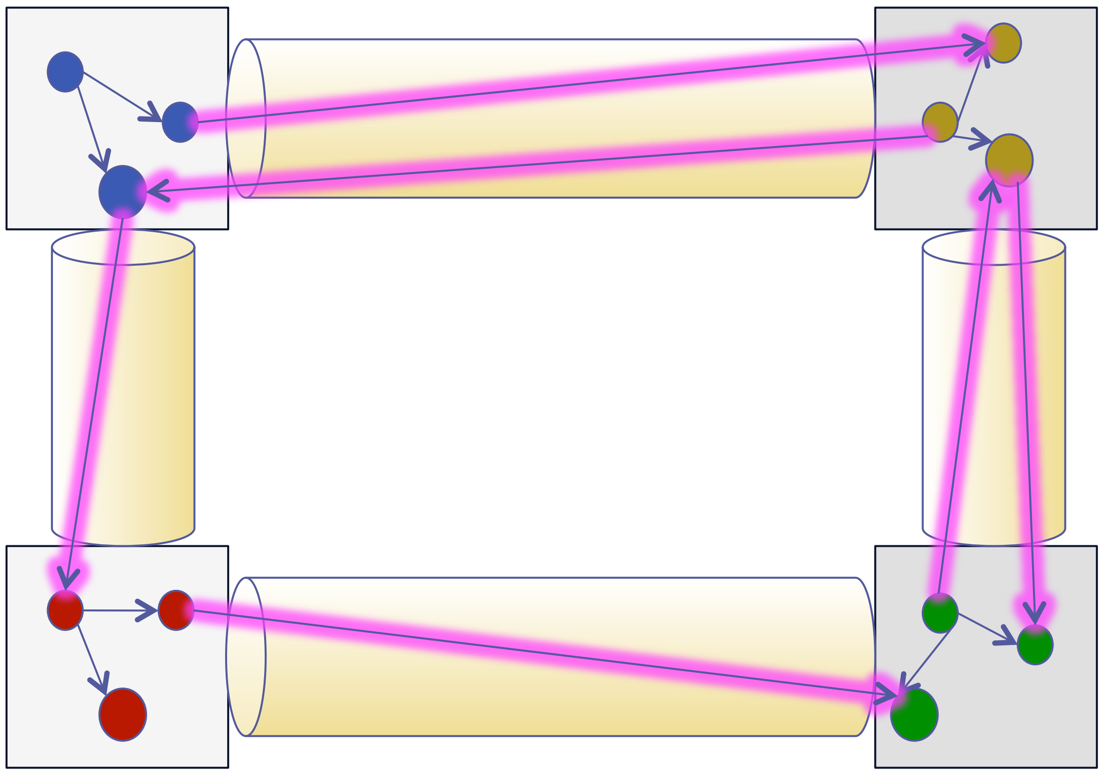
Money as "factorial" of secure coding

No explicit crypto

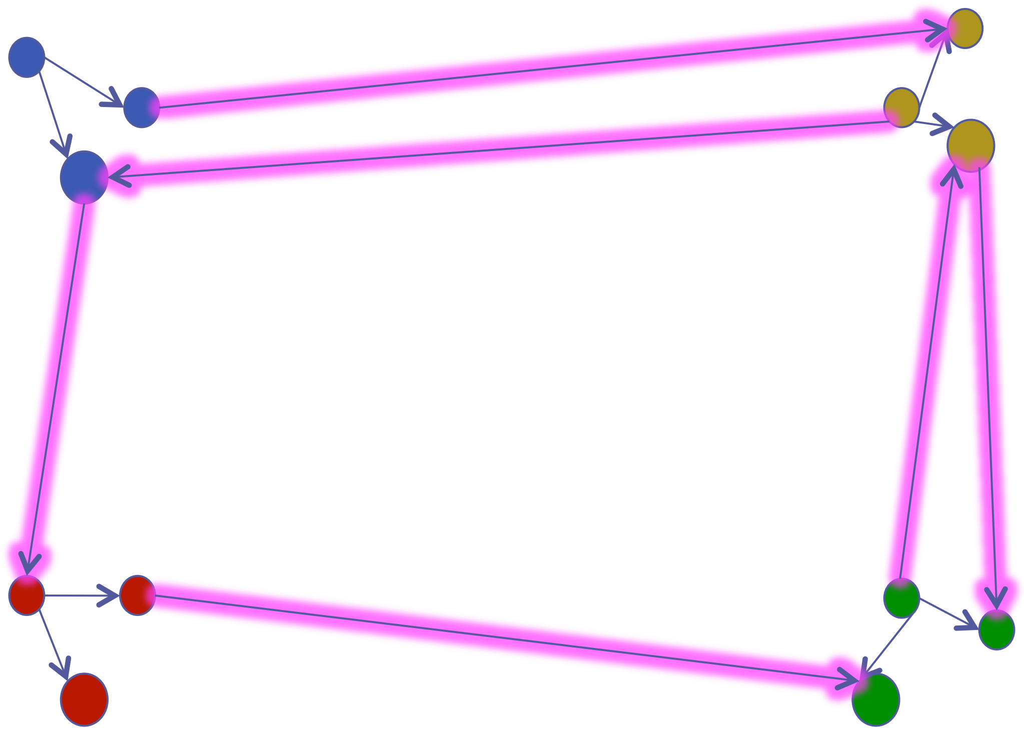


```
function makeMint() {
  var amp = WeakMap();
  return function mint(balance) {
    var purse = def({
      getBalance: function() { return balance; },
      makePurse: function() { return mint(0); },
      deposit: function(amount, src) {
        Nat(balance + amount);
        amp.get(src)(Nat(amount));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

A Web of Distributed Objects



A Web of Distributed Objects



Questions?

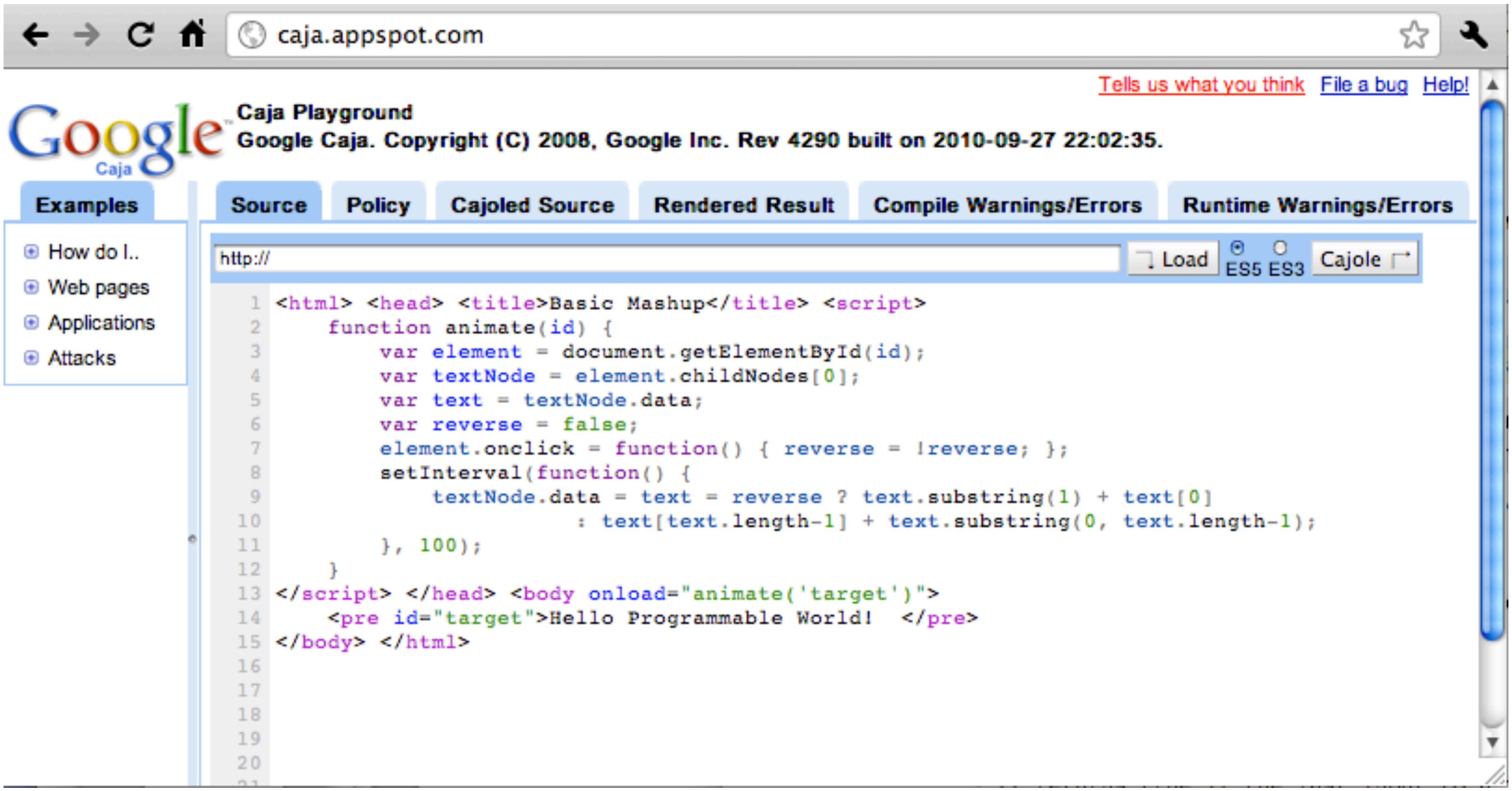
Caja Roadmap

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

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>
```

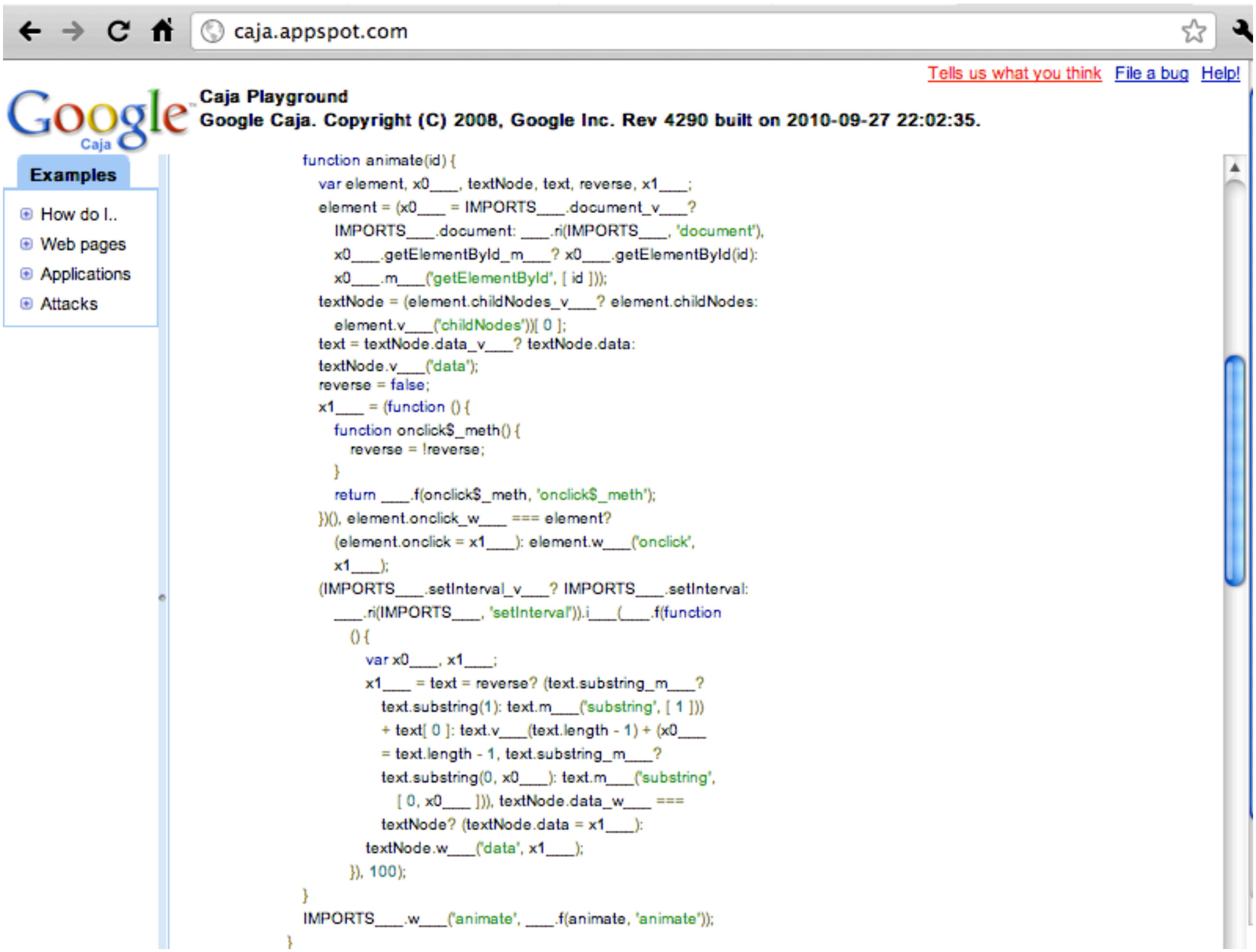
Running ES5 & SES on old browsers



Caja Playground
Google Caja. Copyright (C) 2008, Google Inc. Rev 4290 built on 2010-09-27 22:02:35.

ES5 [✓] ES3

```
<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 Playground
Google Caja. Copyright (C) 2008, Google Inc. Rev 4290 built on 2010-09-27 22:02:35.

```
  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'));
}
```

Dr. SES
Distributed Resilient Secure EcmaScript

	Shared State	**Message Passing**
Blocking	C++/pthreads Java, C#, Mozart/Oz JoCAML, Polyphonic C#	*Blocking receive* CSP, Occam, CCS Erlang, Scala, Go
Non-blocking	*Soft Transactional Mem* Argus, Fortress, X10	*Comm Event Loops* Actors, AmbientTalk E, Waterken **Ajax**

Dr. SES
Distributed Resilient Secure EcmaScript

	Shared State	**Message Passing**
Blocking	C++/pthreads Java, C#, Mozart/Oz JoCAML, Polyphonic C#	*Blocking receive* CSP, Occam, CCS Erlang, Scala, Go
Non-blocking	*Soft Transactional Mem* Argus, Fortress, X10	*Comm Event Loops* Actors, AmbientTalk E, Waterken **Ajax**

No conventional deadlocks or memory races

Dr. SES
Distributed Resilient Secure EcmaScript

	Shared State	**Message Passing**
Blocking	C++/pthreads Java, C#, Mozart/Oz JoCAML, Polyphonic C#	*Blocking receive* CSP, Occam, CCS Erlang, Scala, Go
Non-blocking	*Soft Transactional Mem* Argus, Fortress, X10	*Comm Event Loops* Actors, AmbientTalk E, Waterken **Ajax, Dr. SES**

No conventional deadlocks or memory races

```
var result = bob.foo(carol);        // do it immediately

var resultP = bobP ! foo(carol);    // do it eventually
```