

wasm as an ocap machine

Mark S. Miller
Tea and Crumpets, 5 Dec 2017

EarthEngine's SES Plugins

The screenshot displays the Google Earth Engine Playground interface. The top navigation bar includes the Google Earth Engine logo, a search bar, and user information (Help, nicholas.clinton). The left sidebar shows a list of methods under the 'Scripts' tab, including ee.Algorithms, ee.Array, ee.Blob, ee.Date, ee.DateRange, ee.Dictionary, ee.ErrorMargin, ee.Feature, ee.FeatureCollection, ee.Filter, and ee.Geometry. The main area is titled 'Playground.js' and contains a script for loading the most recent MODIS composite, sorting it by time, and printing it to the console. The script also defines a Simple Image Display (sld) for visualizing the data. The right sidebar shows the 'Inspector' tab, which displays the metadata for the loaded MODIS composite and DEM data. The bottom section shows a map view of the world, with a 'Layers' panel on the right and a 'Map' button. The map is overlaid with a large, semi-transparent blue grid pattern.

Google Earth Engine

Search places and datasets...

Scripts Docs

Filter methods...

ee.Algorithms

ee.Array

ee.Blob

ee.Date

ee.DateRange

ee.Dictionary

ee.ErrorMargin

ee.Feature

ee.FeatureCollection

ee.Filter

ee.Geometry

Playground.js

Get Link Save Run Reset

```
1 Imports (5 entries)
2 // load the most recent MODIS composite
3 var modis = ee.Image(imageCollection
4   .sort('system:time_start', false)
5   .first());
6
7 // print metadata to the console
8 print(modis);
9
10 var sld = "\
11   <RasterSymbolizer>\
12     <ContrastEnhancement><Normalize/></ContrastEnhancement>\
13     <ChannelSelection>\
14       <RedChannel>\
15         <SourceChannelName>sur_refl_b01</SourceChannelName>\
16       </RedChannel>\
17       <GreenChannel>\
18         <SourceChannelName>sur_refl_b04</SourceChannelName>\
```

Inspector Console Tasks

Point (13.54, 23.56) at 20Km/px

Pixels

- MODIS composite: Image (3 bands)
- DEM: Image (2 bands)

Objects

- MODIS composite: Image (3 bands)
 - type: Image
 - bands: List (3 elements)
 - properties: Object (5 properties)
- DEM: Image NOAA/NGDC/ETOP01 (2 bands)
 - type: Image
 - id: NOAA/NGDC/ETOP01
 - version: 1406914481423000
 - bands: List (2 elements)
 - properties: Object (17 properties)

Layers Map Satellite Deep

Google

Map data ©2015 Google, INEGI 1000 km Terms of Use

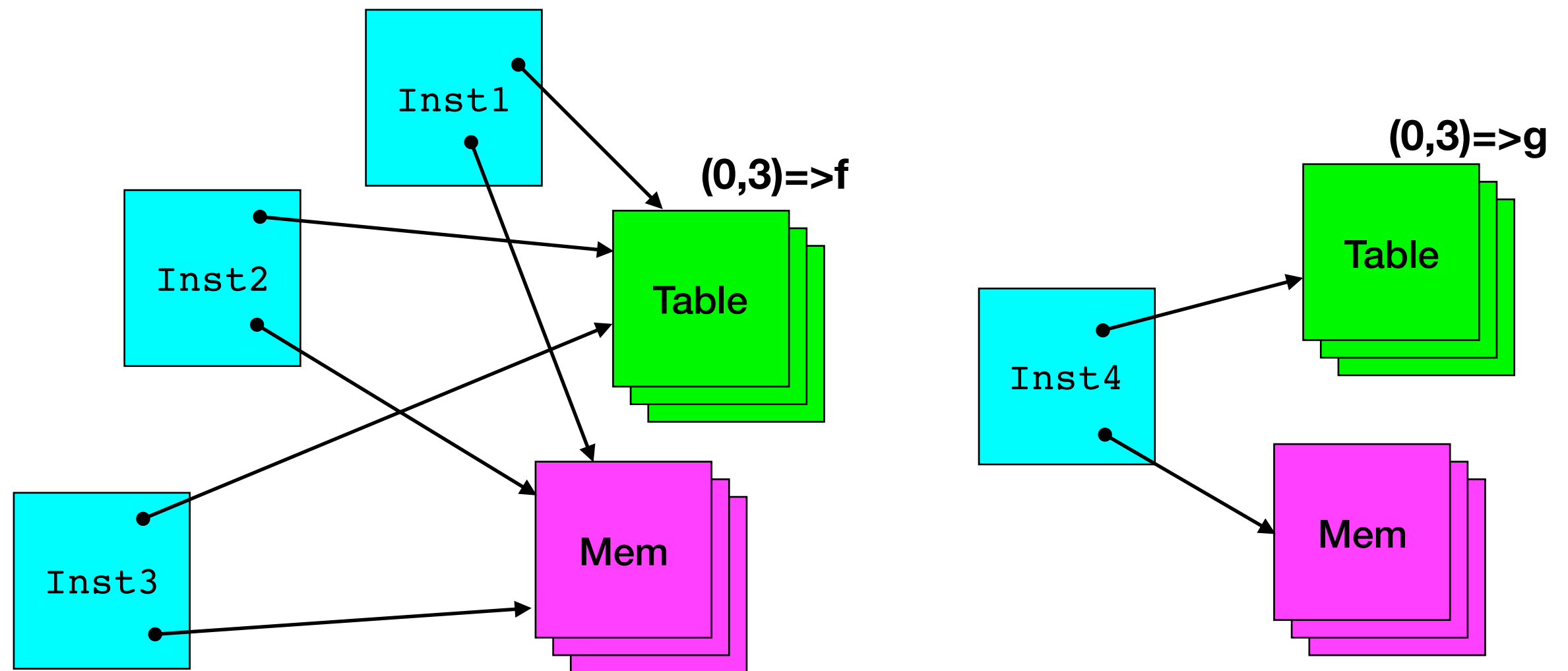
Apps with Fallible Plugins

Third party plugins add value, usually.
May go haywire.

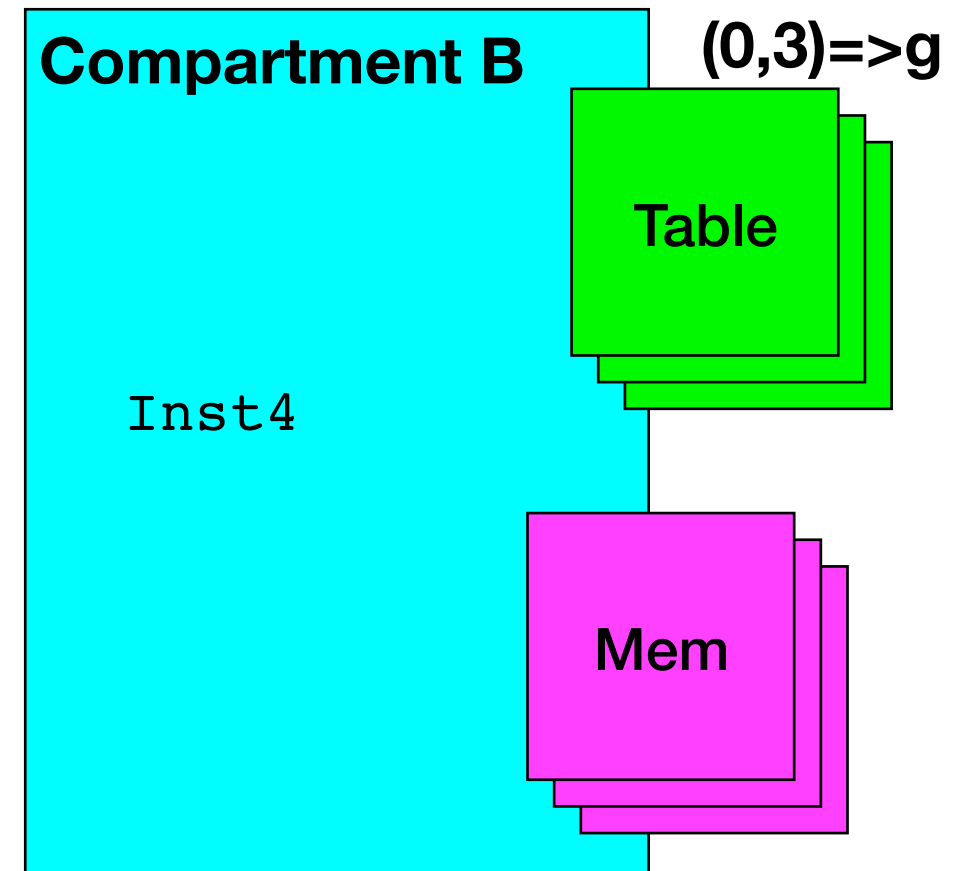
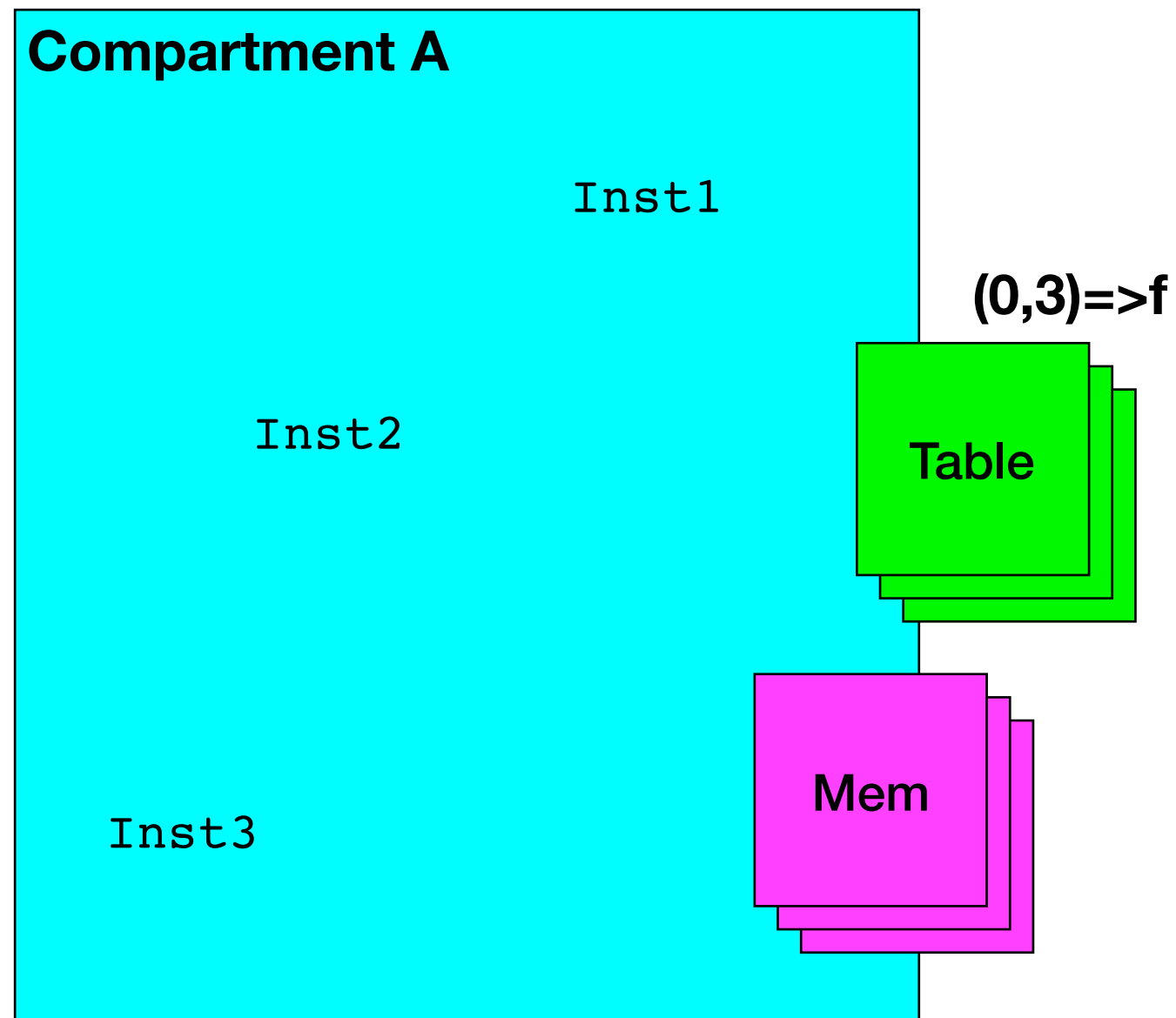
Fault containment \rightarrow ocap safety
wasm already perfect sandbox

Rich defensive APIs \rightarrow ocap expressiveness
 $\text{wasm} \subset \mathbf{\text{wasm-linkage}} \subset \text{wasm-ocap} \subset \text{wasm-gc}$

Number passing adequate?



Compartment pattern



wasm

```
num_type      ::= i32 | i64 | f32 | f64
ref_type      ::= (ref <def_type>) | intref | anyref | anyfunc
value_type    ::= <num_type> | <ref_type>

packed_type   ::= i8 | i16
storage_type  ::= <value_type> | <packed_type>
field_type    ::= <storage_type> | (mut <storage_type>)

data_type     ::= (struct <field_type>*) | (array <field_type>)
func_type     ::= (func <value_type>* <value_type>*)
def_type      ::= <func_type> | <data_type>
```

Unmanaged objects

exporting/importing opaque unforgeable functions that dynamically pass numbers.

Per compartment memory discipline.

wasm-linkage baseline

```
num_type      ::= i32 | i64 | f32 | f64
ref_type      ::= (ref <def_type>) | intref | anyref | anyfunc
value_type    ::= <num_type> | <ref_type>
```

```
packed_type ::= i8 | i16
storage_type ::= <value_type> | <packed_type>
field_type ::= <storage_type> | (mut <storage_type>)
```

```
data_type ::= (struct <field_type>*) | (array <field_type>)
func_type  ::= (func <value_type>* <value_type>*)
def_type   ::= <func_type> | <data_type>
```

Unmanaged objects
dynamically passing
opaque unforgeable refs-to-function.
Per compartment memory discipline.

wasm-linkage baseline

```
num_type      ::= i32 | i64 | f32 | f64
ref_type      ::= (ref <def_type>) | intref | anyref | anyfunc
value_type    ::= <num_type> | <ref_type>
```

```
packed_type ::= i8 | i16
storage_type ::= <value_type> | <packed_type>
field_type ::= <storage_type> | (mut <storage_type>)
```

```
data_type ::= (struct <field_type>*) | (array <field_type>)
func_type  ::= (func <value_type>* <value_type>*)
def_type   ::= <func_type> | <data_type>
```

Unmanaged objects
dynamically passing
opaque unforgeable refs-to-function.
Per compartment memory discipline.

wasm-gc

```
num_type      ::= i32 | i64 | f32 | f64
ref_type      ::= (ref <def_type>) | intref | anyref | anyfunc
value_type    ::= <num_type> | <ref_type>
```

```
packed_type   ::= i8 | i16
storage_type  ::= <value_type> | <packed_type>
field_type    ::= <storage_type> | (mut <storage_type>)
```

```
data_type     ::= (struct <field_type>*) | (array <field_type>)
func_type     ::= (func <value_type>* <value_type>*)
def_type      ::= <func_type> | <data_type>
```

Typed managed objects
dynamically passing
opaque unforgeable refs.
Dynamically allocated and garbage collected.

wasm-linkage baseline

ref-to-function is two words

- pointer to function code
- pointer to module instance

passed by value

- on the stack (params, locals, operand stack)
- read and written as table entries

function code gets

- pointer to module instance
- passed arguments

wasm-linkage beyond baseline

- Host bindings
- Memory-range capabilities
- Table-range capabilities
- Generalized fat pointers
- Unmanaged closures

Unmanaged closure

ref-to-closure is three words

- pointer to function code
- pointer to module instance
- i32 facet-id

function code gets

- pointer to module instance
- i32 facet-id
- passed arguments

Unmanaged!

Unmanaged **fat** closure

Closure's "type" implies both **size** and signature

- pointer to function code
- pointer to module instance
- **value_type** facet-id

function code gets

- pointer to module instance
- **value_type** facet-id
- passed arguments

Closure as unmanaged curried function

Substrate Independent Cap Logic

Hardware	Cap, Intel 432, CHERI
OS	DVH, KeyKOS, Capsicum, Midori, seL4
Language	Gedanken, E, Joe-E, Emily, M#, Dr.SES
Crypto Protocol	DCCS, CapTP, Waterken, Cap'n Proto
Offline Certs	SPKI/SDSI, CapCert, Macaroons, Ids-ocap
Blockchain	Gravity, Dfinity
User Interface	CapDesk, Belay

Substrate Independent Cap Logic

Hardware

Cap, Intel 432, **CHERI**

OS

DVH, KeyKOS, Capsicum, **Midori**, **seL4**

Language

Gedanken, E, Joe-E, Emily, **M#**, **Dr.SES**

Crypto Protocol

DCCS, CapTP, Waterken, Cap'n Proto

Offline Certs

SPKI/SDSI, CapCert, Macaroons, Ids-ocap

Blockchain

Gravity, Dfinity

User Interface

CapDesk, Belay

wasm has ocap safety

Memory safety

Encapsulation

Defensible “objects” (compartments)

External effects **only** by using held references

No powerful references by default

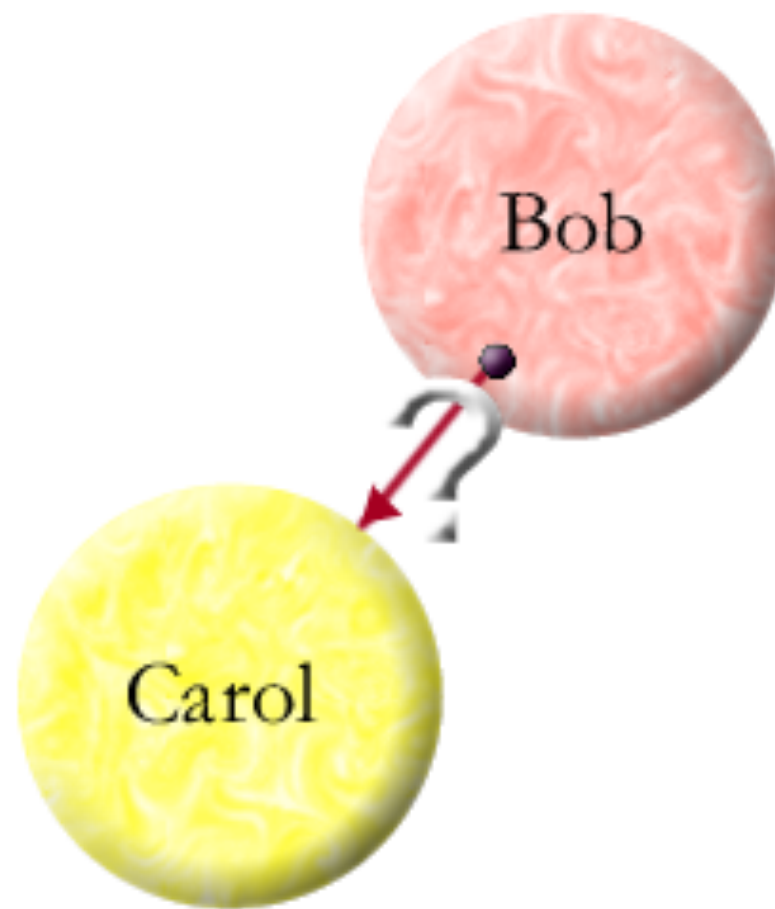
Perfect sandbox

WA.compile: bytes -> Module: Imports -> Instance

Module imports: name => name => ref

Instance exports: name => ref

How do I designate thee?



by Introduction

ref to Carol

ref to Bob

decides to share

by Parenthood

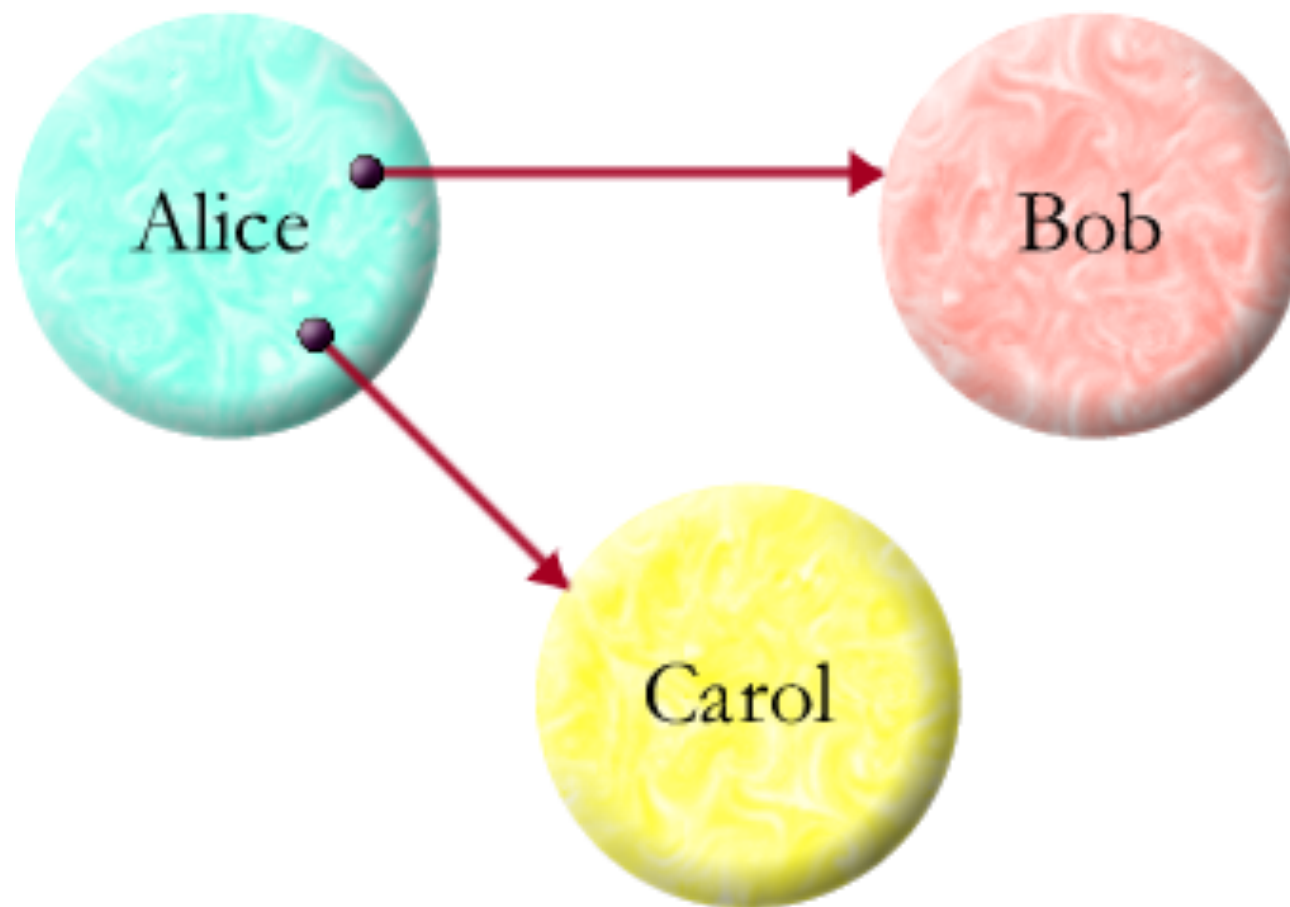
by Endowment

by Initial Conditions

How might object Bob come to know of object Carol?

How do I designate thee?

Alice says: `bob.foo(carol)`



by Introduction

ref to Carol

ref to Bob

decides to share

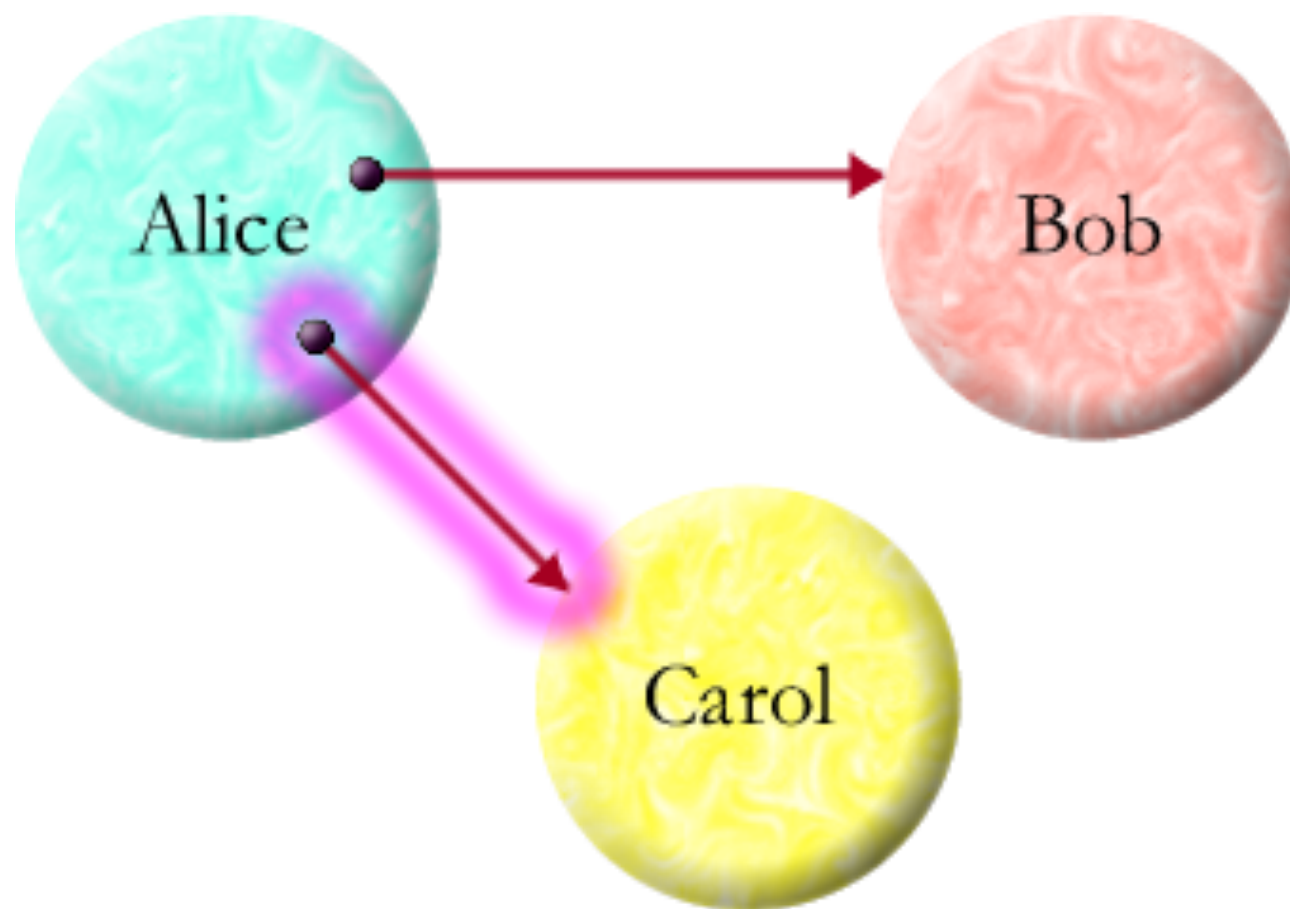
by Parenthood

by Endowment

by Initial Conditions

How do I designate thee?

Alice says: `bob.foo(carol)`



by Introduction

ref to Carol

ref to Bob

decides to share

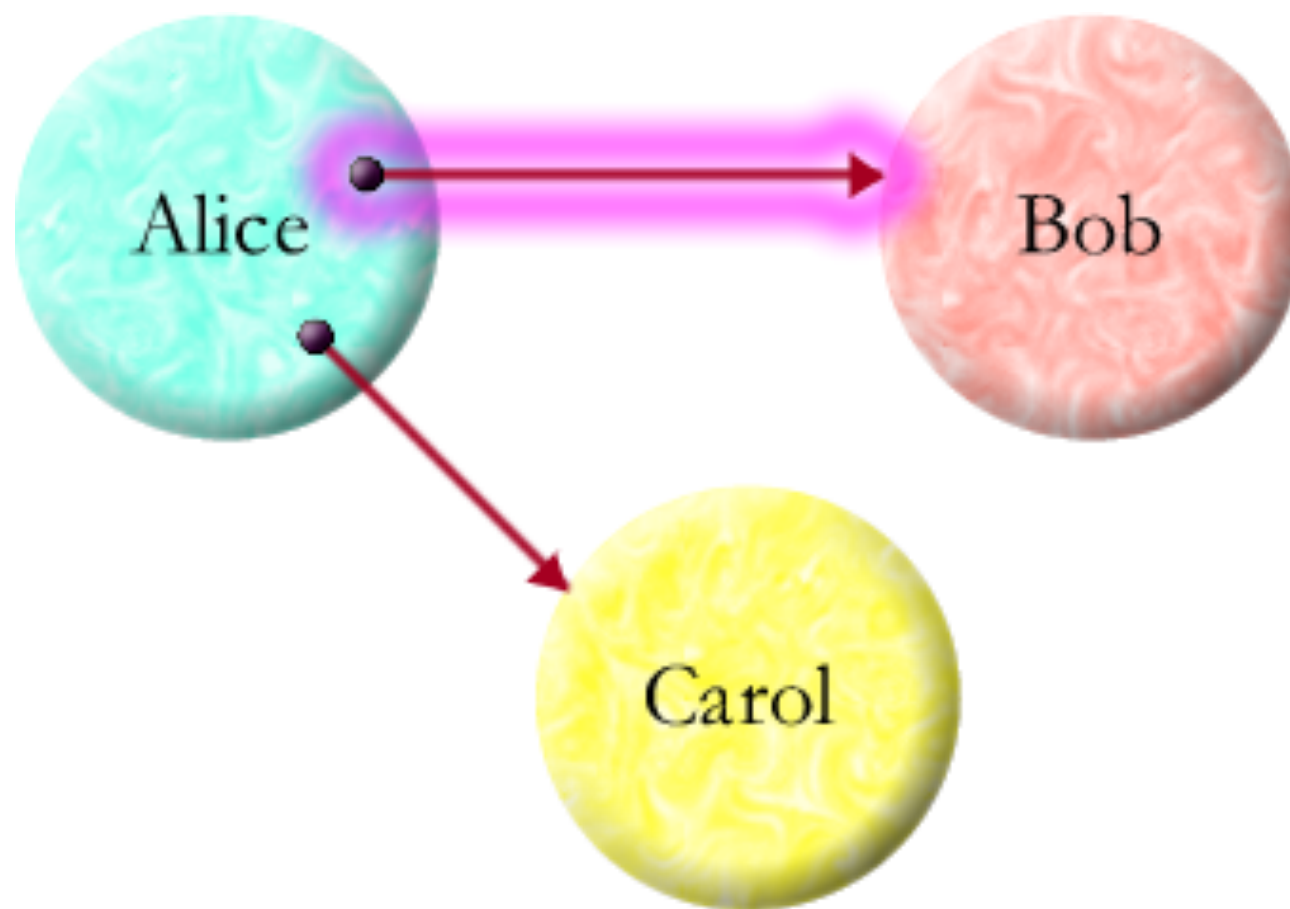
by Parenthood

by Endowment

by Initial Conditions

How do I designate thee?

Alice says: `bob.foo(carol)`



by Introduction

ref to Carol

ref to Bob

decides to share

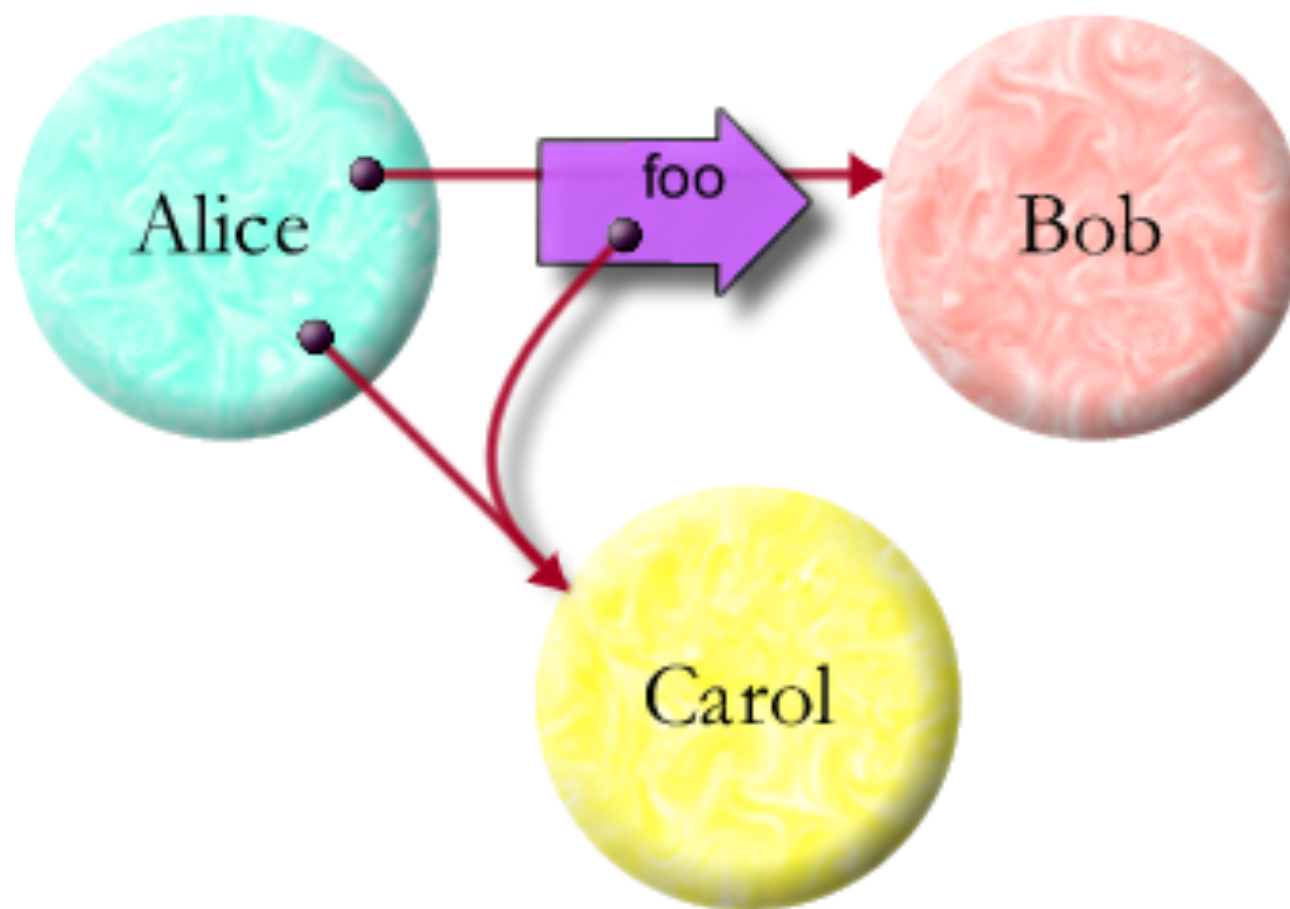
by Parenthood

by Endowment

by Initial Conditions

How do I designate thee?

Alice says: `bob.foo(carol)`



by Introduction

ref to Carol

ref to Bob

decides to share

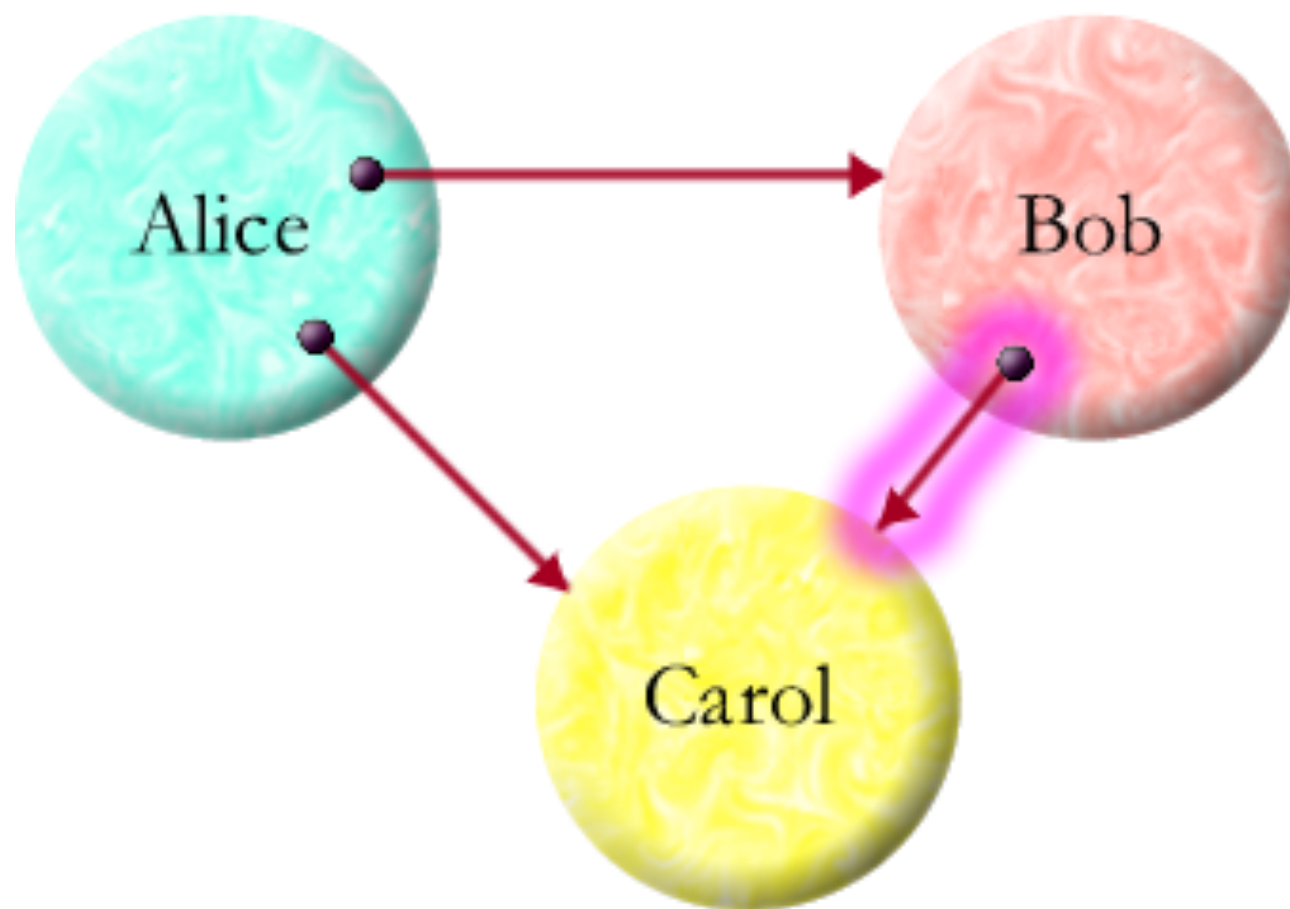
by Parenthood

by Endowment

by Initial Conditions

How do I designate thee?

Alice says: `bob.foo(carol)`



by Introduction
ref to Carol
ref to Bob
decides to share

by Parenthood
by Endowment
by Initial Conditions

How do I designate thee?

Bob says: *var carol* = { ... };



by Introduction

ref to Carol

ref to Bob

decides to share

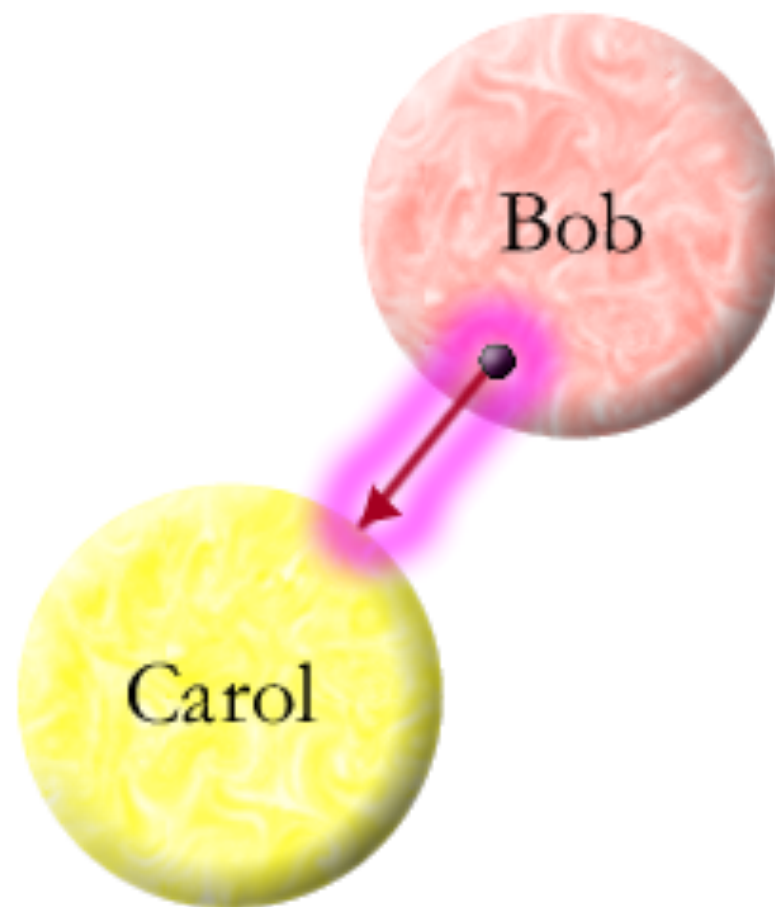
by Parenthood

by Endowment

by Initial Conditions

How do I designate thee?

Bob says: `var carol = { ... };`



by Introduction
ref to Carol
ref to Bob
decides to share

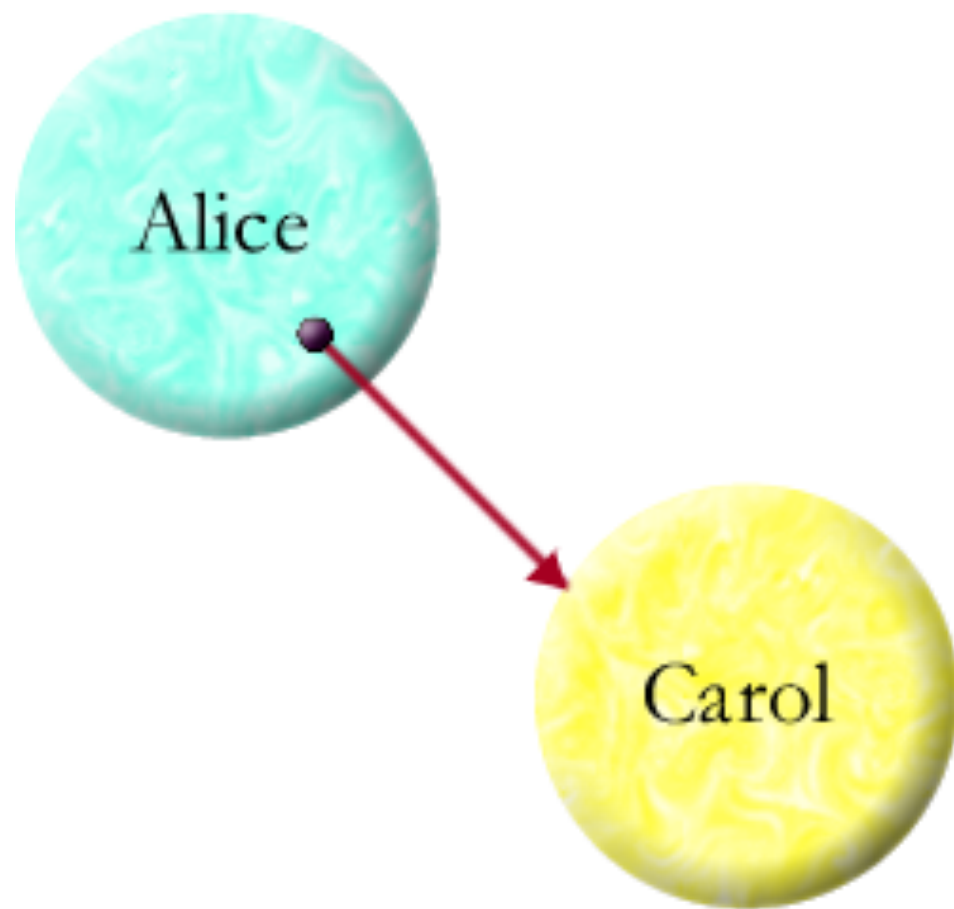
by Parenthood

by Endowment

by Initial Conditions

How do I designate thee?

Alice says: `var bob = { ... carol ... };`



by Introduction

ref to Carol

ref to Bob

decides to share

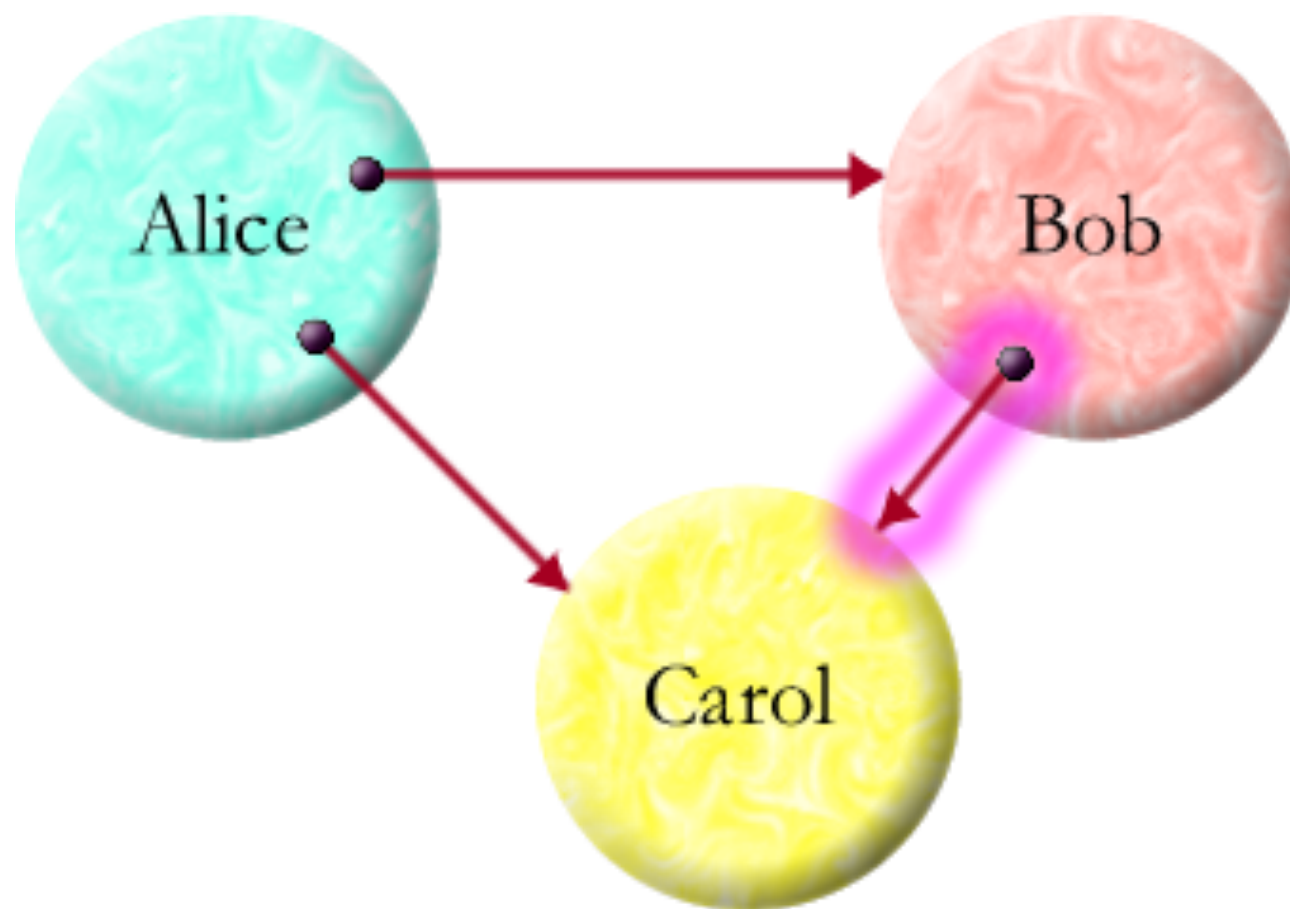
by Parenthood

by Endowment

by Initial Conditions

How do I designate thee?

Alice says: `var bob = { ... carol ... };`



by Introduction

ref to Carol

ref to Bob

decides to share

by Parenthood

by Endowment

by Initial Conditions

How do I designate thee?

At t_0 :

by Introduction

ref to Carol

ref to Bob

decides to share

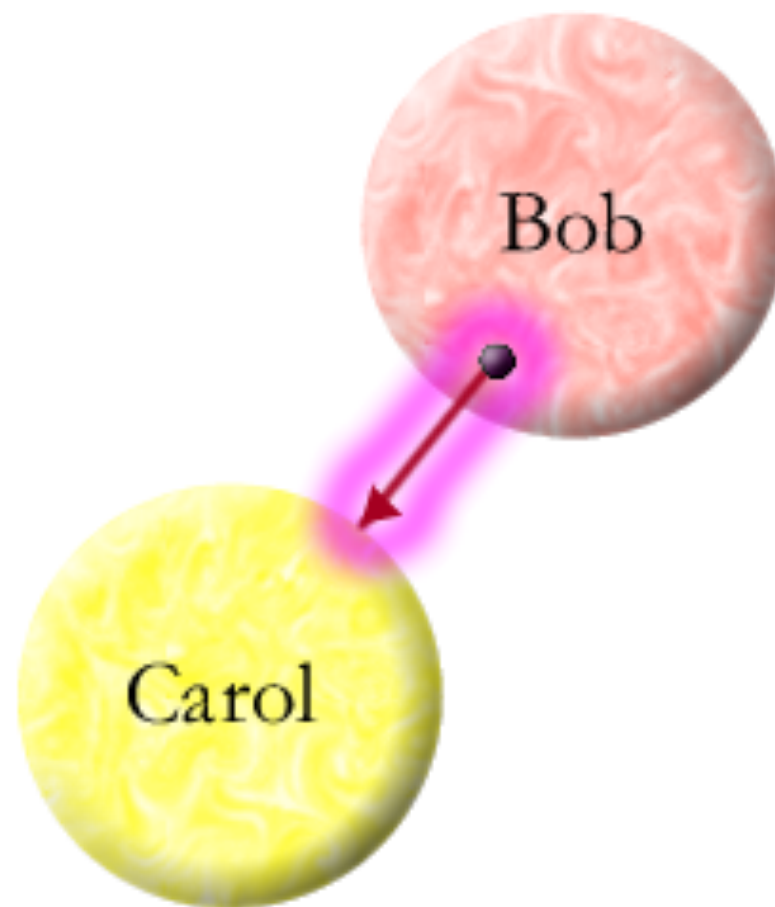
by Parenthood

by Endowment

by Initial Conditions

How do I designate thee?

At t_0 :



by Introduction

ref to Carol

ref to Bob

decides to share

by Parenthood

by Endowment

by Initial Conditions

From wasm-linkage to wasm-ocap

by Introduction
ref to Carol
ref to Bob
decides to share

by Parenthood

by Endowment

by Initial Conditions

WA.compile: bytes -> Module: Imports -> Instance

Module imports: name => name => ref

Instance exports: name => ref

From wasm-linkage to wasm-ocap

“For free”

Trivially provided by embedder
via **host bindings**.

by Introduction

ref to Carol

ref to Bob

decides to share

by Parenthood

by Endowment

by Initial Conditions

WA.compile: bytes -> Module: Imports -> Instance

Module imports: **(name, name) -> ref**

Instance exports: **name -> ref**

From objects to ocaps

Object expressiveness

- + Memory safety
- + Encapsulation
- + Defensible objects
- + External effects only by using held references
- + No powerful references by default

From objects to ocaps

Object expressiveness

- + Memory safety
- + Encapsulation
- + Defensible objects
- + External effects only by using held references
- + No powerful references by default

Reference graph \equiv Access graph

Reachability limits effects

Abstraction boundary \equiv Enforcement mechanism

Abstraction mechanisms for access control