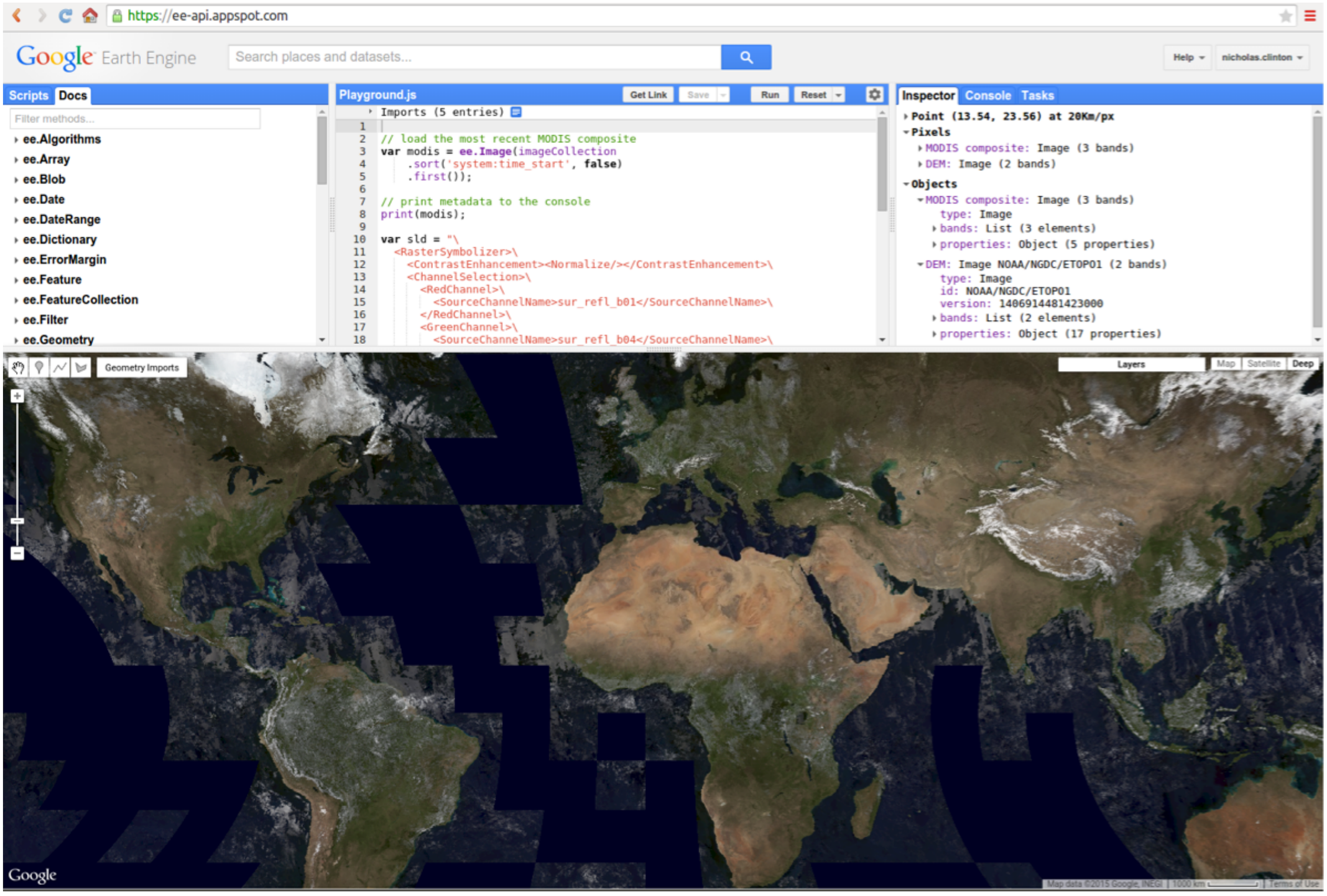
wasm as an ocap machine

Mark S. Miller

Tea and Crumpets, 5 Dec 2017

EarthEngine's SES Plugins



Apps with Fallible Plugins

Third party plugins add value, usually.

May go haywire.

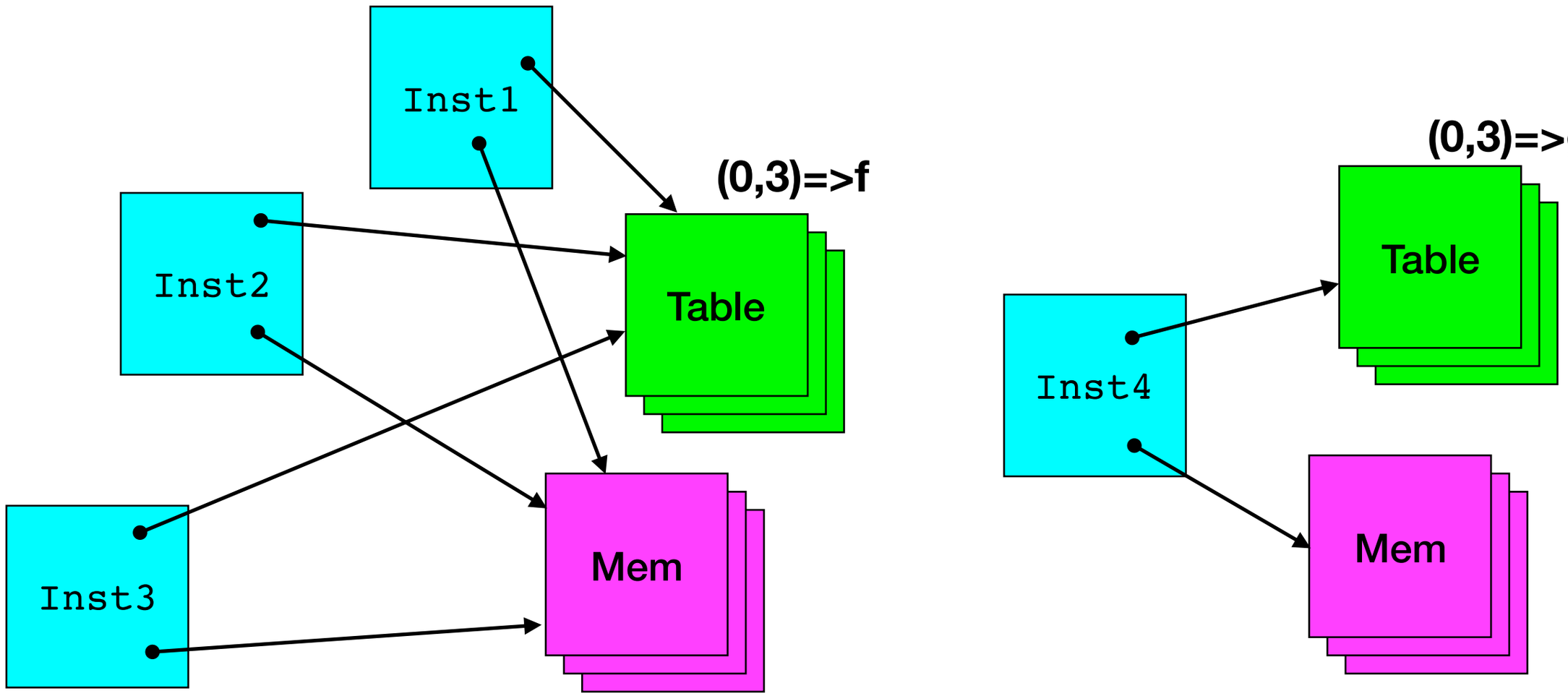
Fault containment —> ocap safety

wasm already perfect sandbox

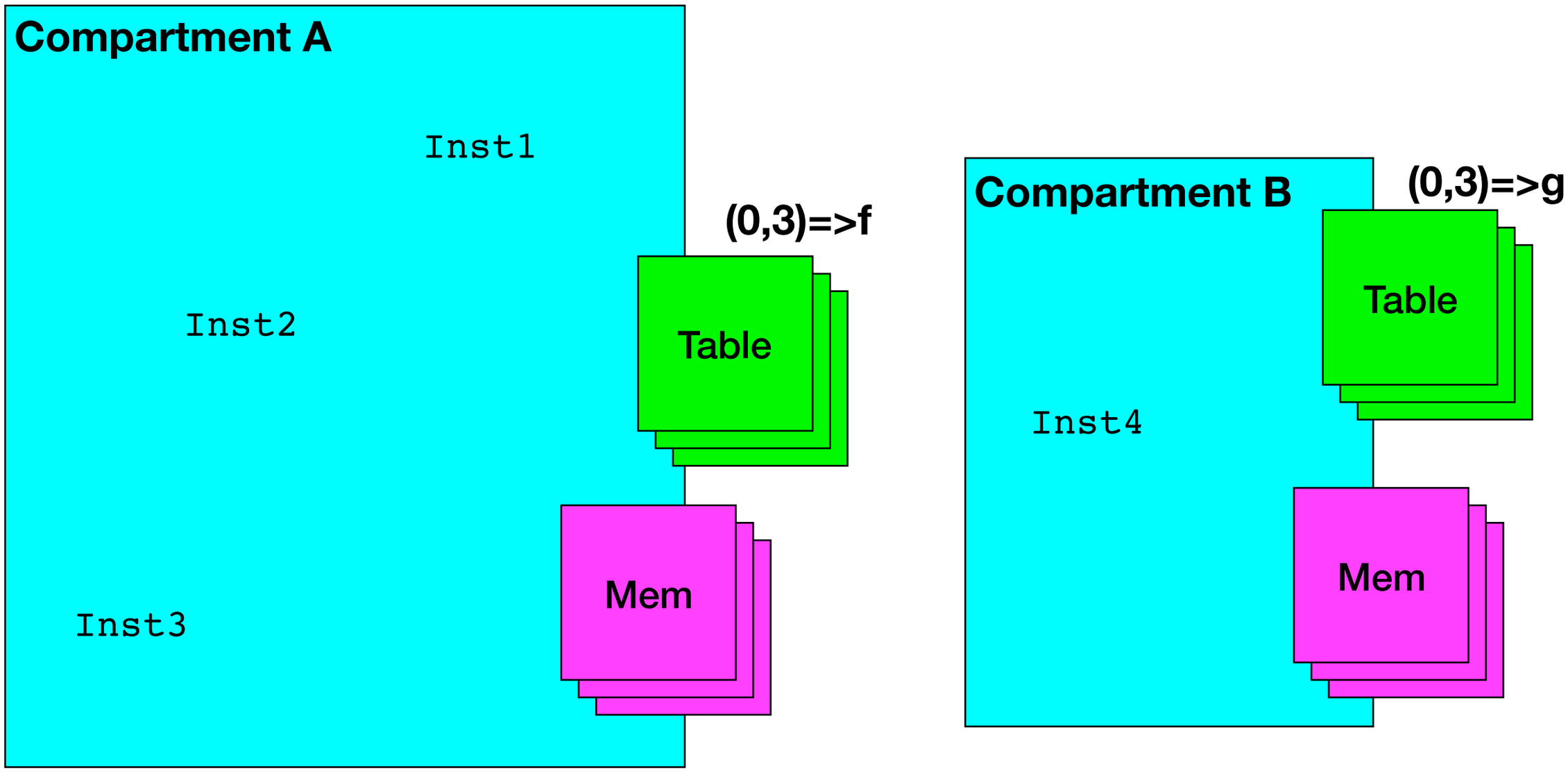
Rich defensive APIs —> ocap expressiveness

wasm ⊂ **wasm-linkage** ⊂ wasm-ocap ⊂ 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, lds-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, lds-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

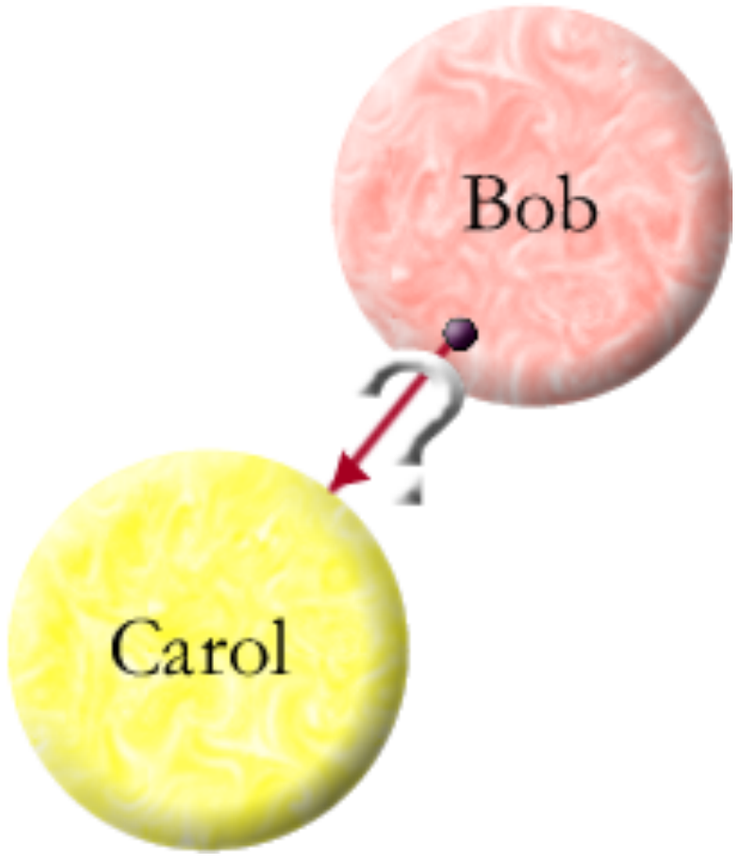
Perfect sandbox

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

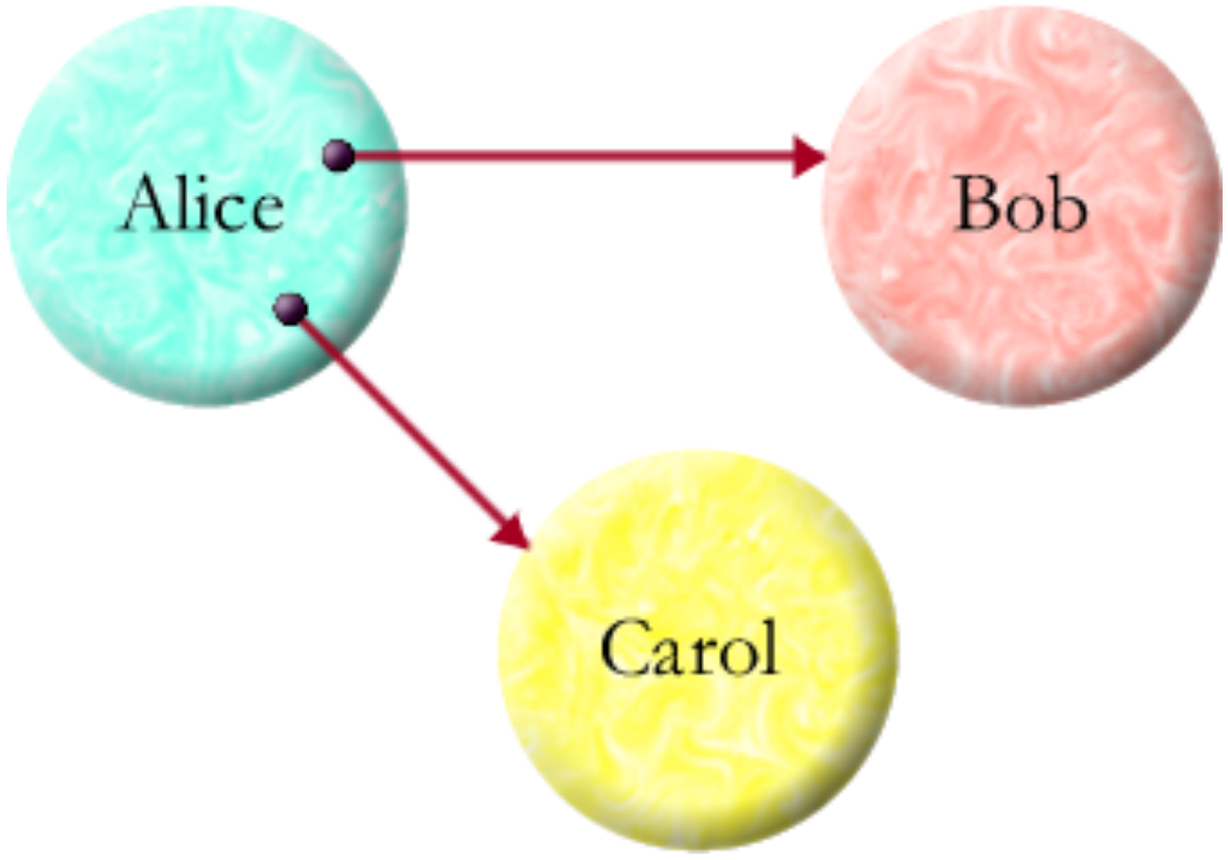
by Endowment

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

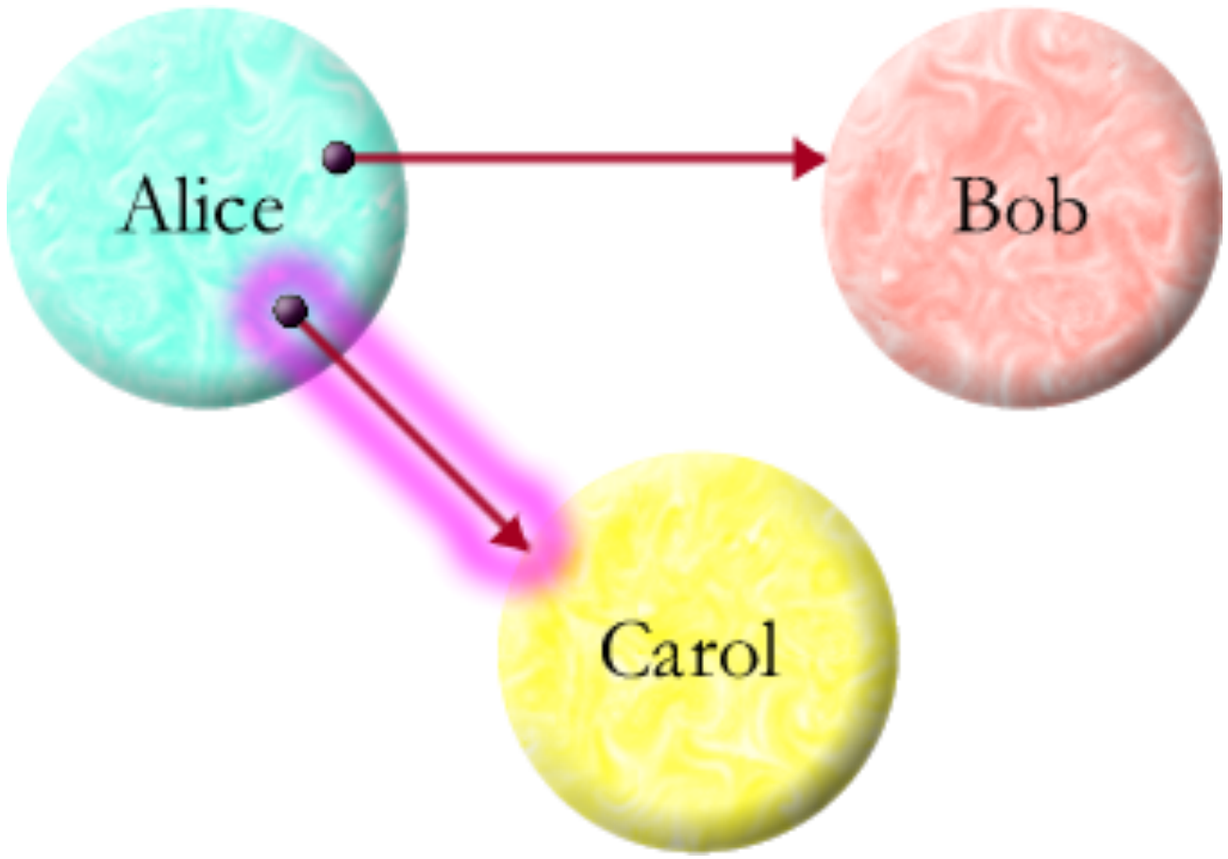
by Parenthood

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

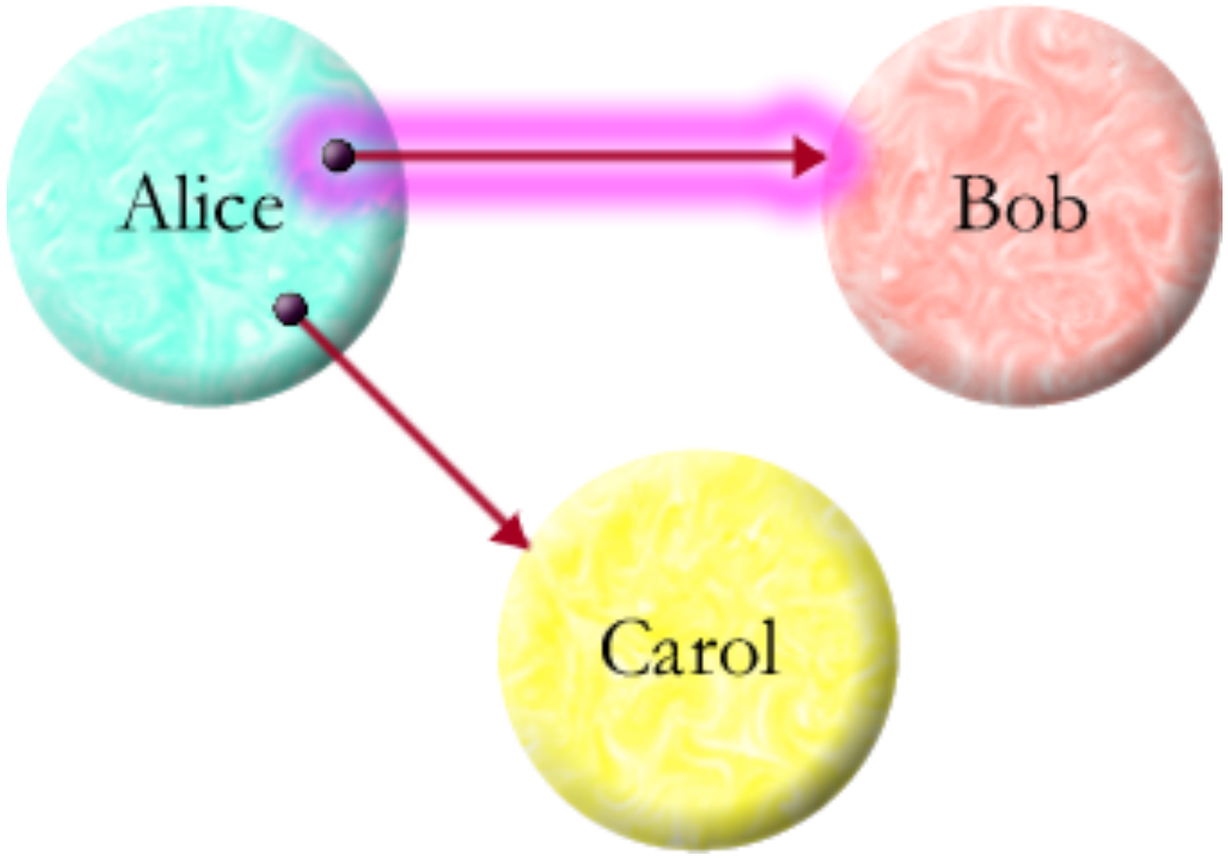
by Parenthood

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

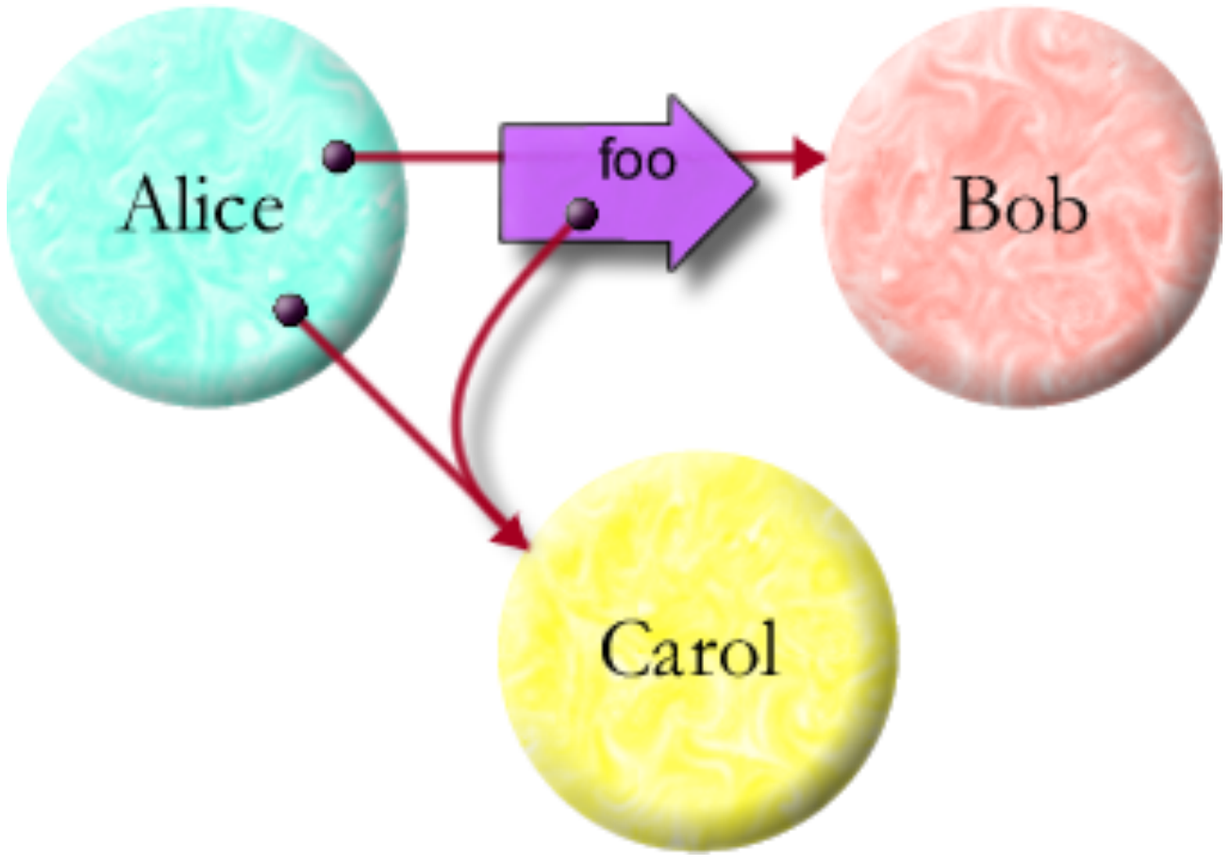
by Parenthood

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**

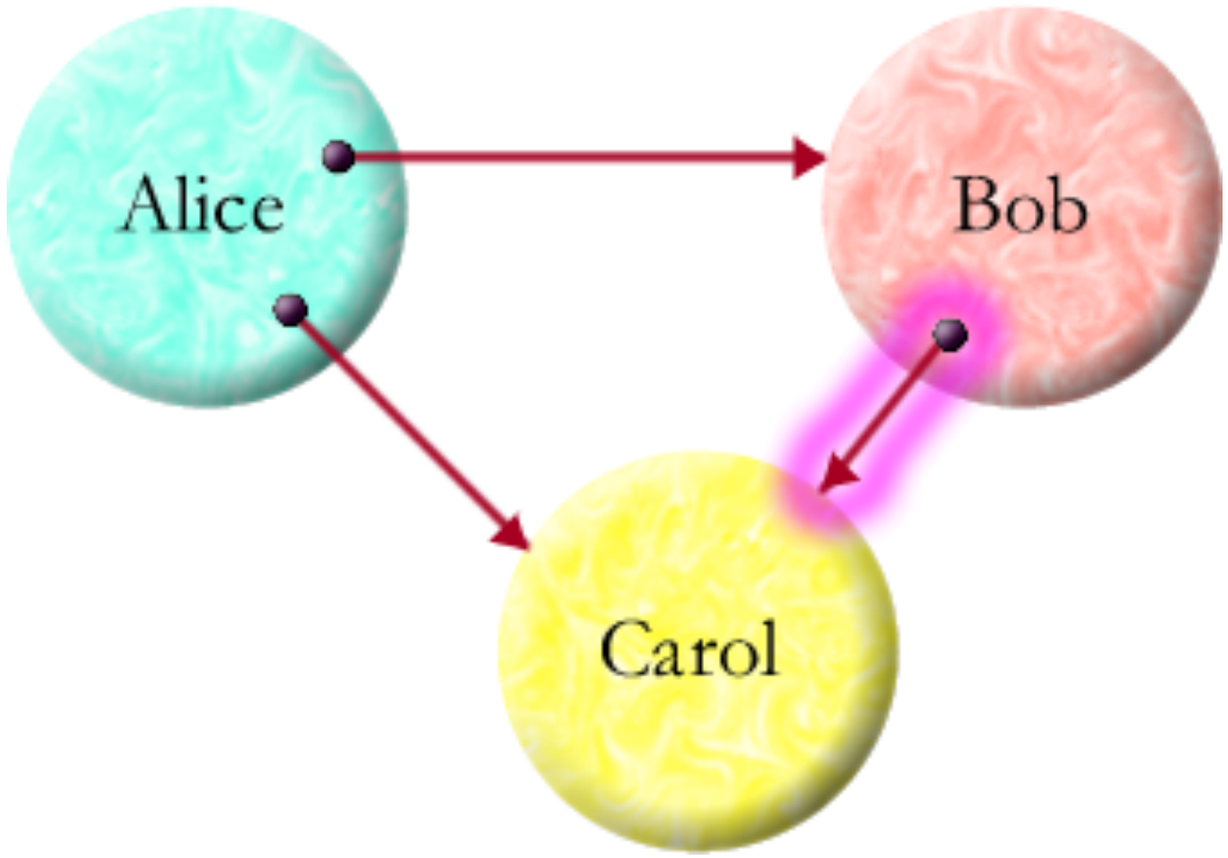
by Parenthood

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

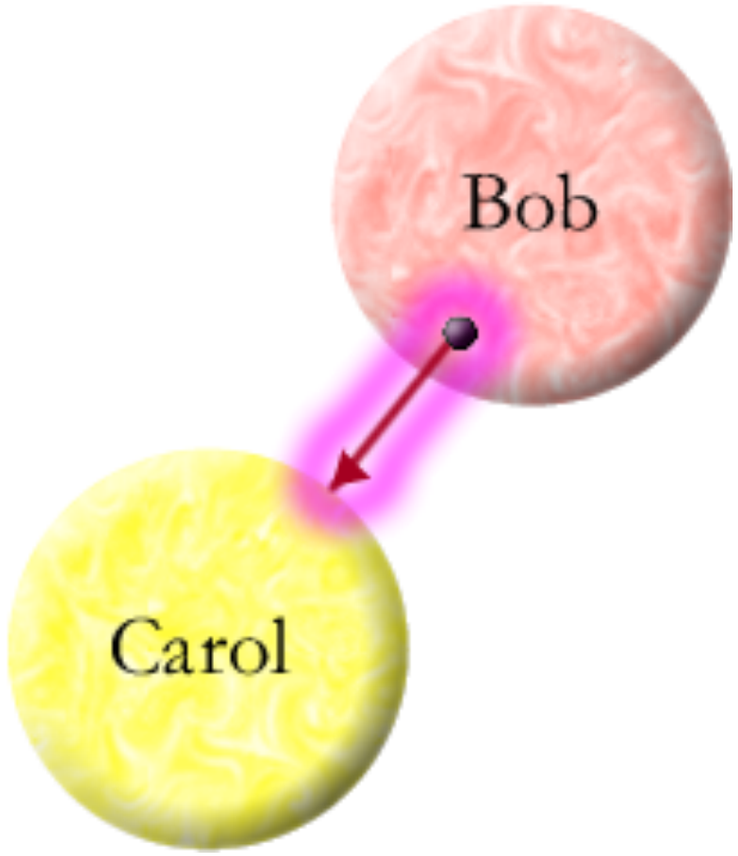
by Parenthood

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

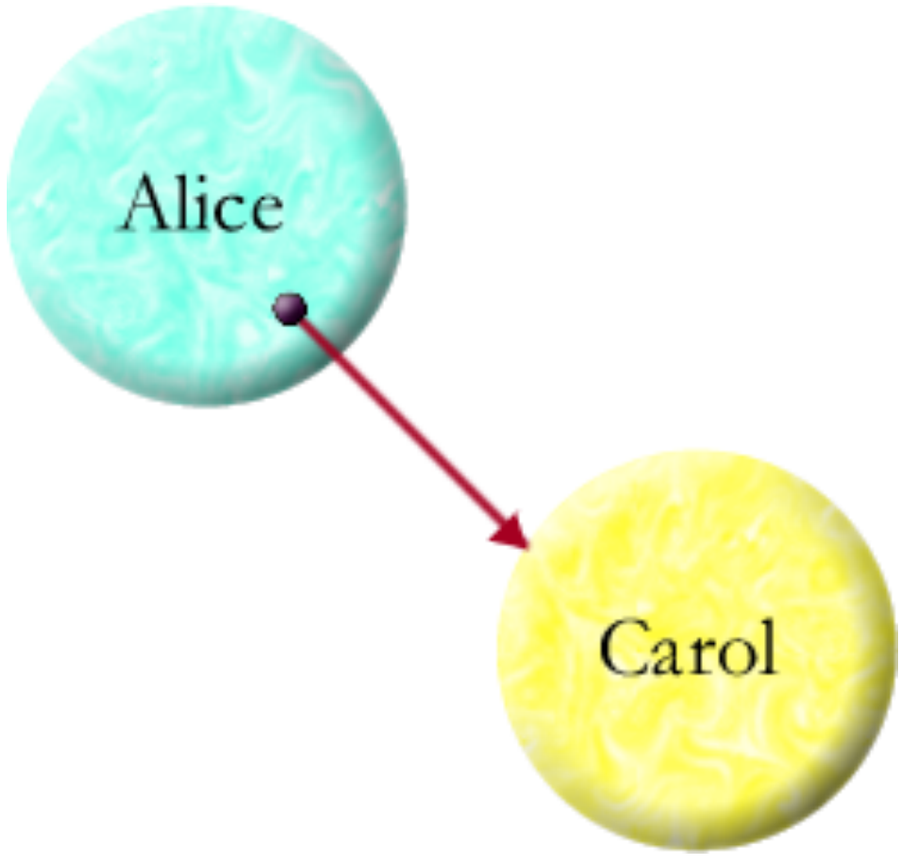
by Parenthood

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

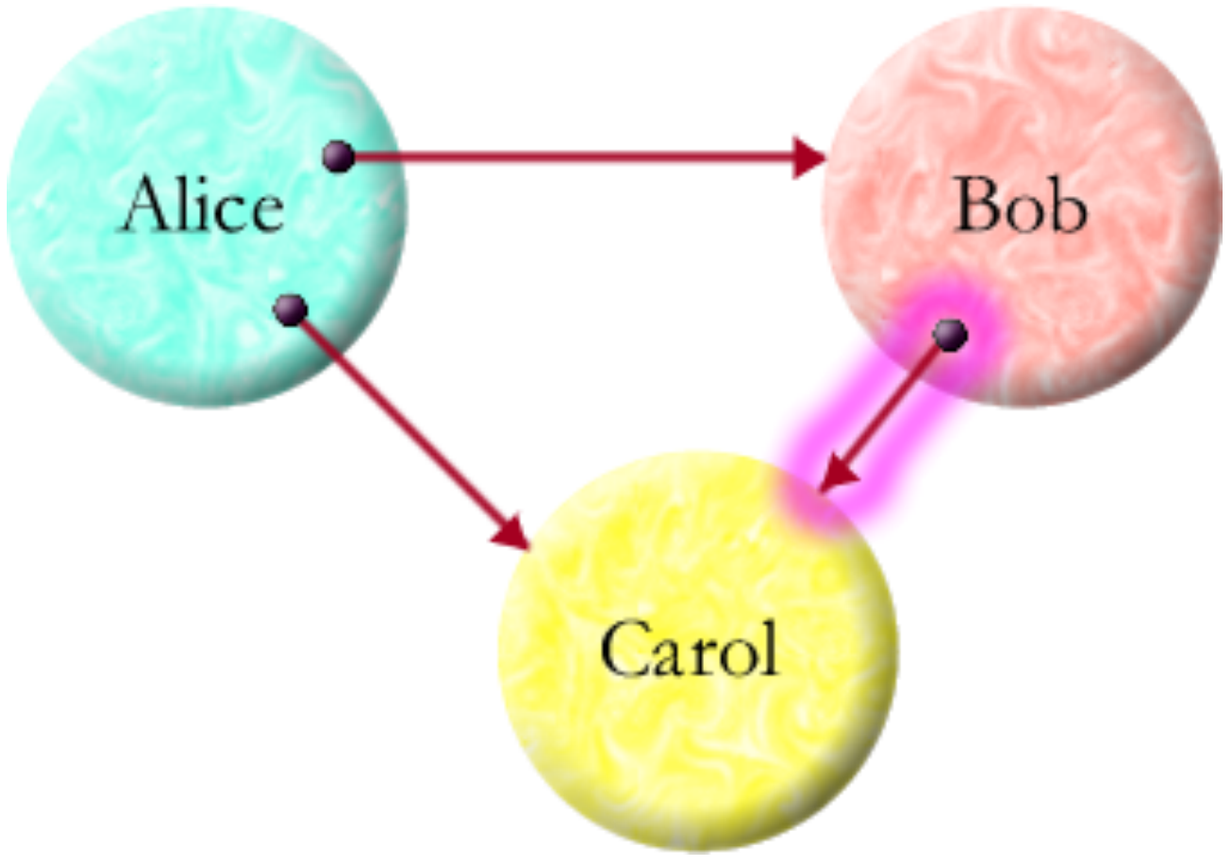
by Parenthood

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

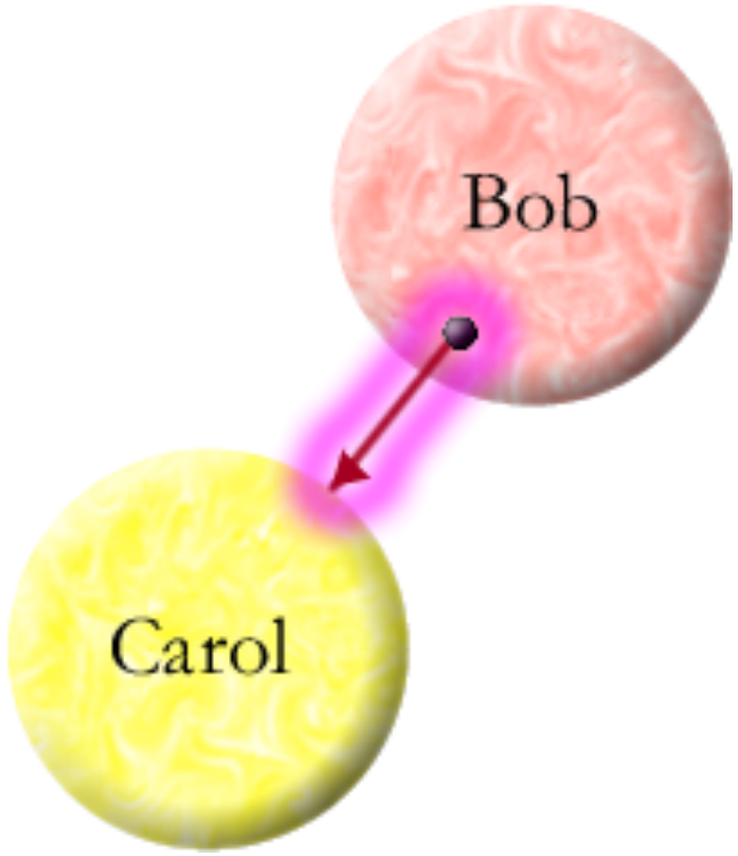
by Parenthood

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 ≡ Access graph

Reachability limits effects

Abstraction boundary ≡ Enforcement mechanism

Abstraction mechanisms for access control