# V8

Vyacheslav Egorov
vegorov@google.com, me@mrale.ph

V8 is a JavaScript VM
inside **Chrome** and **node.js**

# Being *a JavaScript VM* is not an achievement

Being a ***fast*** one

- IS

# What is the challenge?

```
// Adding integers.
function add(a, b) {
  return a + b;
}
```

```
// Adding doubles.
function add(a, b) {
  return a + b;
}
```

```
// Concatenating strings.
function add(a, b) {
  return a + b;
}
```

```
// Arrays are just objects with
// properties "0", "1", "2", ...
Array.prototype[1] = "ha!";
var arr = [0, /* hole */ , 2];
arr[1] // => "ha!"
```

```
// Making a "class".
function Dog(name, breed) {
  this.name  = name;
  this.breed = breed;
}

Dog.prototype.woof = function () {
  /* ... */
};
```

```
// Inheriting from a class.
function Dog(name, breed) {
  Animal.call(this, name);
  this.breed = breed;
}

Dog.prototype =
    Object.create(Animal.prototype);
```

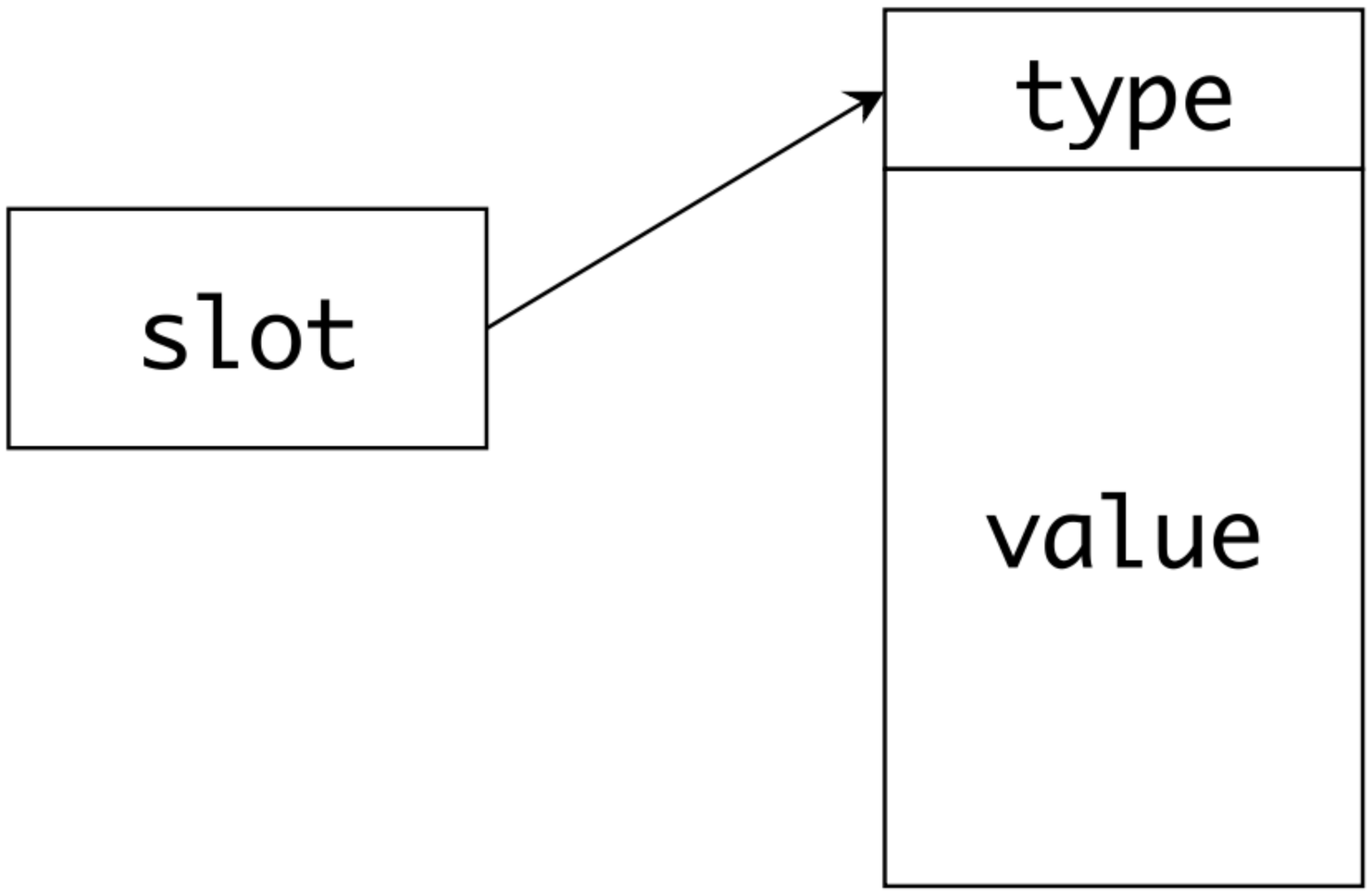
```
// Another way to create a dog.
var dog = {
   name: name,
   breed: breed,
   woof: function () { /* ... */ }
};
```

```
// Yet another way.
function makeADog(name, breed) {
  // name and breed are now "private"
  return {
    woof: function () { /* ... */ }
  }
}
```

```
// Yet another way.
var Dog = {
  woof: function () { /* ... */ }
};

function makeADog() {
  var dog = Object.create(Dog);
  dog.name = name;
  dog.breed = name;
  return dog;
}
```

- Representation
- Resolution
- Redundancy

# REPRESENTATION


slot
type
value

# Use *tagging* to avoid boxing everything

Pointer has last bit unused due to alignment

pointer: **xx...x1**

```
function tag(ptr) { // ptr & 1 === 0
  return ptr | 1;
}

function untag(val) {
  return val & ~1;
}

function isPtr(val) {
  return (val & 1) === 1;
}
```

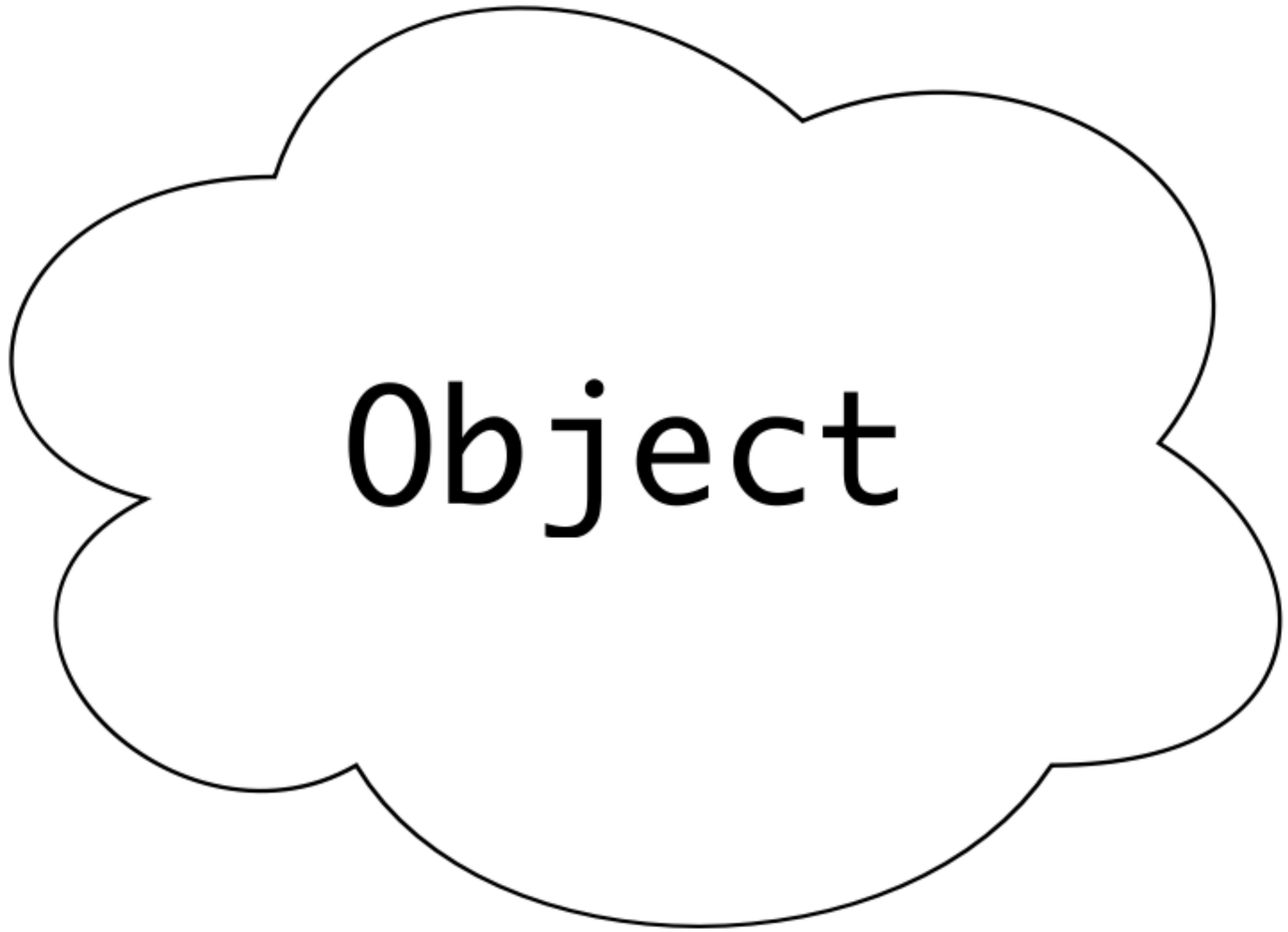
*small* integer: **xx...x0**

```
function tag(val) {
  return val << 1;
}

function untag(val) {
  return val >> 1;
}

function isSmi(val) {
  return (val & 1) === 0
}
```

Double arithmetic

⇒ boxing

# Objects?


Object

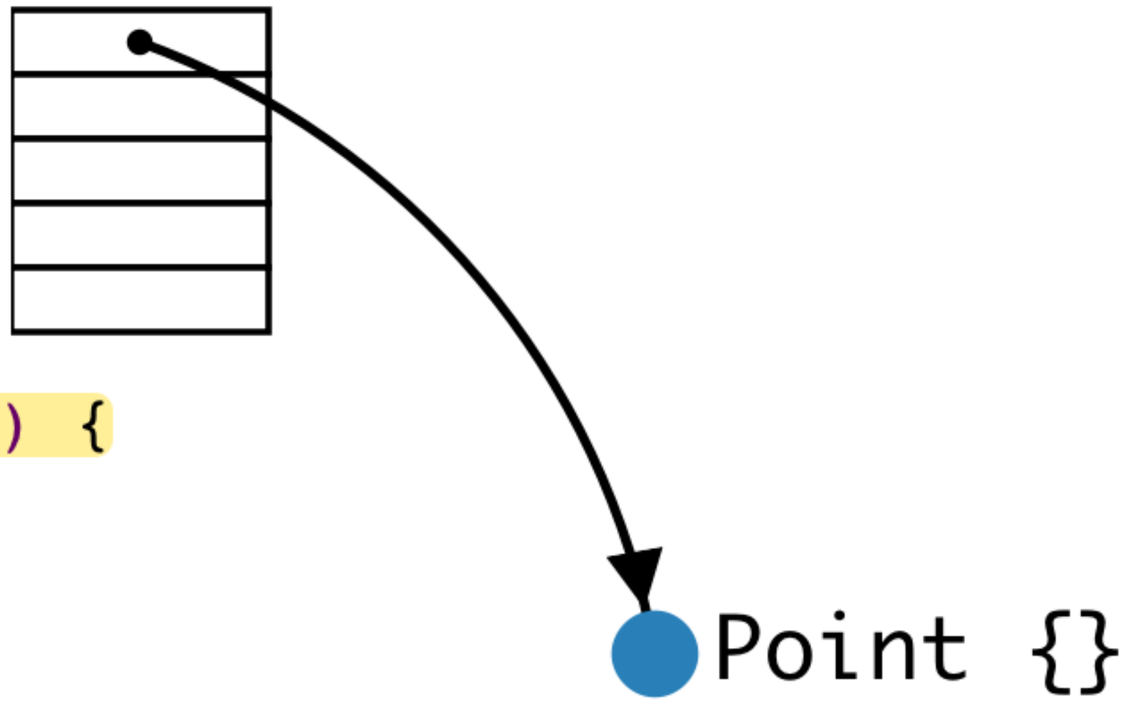
# Hidden classes ≈ *maps* from Self VM

```
function Point(x, y) {
  this.x = x;
  this.y = y;
}
```

```
var p1 = new Point(1, 2);
var p2 = new Point(3, 4);
```

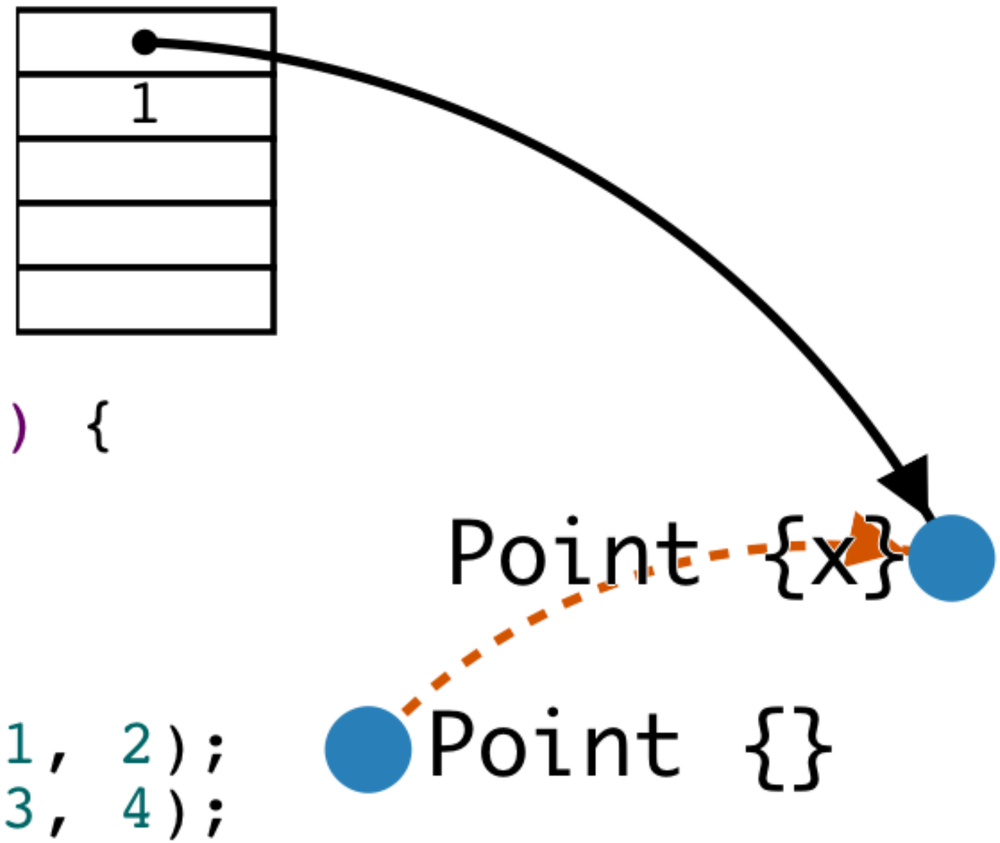
```
function Point(x, y) {
  this.x = x;
  this.y = y;
}
```

```
var p1 = new Point(1, 2);
var p2 = new Point(3, 4);
```



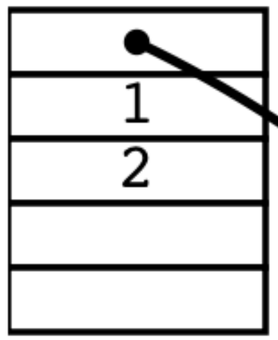

```
function Point(x, y) {
  this.x = x;
  this.y = y;
}
```

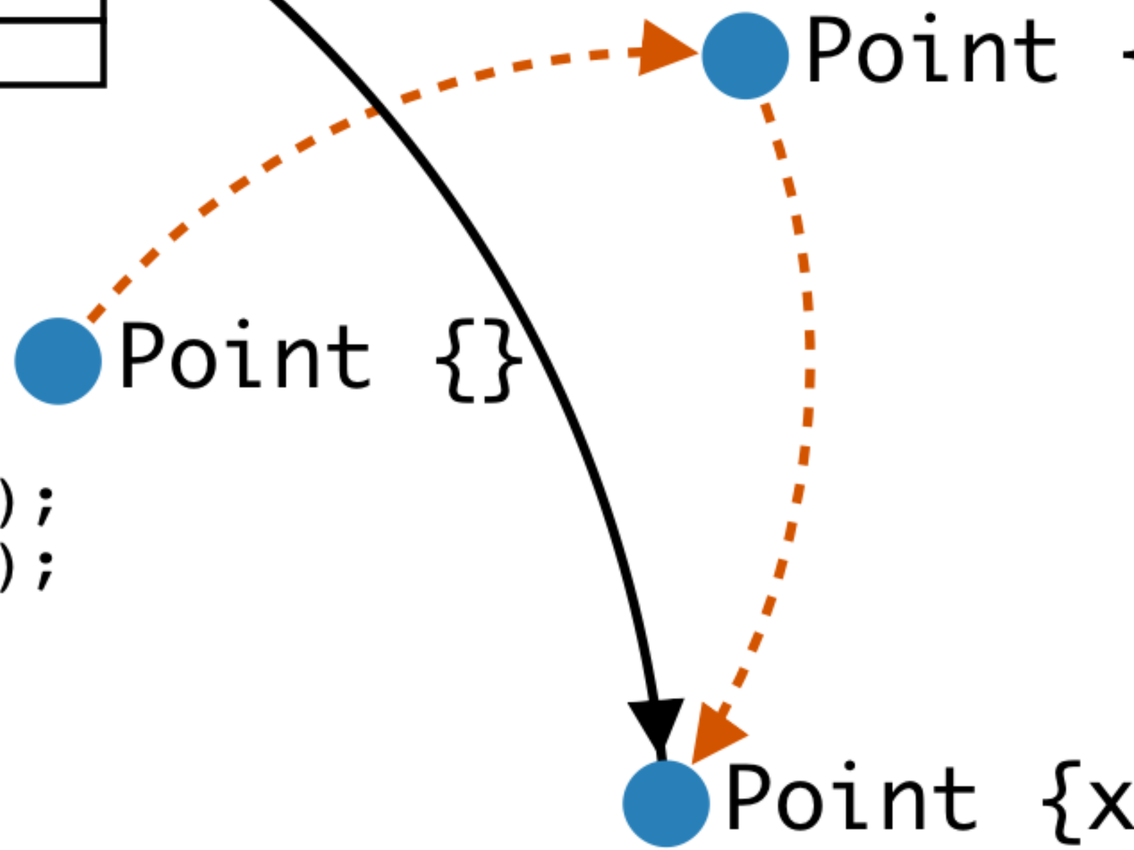
```
var p1 = new Point(1, 2);
var p2 = new Point(3, 4);
```

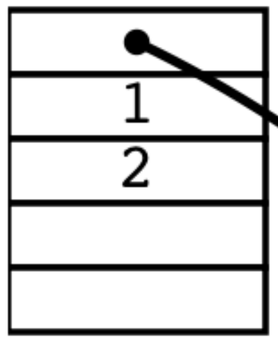


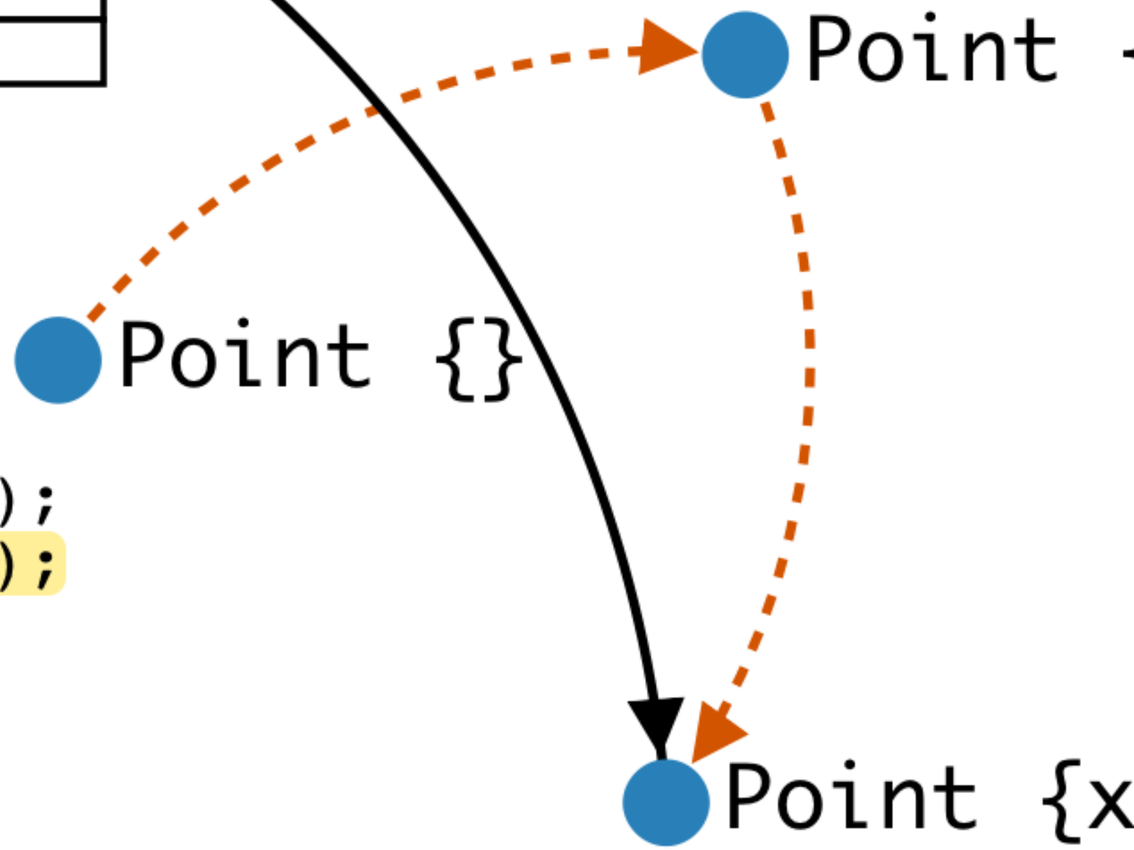

```
function Point(x, y) {
  this.x = x;
  this.y = y;
}

var p1 = new Point(1, 2);
var p2 = new Point(3, 4);
```

```
function Point(x, y) {
  this.x = x;
  this.y = y;
}

var p1 = new Point(1, 2);
var p2 = new Point(3, 4);
```



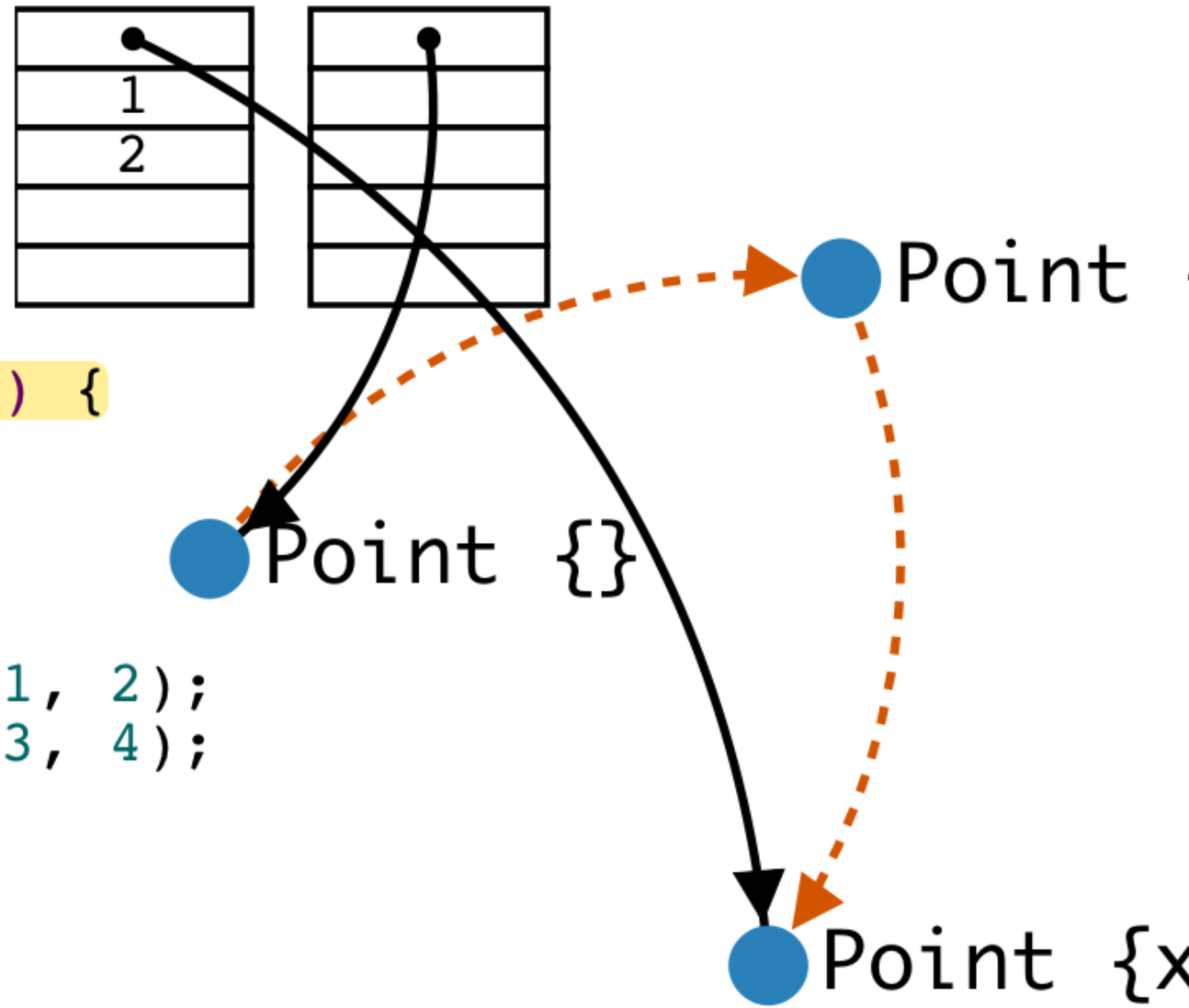





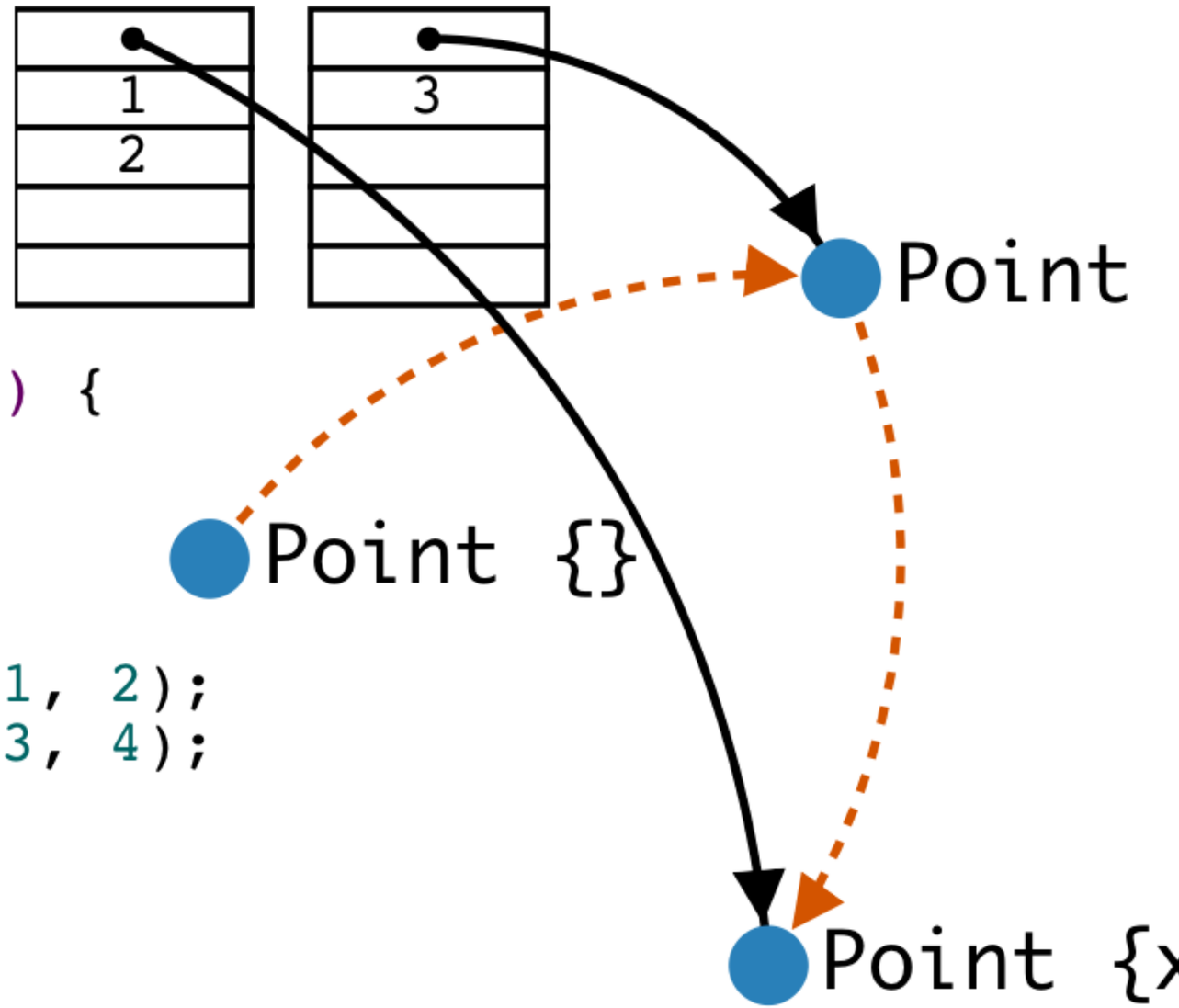




```
function Point(x, y) {
  this.x = x;
  this.y = y;
}

var p1 = new Point(1, 2);
var p2 = new Point(3, 4);
```

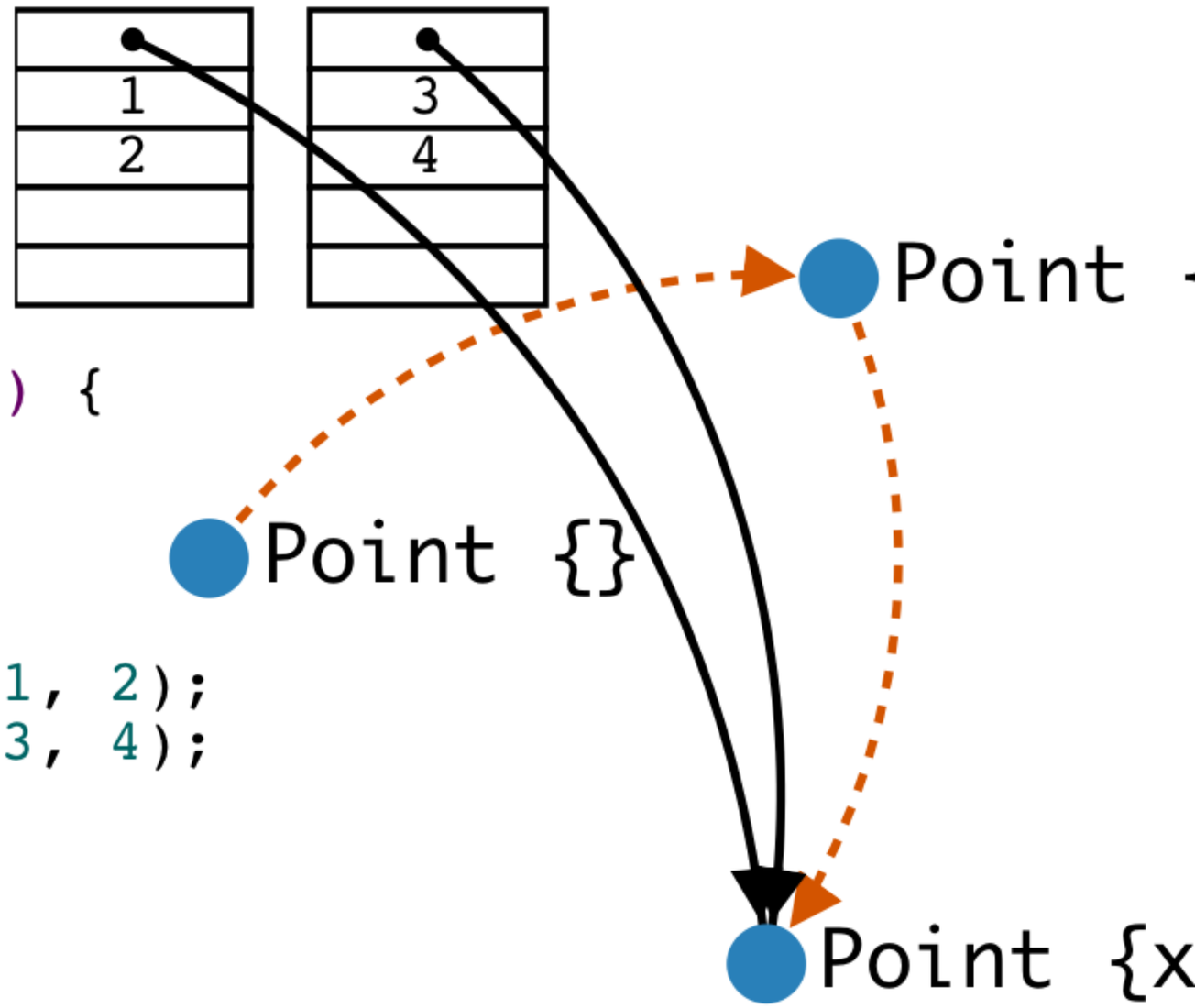
```
function Point(x, y) {
  this.x = x;
  this.y = y;
}

var p1 = new Point(1, 2);
var p2 = new Point(3, 4);
```



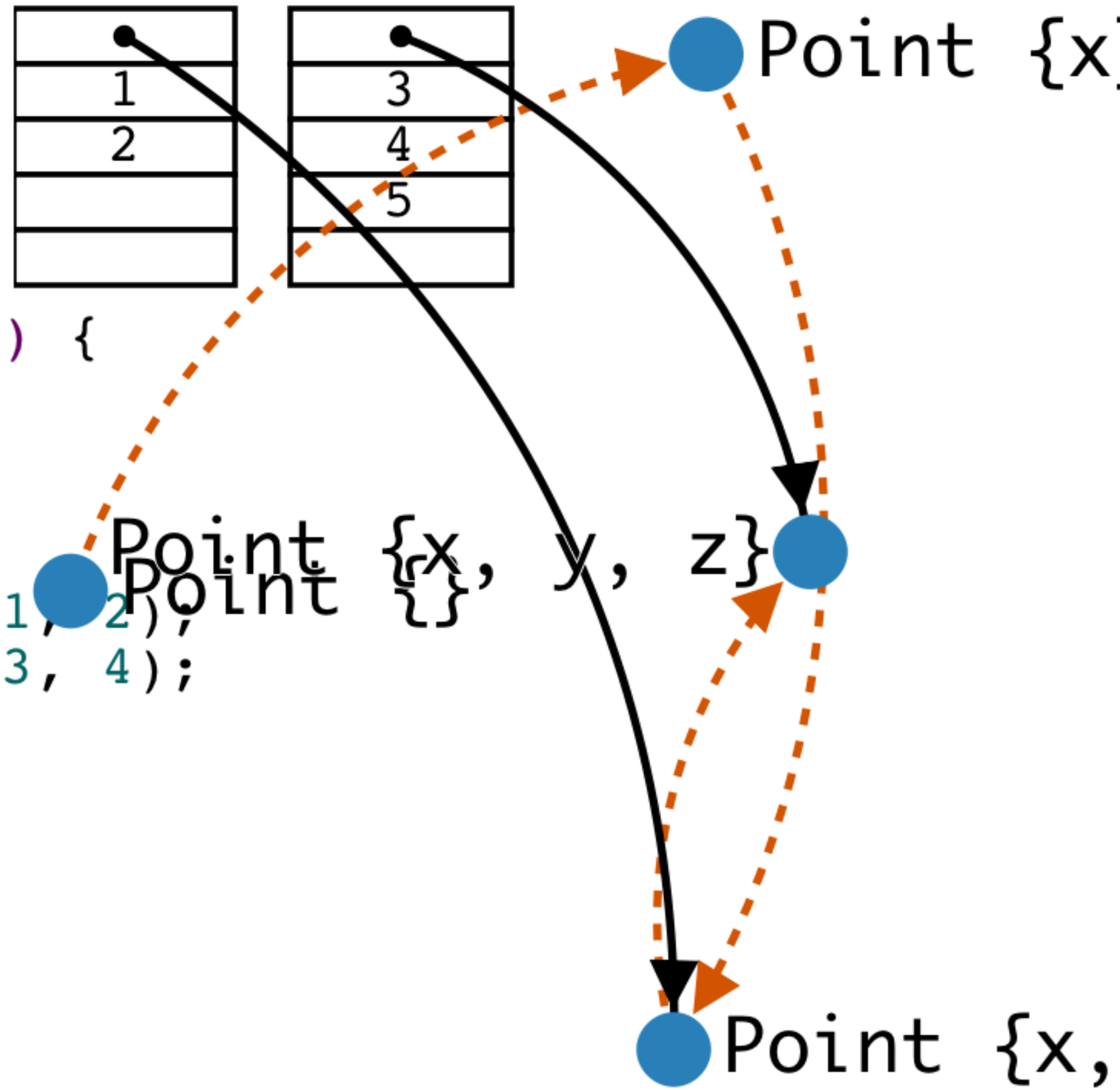
```
function Point(x, y) {
  this.x = x;
  this.y = y;
}

var p1 = new Point(1, 2);
var p2 = new Point(3, 4);
```



```
function Point(x, y) {
  this.x = x;
  this.y = y;
}

var p1 = new Point(1, 2);
var p2 = new Point(3, 4);
```

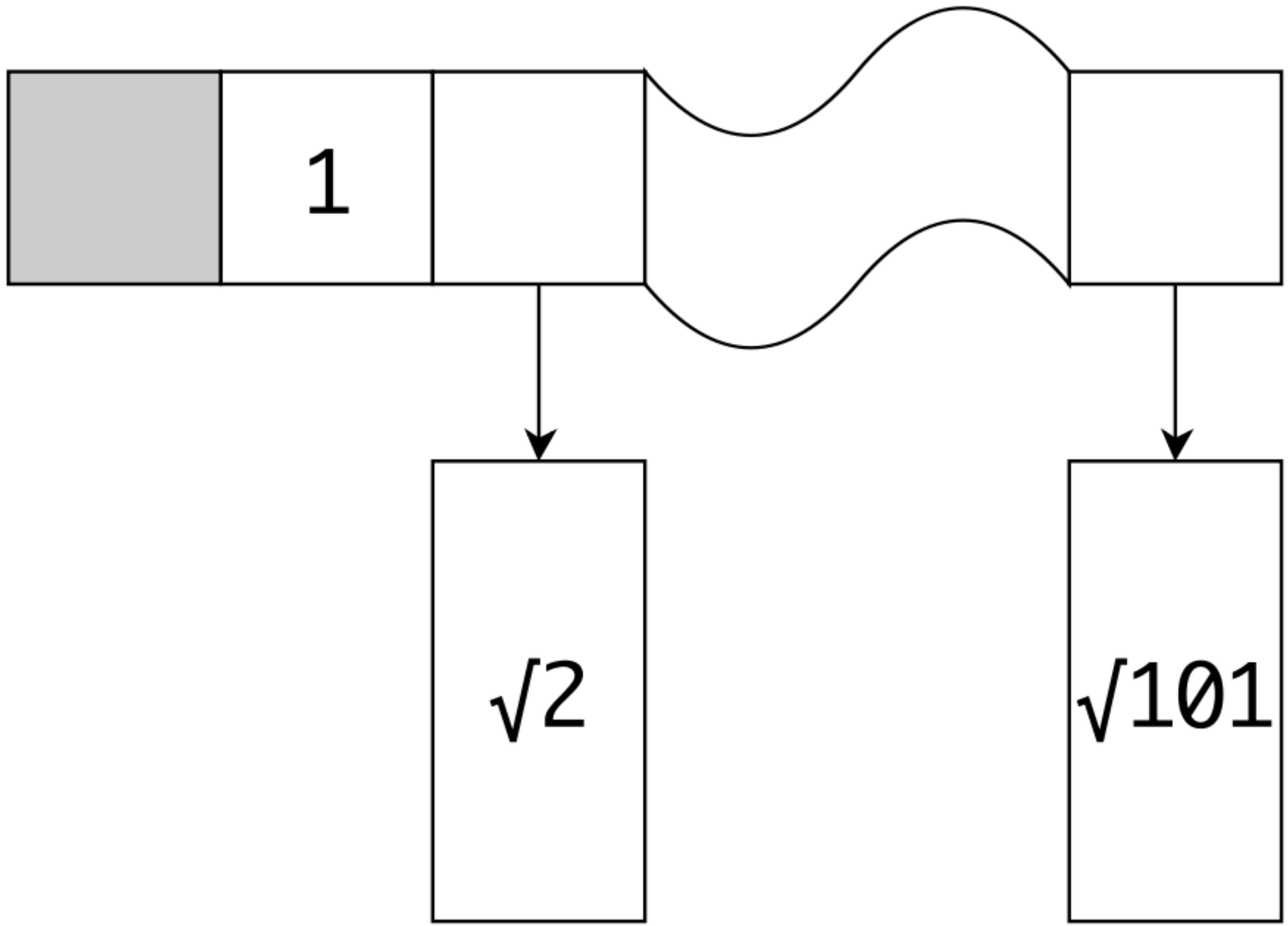


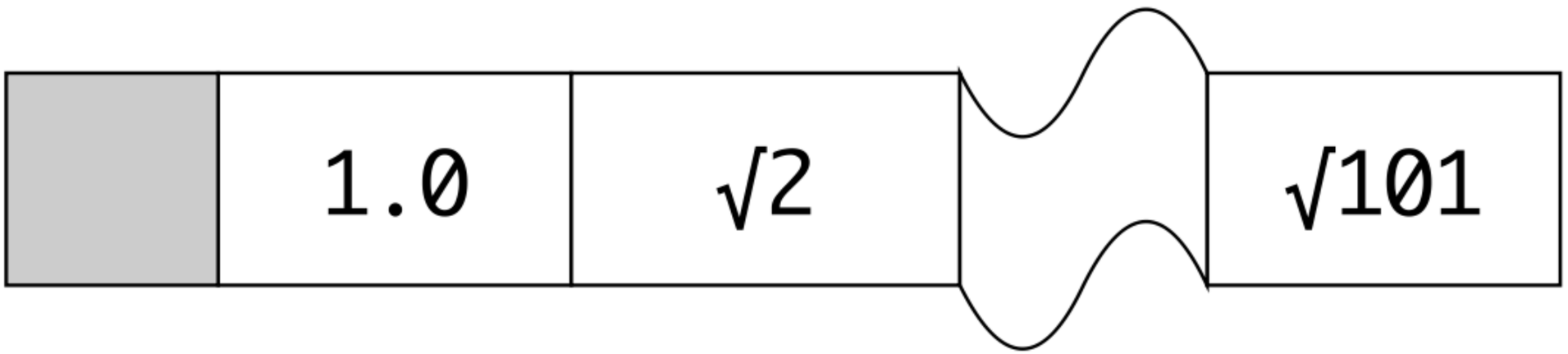
```
function Point(x, y) {
  this.x = x;
  this.y = y;
}

var p1 = new Point(1, 2);
var p2 = new Point(3, 4);
p2.z = 5;
```



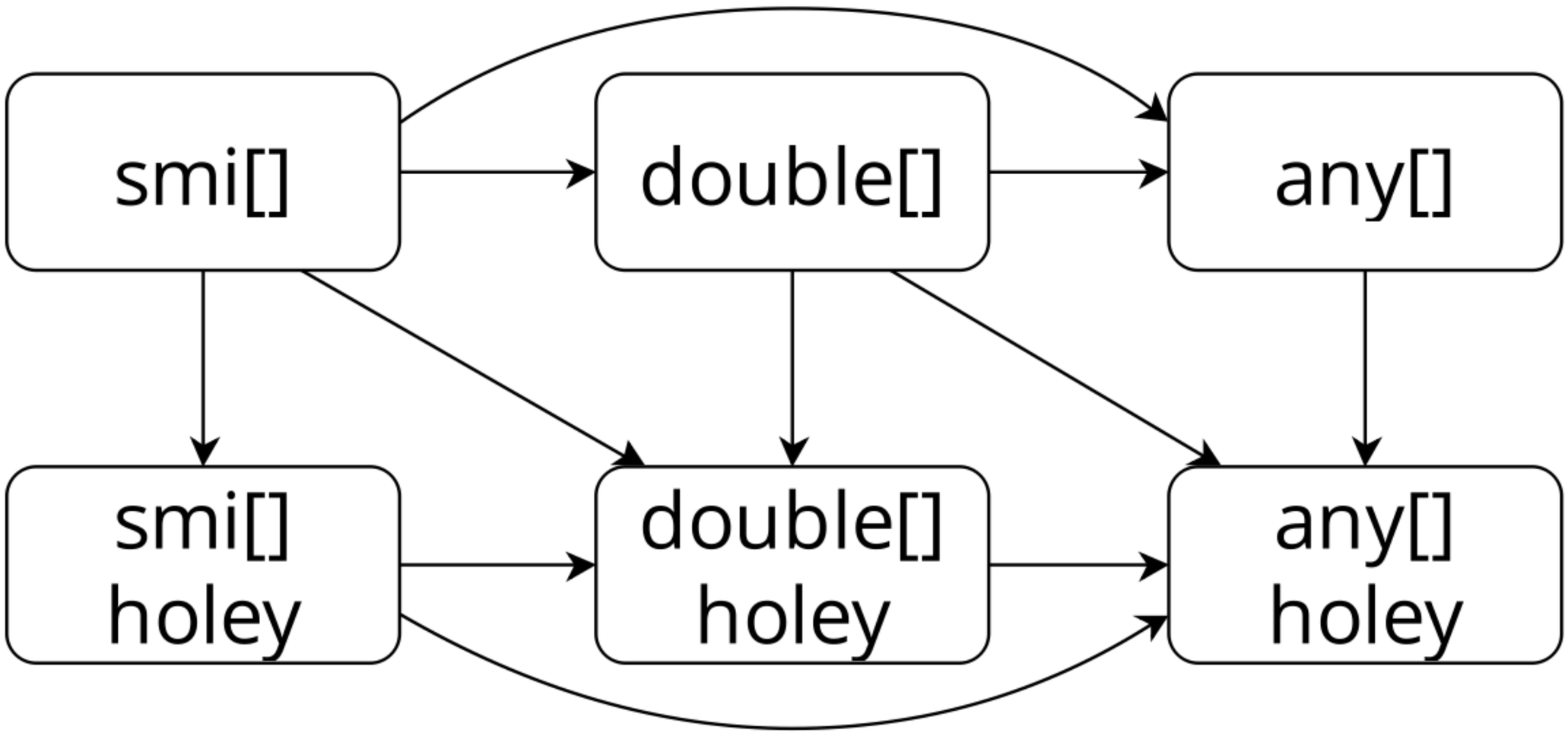
Approximates
*static* structure
*dynamic*ally.

```
var arr = [];
for (var i = 0; i < 101; i++)
arr[i] = Math.sqrt(i);
```


1
√2
√101


1.0
√2
√101

# Track *denseness* and (un)box!


smi[]
double[]
any[]
smi[]
holey
double[]
holey
any[]
holey

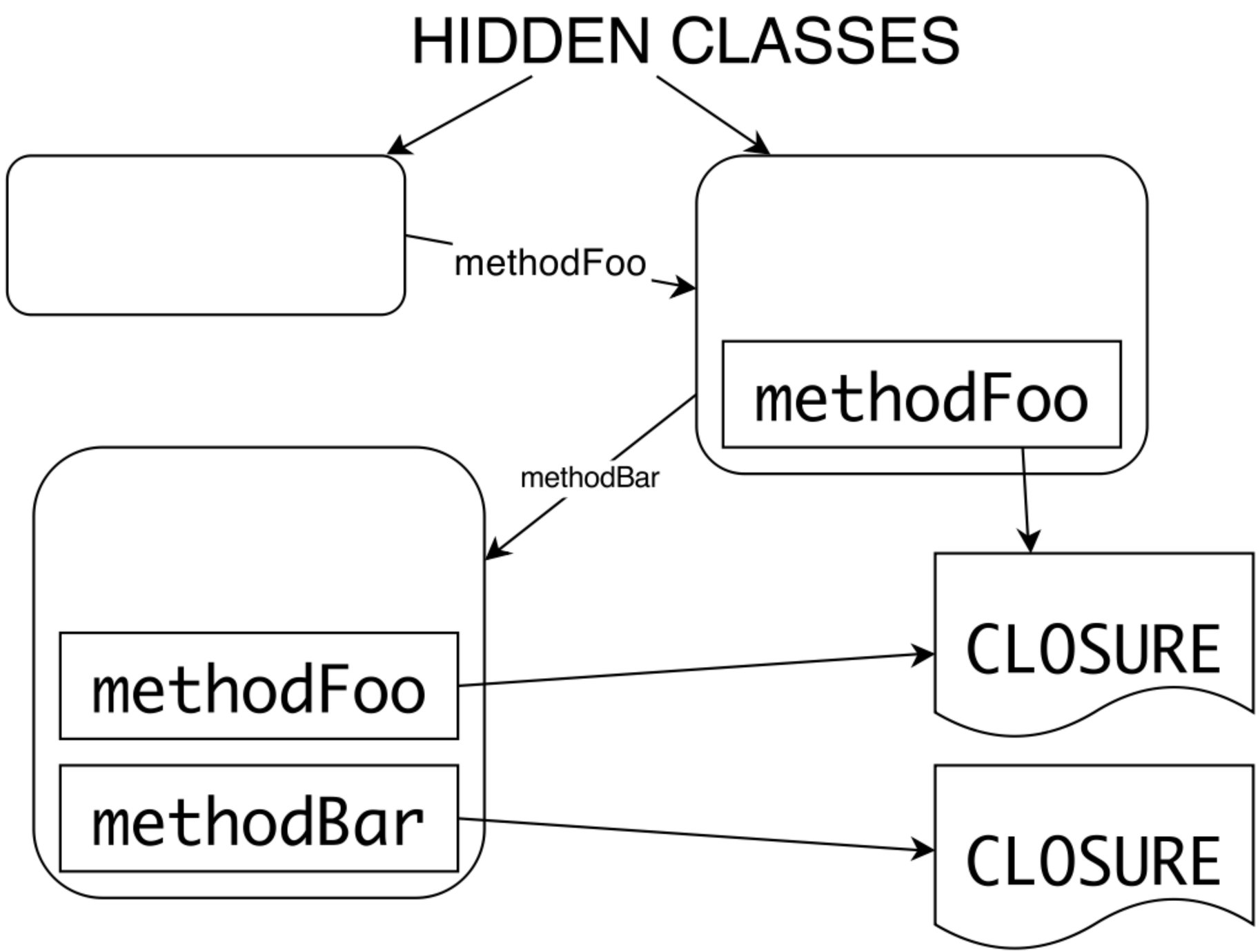
# Same for properties

```
// Want "real" fast method calls
C.prototype.methodFoo = function () {
  /* ... */
};
C.prototype.methodBar = function () {
  /* ... */
};

obj.methodFoo();
```

```
// Don't want (pseudo-code)
m = LoadProperty(obj, "methodFoo")
CheckIfFunction(m)
Invoke(m, obj)
```

```
// Want (pseudo-code)
CheckClass(obj, klass0);
Invoke(methodFoo, obj);
```

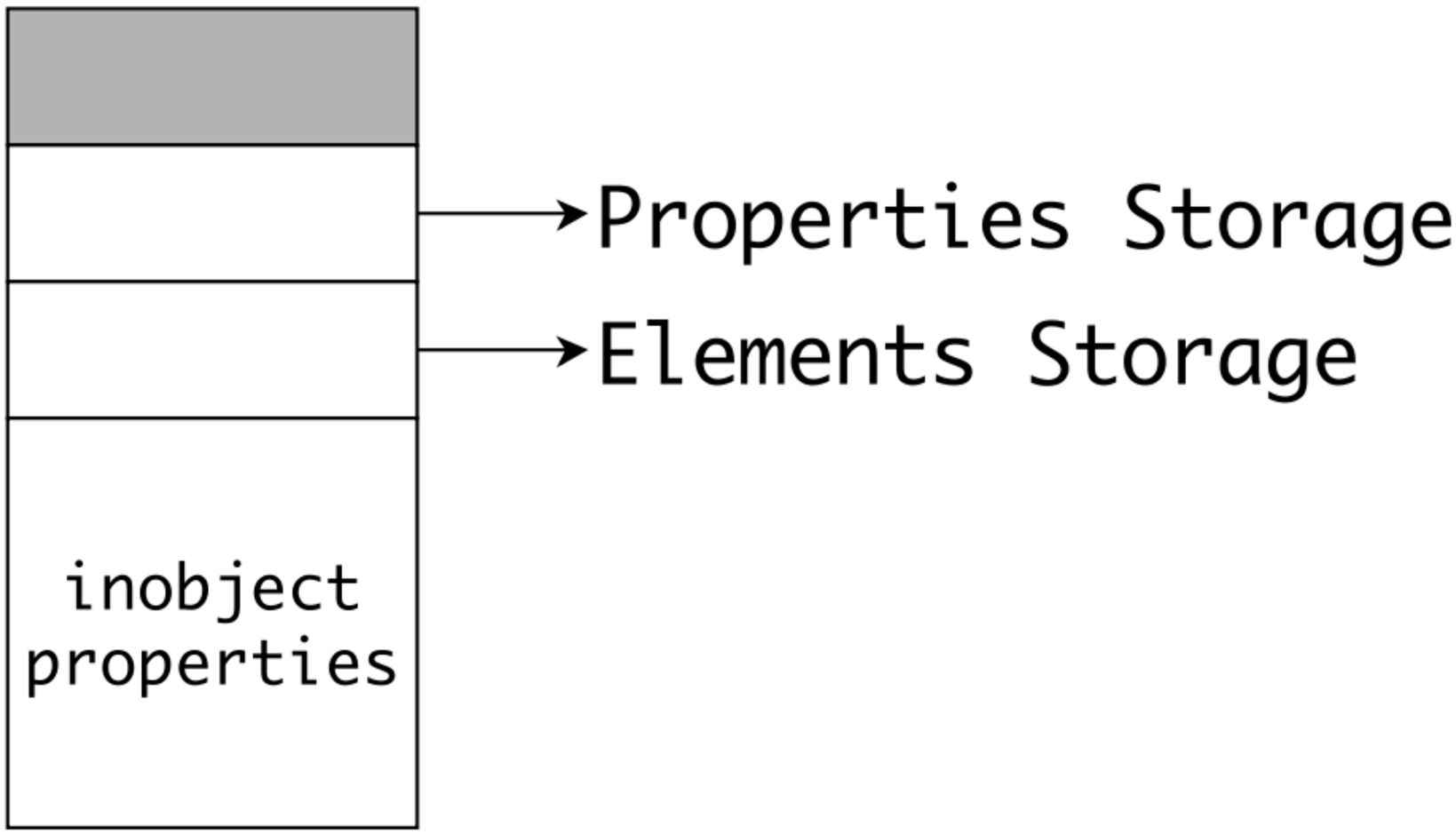
# *Promote* functions to hidden class


HIDDEN CLASSES
methodFoo
methodFoo
methodBar
methodFoo
methodBar
CLOSURE
CLOSURE

# Catch: The *whole* closure is promoted!

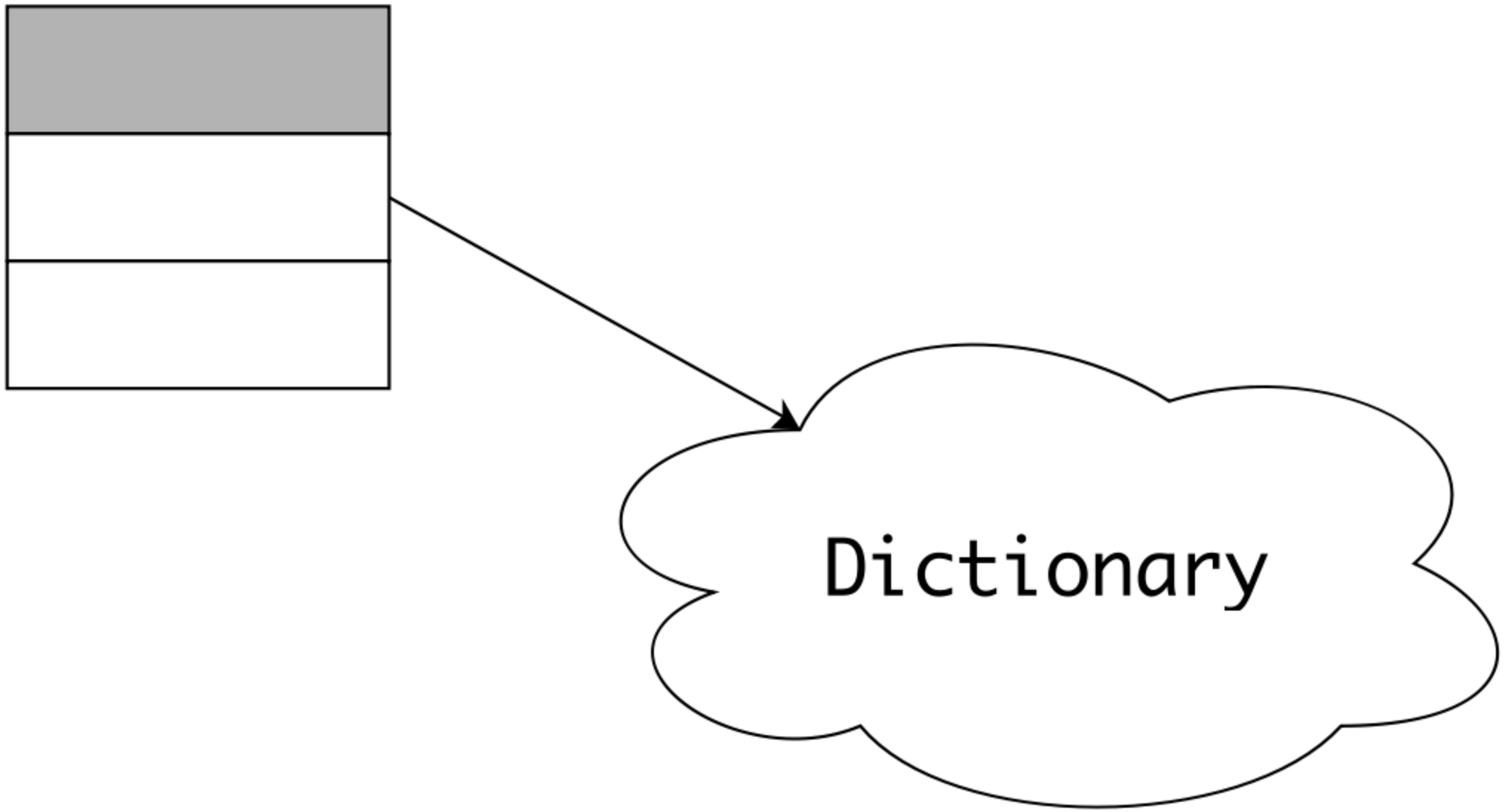
```
// What is "perf-unfriendly" here?
function buyDog() {
  return {
    woof: function () {
      /* ... */
    }
  }
}
```

different `woof` ⇒
different classes


Properties Storage
Elements Storage
inobject
properties

# objects are not always structures

- too many properties
- `delete obj.prop`


Dictionary

# RESOLUTION

```
function Load(receiver, property) {
 var O = ToObject(receiver);
 var P = ToString(property);
 var desc = O.[[GetProperty]](P);
 if (desc === $undefined) return $undefined;
 if (IsDataDescriptor(desc)) return desc.Value;
 assert(IsAccessorDescriptor(desc));
 var getter = desc.Get;
 if (getter === $undefined) return $undefined;
 return getter.[[Call]](receiver);
}
```

```
JSObject.prototype.[[GetProperty]] = function (P) {
 var prop = this.[[GetOwnProperty]](P);
 if (prop !== $undefined) return prop;
 var proto = this.[[Proto]];
 if (proto === $null) return $undefined;
 return proto.[[GetPropery]](P);
};

JSObject.prototype.[[GetOwnProperty]] = function (P) {
 return this.properties.get(P);
};
```

```
function Load(receiver, property) {
 var O = ToObject(receiver);
 var P = ToString(property);
 var desc = O.[[GetProperty]](P);
 if (desc === $undefined) return $undefined;
 if (IsDataDescriptor(desc)) return desc.Value;
 assert(IsAccessorDescriptor(desc));
 var getter = desc.Get;
 if (getter === $undefined) return $undefined;
 return getter.[[Call]](receiver);
}
```

```
JSObject.prototype.[[GetProperty]] = function (P) {
 var prop = this.[[GetOwnProperty]](P);
 if (prop !== $undefined) return prop;
 var proto = this.[[Proto]];
 if (proto === $null) return $undefined;
 return proto.[[GetPropery]](P);
};

JSObject.prototype.[[GetOwnProperty]] = function (P) {
 return this.properties.get(P);
};
```

```
function Load(receiver, property) {
 var O = ToObject(receiver);
 var P = ToString(property);
 var desc = O.[[GetProperty]](P);
 if (desc === $undefined) return $undefined;
 if (IsDataDescriptor(desc)) return desc.Value;
 assert(IsAccessorDescriptor(desc));
 var getter = desc.Get;
 if (getter === $undefined) return $undefined;
 return getter.[[Call]](receiver);
}
```

```
JSObject.prototype.[[GetProperty]] = function (P) {
 var prop = this.[[GetOwnProperty]](P);
 if (prop !== $undefined) return prop;
 var proto = this.[[Proto]];
 if (proto === $null) return $undefined;
 return proto.[[GetPropery]](P);
};

JSObject.prototype.[[GetOwnProperty]] = function (P) {

return this.properties.get(P);
};
```

tons of code
+
dictionary lookup

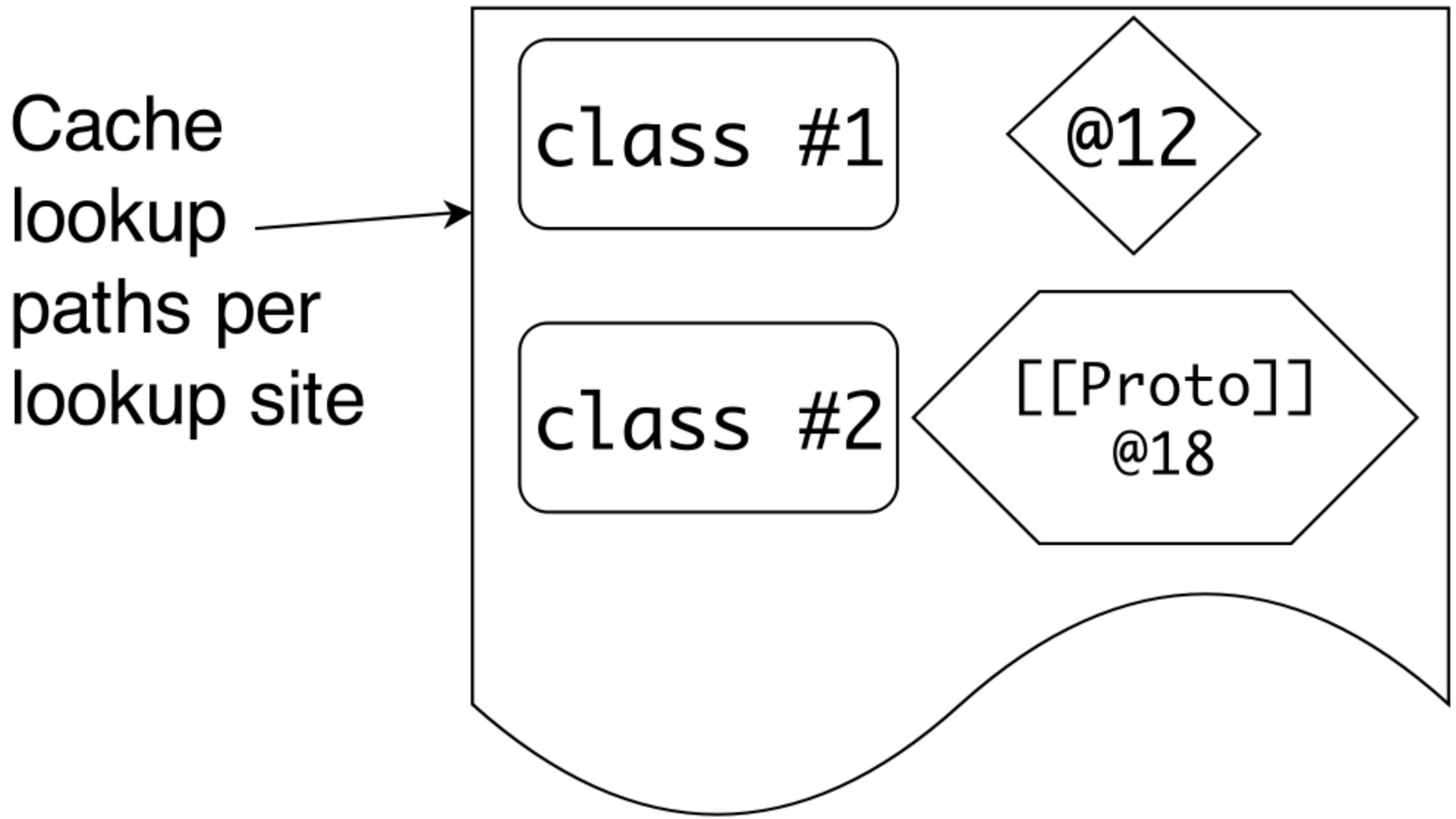
can we do better?

we have to do *first* lookup

do we have to do *second*?

same hidden class ⇒
same structure

# obj.foo


Cache lookup paths per lookup site
class #1
@12
class #2
[[Proto]]
@18

# Inline Caching

```
;; Compiled code
mov eax, obj
mov ecx, "foo"
call LoadIC_Initialize
```

```
// Runtime system.
function LoadIC_Initialize(obj, prop) {
  var lookupResult = obj.lookup(prop);
  patch(lookupResult.compile());
  return lookupResult.value;
}
```

```
// Runtime system.
function LoadIC_Initialize(obj, prop) {
  var lookupResult = obj.lookup(prop);
  patch(lookupResult.compile());
  return lookupResult.value;
}
```

```
;; Compiled LoadIC Stub
0xabcdef:
cmp [eax - 1], klass0
jnz LoadIC_Miss
mov eax, [eax + 11]
ret
```

```
;; Compiled code
mov eax, obj
mov ecx, "foo"
call 0xabcdef ;; patched!
```

```
;; Compiled LoadIC Stub
0xabcdef:
cmp [eax - 1], klass0
jnz LoadIC_Miss
mov eax, [eax + 11]
ret
```

```
;; Compiled code
mov eax, obj
mov ecx, "foo"
call 0xabcdef ;; patched!
```

```
;; Compiled LoadIC Stub
0xabcdef:
cmp [eax - 1], klass0
jnz LoadIC_Miss
mov eax, [eax + 11]
ret
```

```
;; Compiled code
mov eax, obj
mov ecx, "foo"
call 0xabcdef ;; patched!
```

```
;; Compiled LoadIC Stub
0xabcdef:
cmp [eax - 1], klass0
jnz LoadIC_Miss
mov eax, [eax + 11]
ret
```

```
;; Compiled code
mov eax, obj
mov ecx, "foo"
call 0xabcdef ;; patched!
```

```
;; Compiled LoadIC Stub
0xabcdef:
cmp [eax - 1], klass0
jnz LoadIC_Miss
mov eax, [eax + 11]
ret
```

```
;; Compiled code
mov eax, obj
mov ecx, "foo"
call 0xabcdef ;; patched!
```

# Everything is an IC stub

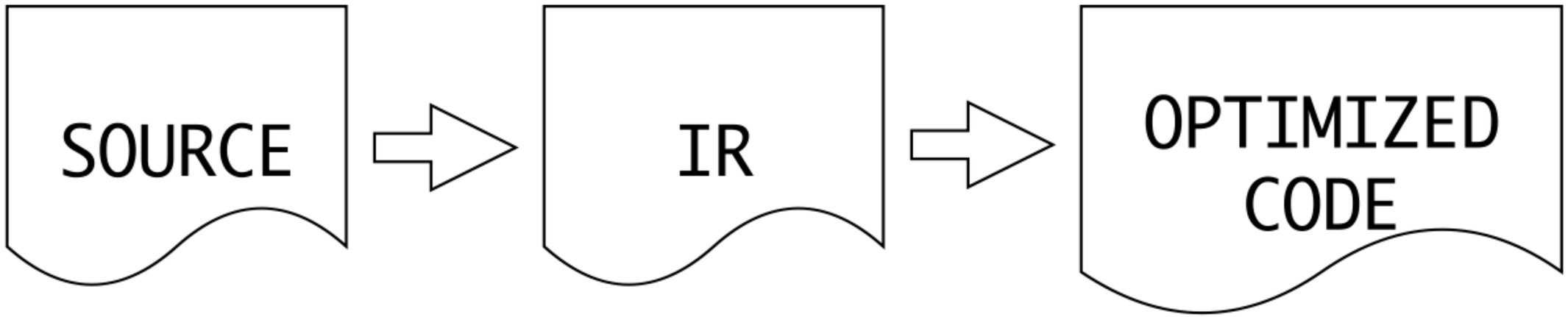
- property accesses
- element accesses
- method calls
- special method calls
- global variables lookup
- arithmetic operations

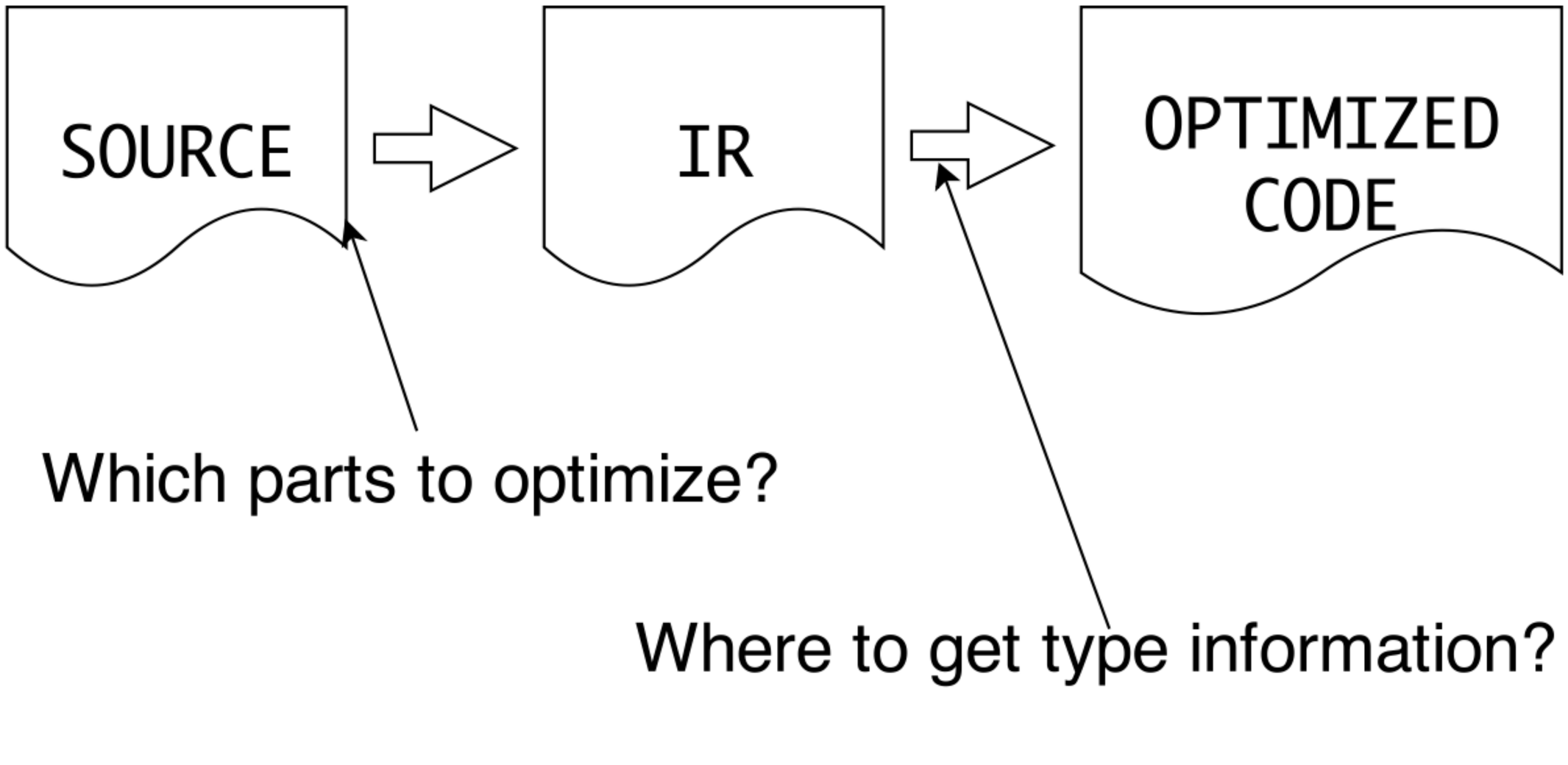
```
Vector.prototype.length = function () {
  return Math.sqrt(this.x * this.x +
                   this.y * this.y);
};
```

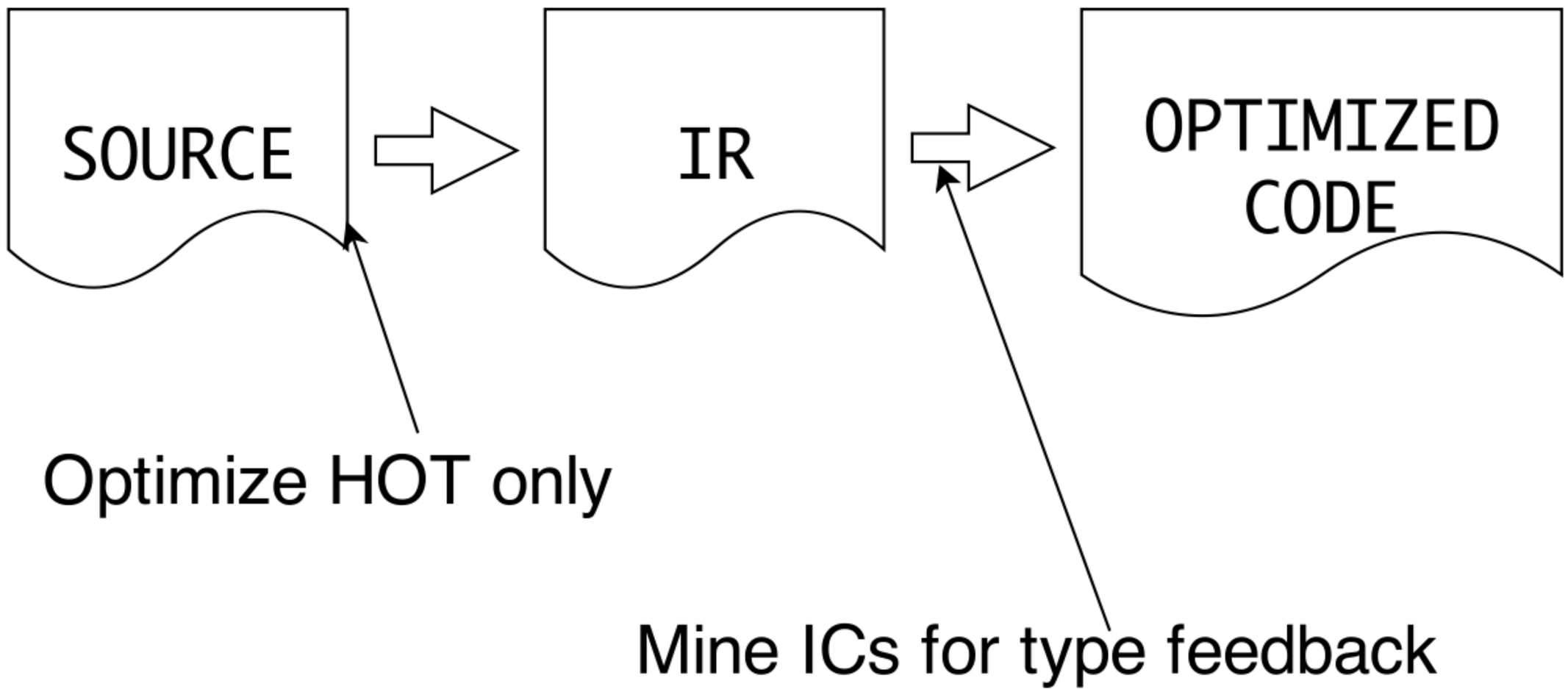
```
Vector.prototype.length = function () {
  return Math.sqrt(this.x * this.x +
                   this.y * this.y);
};
```

# REDUNDANCY

# Need an optimizing compiler


SOURCE
IR
OPTIMIZED
CODE


SOURCE
IR
OPTIMIZED CODE
Which parts to optimize?
Where to get type information?


SOURCE
IR
OPTIMIZED CODE
Optimize HOT only
Mine ICs for type feedback

# Crankshaft

1. Compile unoptimized code
2. Feed hot functions into optimizer
3. Speculate types based on IC states
4. Apply classic optimizations
5. Emit optimized code

# Crankshaft

1. Compile unoptimized code
2. Feed hot functions into optimizer
3. **Speculate** types based on IC states
4. Apply classic optimizations
5. Emit optimized code

# Checks inserted into code verify speculations

Failed check ⇒ jump to unoptimized code

```
Vector.prototype.length = function () {
  return Math.sqrt(this.x * this.x +
                   this.y * this.y);
};
```

```
CheckMap v0, klass
v1 = Load v0, @12
CheckMap v0, klass
v2 = Load v0, @12
d3 = TaggedToDouble v1
d4 = TaggedToDouble v2
d5 = Mul d3, d4
CheckMap v0, klass
v6 = Load v0, @16
CheckMap v0, klass
v7 = Load v0, @16
```

```
CheckMap v0, klass
v1 = Load v0, @12
d3 = TaggedToDouble v1
d5 = Mul d3, d3
v6 = Load v0, @16
d8 = TaggedToDouble v6
d10 = Mul d8, d8
d11 = Add d5, d10
```

- inlining
- GVN
- LICM
- DCE
- representation selection
- uint32 optimization
- escape analysis
- type inference
- range inference
- bounds check elimination

# There are tools to peak into optimizer

`--trace-hydrogen --trace-deopt --print-opt-code --code-comments`

no UI :-(