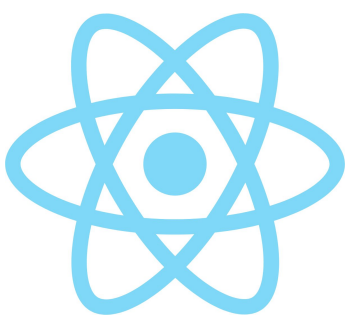
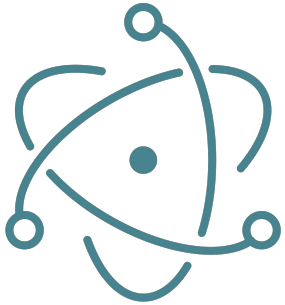
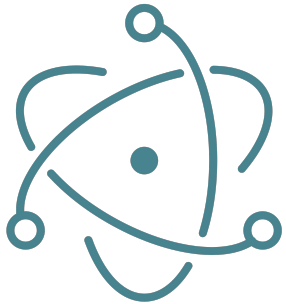
JavaScript engine fundamentals: the **good**, the **bad**, and the **ugly**

[@mathias, @bmeurer].join(@v8js)





@bmeurer && @mathias



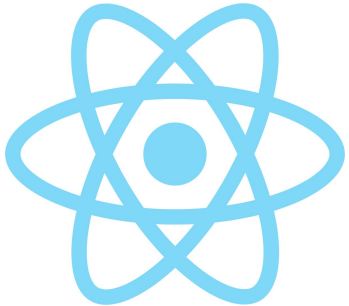
@bmeurer && @mathias

SpiderMonkey

@bmeurer && @mathias

Chakra

JavaScriptCore



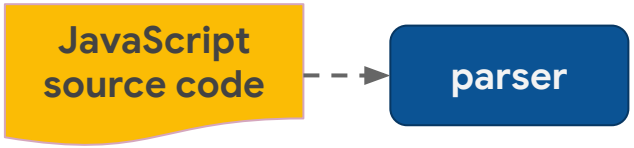
@bmeurer && @mathias

```
$ npm i -g jsvu
$ jsvu # Install/update JS engine binaries
```

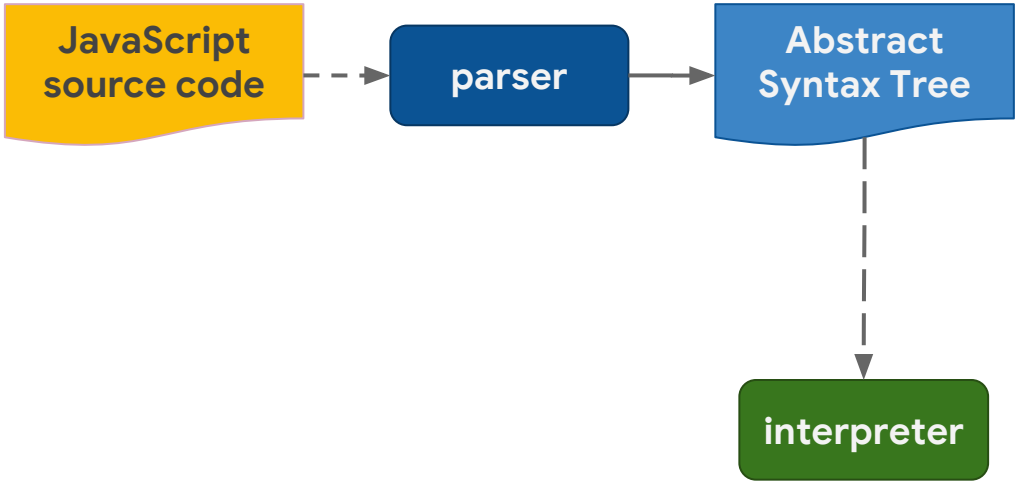
```
$ npm i -g jsvu
$ jsvu # Install/update JS engine binaries
$ v8
$ spidermonkey
$ chakra
$ jsc
```

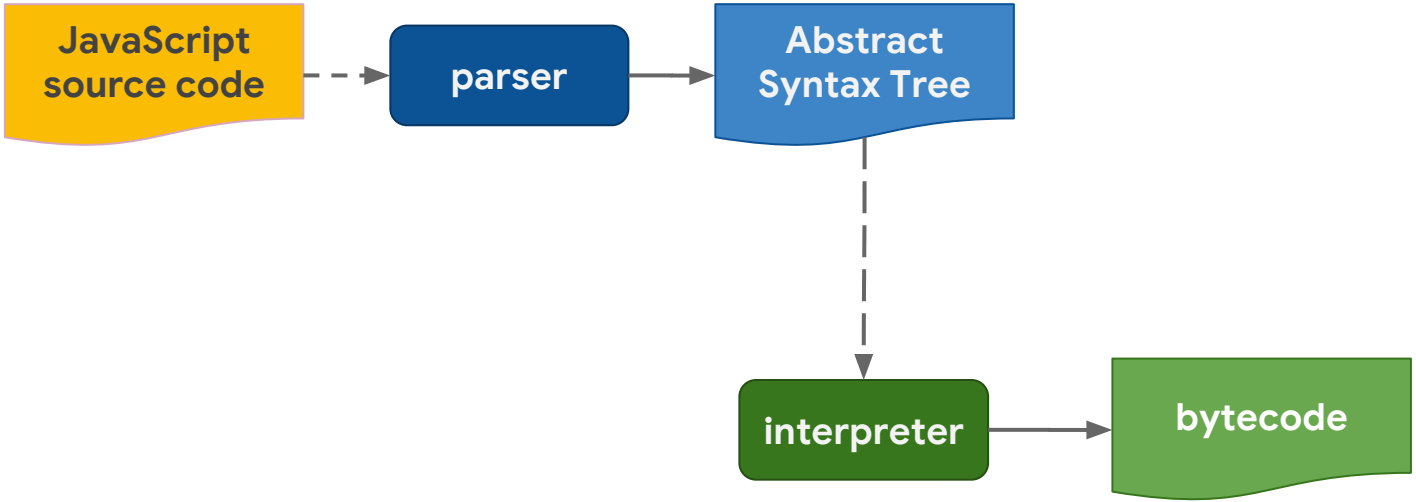
@bmeurer && @mathias

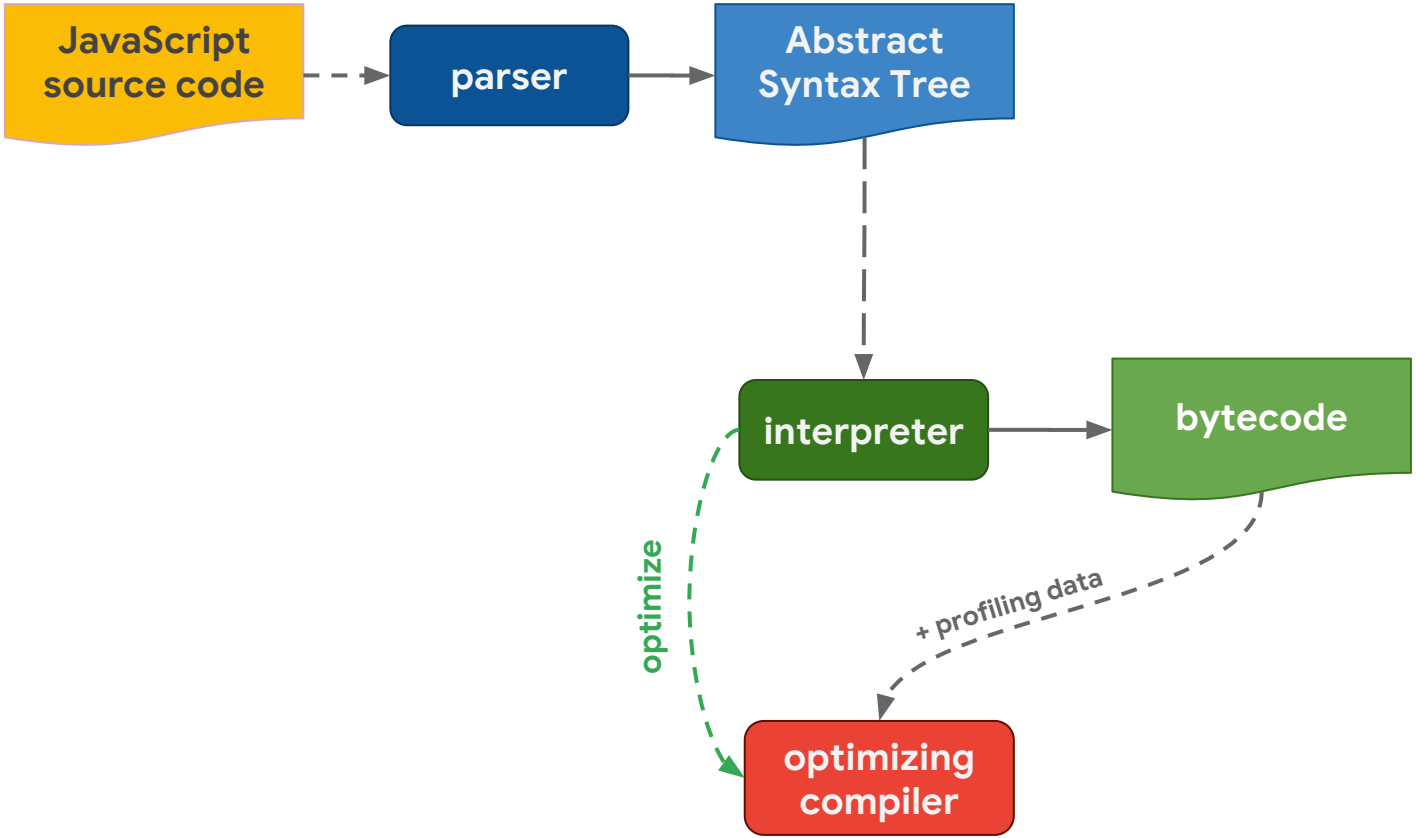
JavaScript source code


JavaScript
source code
parser

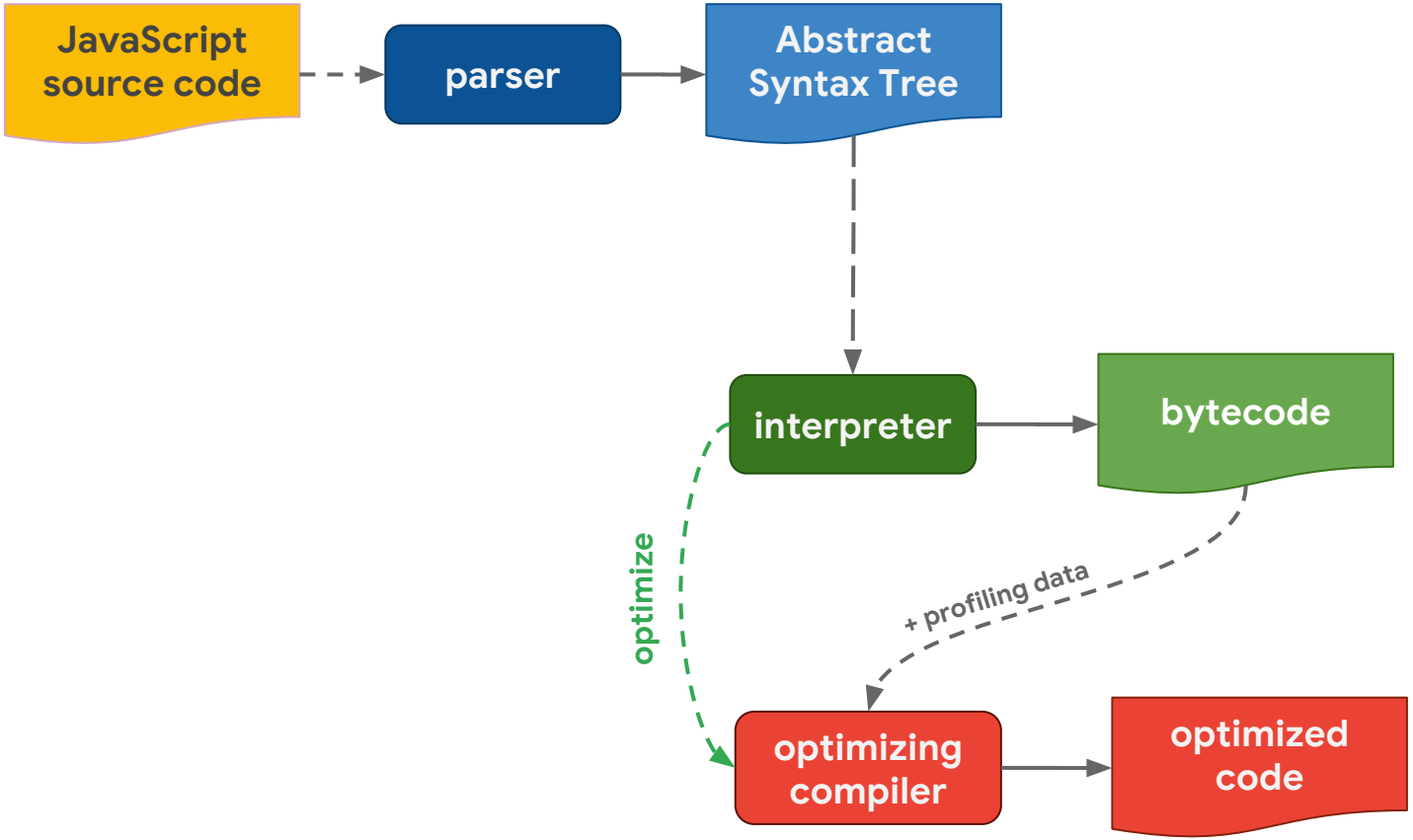

JavaScript
source code
parser
Abstract
Syntax Tree


JavaScript source code
parser
Abstract Syntax Tree
interpreter

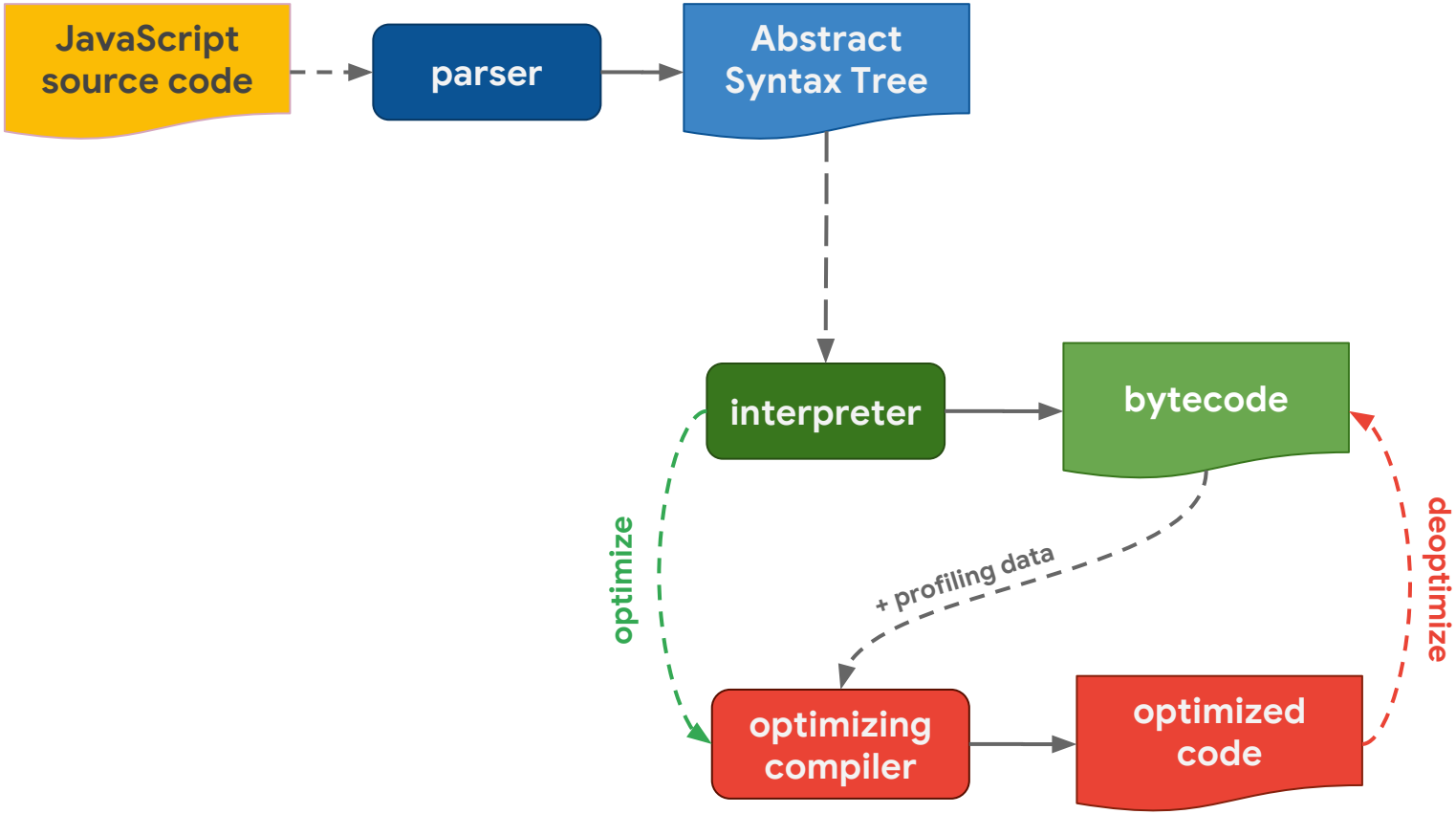

JavaScript source code
parser
Abstract Syntax Tree
interpreter
bytecode



@bmeurer && @mathias

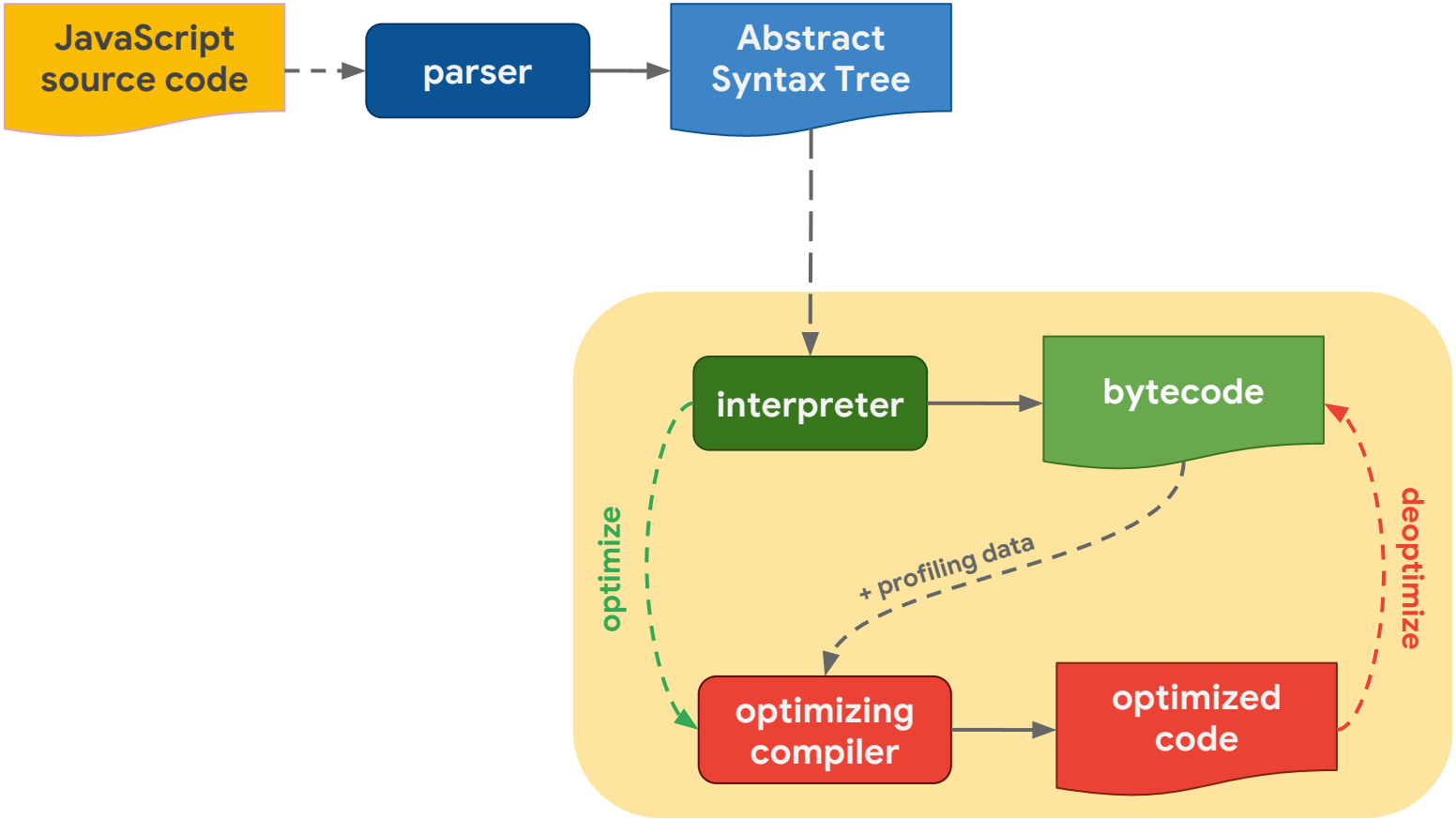


@bmeurer && @mathias

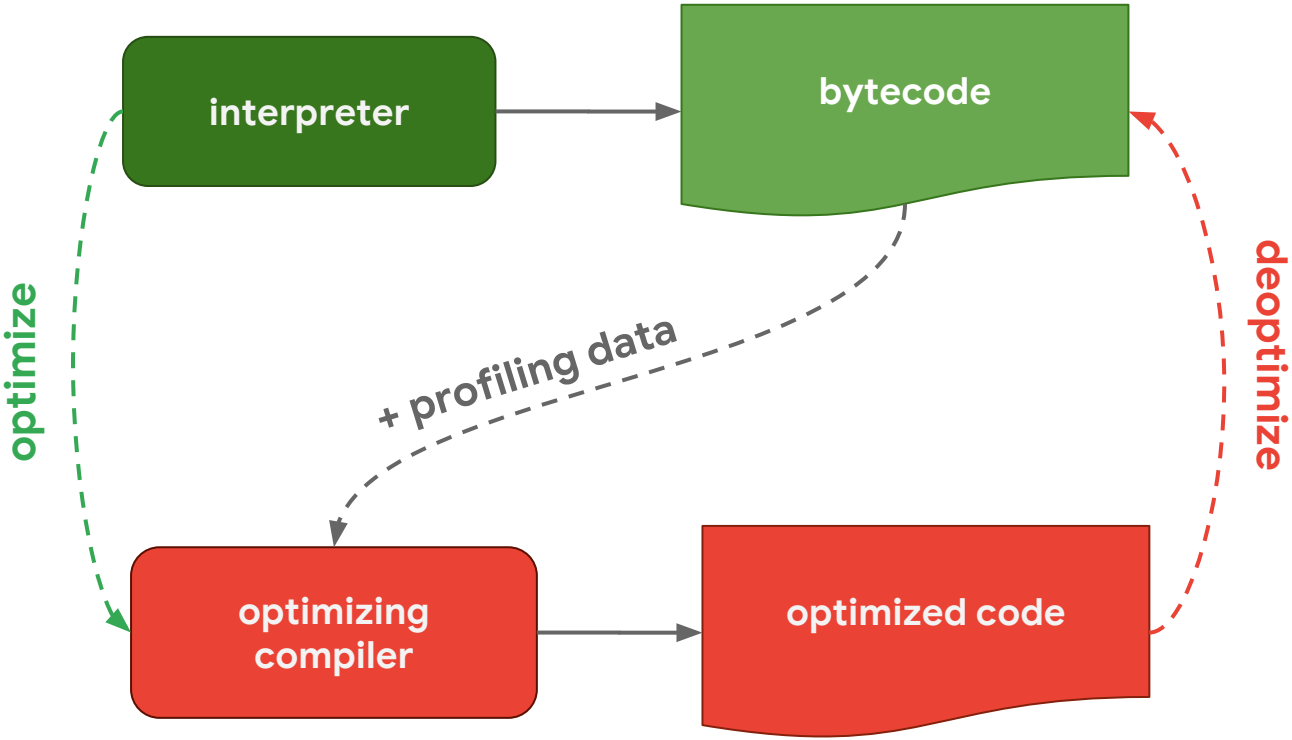


@bmeurer && @mathias

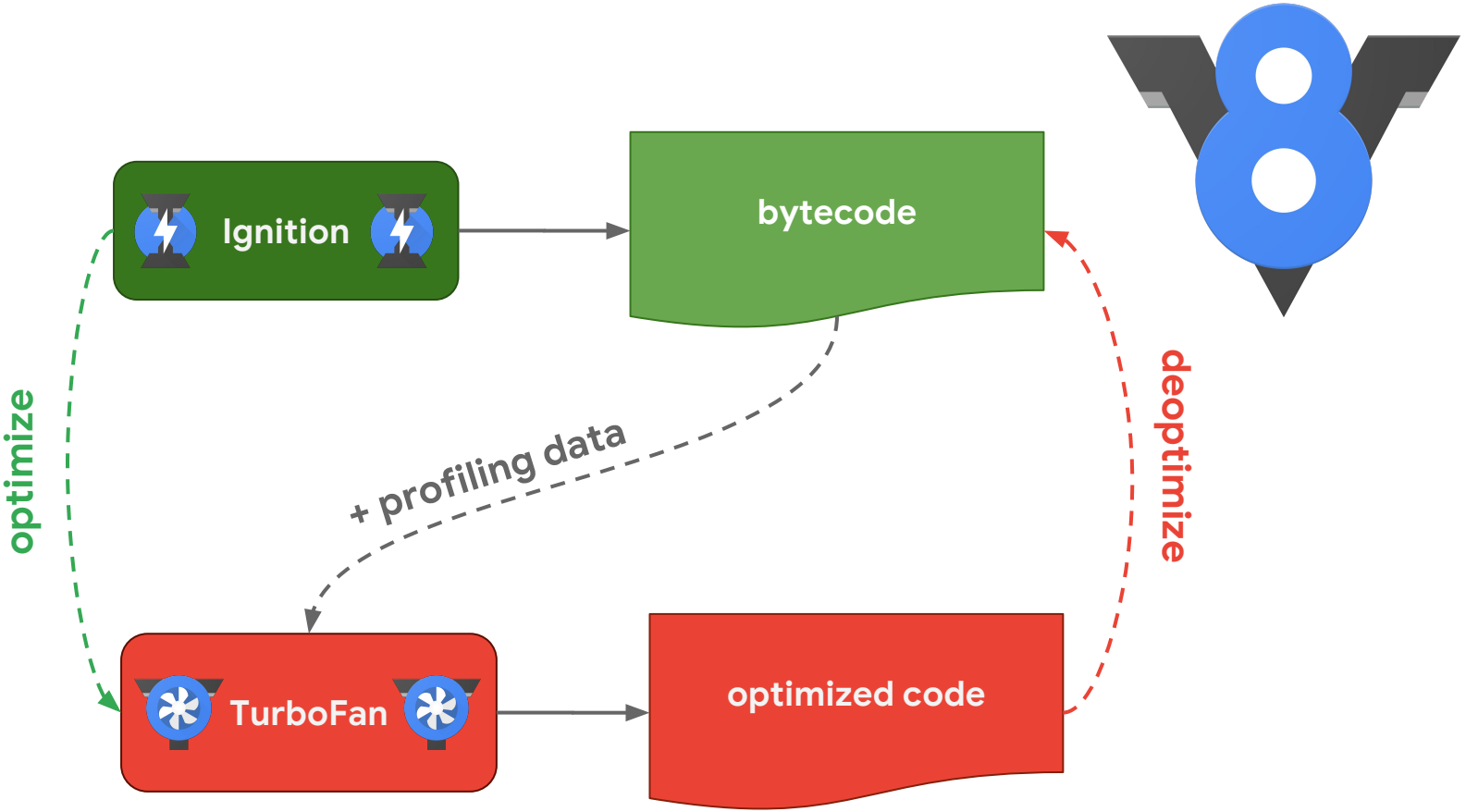
JavaScript engines
have a lot in common



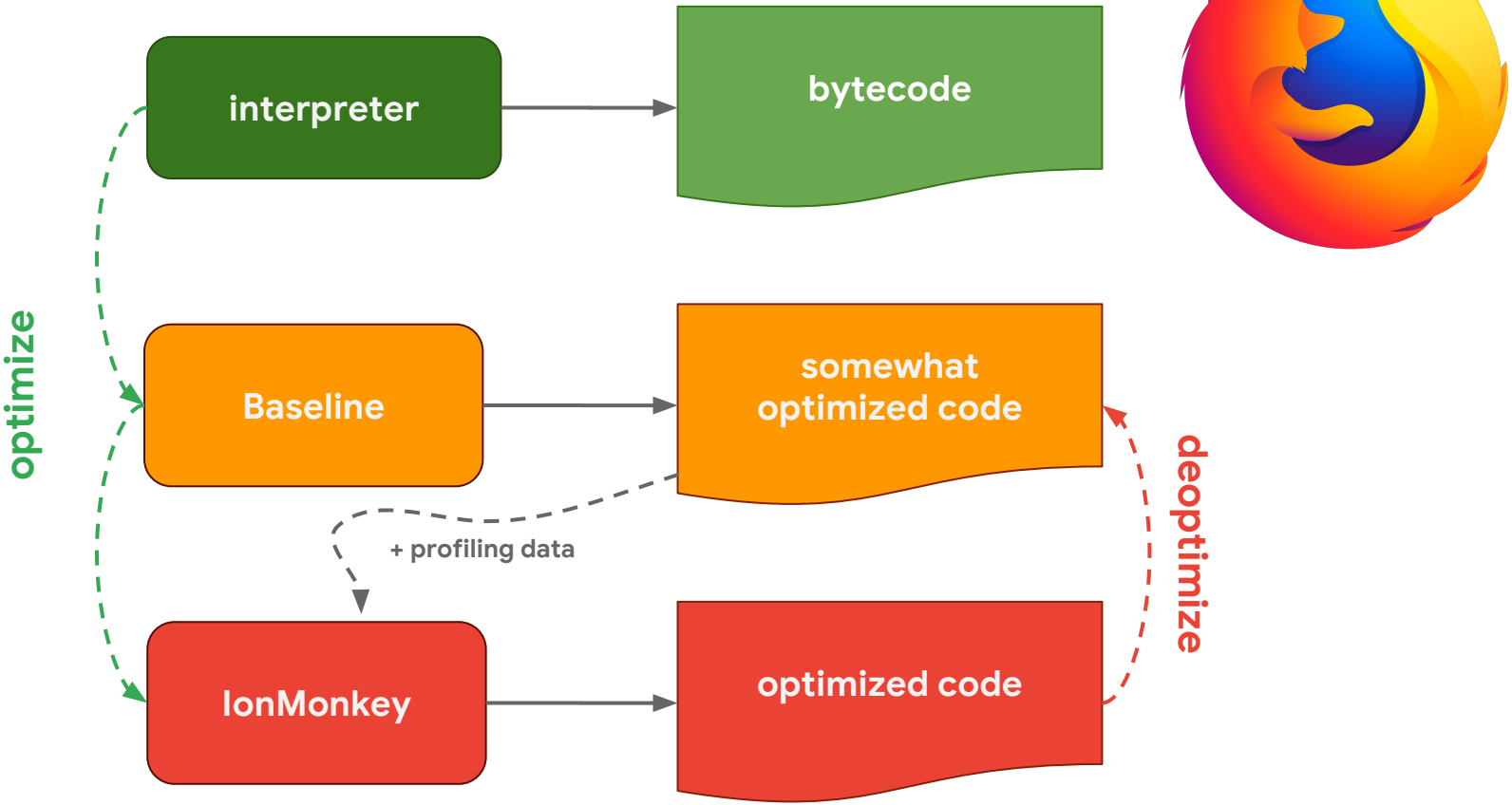
@bmeurer && @mathias



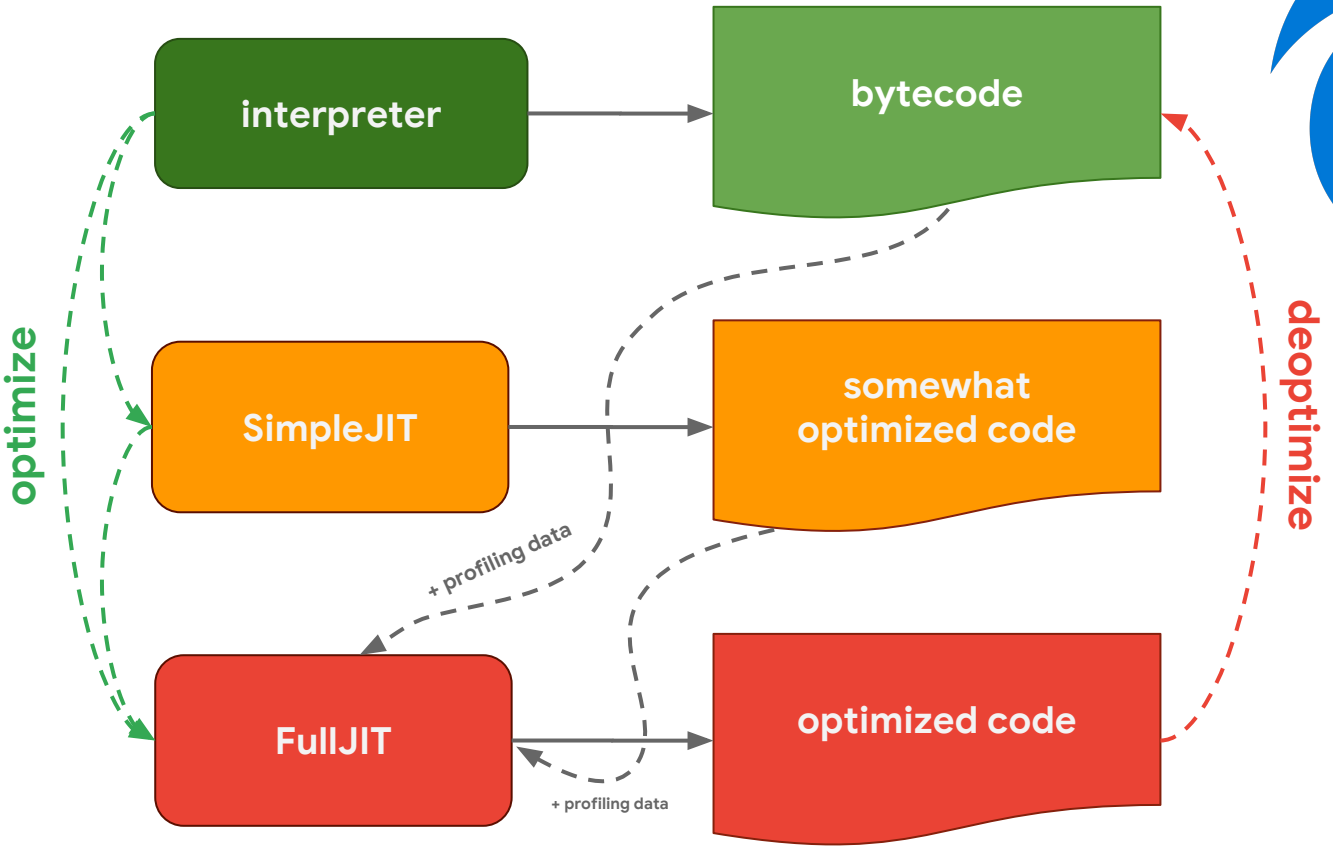
@bmeurer && @mathias



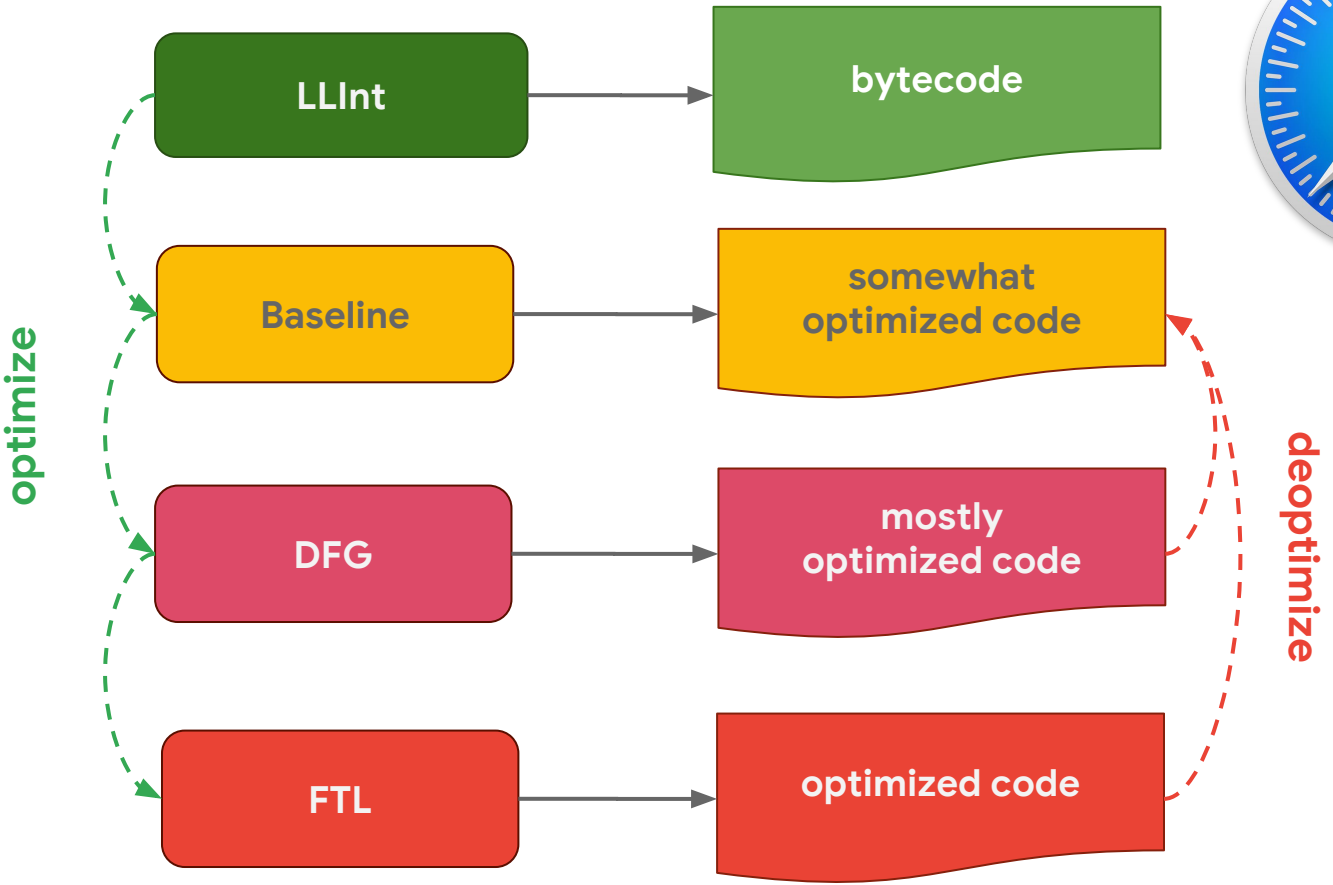
@bmeurer && @mathias



@bmeurer && @mathias



@bmeurer && @mathias

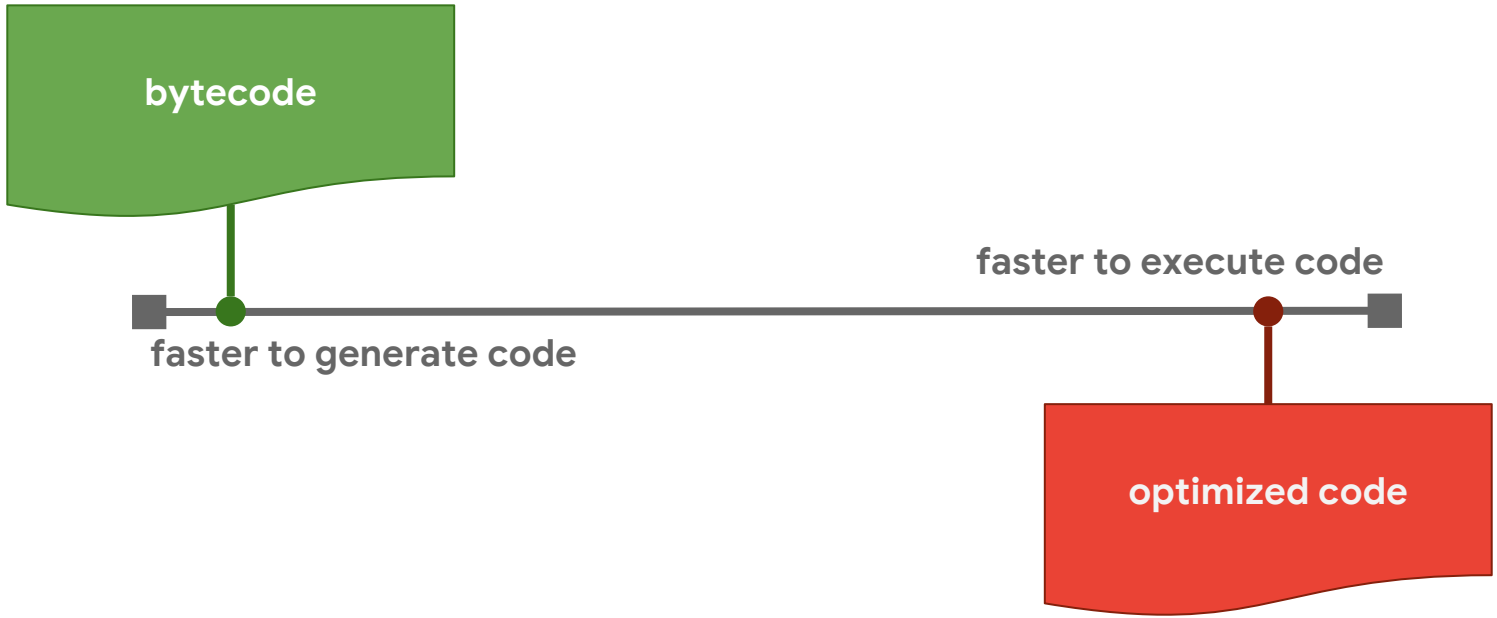


@bmeurer && @mathias

JavaScript engines differ in their optimization tiers


faster to execute code
faster to generate code


bytecode
faster to execute code
faster to generate code

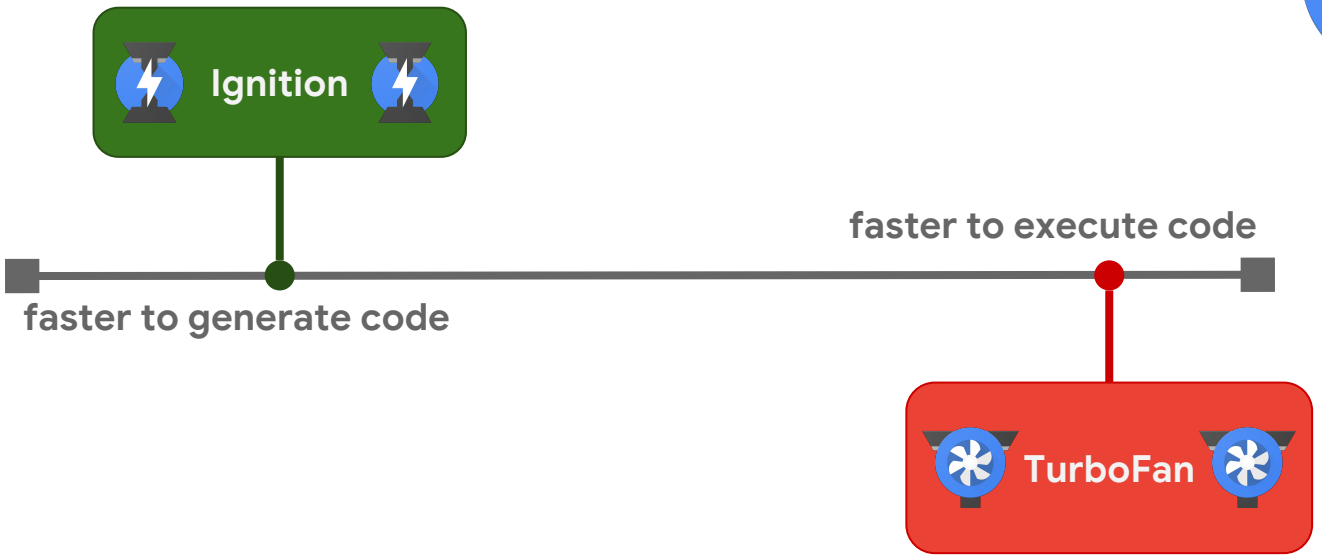

bytecode
faster to execute code
faster to generate code
optimized code


faster to execute code
faster to generate code


Ignition
faster to execute code
faster to generate code

#humblebrag

@bmeurer && @mathias

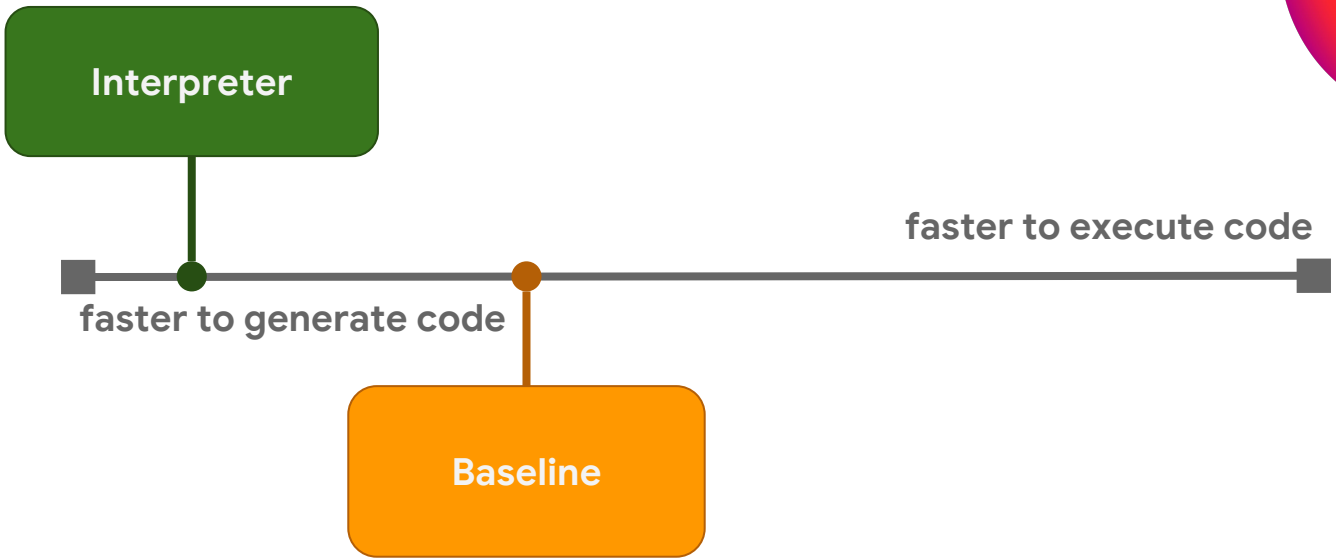


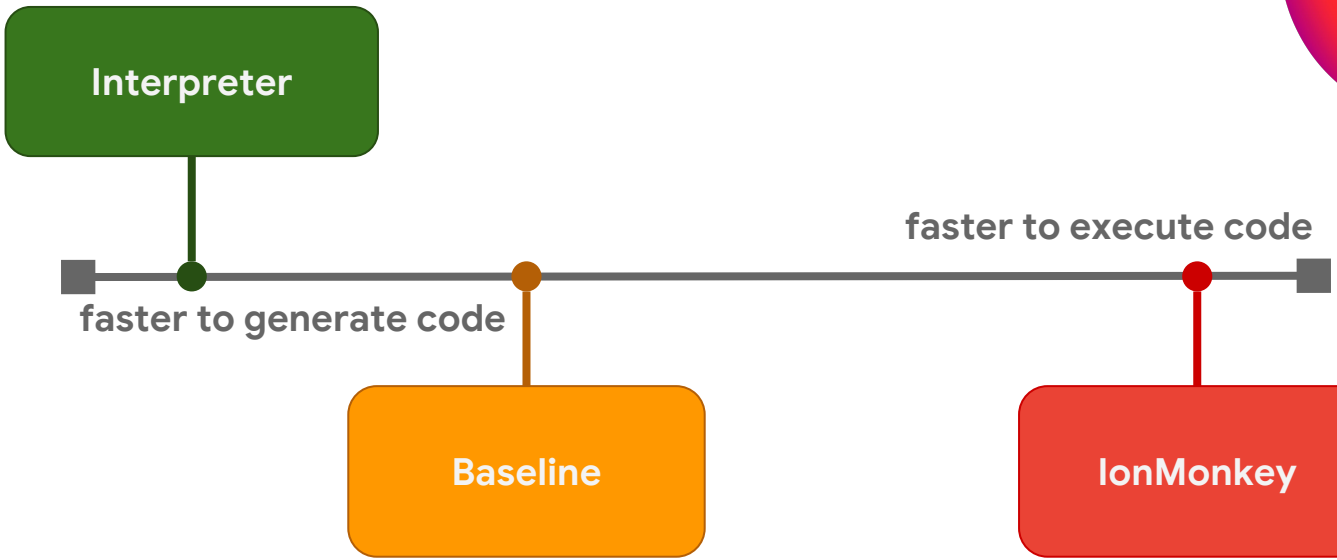
@bmeurer && @mathias

faster to execute code

faster to generate code

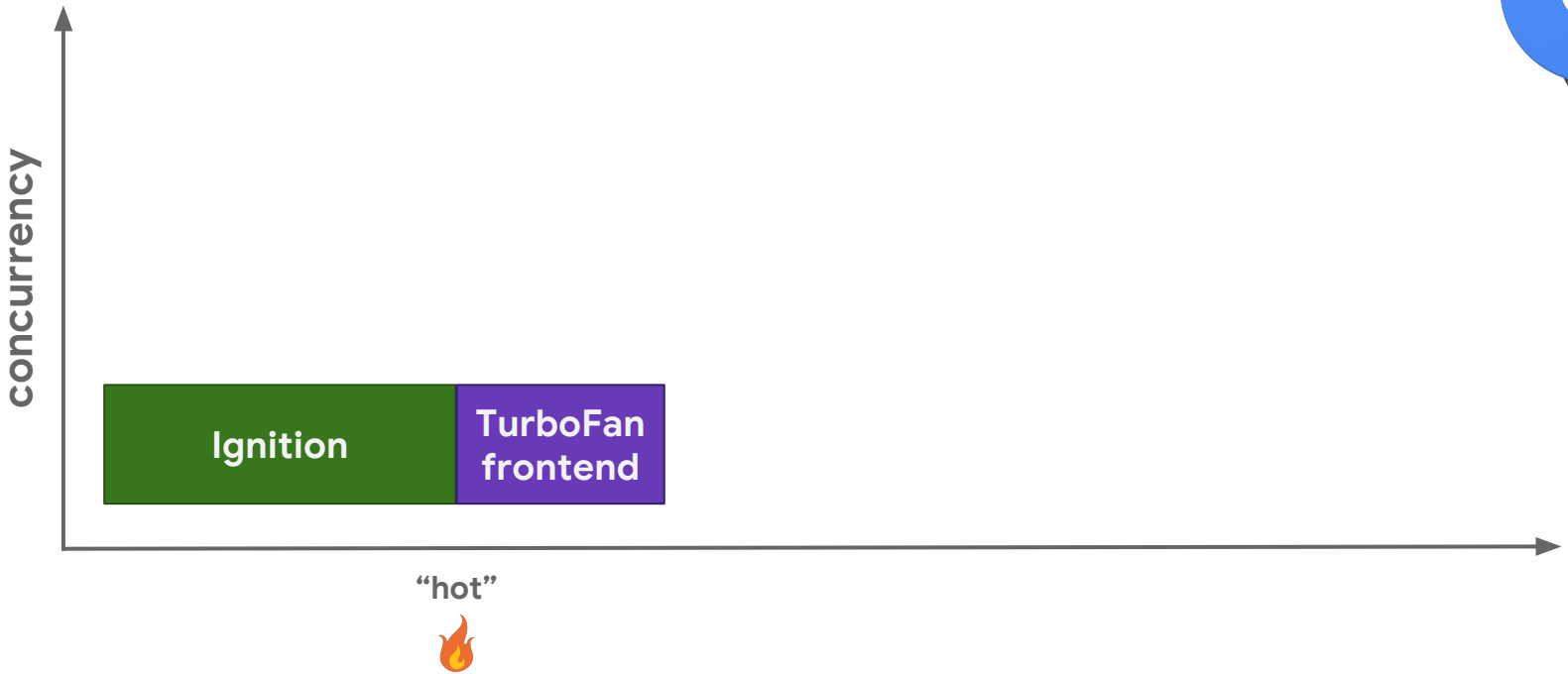

Interpreter
faster to execute code
faster to generate code


Interpreter
faster to execute code
faster to generate code
Baseline

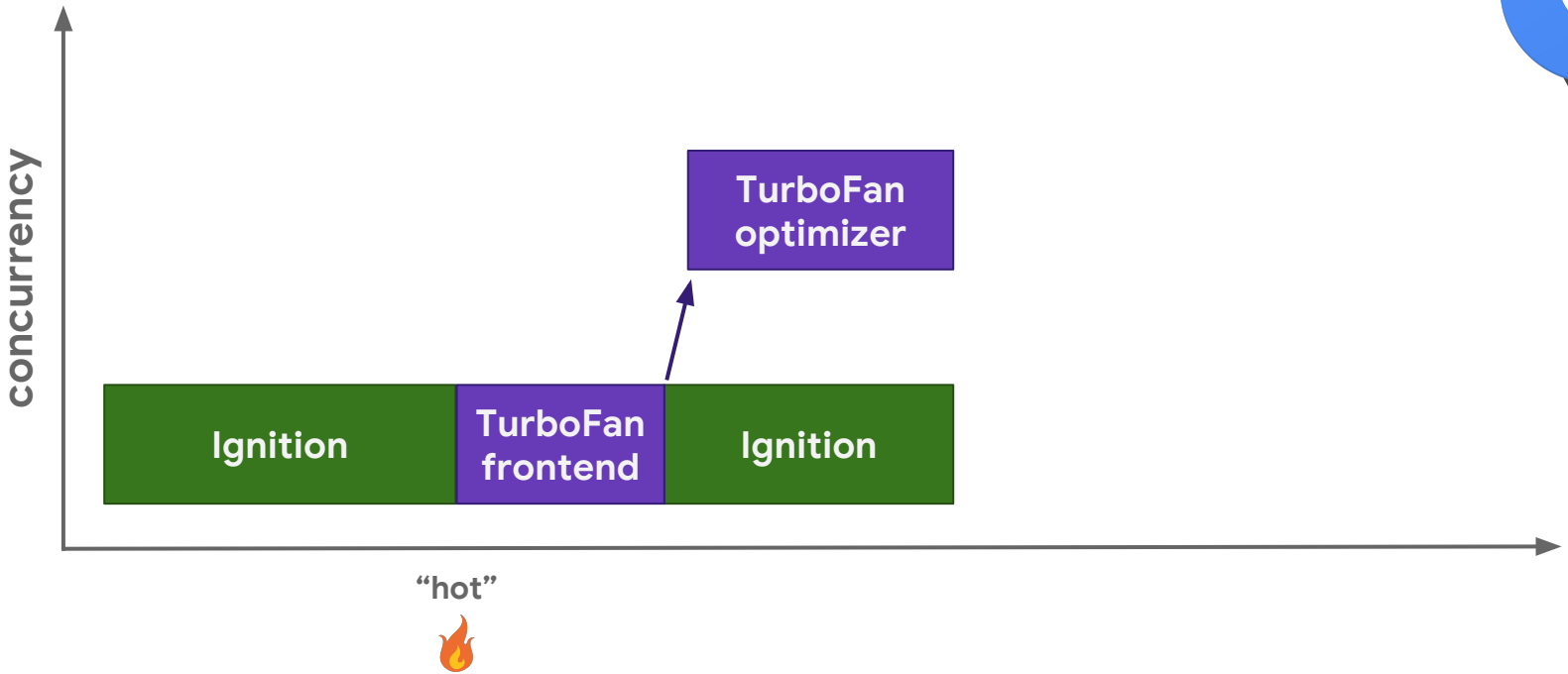

Interpreter
faster to execute code
faster to generate code
Baseline
IonMonkey

```
let result = 0;
for (let i = 0; i < 4242424242; ++i) {
  result += i;
}
console.log(result);
```

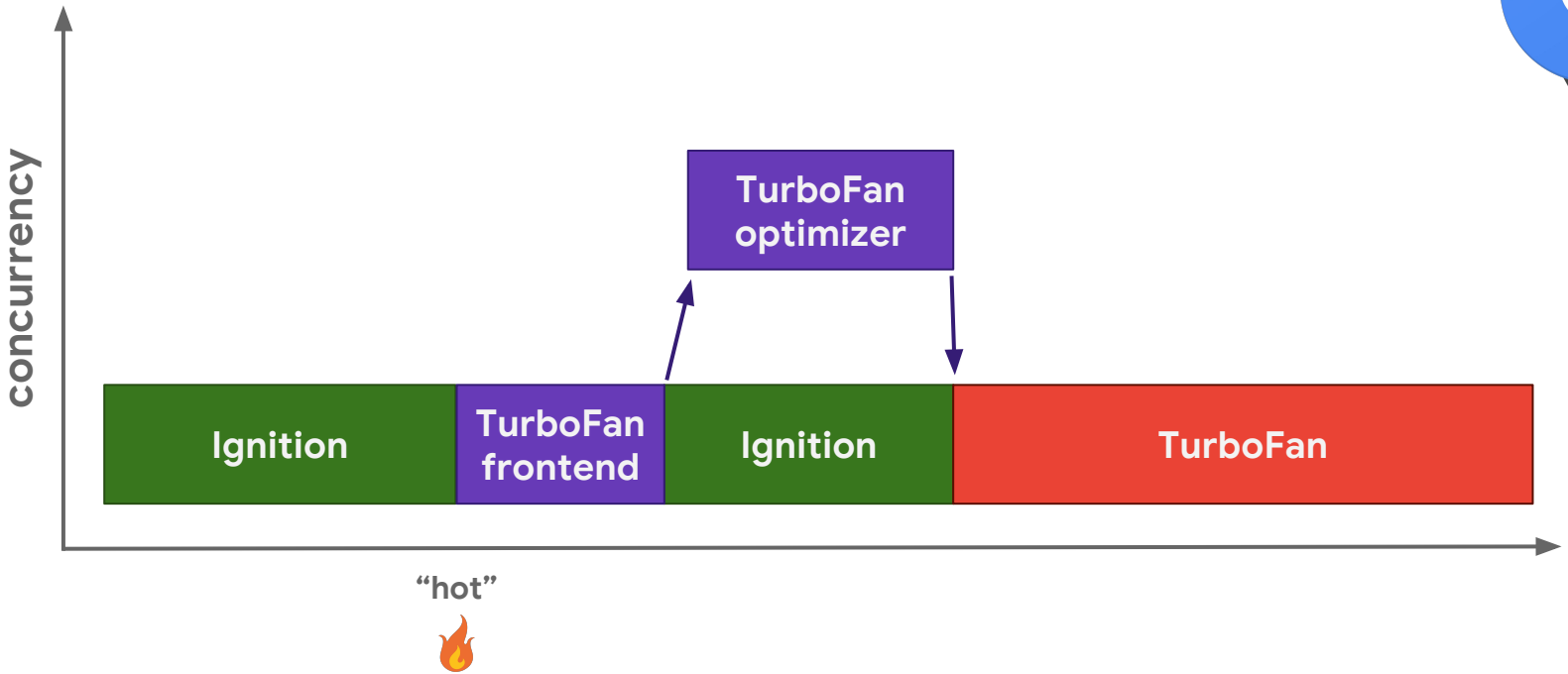

concurrency
Ignition



@bmeurer && @mathias

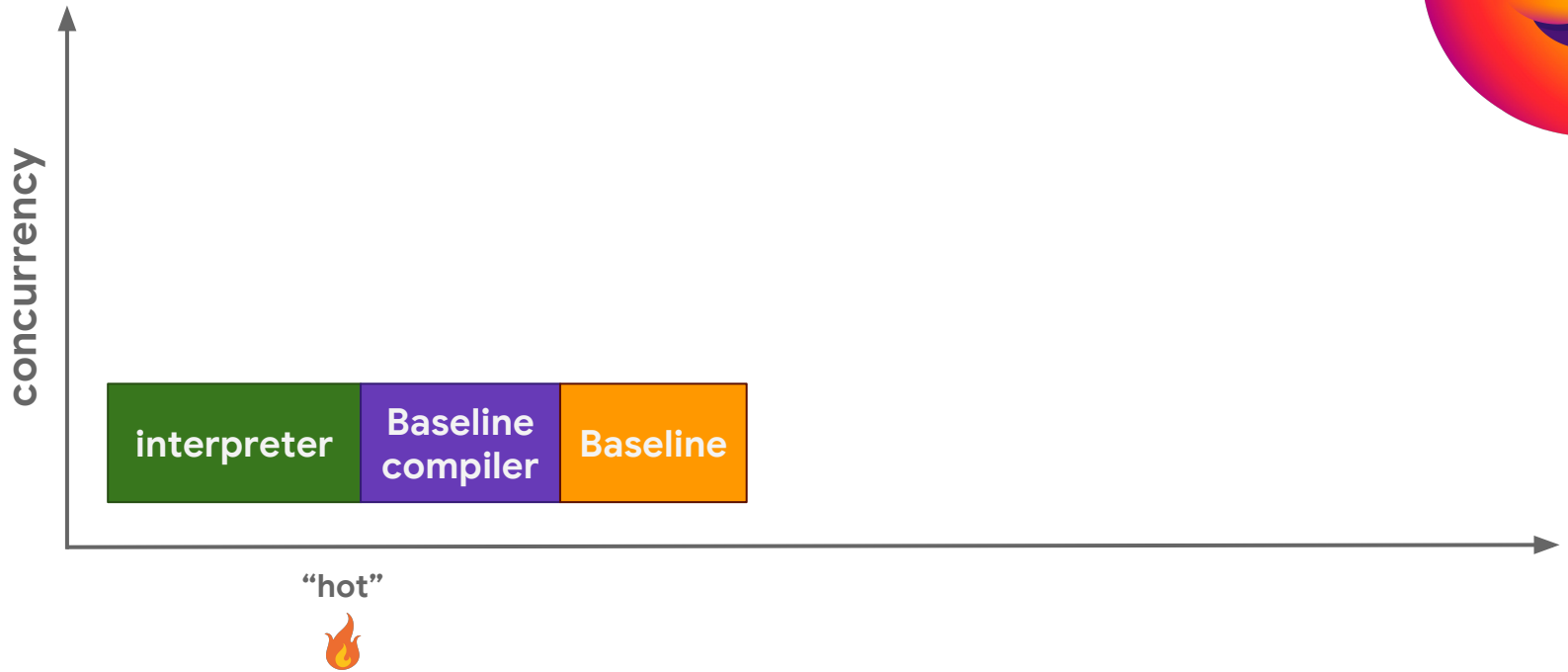


@bmeurer && @mathias

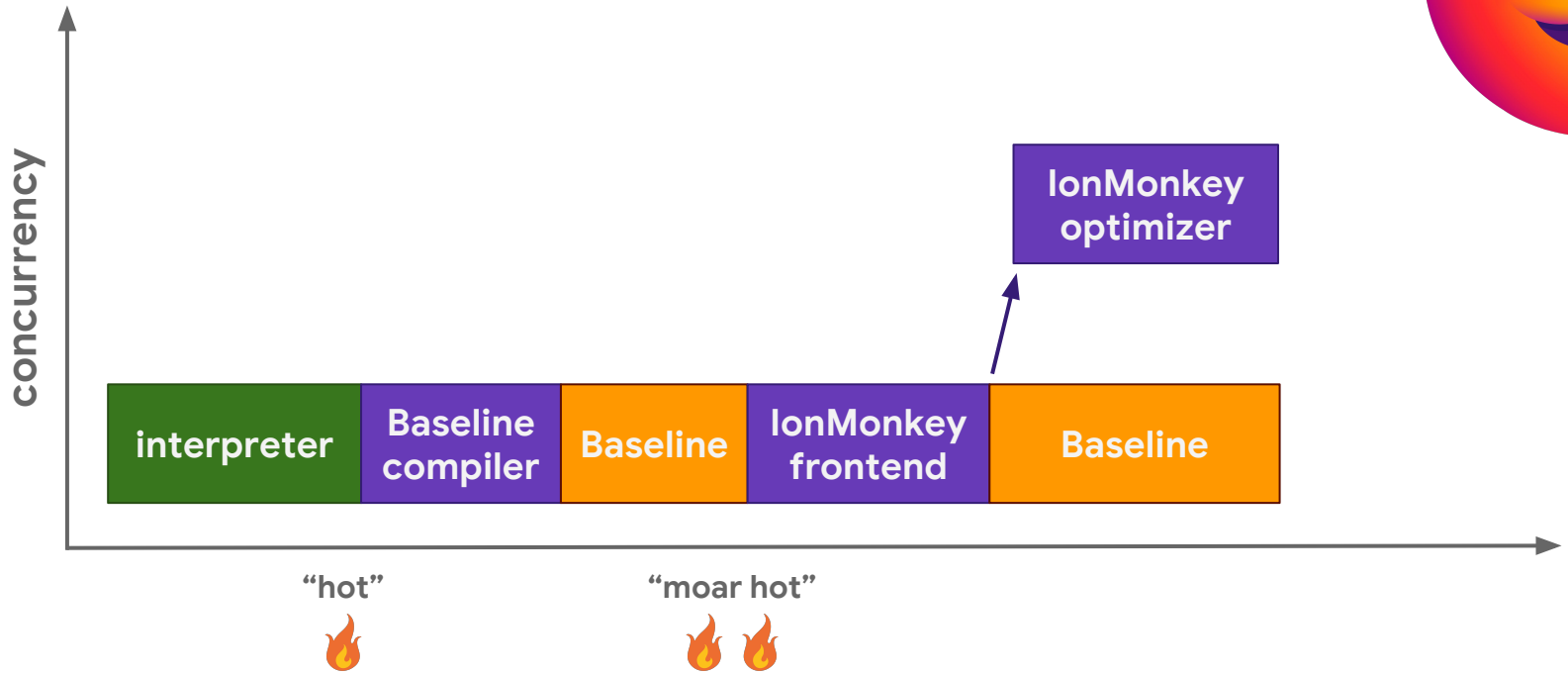


@bmeurer && @mathias

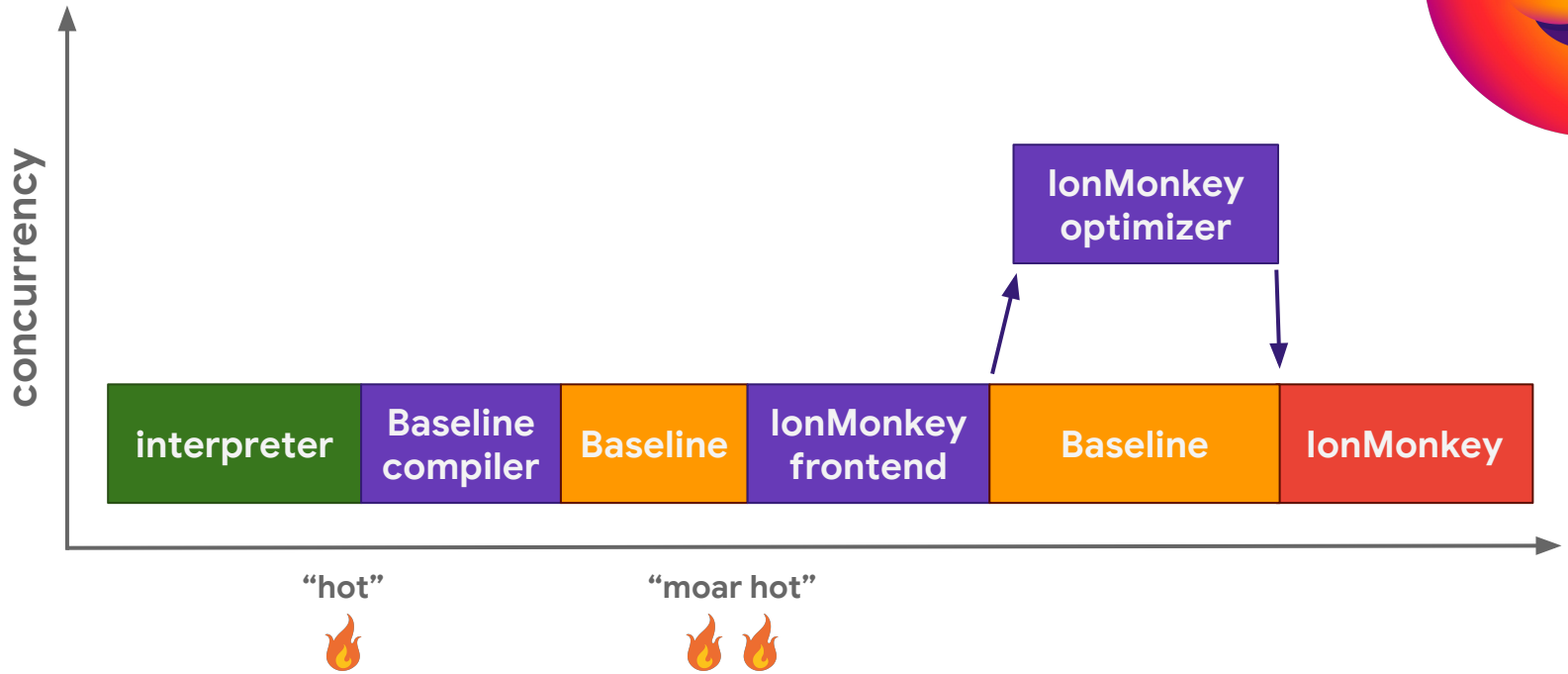

concurrency
interpreter



@bmeurer && @mathias

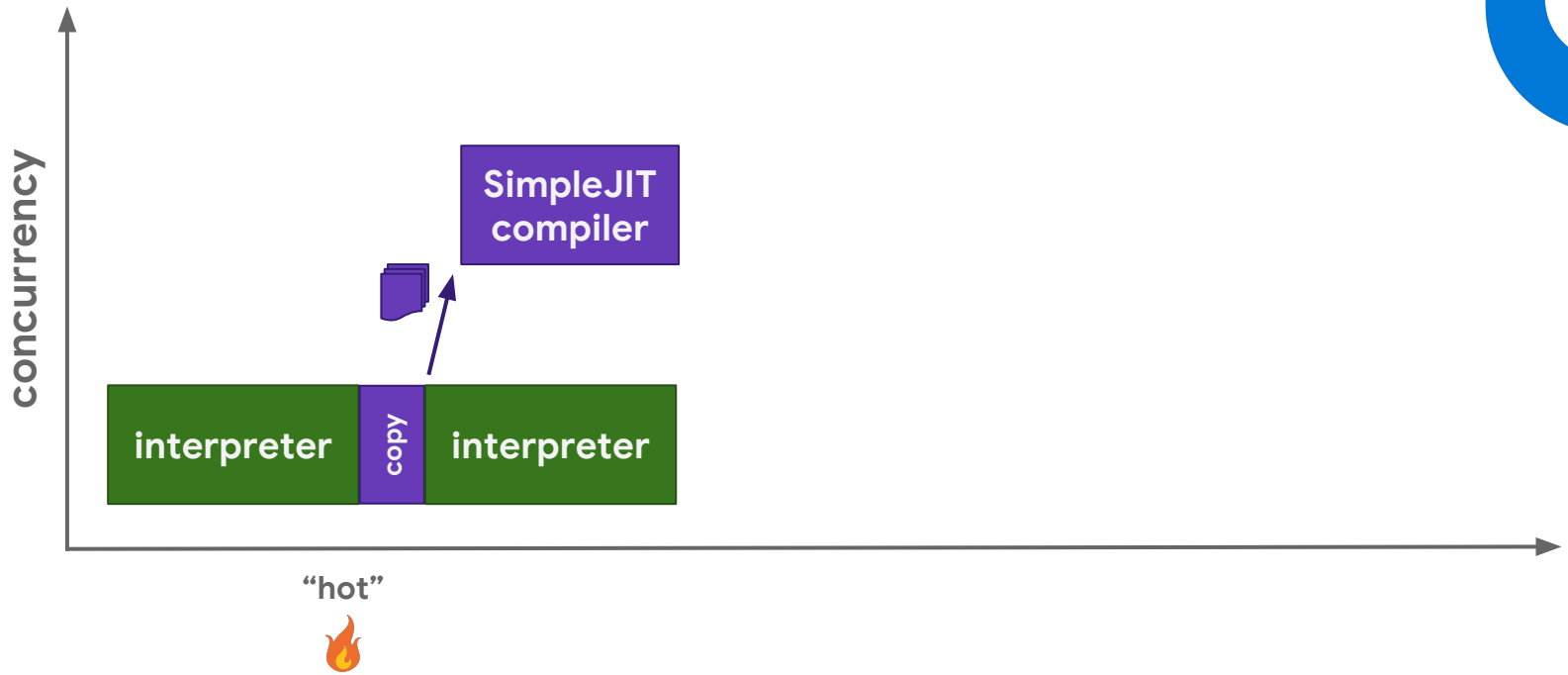


@bmeurer && @mathias

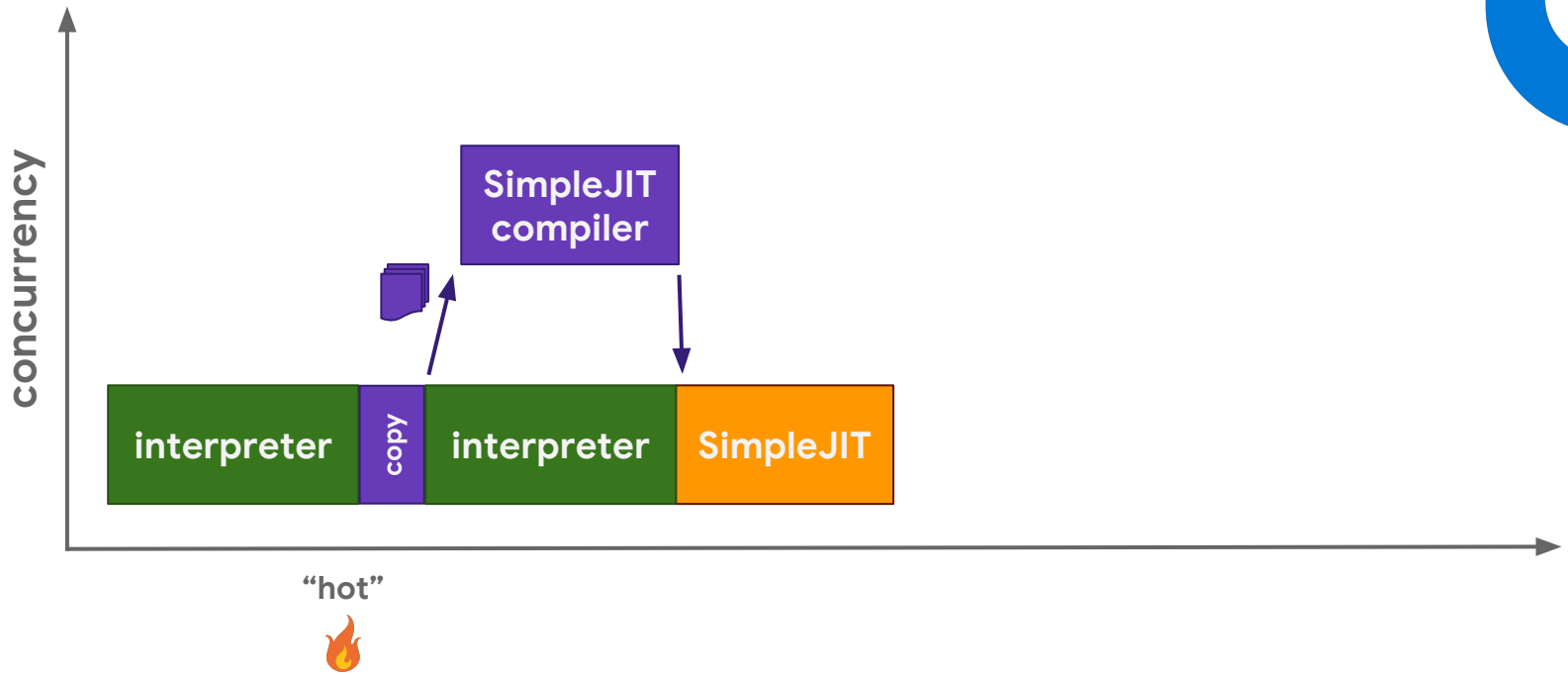


@bmeurer && @mathias

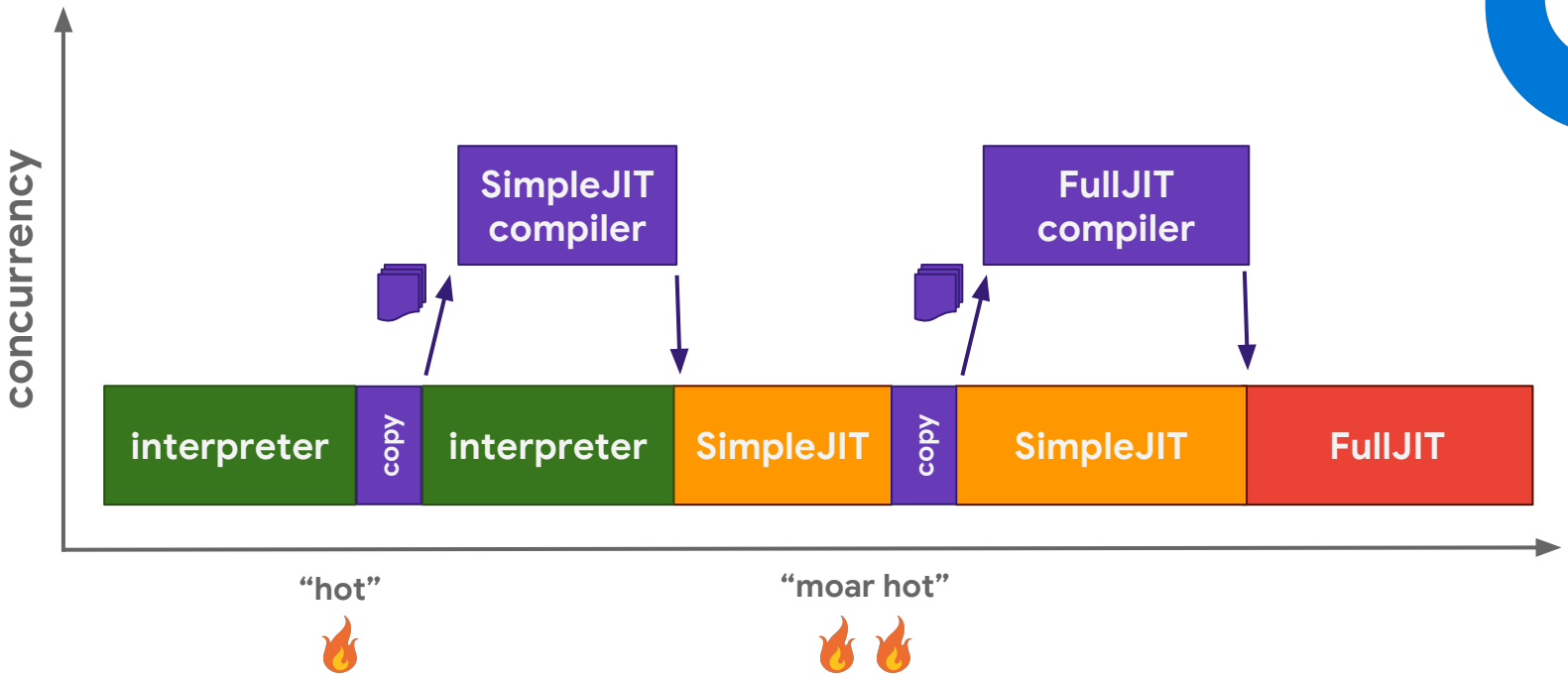

concurrency
interpreter



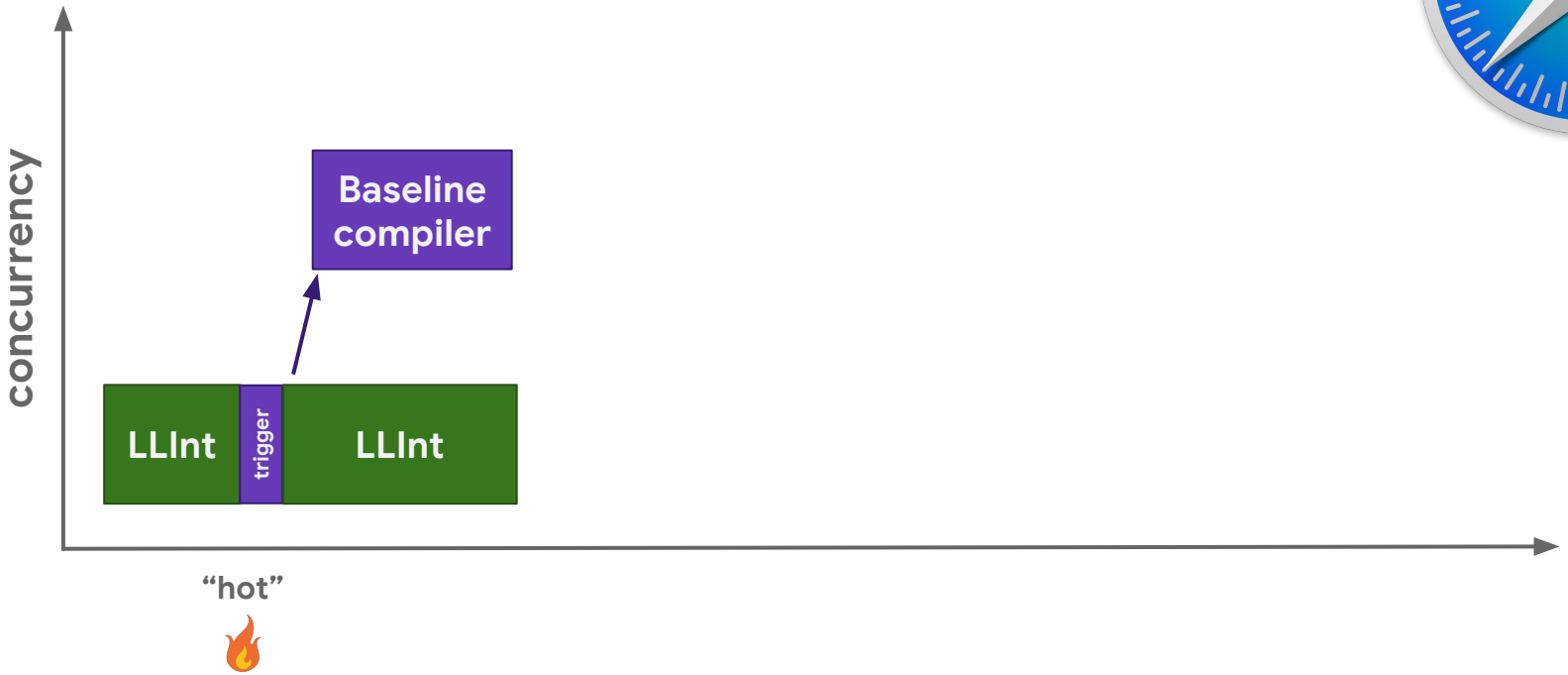
@bmeurer && @mathias



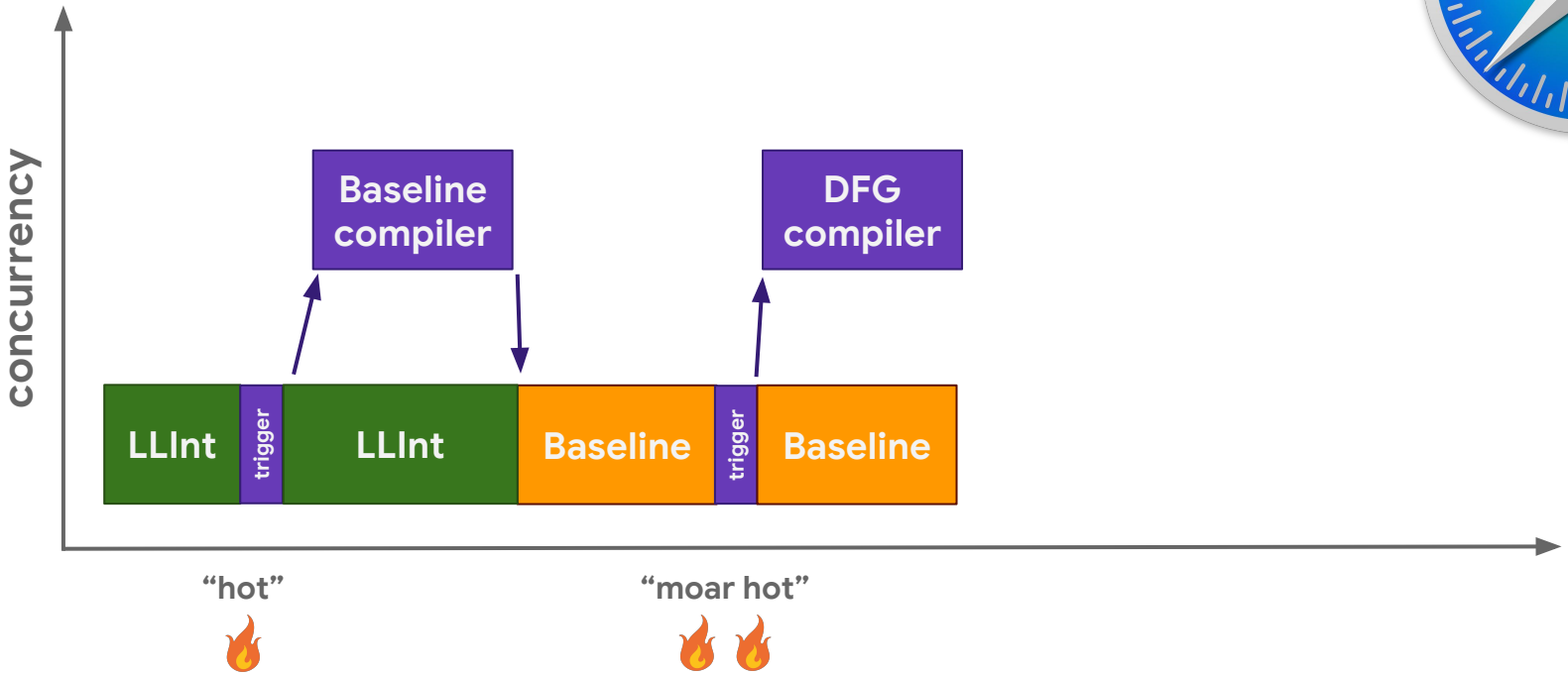
@bmeurer && @mathias



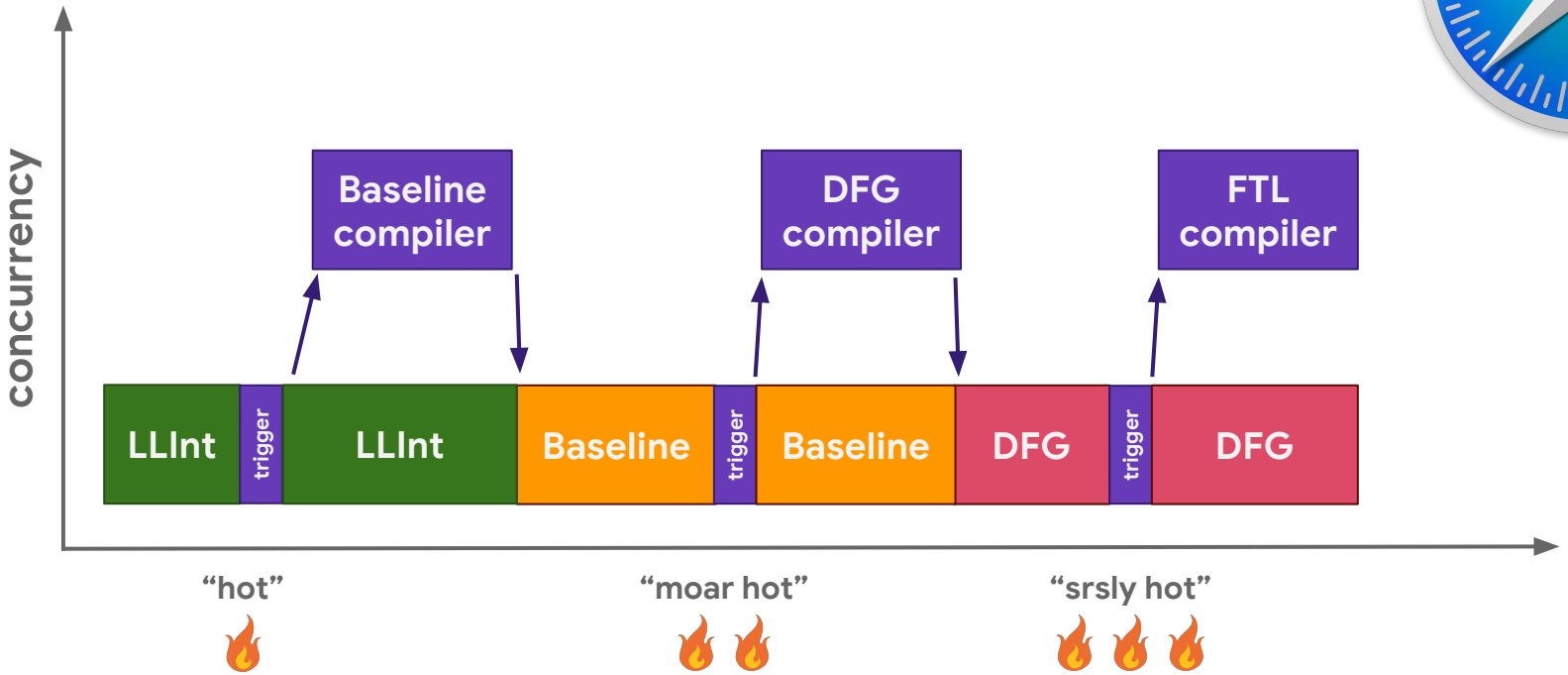
@bmeurer && @mathias



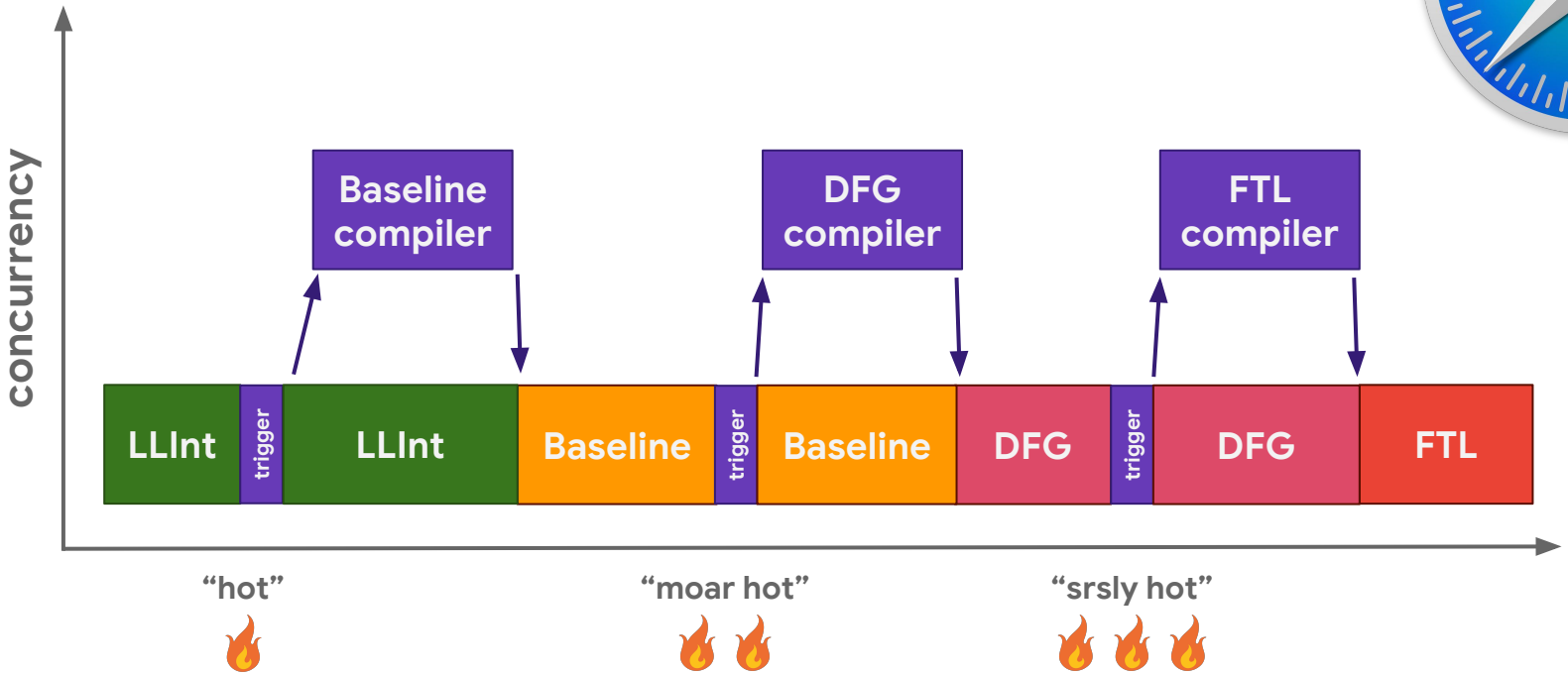
@bmeurer && @mathias



@bmeurer && @mathias



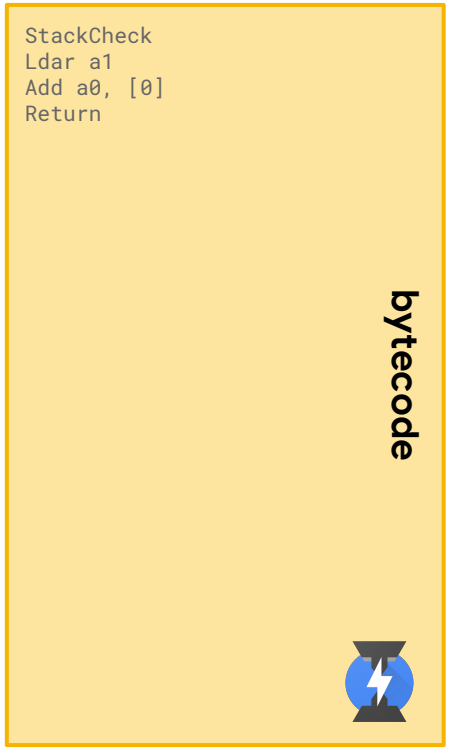
@bmeurer && @mathias



@bmeurer && @mathias

```
function add(x, y) {
  return x + y;
}

add(1, 2);
```

```
function add(x, y) {
  return x + y;
}

add(1, 2);
```



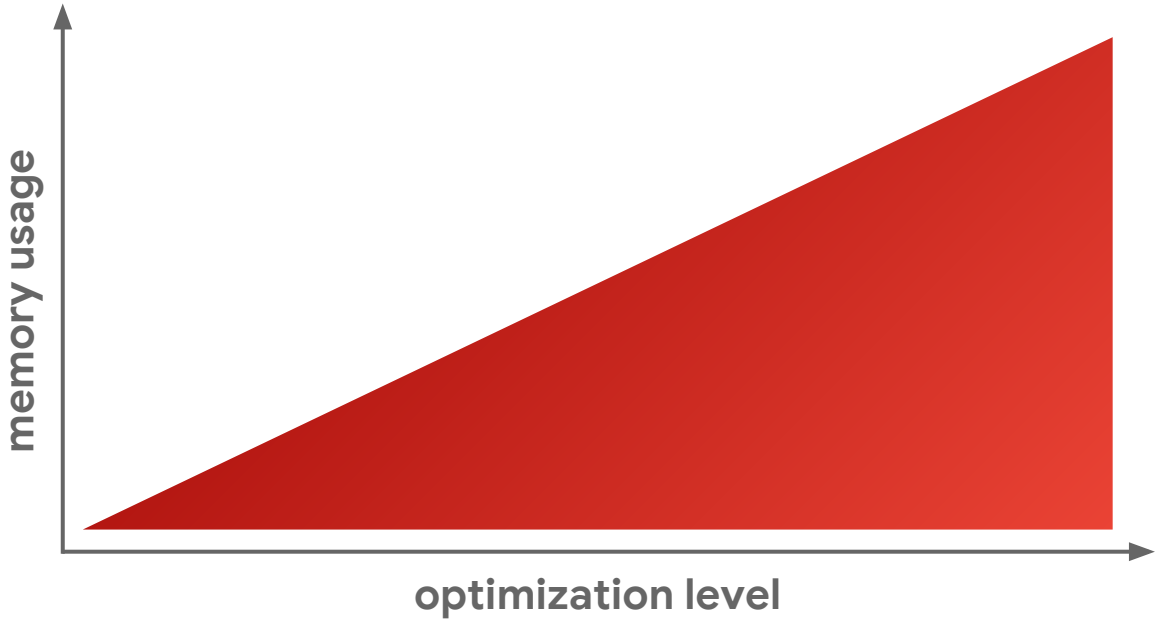
```
function add(x, y) {
  return x + y;
}

add(1, 2);
```

```
StackCheck
Ldar a1
Add a0, [0]
Return
```

bytecode

```
leaq rcx,[rip+0x0]
movq rcx,[rcx-0x37]
testb [rcx+0xf],0x1
jnz CompileLazyDeoptimizedCode
push rbp
movq rbp,rsp
push rsi
push rdi
cmpq rsp,[r13+0xe88]
jna StackOverflow
movq rax,[rbp+0x18]
test al,0x1
jnz Deoptimize
movq rbx,[rbp+0x10]
testb rbx,0x1
jnz Deoptimize
movq rdx,rbx
shrq rdx, 32
movq rcx,rax
shrq rcx, 32
addl rdx,rcx
jo Deoptimize
shlq rdx, 32
movq rax,rdx
movq rsp,rbp
pop rbp
ret 0x18
```

optimized code


memory usage
optimization level

Trade-offs between *generating code quickly* vs. *generating quick code* and *optimization level* vs. *memory usage*

```
const object = {
  x: 5,
  y: 6,
};

// Later:
doSomething(object.x);
```

```
object = {
  x: 5,
  y: 6,
};
```

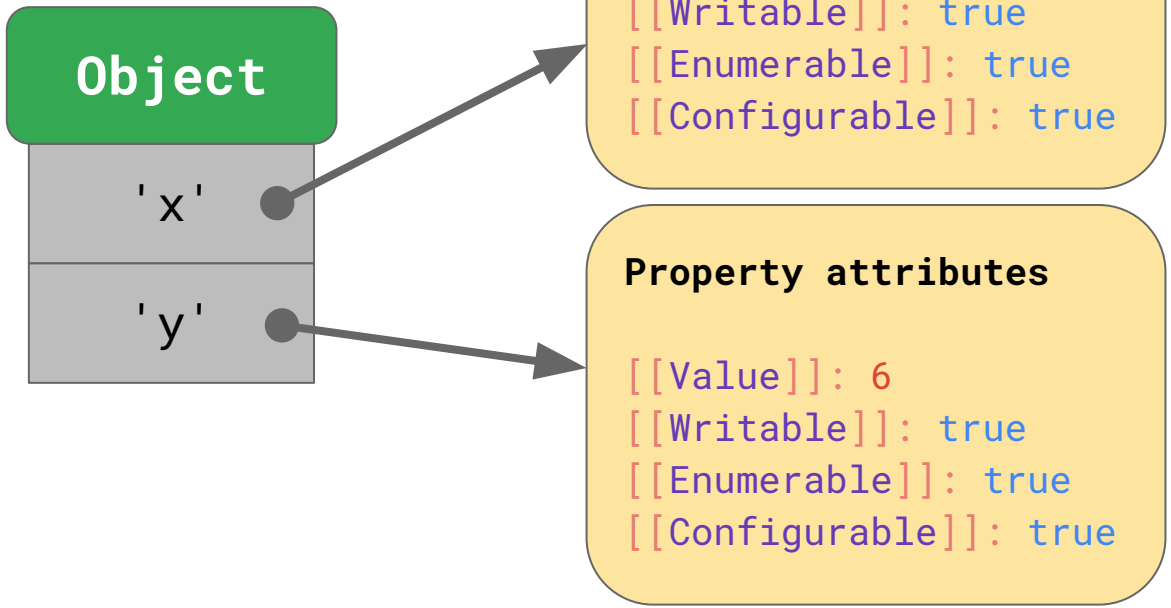

Object
'x'
'y'
Property attributes
[[Value]]: 5
[[Writable]]: true
[[Enumerable]]: true
[[Configurable]]: true
Property attributes
[[Value]]: 6
[[Writable]]: true
[[Enumerable]]: true
[[Configurable]]: true

Table 2: Attributes of a Data Property

Attribute Name	Value Domain	Description
[[Value]]	Any ECMAScript language type	The value retrieved by a get access of the property.
[[Writable]]	Boolean	If **false**, attempts by ECMAScript code to change the property's [[Value]] attribute using [[Set]] will not succeed.
[[Enumerable]]	Boolean	If **true**, the property will be enumerated by a for-in enumeration (see 13.7.5). Otherwise, the property is said to be non-enumerable.
[[Configurable]]	Boolean	If **false**, attempts to delete the property, change the property to be an accessor property, or change its attributes (other than [[Value]], or changing [[Writable]] to **false**) will fail.

Property attributes

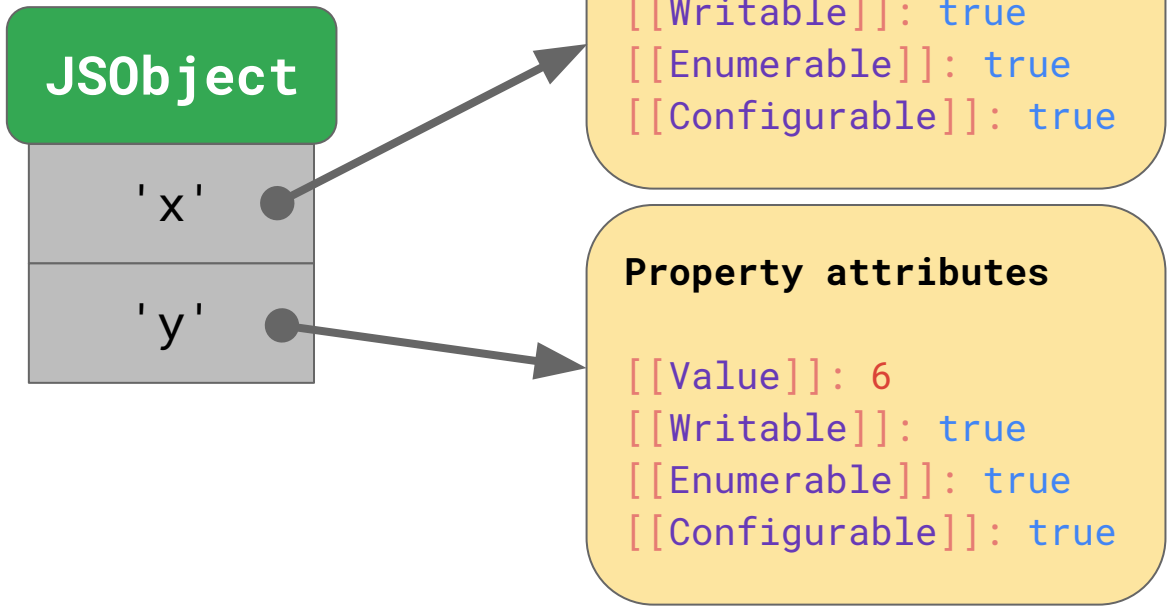
```
[[Value]]: 5
[[Writable]]: true
[[Enumerable]]: true
[[Configurable]]: true
```

```
const object = {
  foo: 'bar',
  baz: 'qux',
};

// Later:
doSomething(object.foo);
```

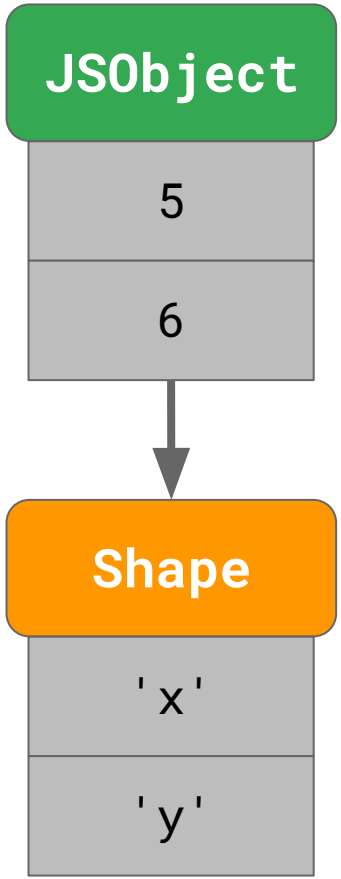
```
const object1 = {
  x: 1,
  y: 2,
};
const object2 = {
  x: 3,
  y: 4,
};
// `object1` and `object2` have the same shape.
```

@bmeurer && @mathias

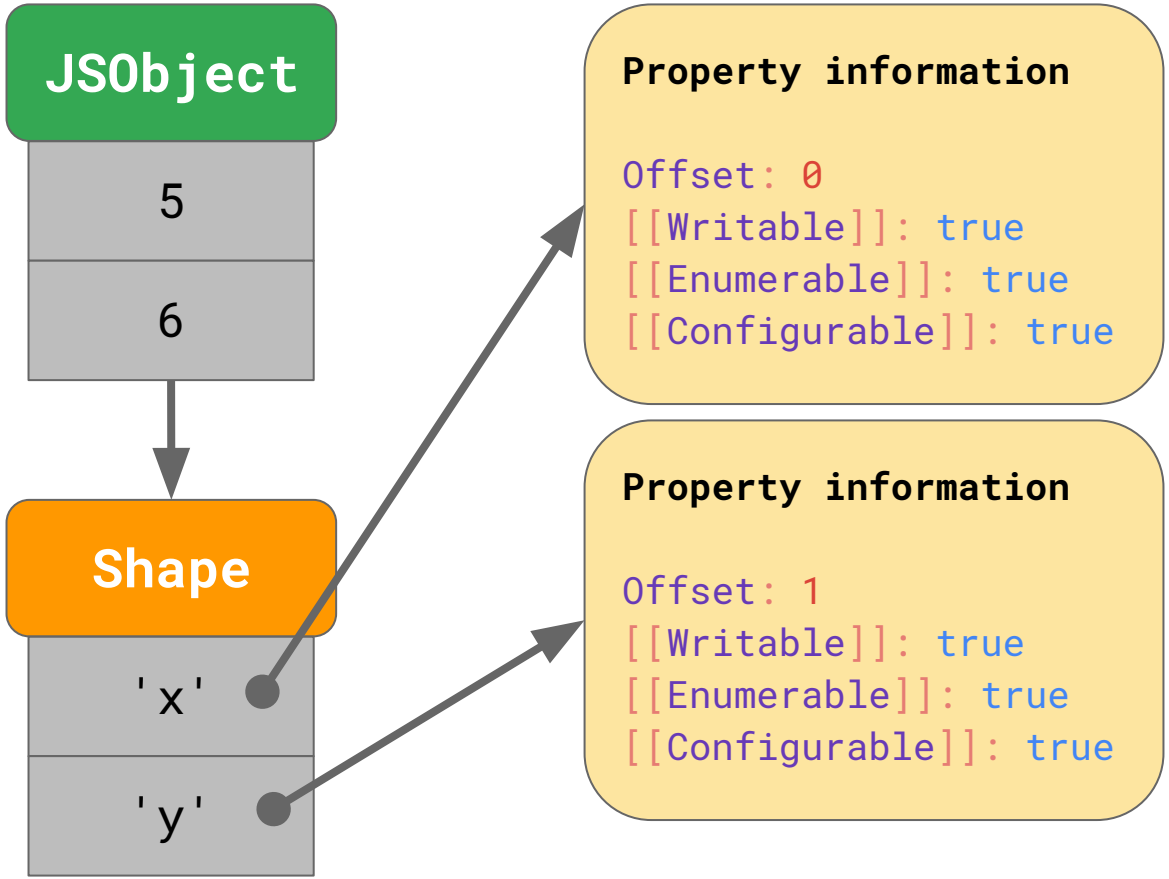
```
object = {
  x: 5,
  y: 6,
};
```

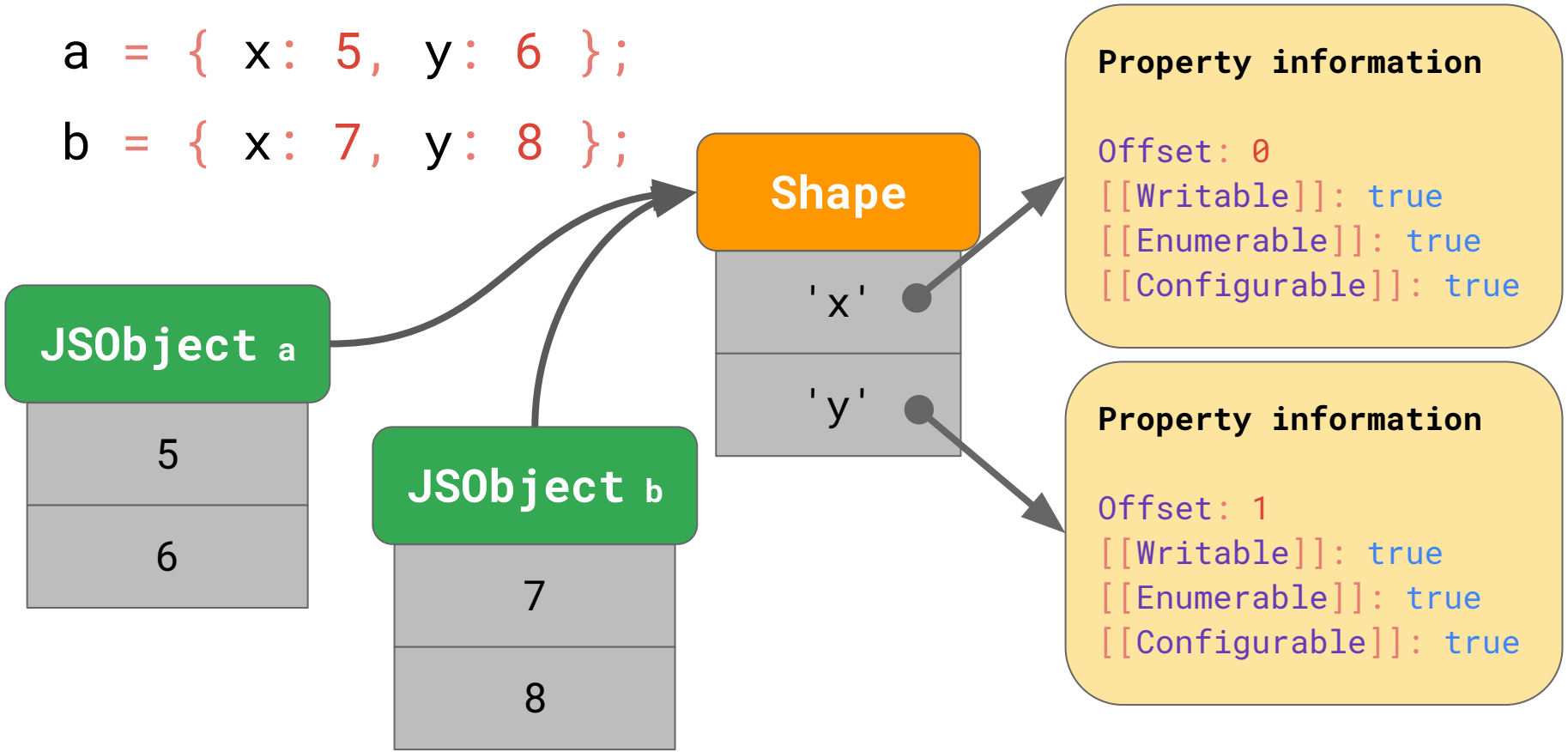

JSObject
'x'
'y'
Property attributes
[[Value]]: 5
[[Writable]]: true
[[Enumerable]]: true
[[Configurable]]: true
Property attributes
[[Value]]: 6
[[Writable]]: true
[[Enumerable]]: true
[[Configurable]]: true

```
object = {
  x: 5,
  y: 6,
};
```



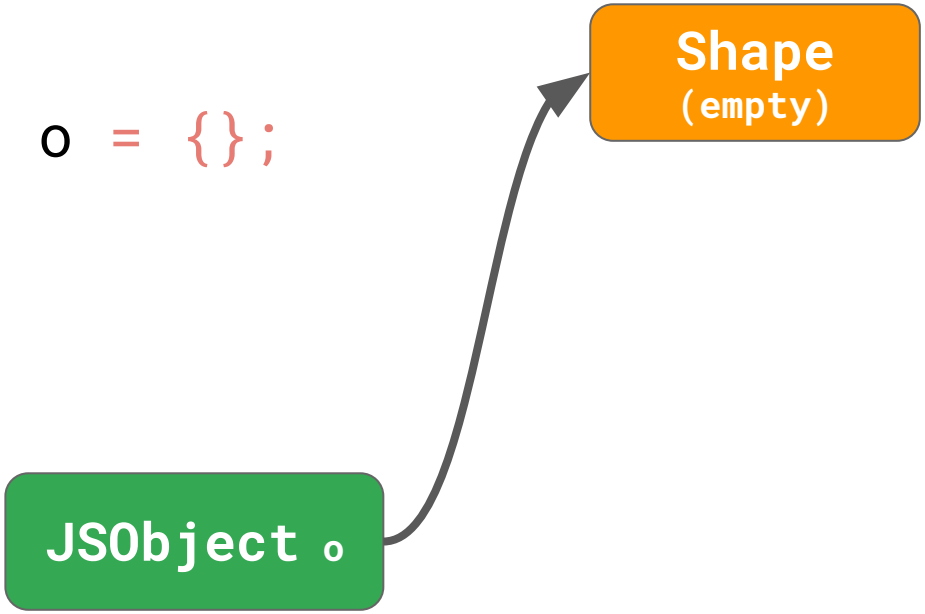
```
object = {
  x: 5,
  y: 6,
};
```

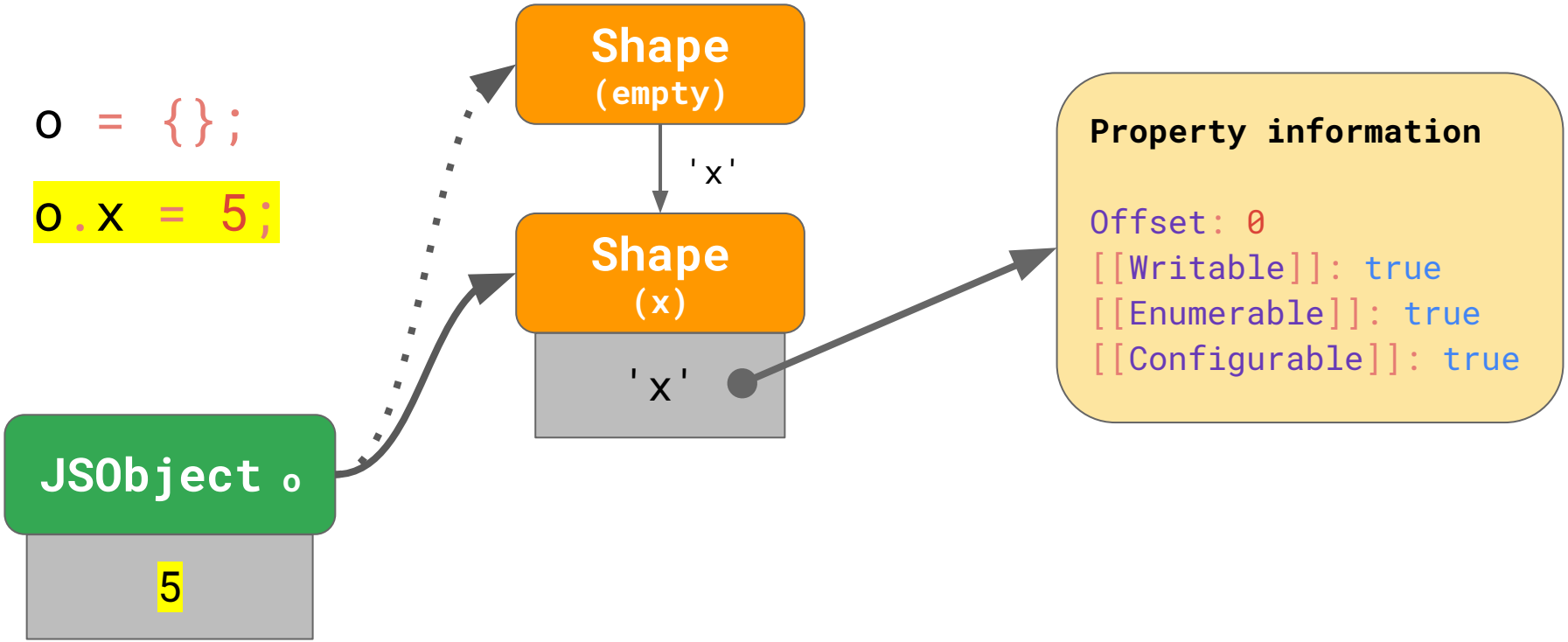

JSObject
5
6
Shape
'x'
'y'
Property information
Offset: 0
[[Writable]]: true
[[Enumerable]]: true
[[Configurable]]: true
Property information
Offset: 1
[[Writable]]: true
[[Enumerable]]: true
[[Configurable]]: true

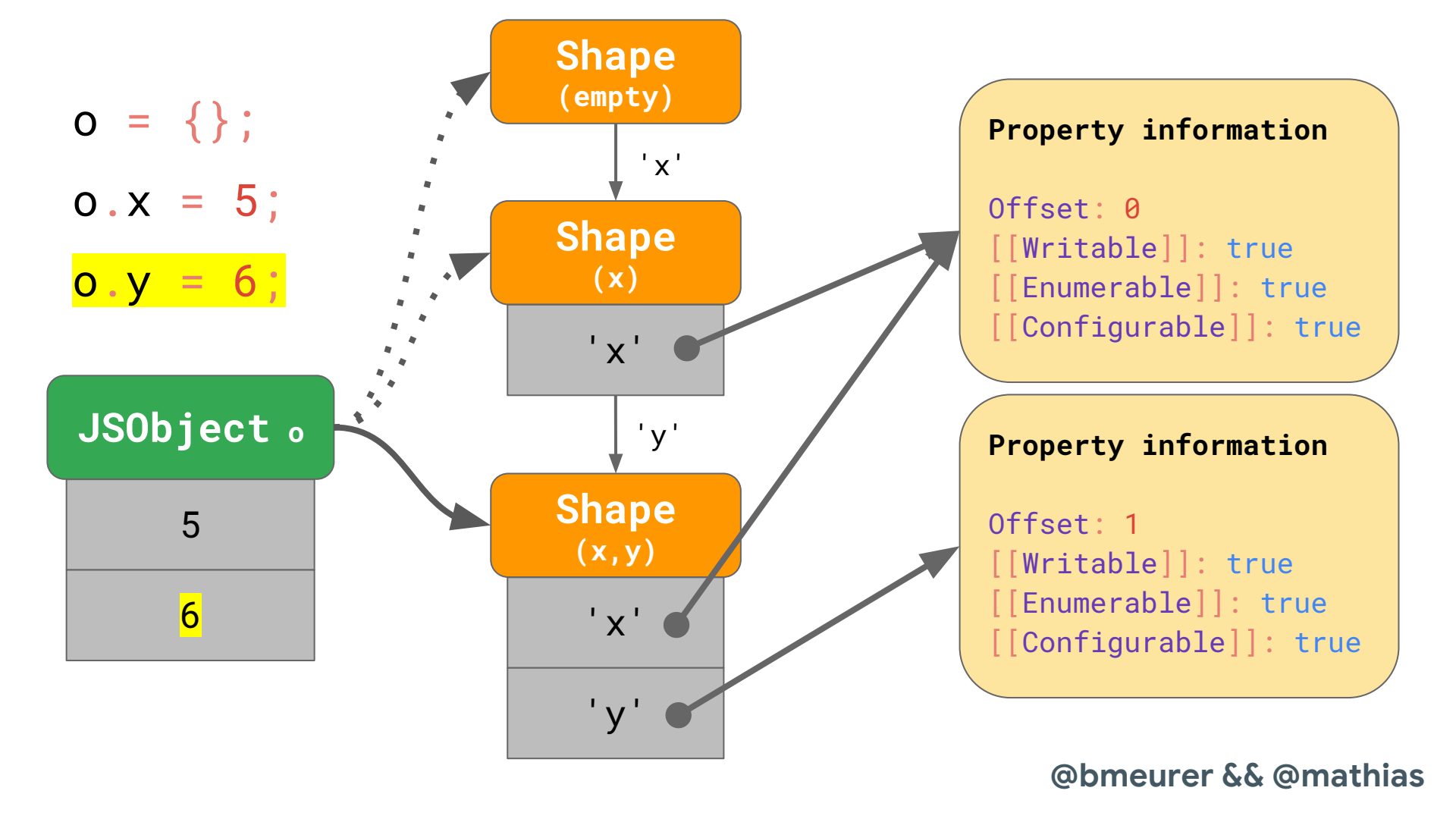

a = { x: 5, y: 6 };
b = { x: 7, y: 8 };
JSObject a
5
6
JSObject b
7
8
Shape
'x'
'y'
Property information
Offset: 0
[[Writable]]: true
[[Enumerable]]: true
[[Configurable]]: true
Property information
Offset: 1
[[Writable]]: true
[[Enumerable]]: true
[[Configurable]]: true

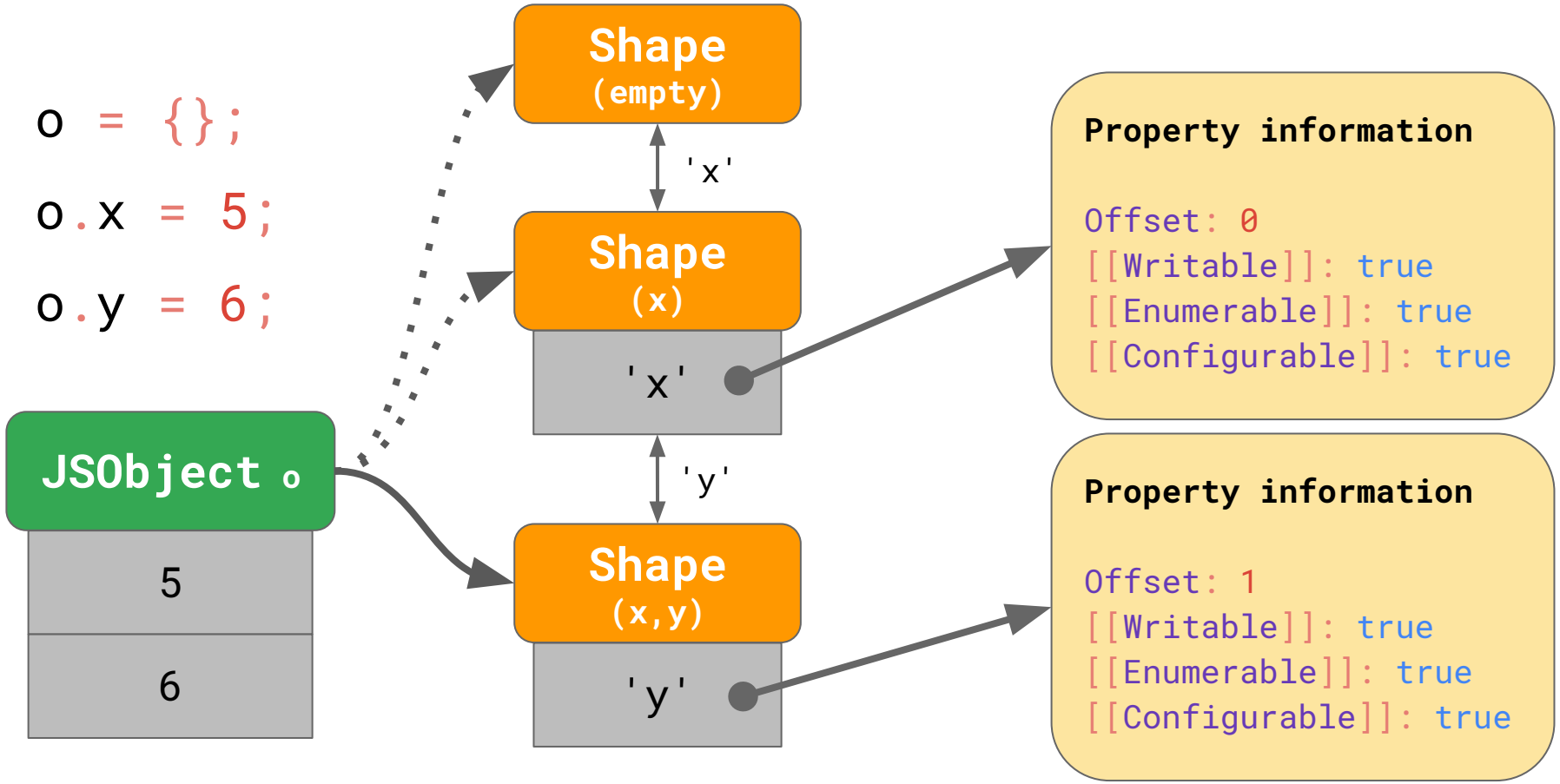
```
const object = {};
object.x = 5;
object.y = 6;
```

```
o = {};
```



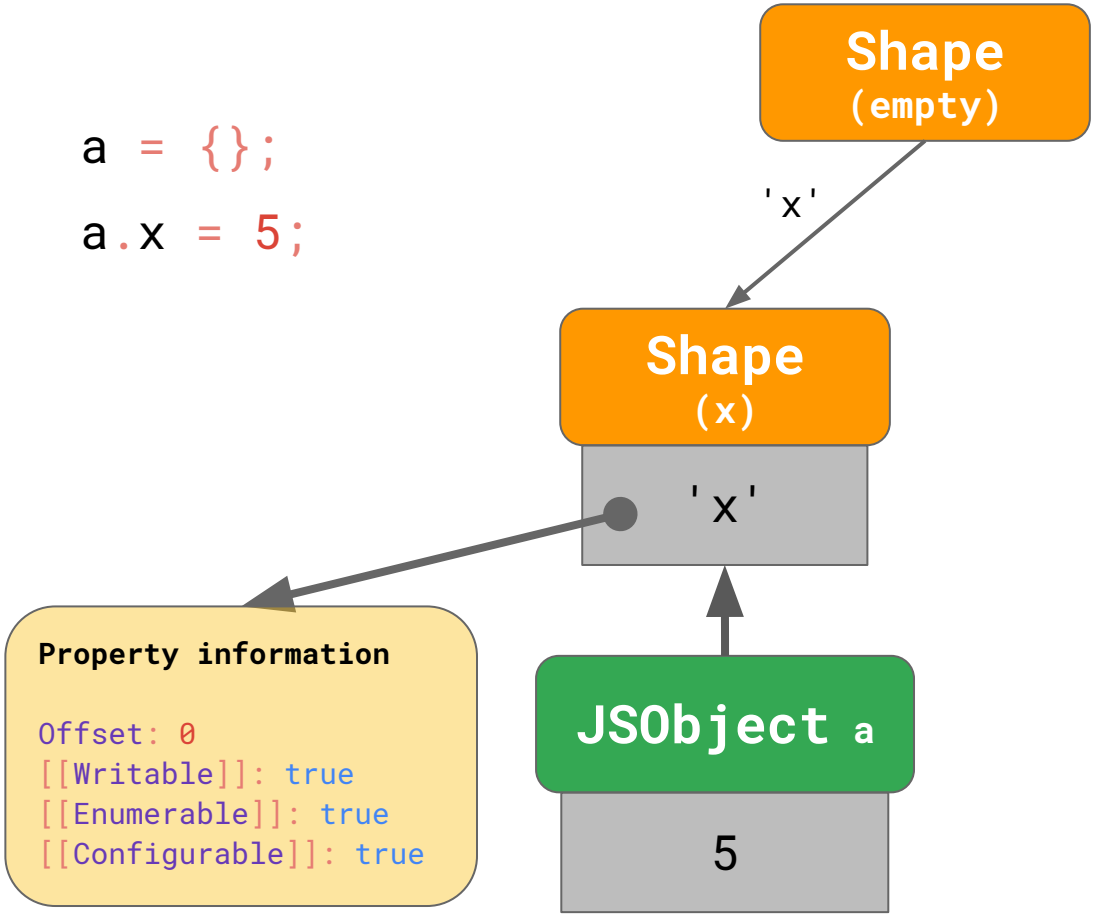

o = {};
o.x = 5;
Shape
(empty)
'x'
Shape
(x)
'x'
JSObject o
5
Property information
Offset: 0
[[Writable]]: true
[[Enumerable]]: true
[[Configurable]]: true


o = {};
o.x = 5;
o.y = 6;
JSObject o
5
6
Shape
(empty)
'x'
Shape
(x)
'x'
'y'
Shape
(x,y)
'x'
'y'
Property information
Offset: 0
[[Writable]]: true
[[Enumerable]]: true
[[Configurable]]: true
Property information
Offset: 1
[[Writable]]: true
[[Enumerable]]: true
[[Configurable]]: true
@bmeurer && @mathias

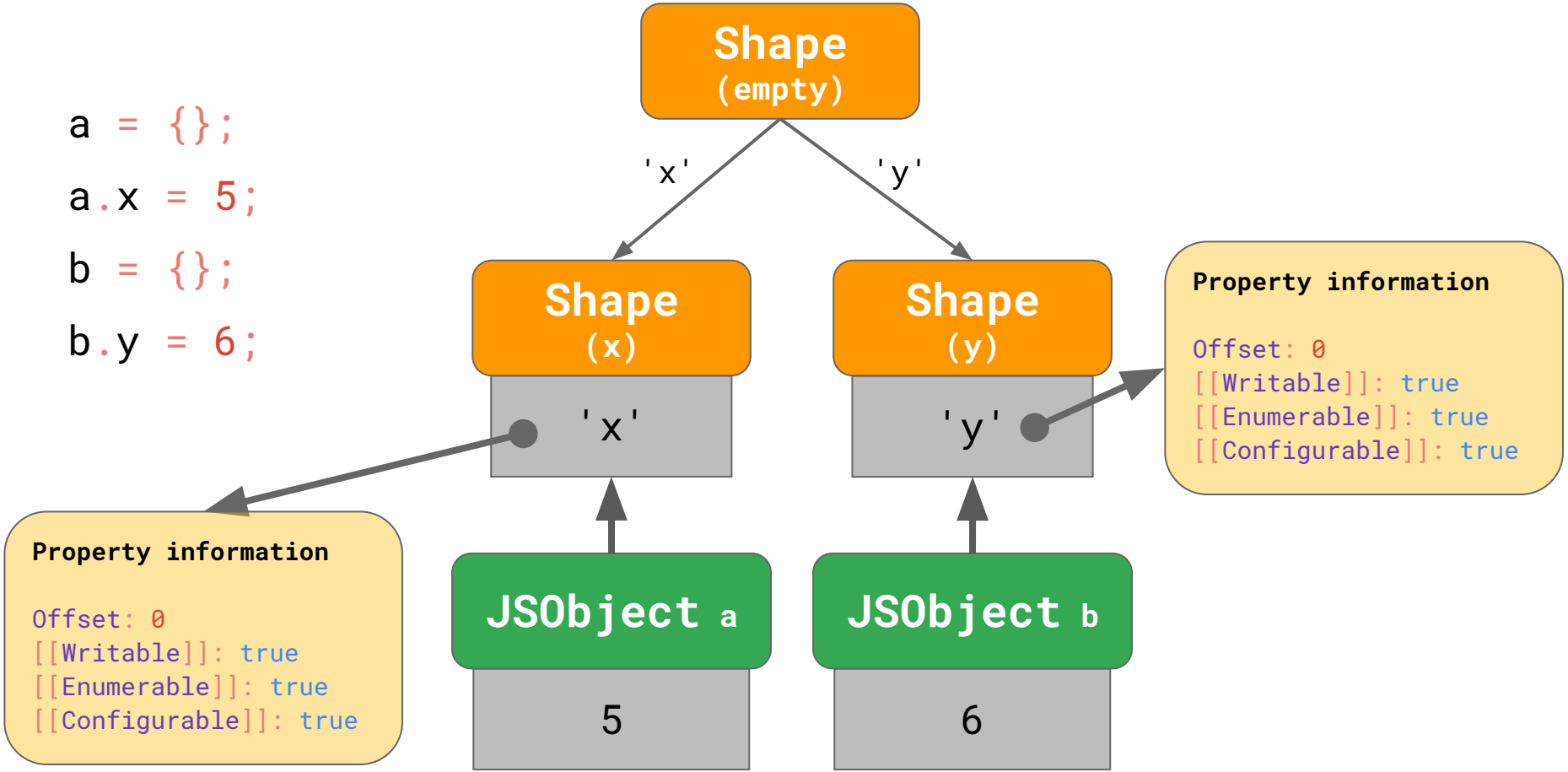

o = {};
o.x = 5;
o.y = 6;
JSObject o
5
6
Shape
(empty)
'x'
Shape
(x)
'x'
'y'
Shape
(x,y)
'y'
Property information
Offset: 0
[[Writable]]: true
[[Enumerable]]: true
[[Configurable]]: true
Property information
Offset: 1
[[Writable]]: true
[[Enumerable]]: true
[[Configurable]]: true

```
const object1 = {};
object1.x = 5;
const object2 = {};
object2.y = 6;
```

```
a = {};
a.x = 5;
```


Shape
(empty)
'x'
Shape
(x)
'x'
Property information
Offset: 0
[[Writable]]: true
[[Enumerable]]: true
[[Configurable]]: true
JSObject a
5

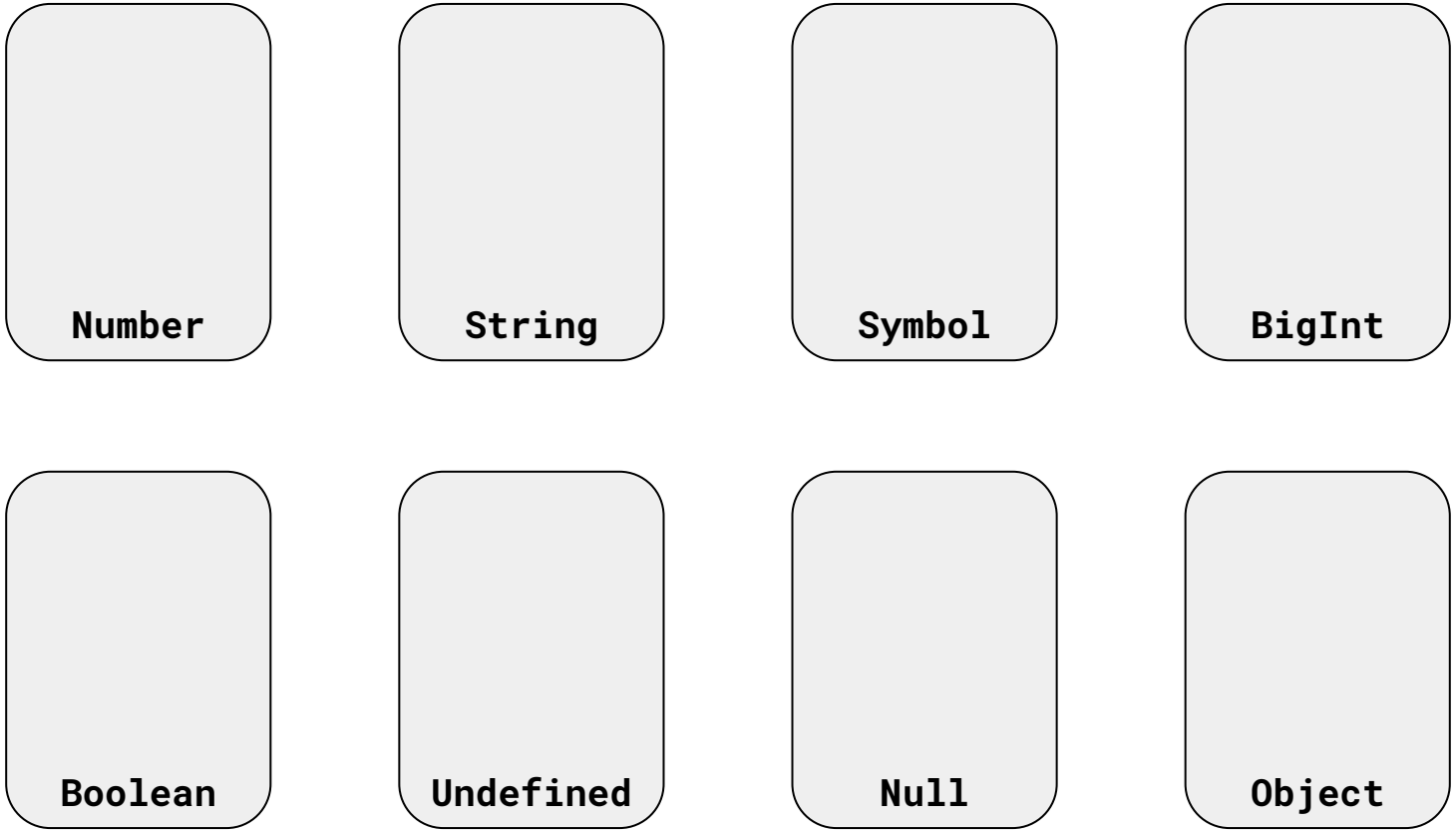
```
a = {};
a.x = 5;
b = {};
b.y = 6;
```


Shape
(empty)
'x'
'y'
Shape
(x)
Shape
(y)
'x'
'y'
Property information
Offset: 0
[[Writable]]: true
[[Enumerable]]: true
[[Configurable]]: true
Property information
Offset: 0
[[Writable]]: true
[[Enumerable]]: true
[[Configurable]]: true
JSObject a
JSObject b
5
6

Engines use object *shapes*
as an optimization

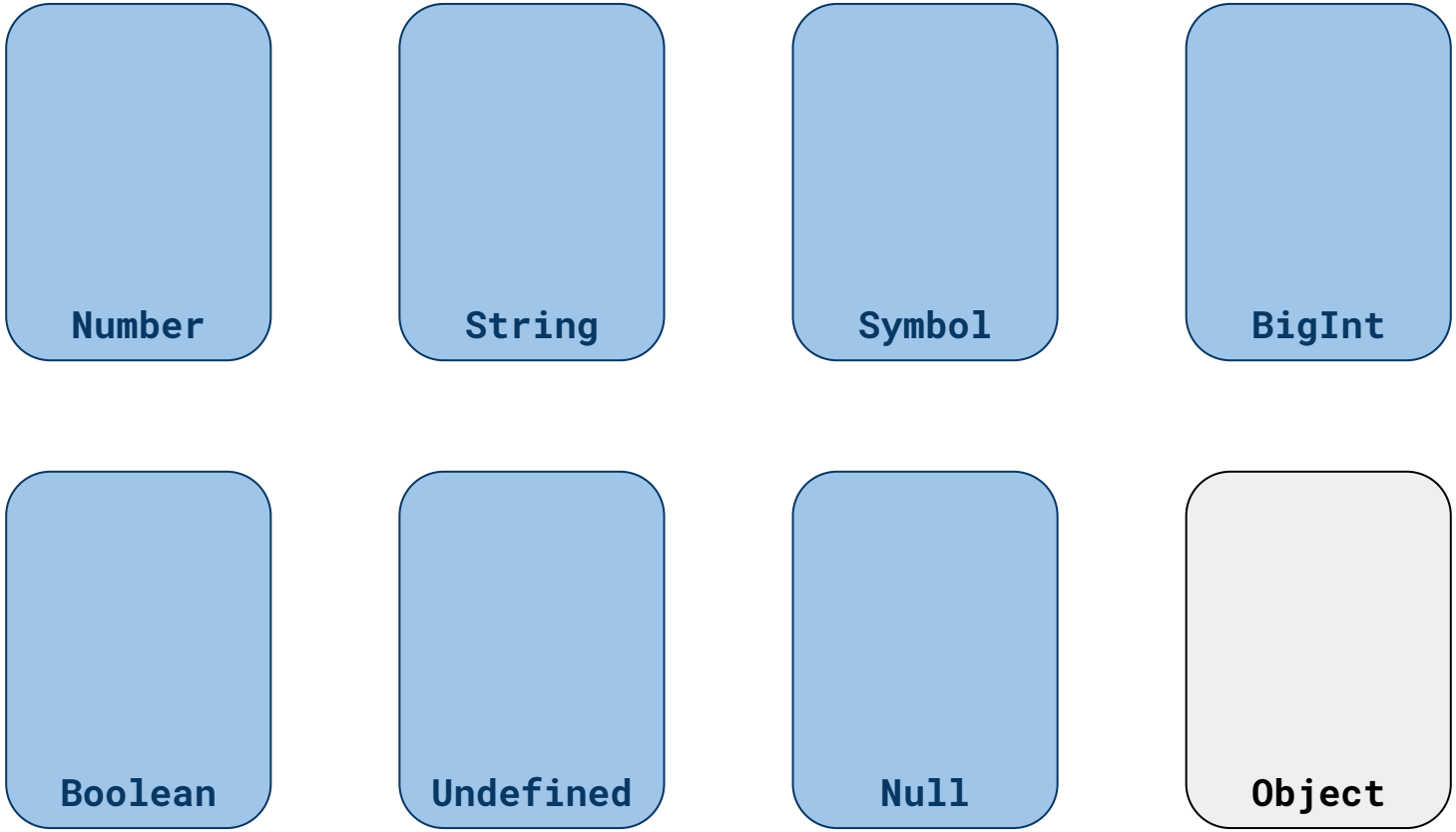
```
const a = 42; typeof a;
// → 'number'
const b = 4.2; typeof b;
// → 'number'
const c = '42'; typeof c;
// → 'string'
const d = { x: 42 }; typeof d;
// → 'object'
```

@mathias && @bmeurer

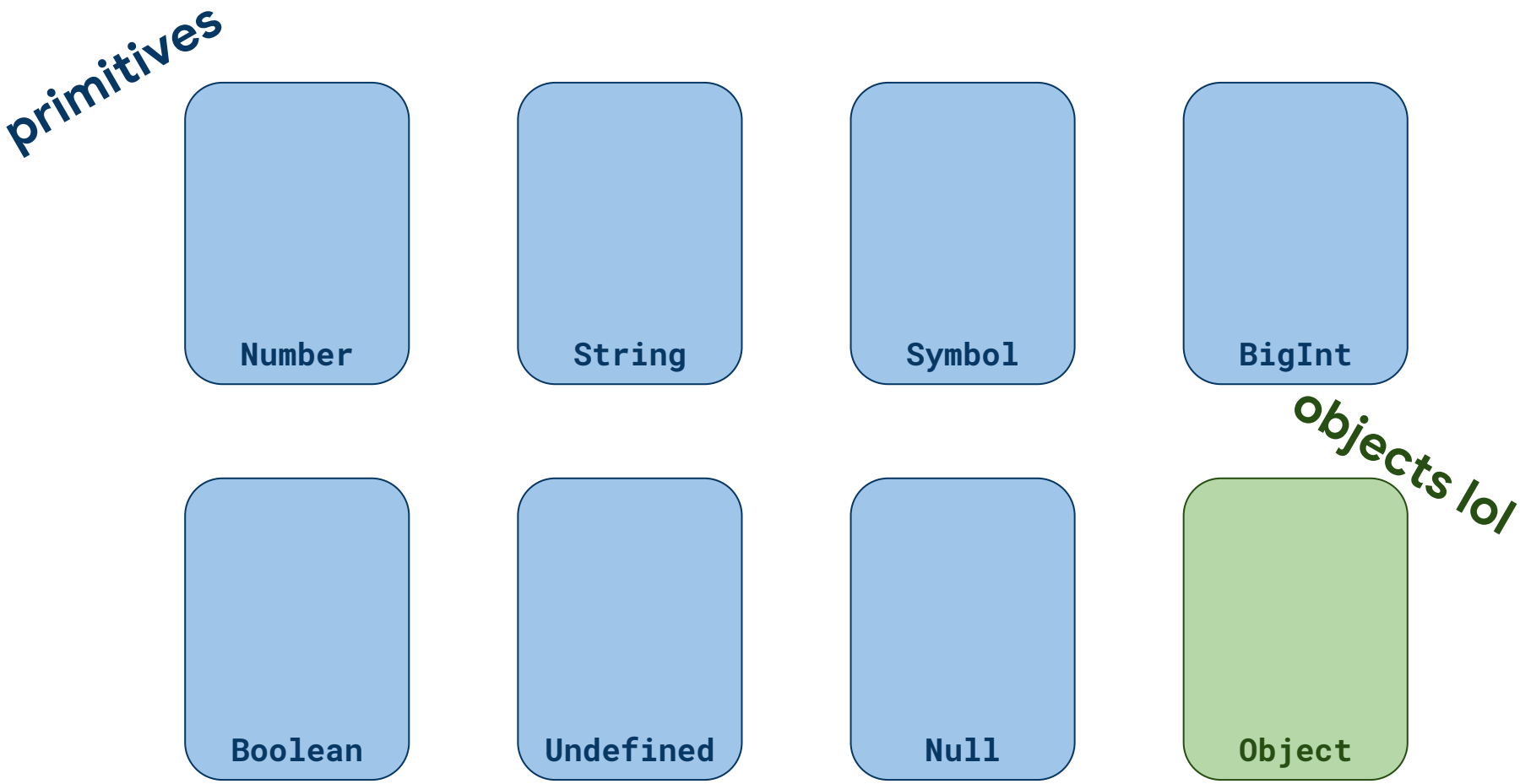


@mathias && @bmeurer

primitives



@mathias && @bmeurer



@mathias && @bmeurer

```
typeof null
```

```
typeof null === 'object'
```

primitives

objects

regular values

primitives

objects

regular values

```
'#agentconf'
```

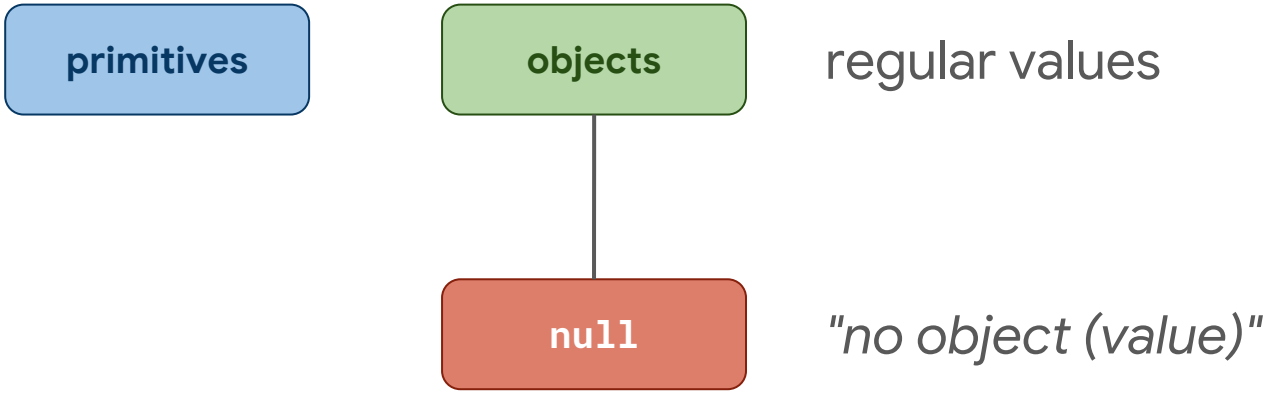
primitives

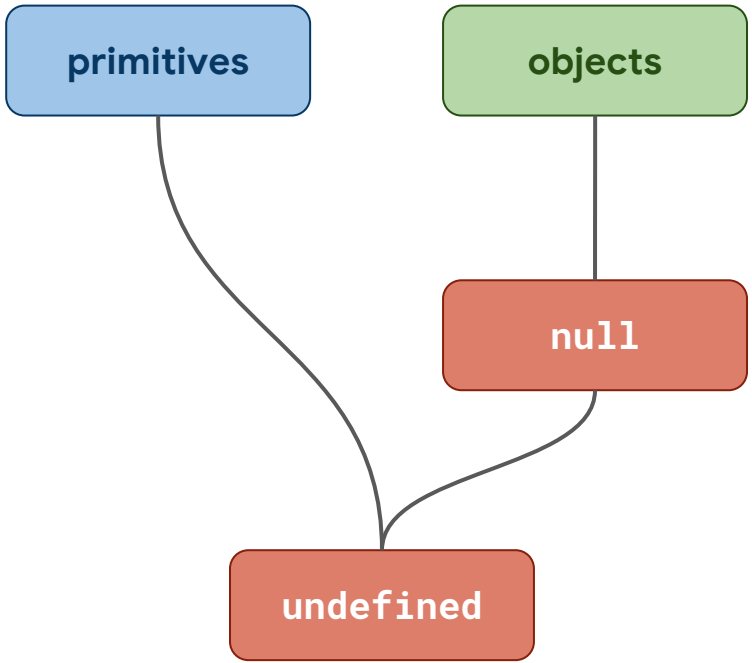
objects

regular values

```
'#agentconf'
```

```
new String('#agentconf')
```

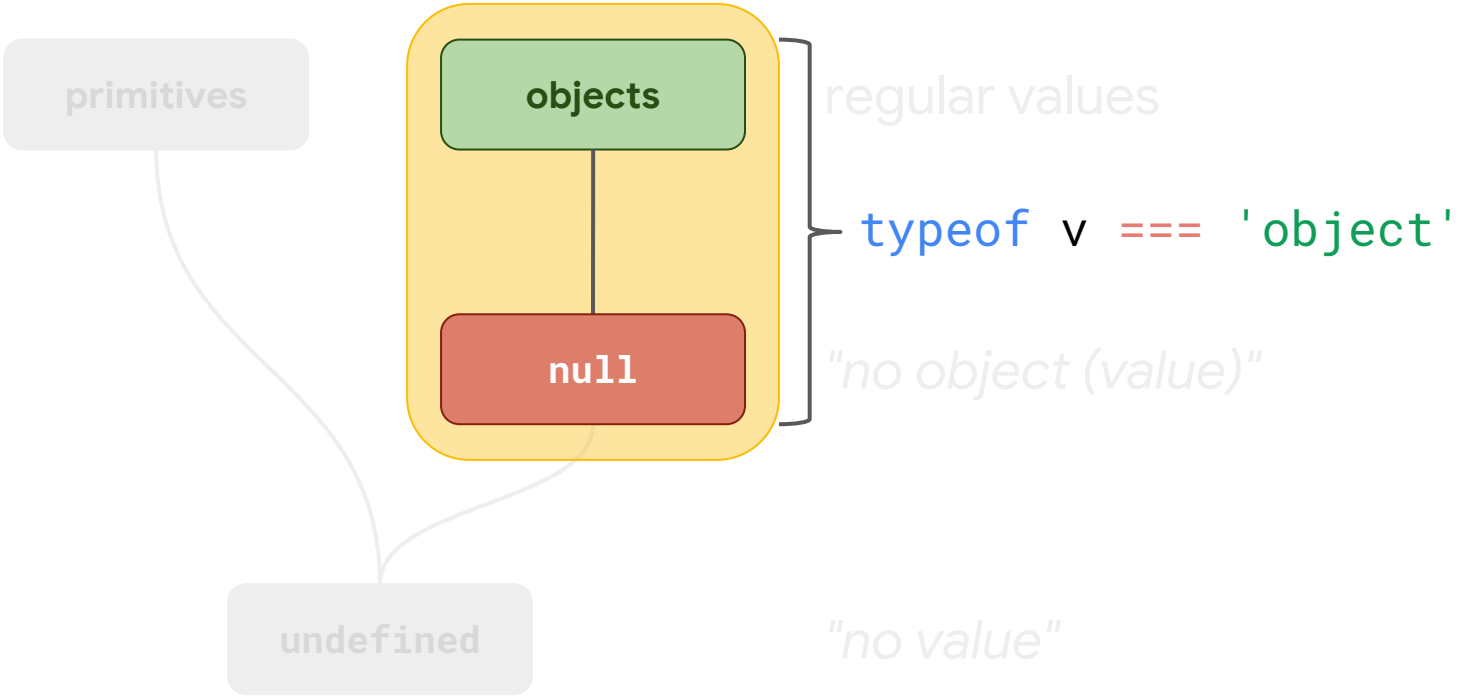

primitives
objects
regular values
null
"no object (value)"



regular values

"no object (value)"

"no value"


primitives
objects
null
undefined
regular values
typeof v === 'object'
"no object (value)"
"no value"

42

```
typeof 42 === 'number'
```

42

42

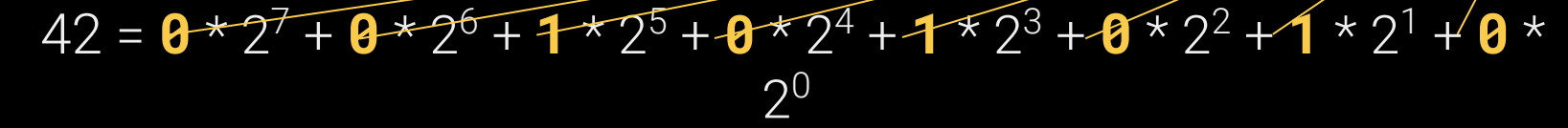
two’s complement 8-bit

`0010 1010`

42

two’s complement 8-bit

0010 1010



42

two's complement 8-bit `0010 1010`

two’s complement 32-bit `0000 0000 0000 0000 0000 0000 0010 1010`

42

two’s complement 8-bit	`0010 1010`
two’s complement 32-bit	`0000 0000 0000 0000 0000 0000 0010 1010`
packed binary-coded decimal (BCD)	`0100 0010`

42

two’s complement 8-bit	`0010 1010`
two’s complement 32-bit	`0000 0000 0000 0000 0000 0000 0010 1010`
packed binary-coded decimal (BCD)	`0100 0010`
32-bit IEEE-754 floating-point	`0100 0010 0010 1000 0000 0000 0000 0000`

42

two's complement 8-bit	0010 1010
two's complement 32-bit	0000 0000 0000 0000 0000 0000 0010 1010
packed binary-coded decimal (BCD)	0100 0010
32-bit IEEE-754 floating-point	0100 0010 0010 1000 0000 0000 0000 0000
64-bit IEEE-754 floating-point	0100 0000 0100 0101 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

@mathias && @bmeurer

42

two's complement 8-bit	0010 1010
two's complement 32-bit	0000 0000 0000 0000 0000 0000 0010 1010
packed binary-coded decimal (BCD)	0100 0010
32-bit IEEE-754 floating-point	0100 0010 0010 1000 0000 0000 0000 0000

ECMAScript standardizes Number as Float64

64-bit IEEE-754 floating-point	0100 0000 0100 0101 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

@mathias && @bmeurer

42

```
array[42]
```

```
array[0]
array[42]
array[2**32-2]
```


32-bit two's complement ftw!
array[42]

array[42]

@mathias && @bmeurer

```
for (let i = 0; i < 1000; ++i) {
  // fast
}

for (let i = 0.1; i < 1000.1; ++i) {
  // slow
}
```

@mathias && @bmeurer

```
const remainder = value % divisor;
```

Integer operations generally execute much faster than floating-point operations

```
2**53 === 2**53+1
```

```
2**53 === 2**53+1
 -1*0 === -0
```

```
2**53 === 2**53+1
 -1*0 === -0
  1/0 === Infinity
```

```
2**53 === 2**53+1
 -1*0 === -0
  1/0 === Infinity
 -1/0 === -Infinity
```

```
2**53 === 2**53+1
 -1*0 === -0
  1/0 === Infinity
 -1/0 === -Infinity
  0/0 === NaN
```

@mathias && @bmeurer

```
-(2**30) // Smi
     -42 // Smi

       0 // Smi

      42 // Smi
 2**30-1 // Smi
```

```
-Infinity // HeapNumber
-(2**30)-1 // HeapNumber

-0 // HeapNumber

4.2 // HeapNumber

2**30 // HeapNumber
Infinity // HeapNumber
NaN // HeapNumber
```

@mathias && @bmeurer

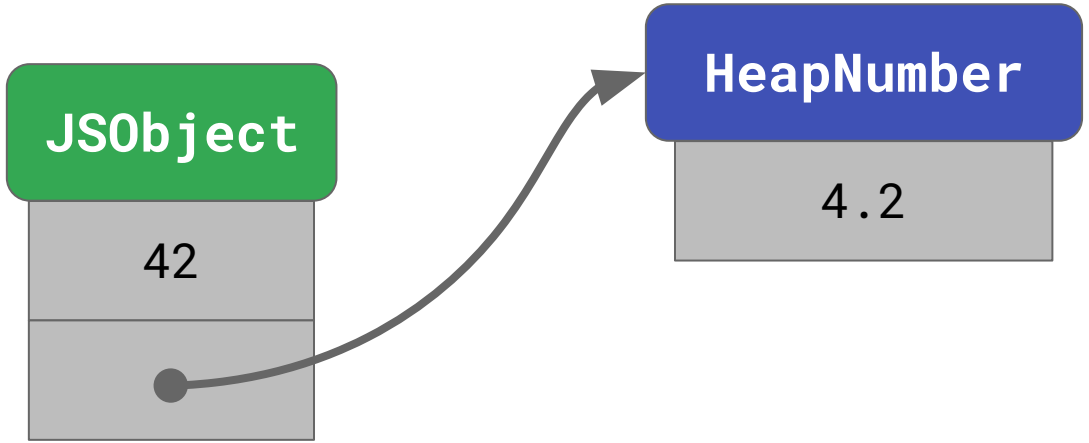
```
-Infinity // HeapNumber
-(2**30)-1 // HeapNumber
  -(2**30) // Smi
        -42 // Smi
         -0 // HeapNumber
          0 // Smi
        4.2 // HeapNumber
         42 // Smi
   2**30-1 // Smi
     2**30 // HeapNumber
  Infinity // HeapNumber
        NaN // HeapNumber
```

@mathias && @bmeurer

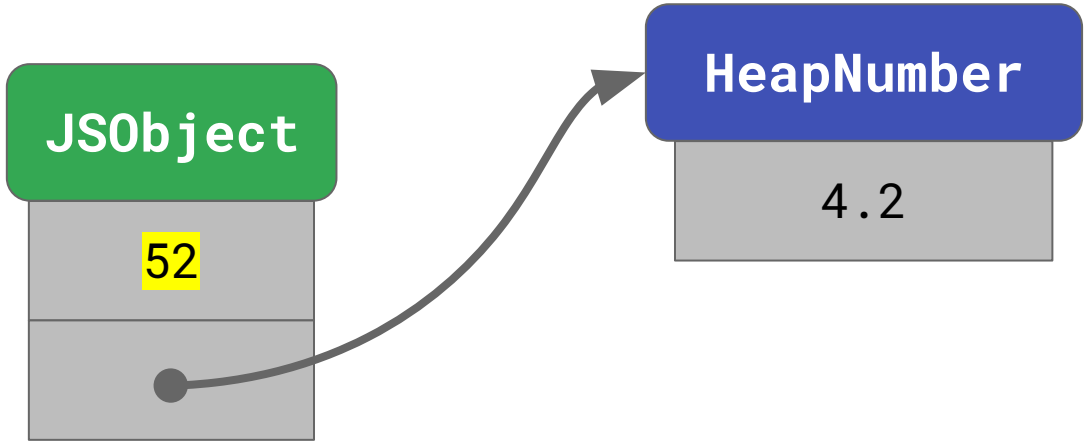
Engines choose different (number) representations as an optimization

```
o = {
  x: 42,
  y: 4.2,
};
```

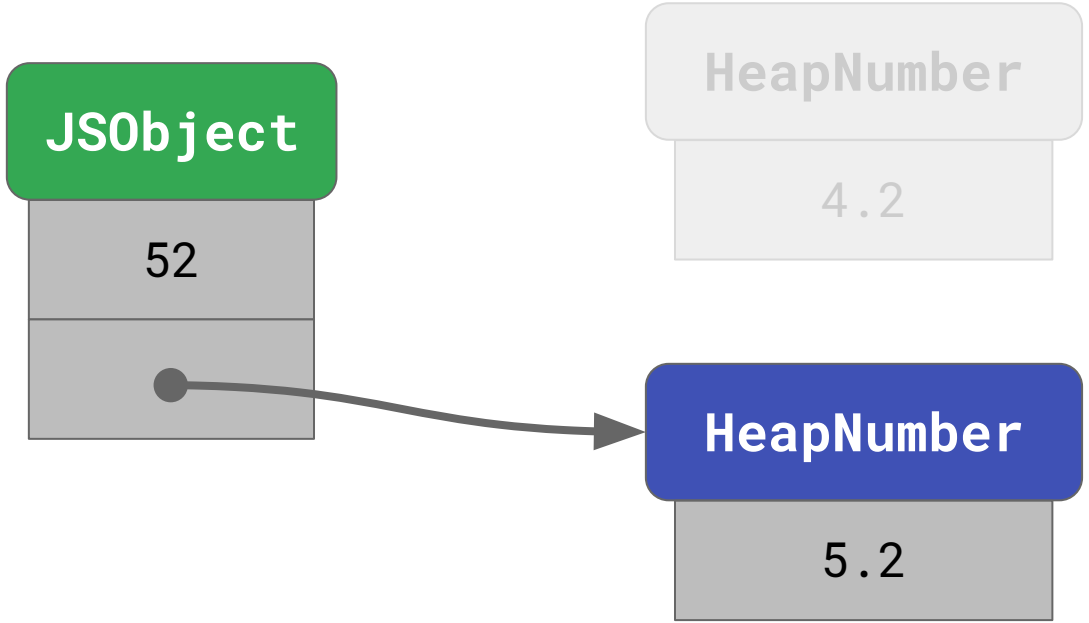
```
o = {
  x: 42,
  y: 4.2,
};
```



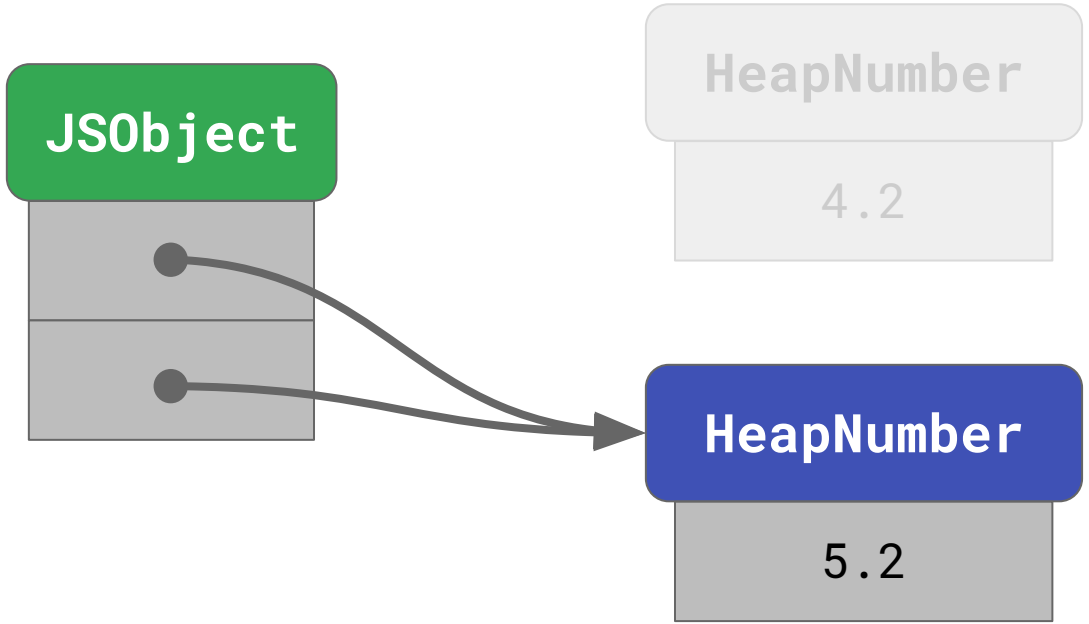
```
o = {
  x: 42,
  y: 4.2,
};

o.x += 10;
```



```
o = {
  x: 42,
  y: 4.2,
};

o.x += 10;
o.y += 1;
```

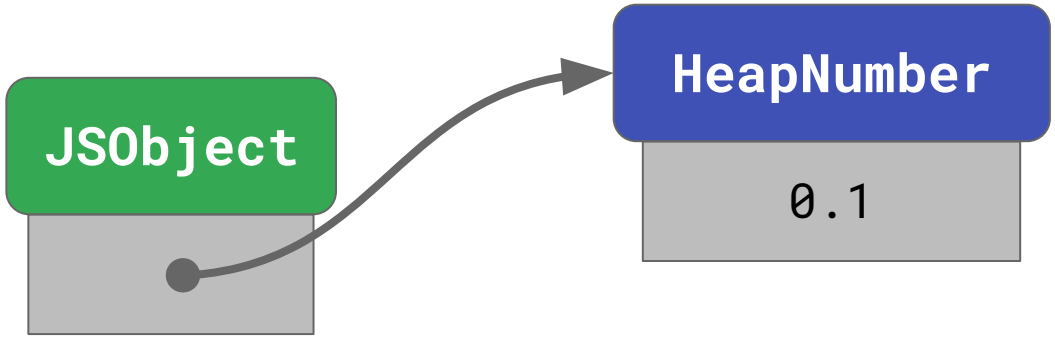


```
o = {
  x: 42,
  y: 4.2,
};

o.x += 10;
o.y += 1;
o.x = o.y;
```

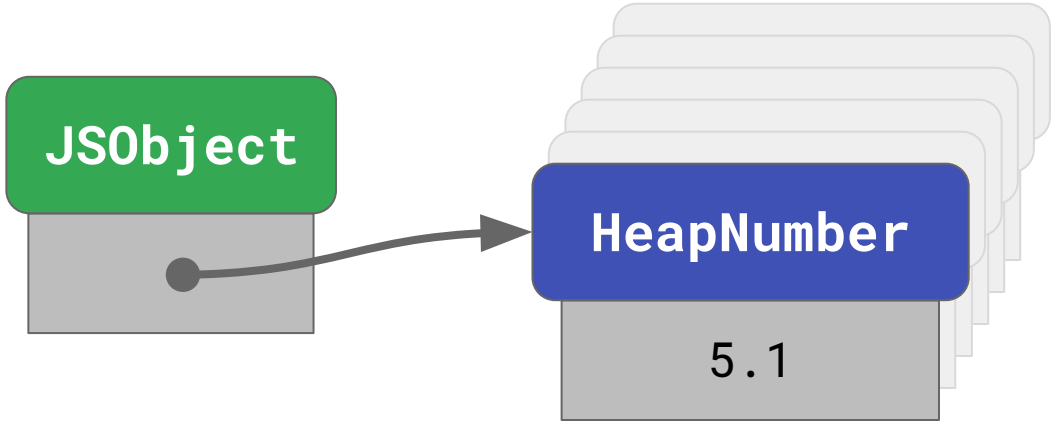


```
const object = { x: 0.1 };

for (let i = 0; i < 5; ++i) {
  object.x += 1;
}
```

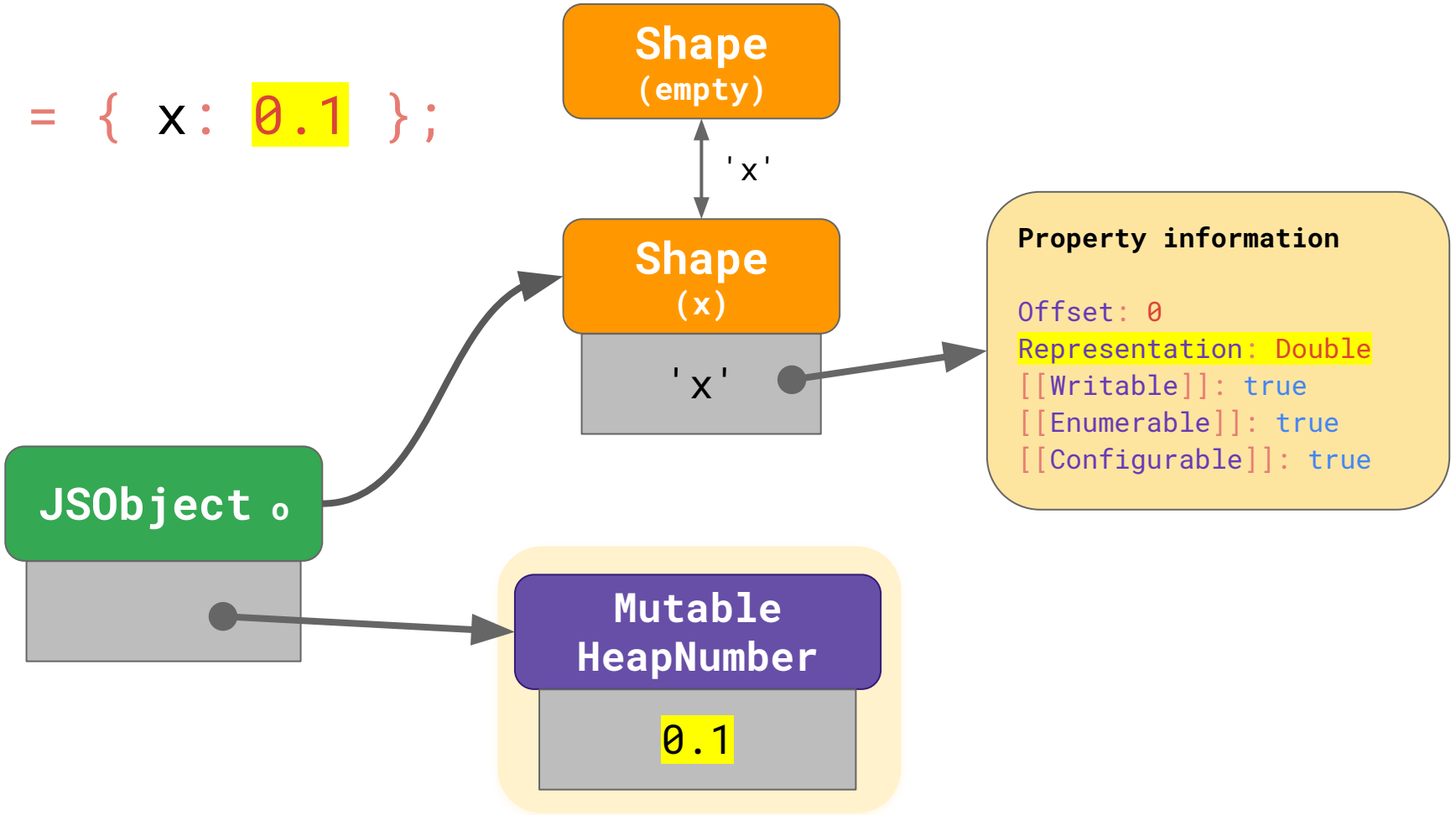
```
o = { x: 0.1 };
```



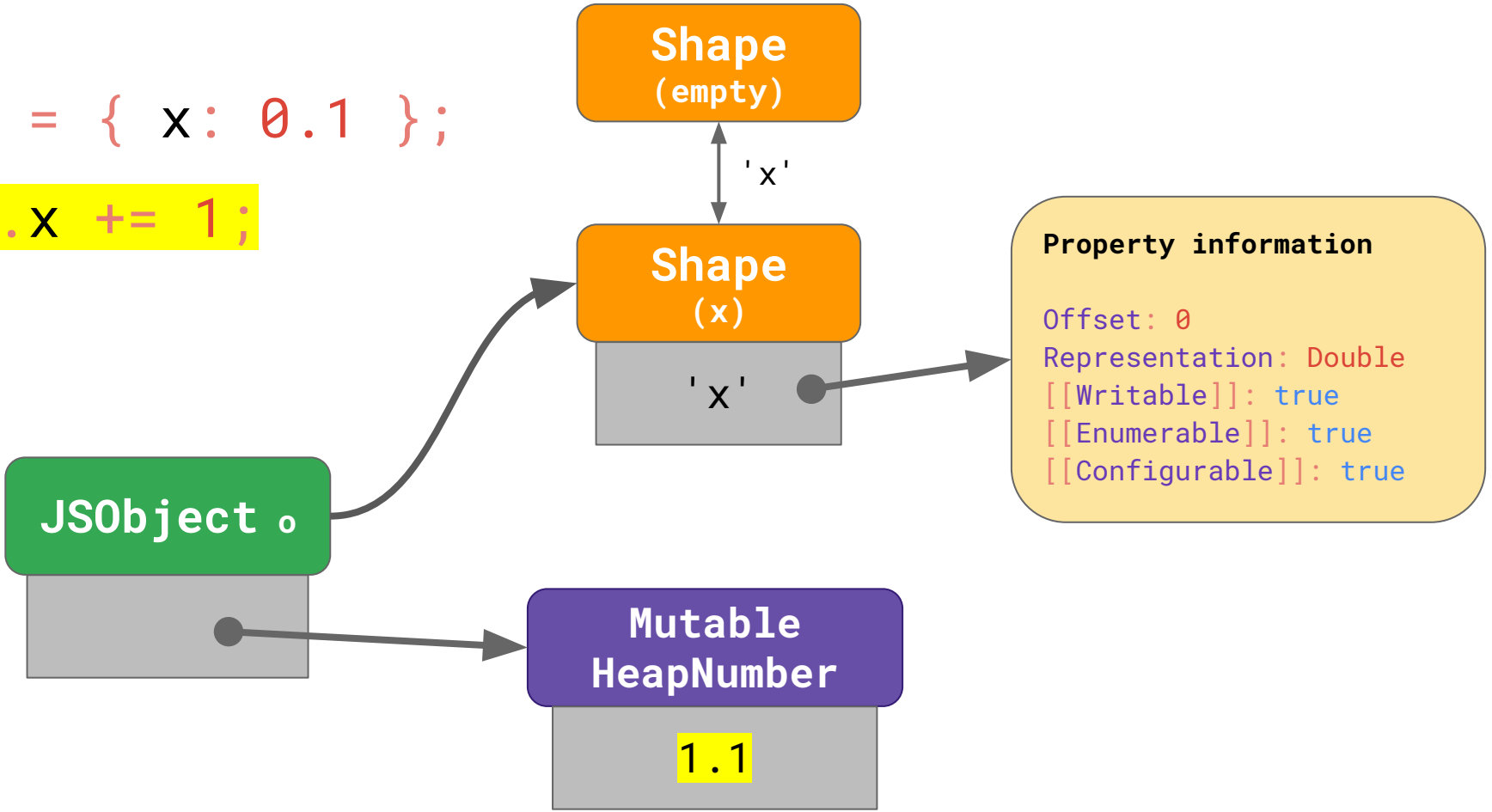
```
o = { x: 0.1 };

for (i = 0; i < 5; ++i) {
  o.x += 1;
}
```

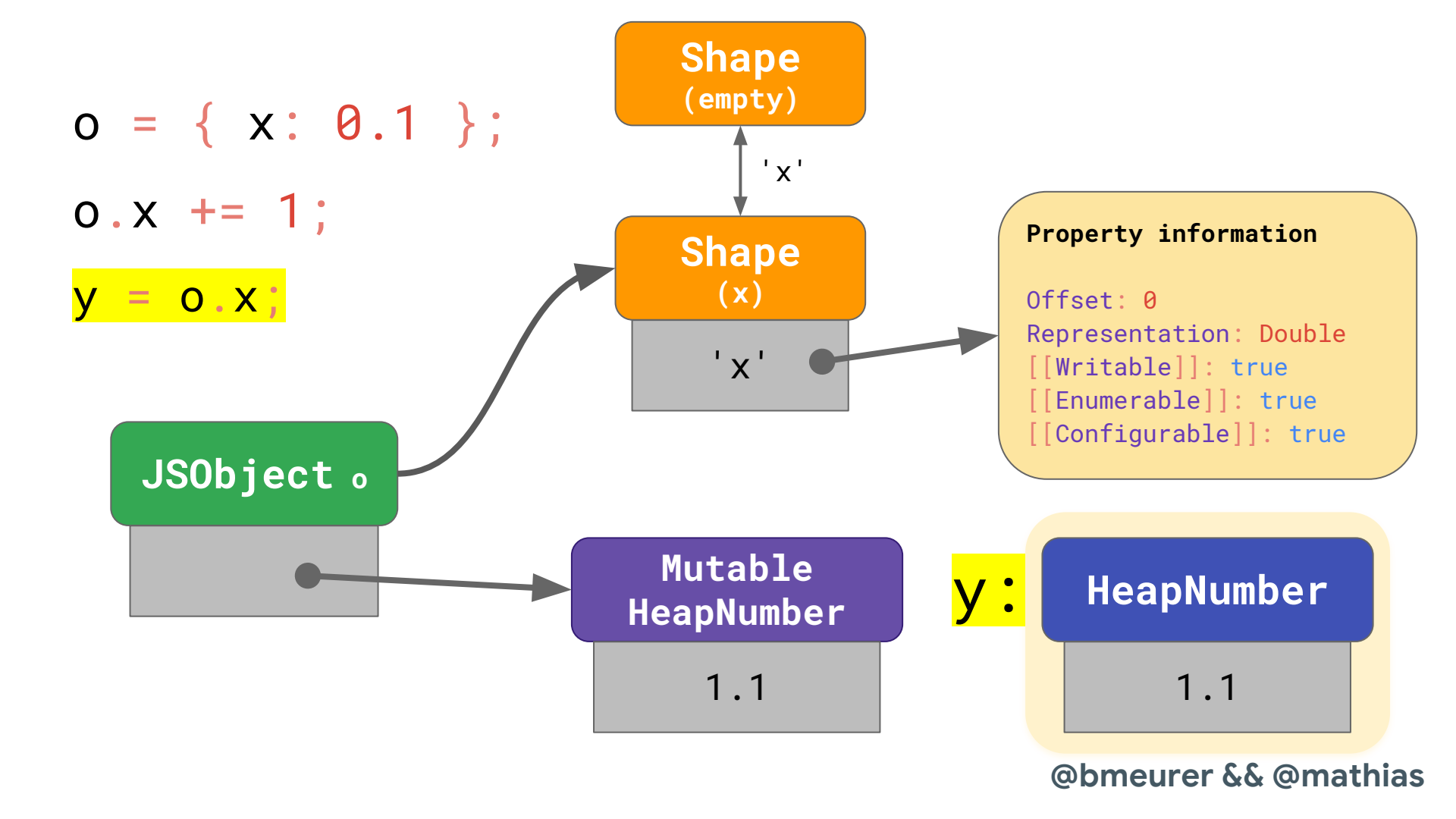


```
o = { x: 0.1 };
```

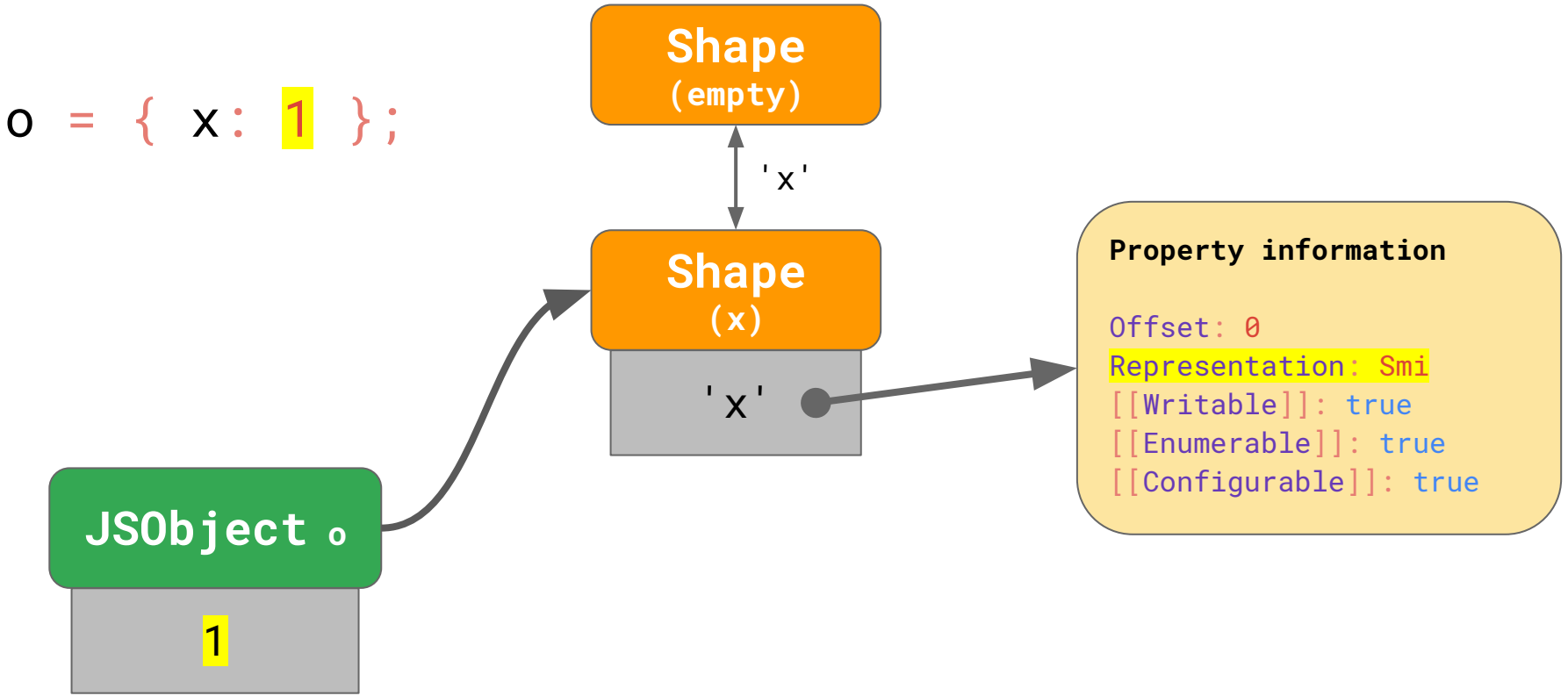

Shape
(empty)
'x'
Shape
(x)
'x'
Property information
Offset: 0
Representation: Double
[[Writable]]: true
[[Enumerable]]: true
[[Configurable]]: true
JSObject o
Mutable
HeapNumber
0.1

```
o = { x: 0.1 };

o.x += 1;
```

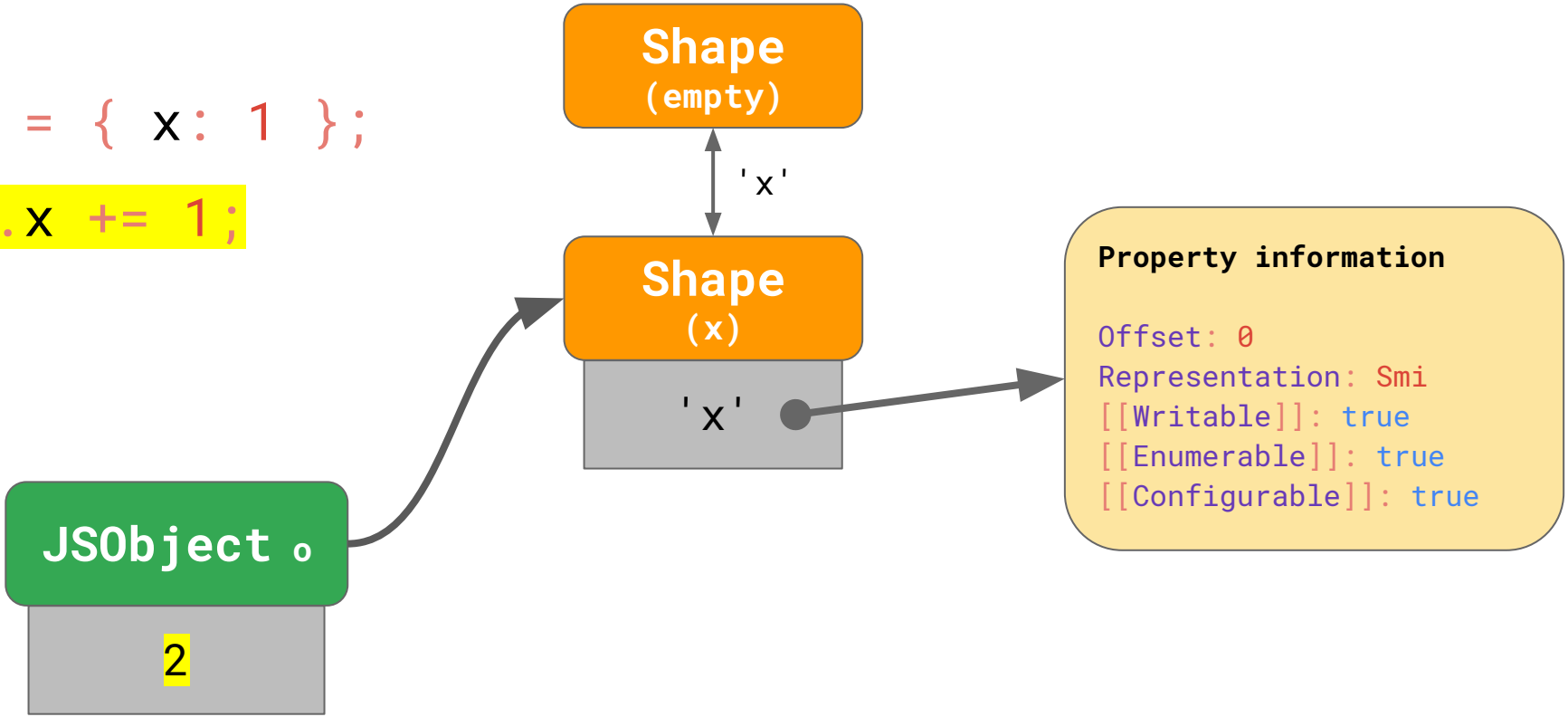

Shape
(empty)
'x'
Shape
(x)
'x'
Property information
Offset: 0
Representation: Double
[[Writable]]: true
[[Enumerable]]: true
[[Configurable]]: true
JSObject o
Mutable
HeapNumber
1.1


o = { x: 0.1 };
o.x += 1;
y = o.x;
Shape
(empty)
'x'
Shape
(x)
'x'
Property information
Offset: 0
Representation: Double
[[Writable]]: true
[[Enumerable]]: true
[[Configurable]]: true
JSObject o
Mutable
HeapNumber
1.1
y:
HeapNumber
1.1
@bmeurer && @mathias

```
const object = { x: 1 };
// → no “boxing” for x in object

object.x += 1;
// → update the value of x inside object
```


o = { x: 1 };
Shape
(empty)
'x'
Shape
(x)
'x'
JSObject o
1
Property information
Offset: 0
Representation: Smi
[[Writable]]: true
[[Enumerable]]: true
[[Configurable]]: true

```
o = { x: 1 };

o.x += 1;
```

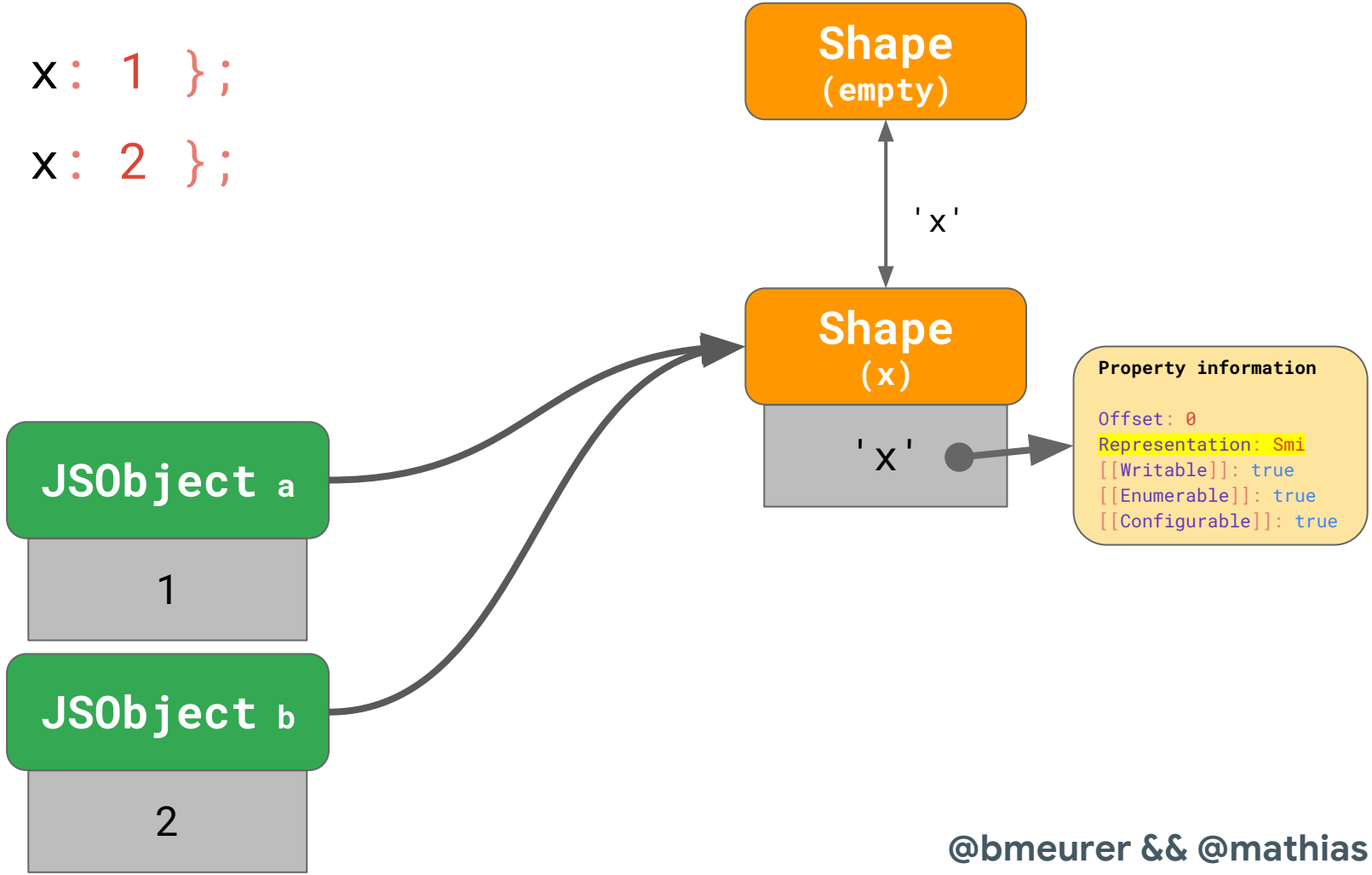

Shape
(empty)
'x'
Shape
(x)
'x'
JSObject o
2
Property information
Offset: 0
Representation: Smi
[[Writable]]: true
[[Enumerable]]: true
[[Configurable]]: true

```
const a = { x: 1 };
const b = { x: 2 };
// → objects have `x` as Smi field now
```

```
const a = { x: 1 };
const b = { x: 2 };
// → objects have `x` as Smi field now

b.x = 0.2;
// → `b.x` is now `Double` representation

y = a.x;
```

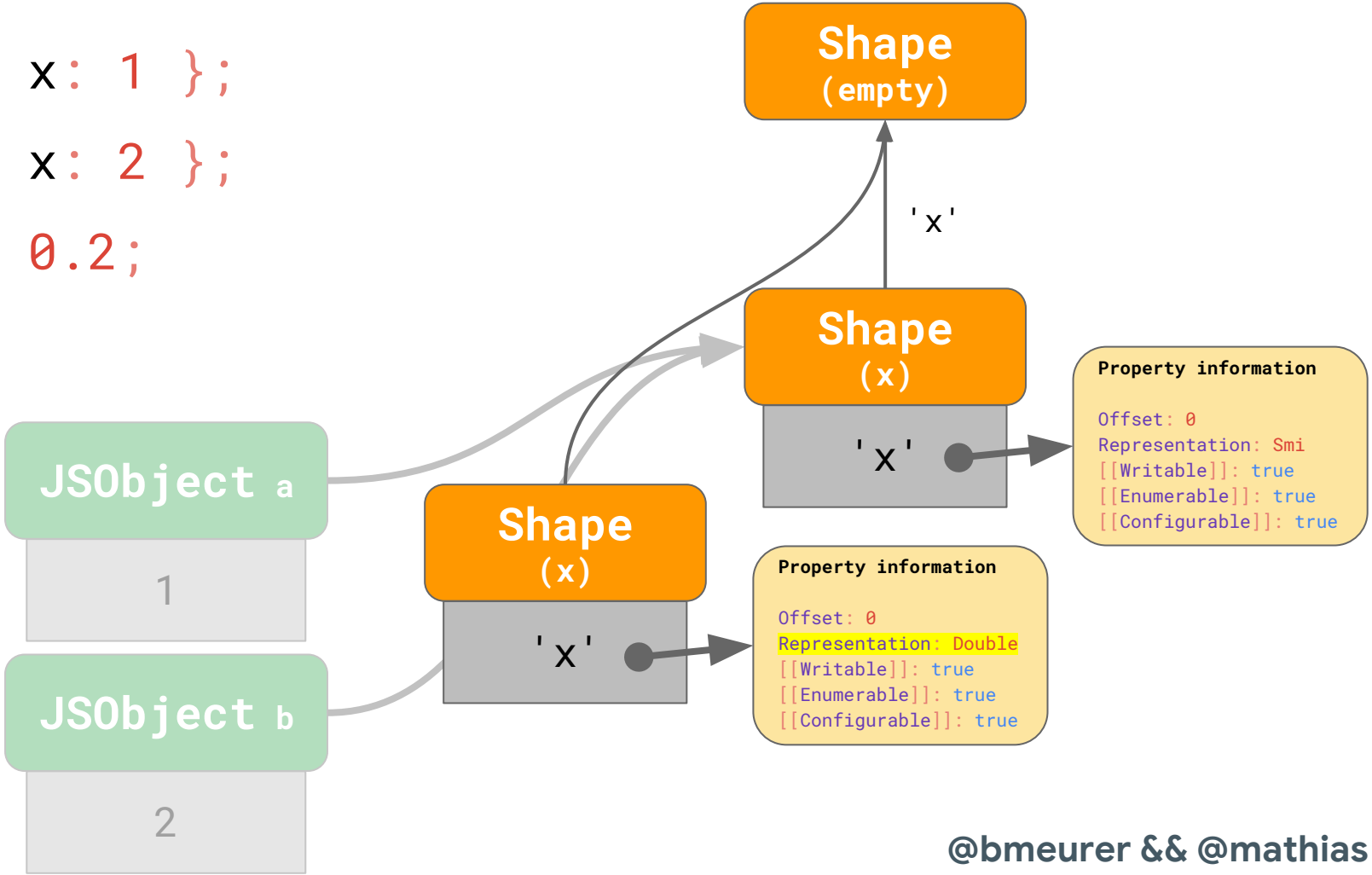
@bmeurer && @mathias

```
a = { x: 1 };
b = { x: 2 };
```



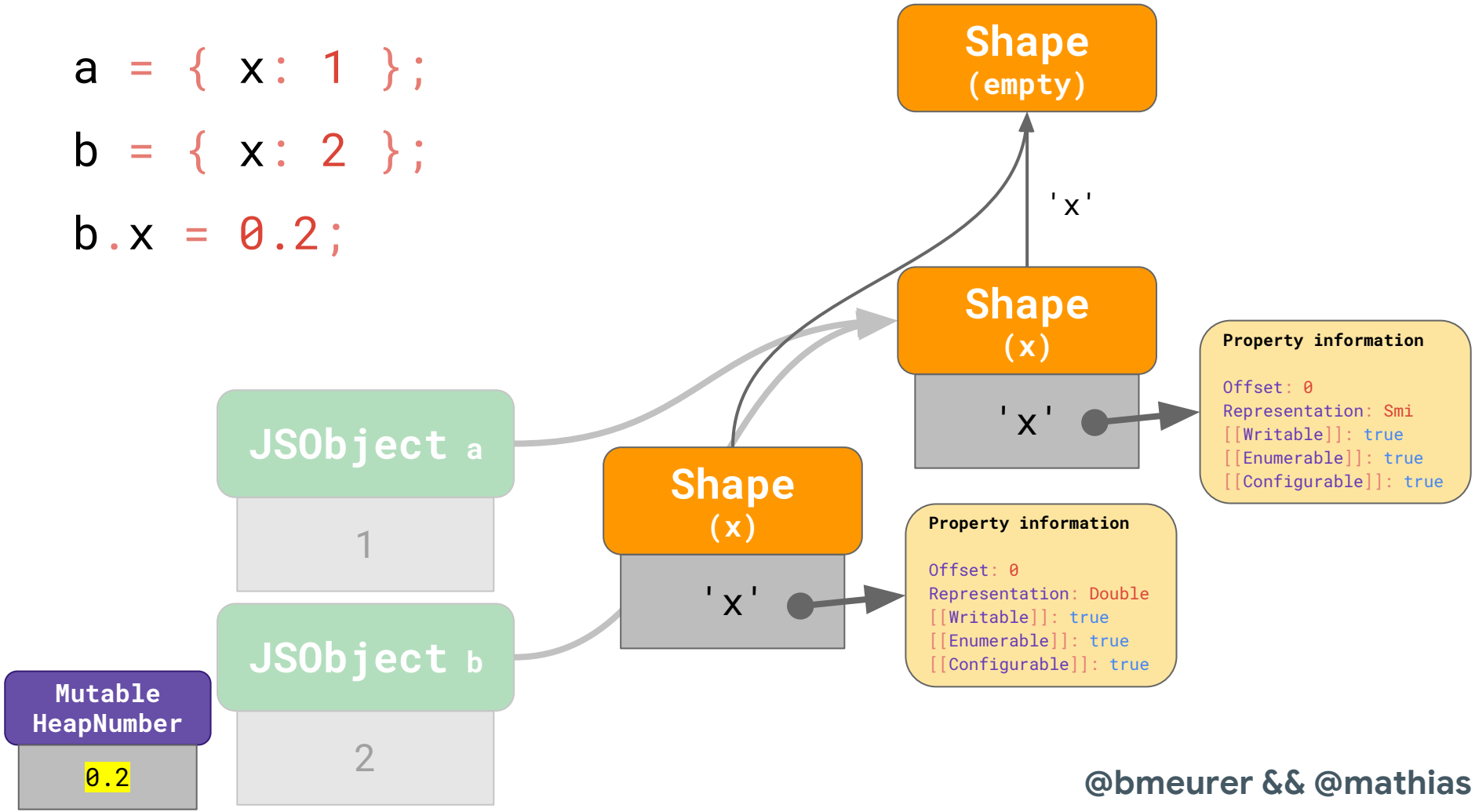
@bmeurer && @mathias

```
a = { x: 1 };
b = { x: 2 };
b.x = 0.2;
```



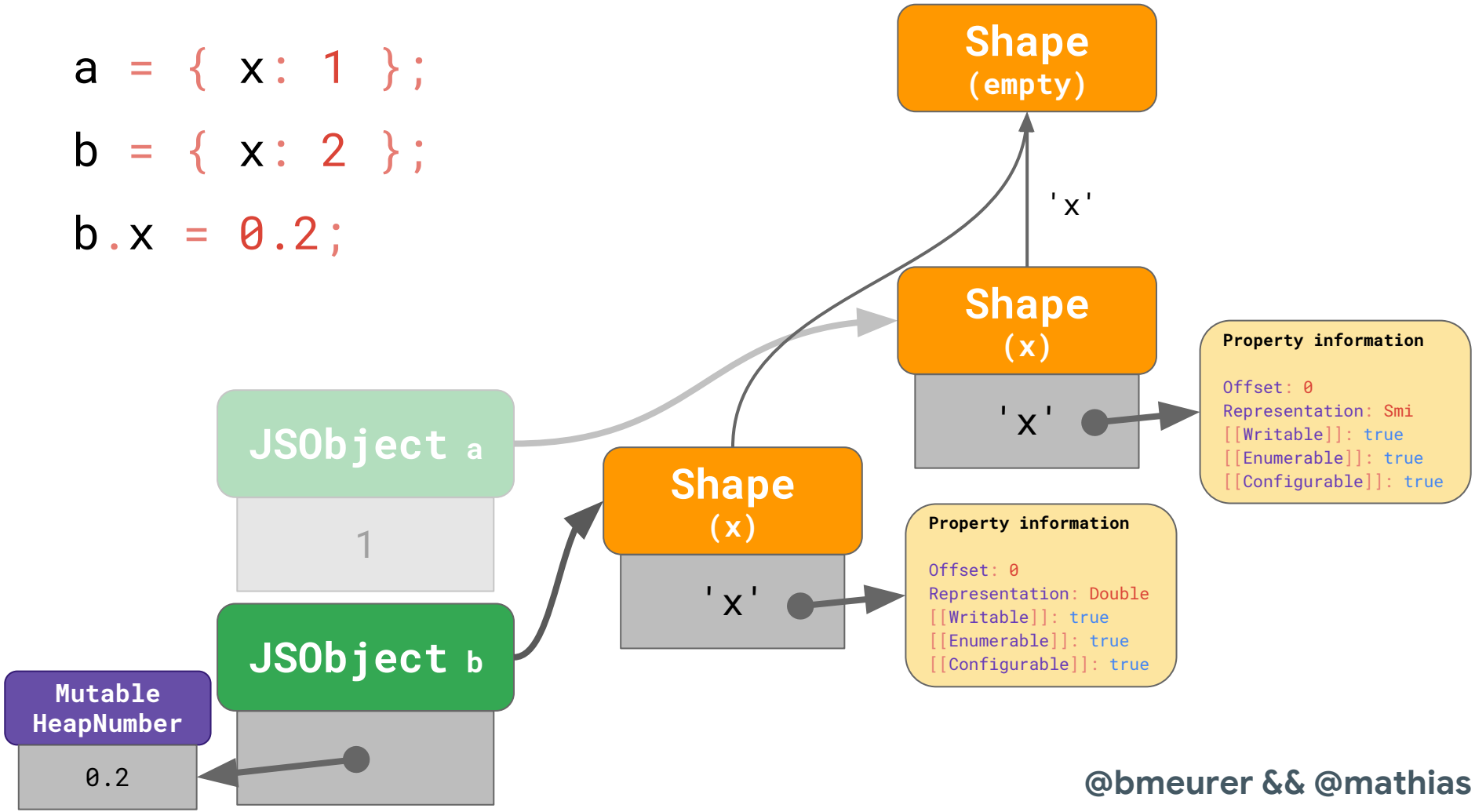
@bmeurer && @mathias

```
a = { x: 1 };
b = { x: 2 };
b.x = 0.2;
```



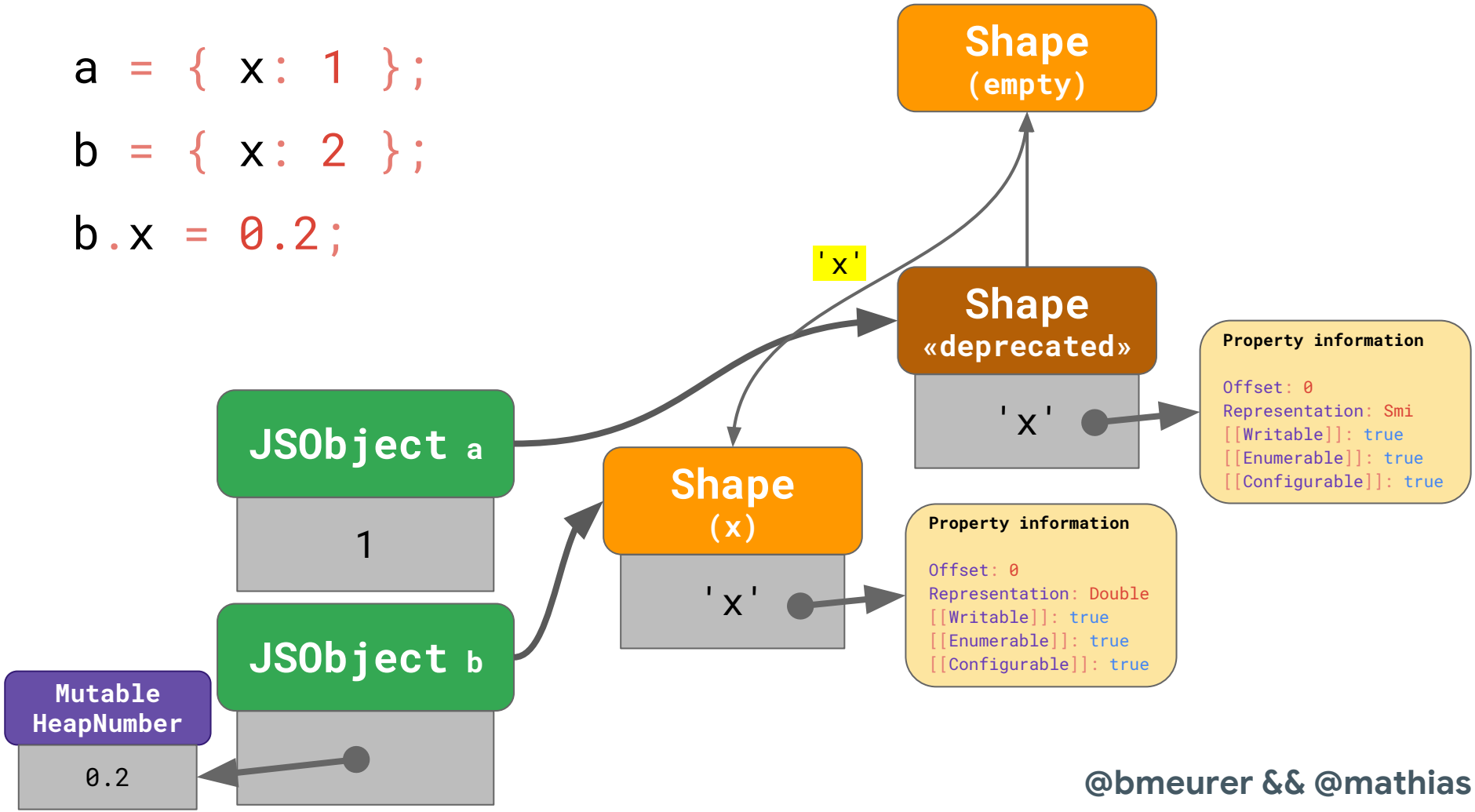
@bmeurer && @mathias

```
a = { x: 1 };
b = { x: 2 };
b.x = 0.2;
```



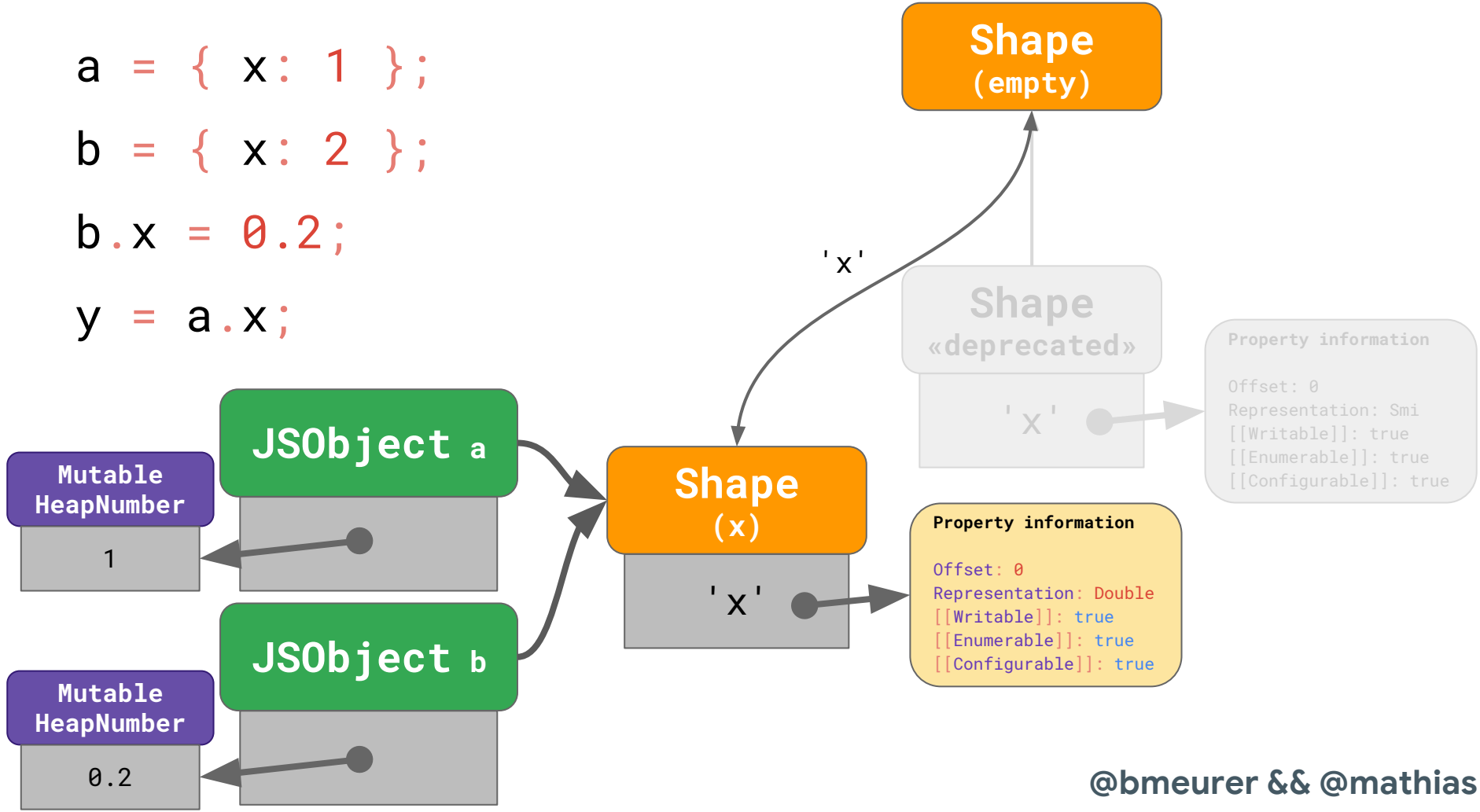
@bmeurer && @mathias

```
a = { x: 1 };
b = { x: 2 };
b.x = 0.2;
```



@bmeurer && @mathias

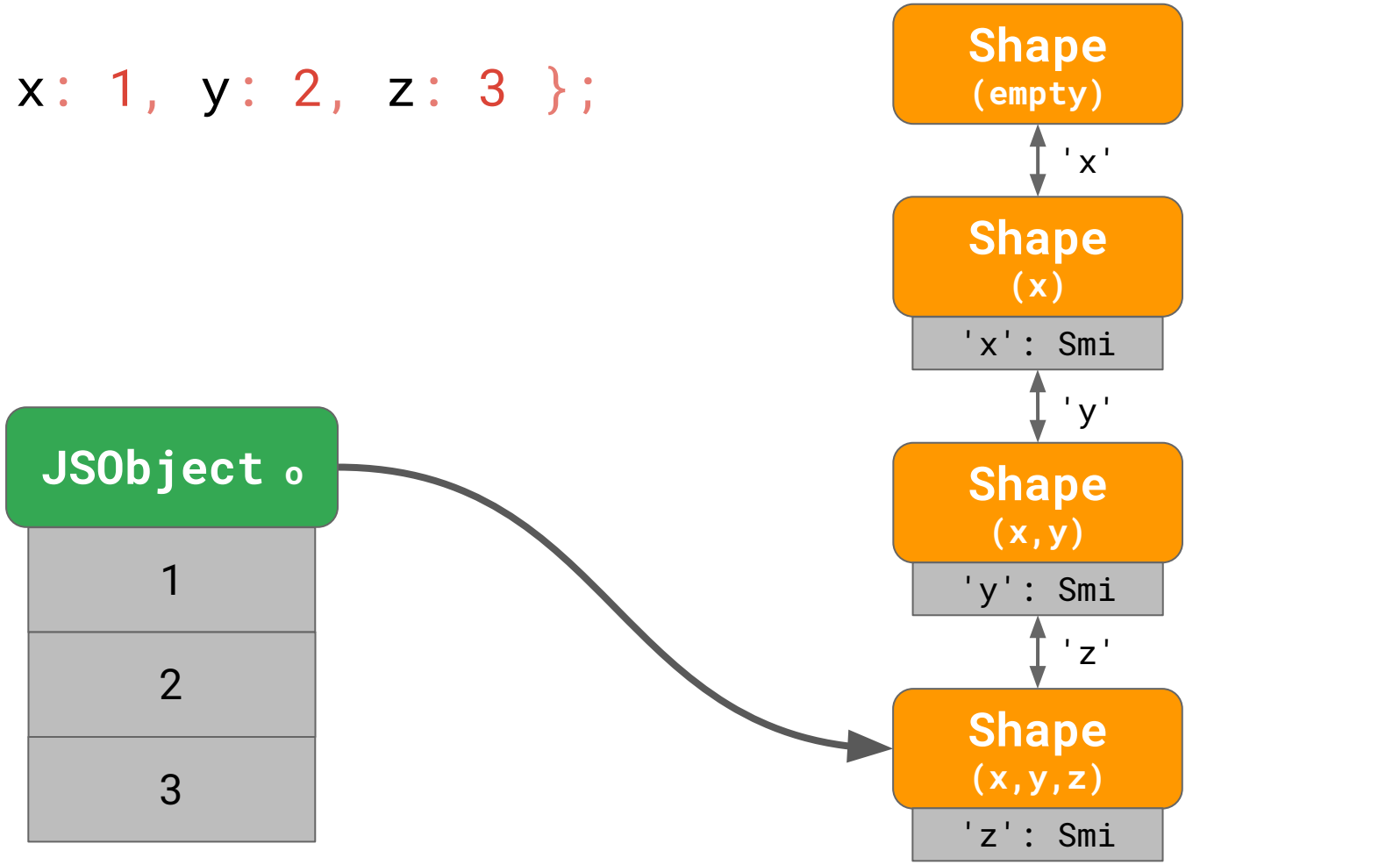
```
a = { x: 1 };
b = { x: 2 };
b.x = 0.2;
y = a.x;
```



@bmeurer && @mathias

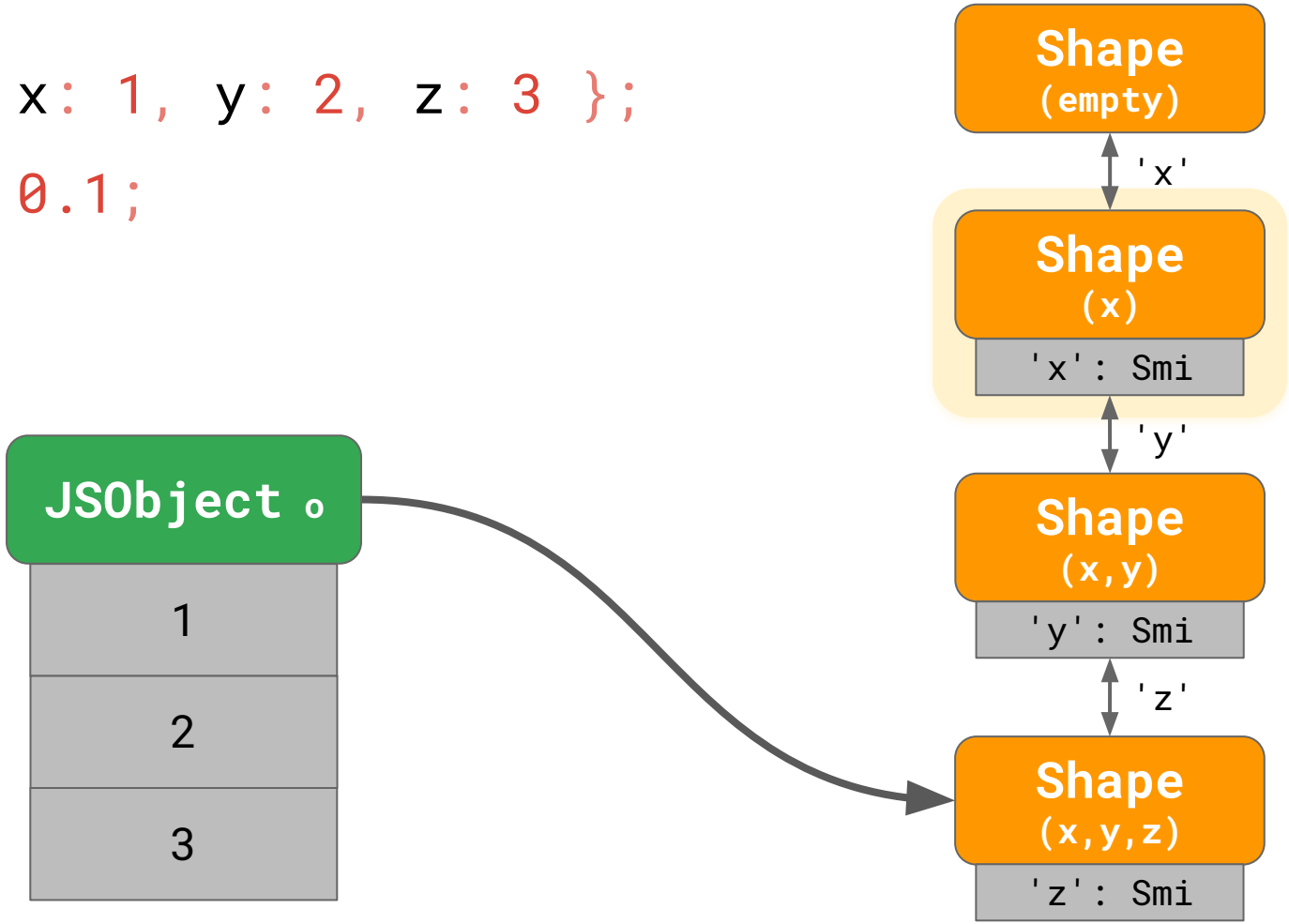
```
const object = {
  x: 1,
  y: 2,
  z: 3,
};

object.y = 0.1;
```

```
o = { x: 1, y: 2, z: 3 };
```



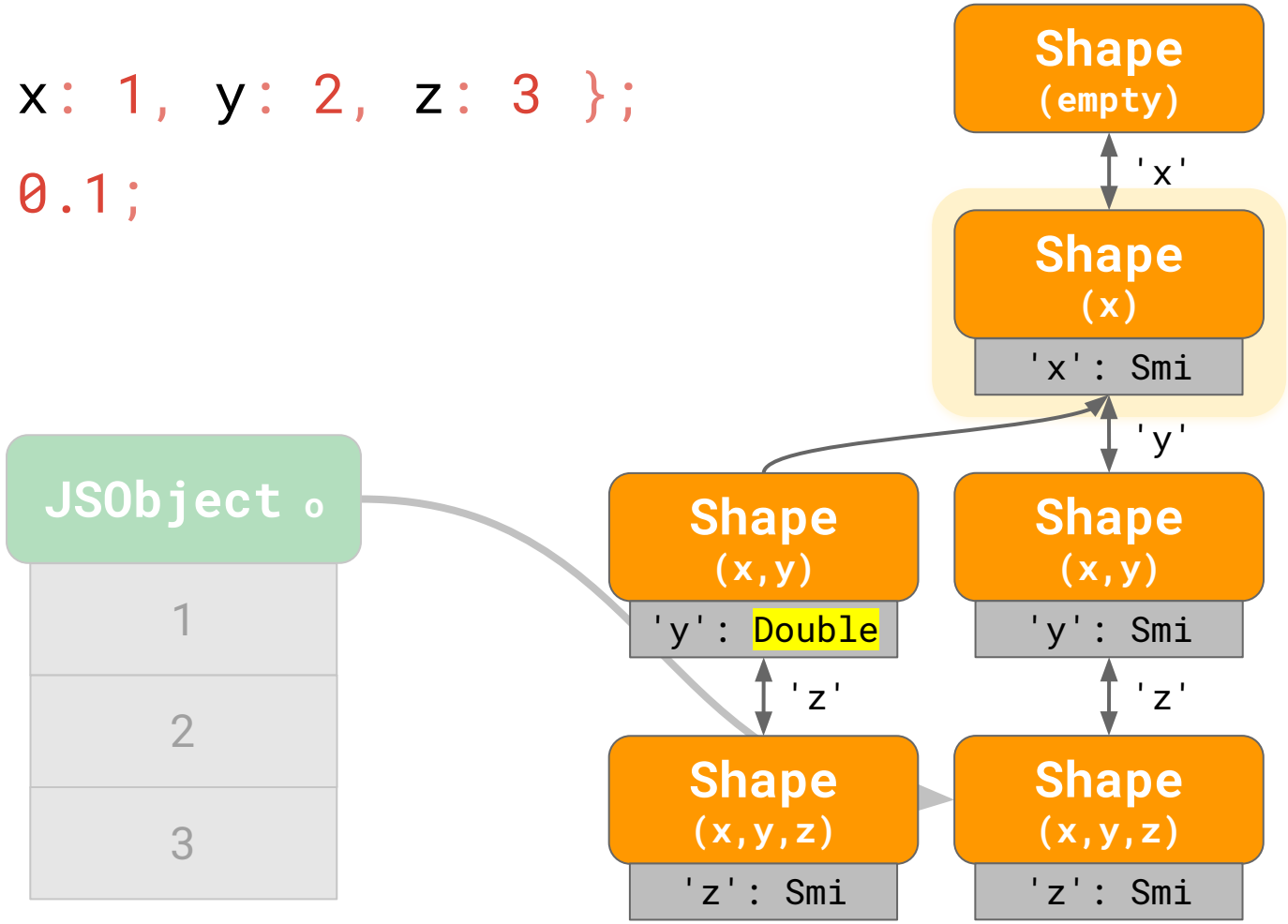
@bmeurer && @mathias

```
o = { x: 1, y: 2, z: 3 };
o.y = 0.1;
```



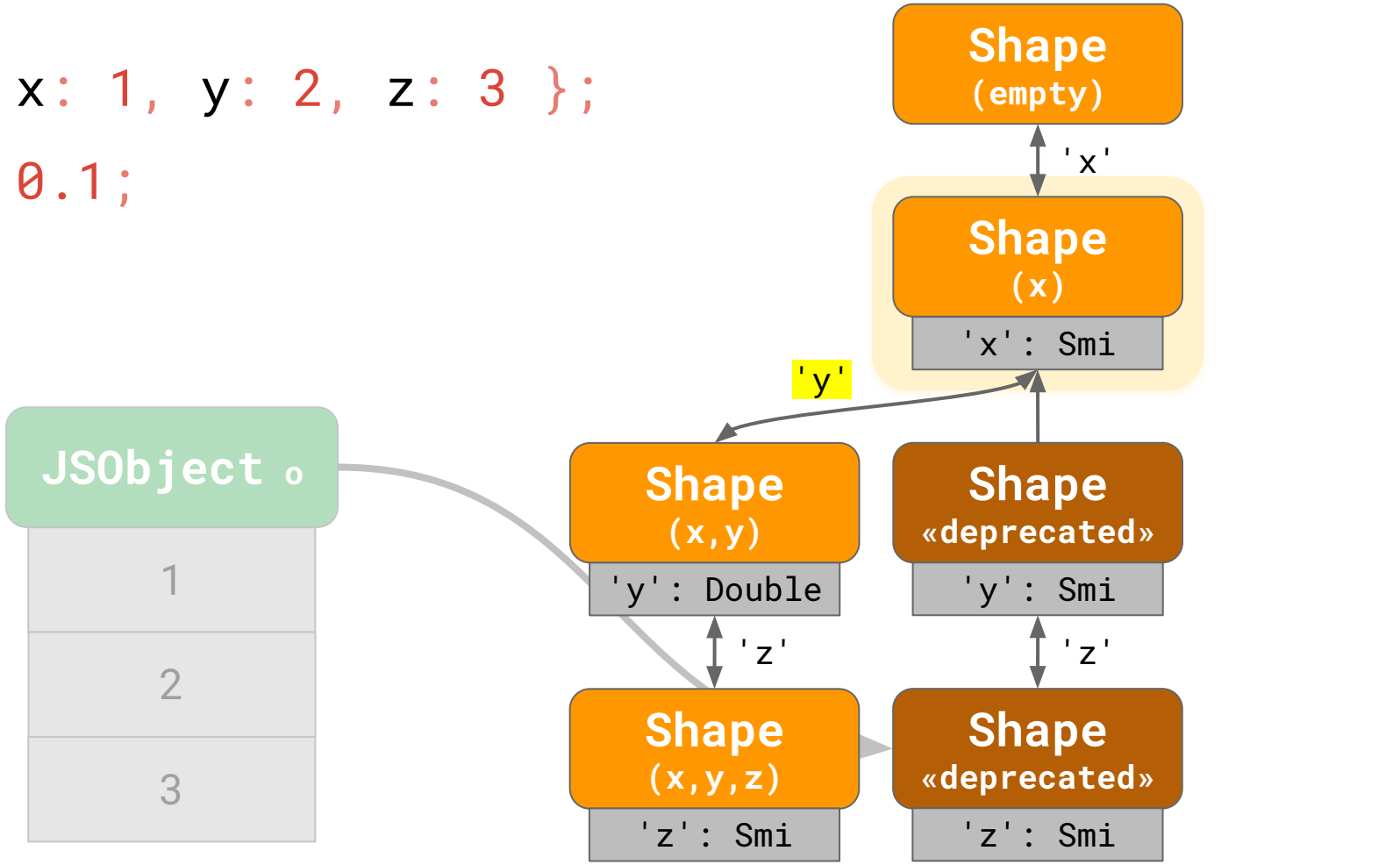
@bmeurer && @mathias

```
o = { x: 1, y: 2, z: 3 };
o.y = 0.1;
```



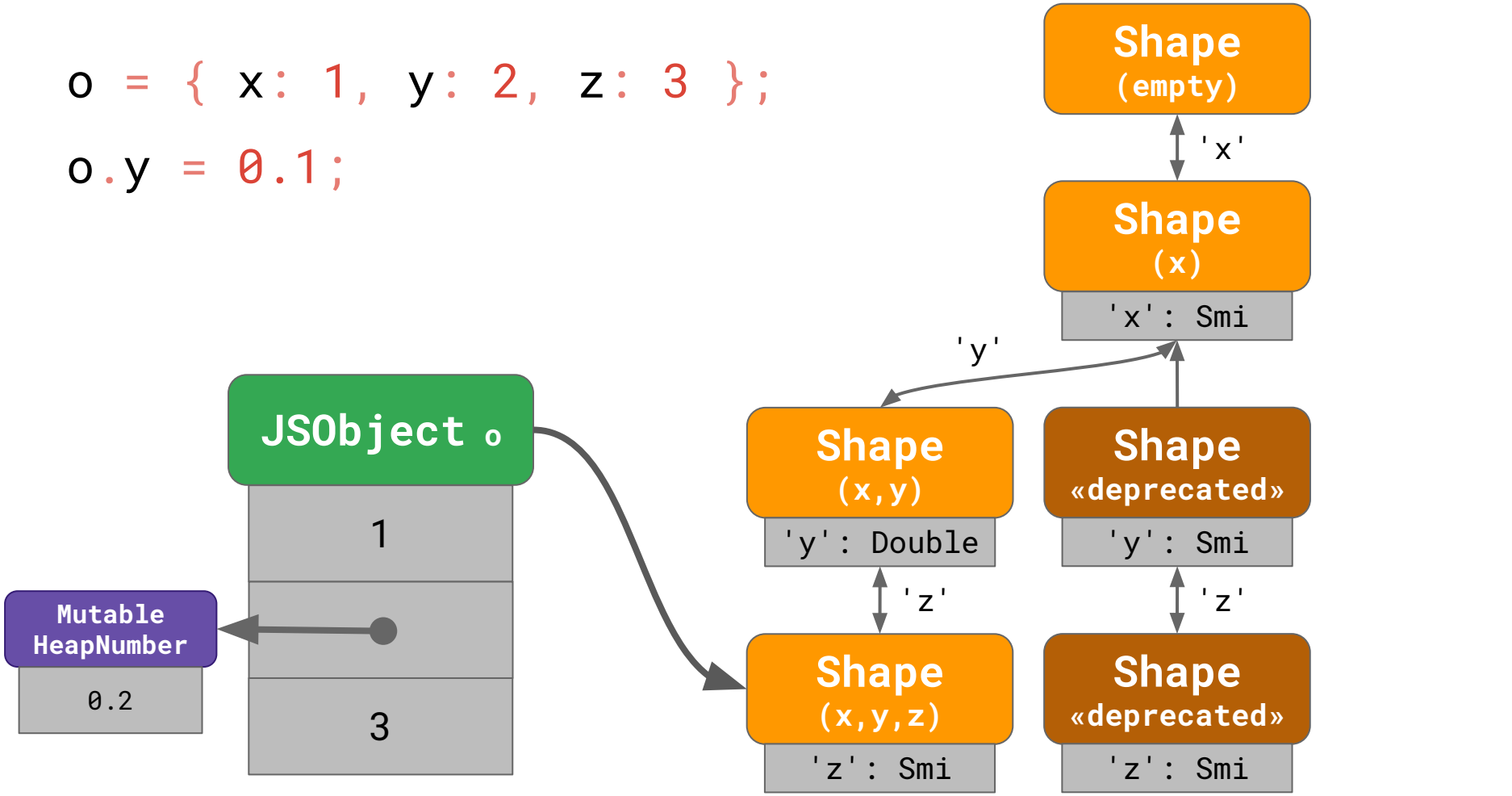
@bmeurer && @mathias

```
o = { x: 1, y: 2, z: 3 };
o.y = 0.1;
```



@bmeurer && @mathias

```
o = { x: 1, y: 2, z: 3 };
o.y = 0.1;
```



@bmeurer && @mathias

Engines optimize property representations transparently

```
const object = { x: 1 };
Object.preventExtensions(object);
object.y = 2;
// TypeError: Cannot add property y;
//            object is not extensible
```

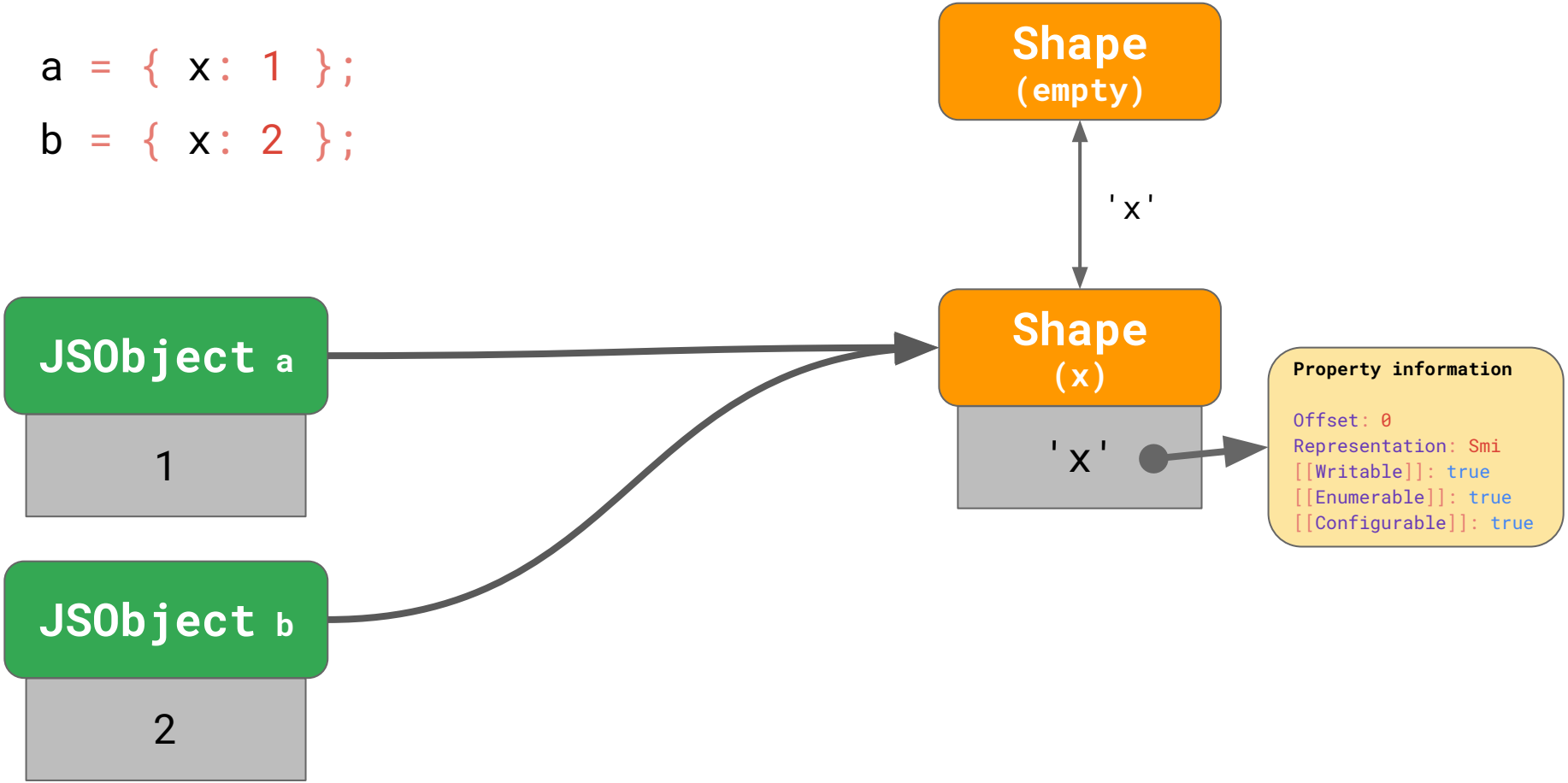
```
const object = { x: 1 };
Object.seal(object);
object.y = 2;
// TypeError: Cannot add property y;
//            object is not extensible
delete object.x;
// TypeError: Cannot delete property x
```

```
const object = { x: 1 };
Object.freeze(object);
object.y = 2;
// TypeError: Cannot add property y;
//            object is not extensible
delete object.x;
// TypeError: Cannot delete property x
object.x = 3;
// TypeError: Cannot assign to read-only property x
```

@bmeurer && @mathias

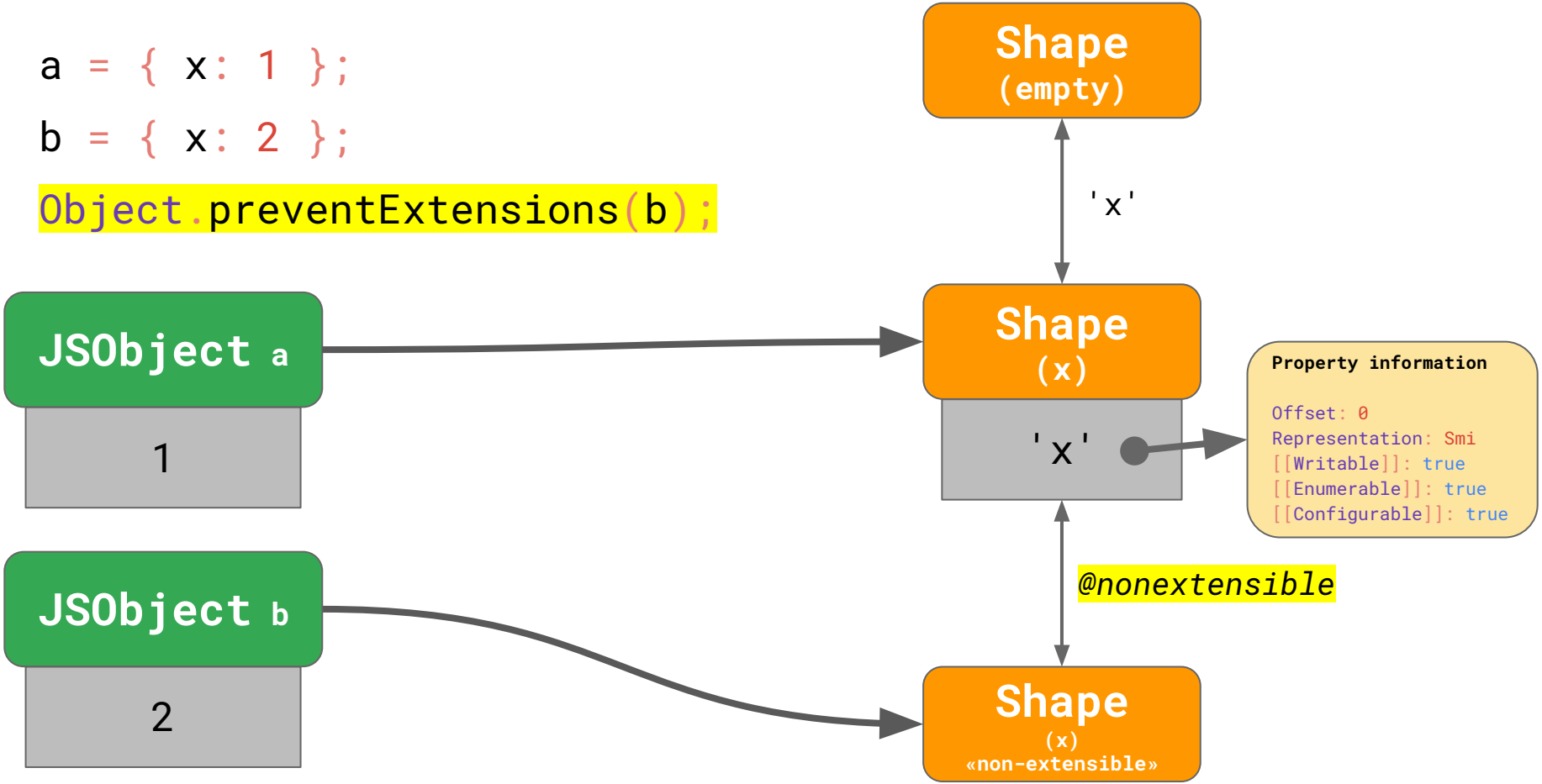
```
const object1 = { x: 1 };
const object2 = { x: 2 };

Object.preventExtensions(object2);
```

```
a = { x: 1 };
b = { x: 2 };
```



@bmeurer && @mathias

```
a = { x: 1 };
b = { x: 2 };
Object.preventExtensions(b);
```



@bmeurer && @mathias

facebook / react

Watch 6,596 | Unstar 119,129 | Fork 21,612

Code | Issues 406 | Pull requests 126 | Projects 0 | Wiki | Insights

performance cliff

Possible v8 ~~de-opt~~ for profiling react-dom bundles #14365

New issue

Closed bvaughn opened this issue on Nov 30, 2018 · 25 comments

bvaughn commented on Nov 30, 2018 • edited — Contributor

I've observed that when React's profiling mode is enabled, updating the profiling fields here and here during the *render* phase seems to cause a significant deopt in `getHostSibling` during the *commit* phase for Chrome only. (Neither Safari nor Firefox seem to be affected by this.)

I have created a repro case with inline annotations here:
https://github.com/bvaughn/react-profiling-v8-deopt-repro#about-this-repro-case

(Note that the above repo is private. Please tag me here if you need access.)

👍 2

bvaughn changed the title ~~Possible v8 de-opt [WIP]~~ **Possible v8 de-opt for profiling react-dom bundles** on Nov 30, 2018

bvaughn commented on Nov 30, 2018 • edited — Contributor

cc **@bmeurer** and **@mathiasbynens**

@bvaughn Can haz repo access? I imagine **@bmeurer** would want access as well.

Assignees
No one assigned

Labels
None yet

Projects
None yet

Milestone
No milestone

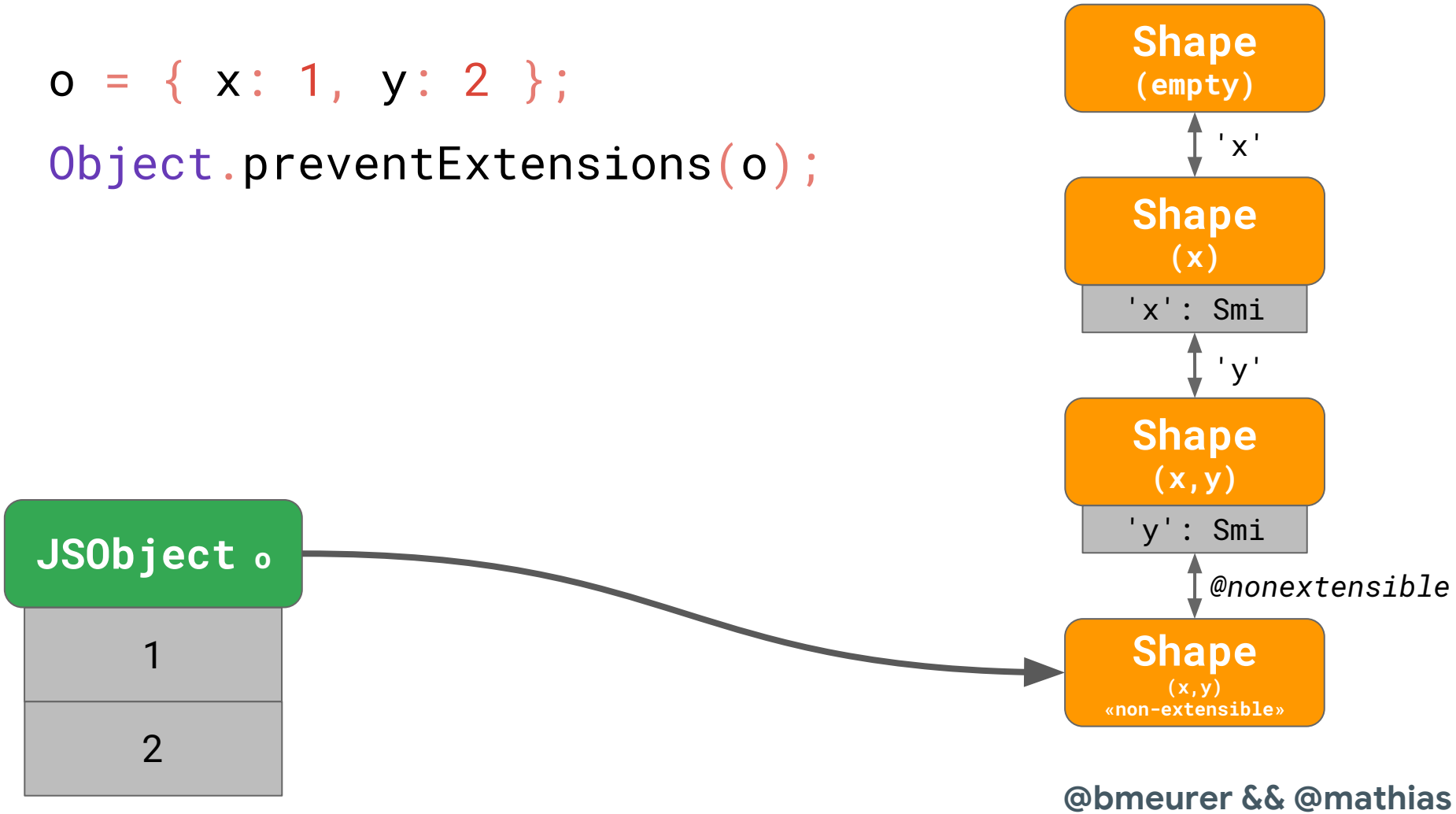
Notifications
Unsubscribe
You're receiving notifications because you were mentioned.

6 participants

facebook/react#14365

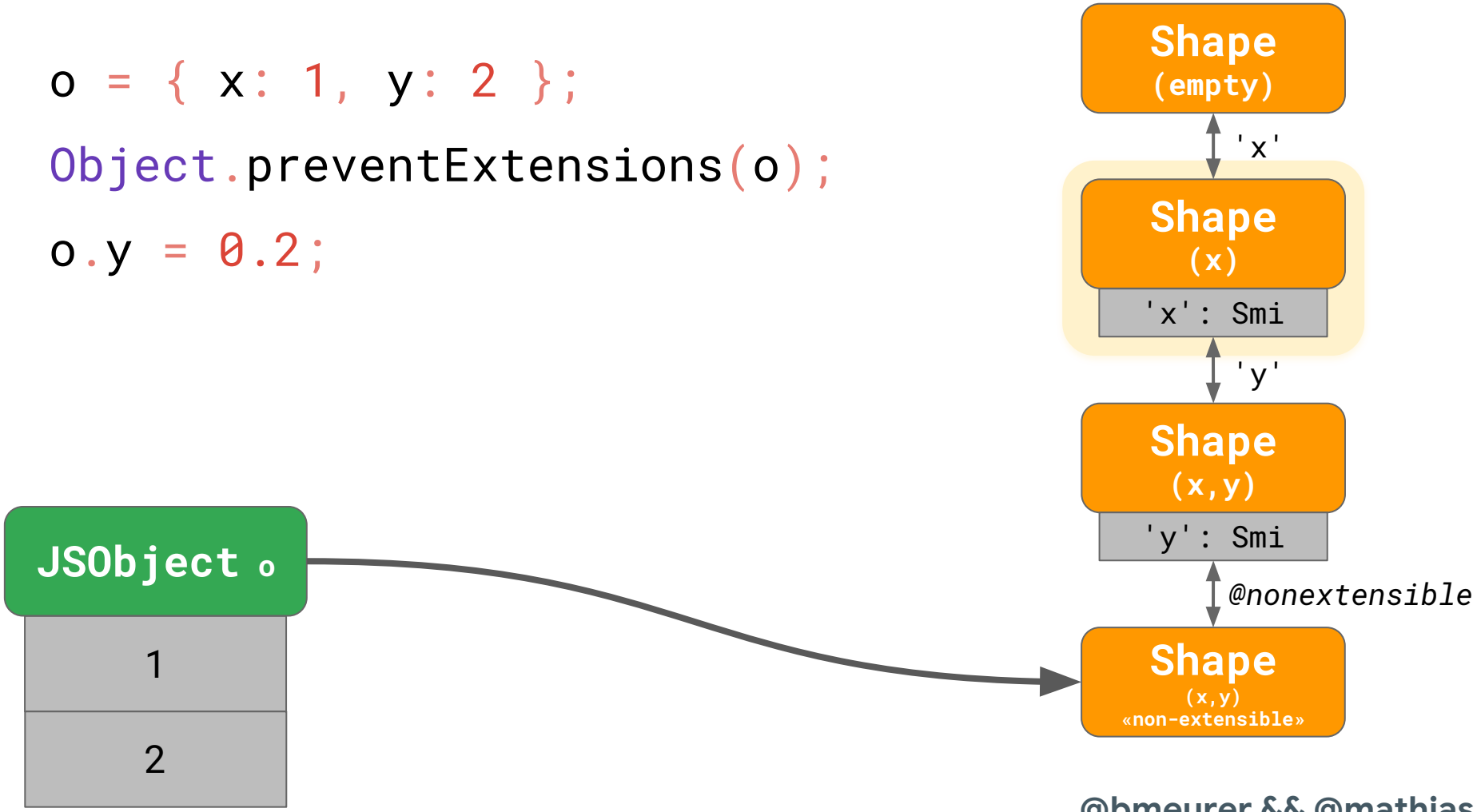
```
const object = { x: 1, y: 2 };
Object.preventExtensions(object);
object.y = 0.2;
```

```
o = { x: 1, y: 2 };
Object.preventExtensions(o);
```



@bmeurer && @mathias

```
o = { x: 1, y: 2 };
Object.preventExtensions(o);
o.y = 0.2;
```



@bmeurer && @mathias

```
o = { x: 1, y: 2 };
Object.preventExtensions(o);
o.y = 0.2;
```

¯_(ツ)_/¯

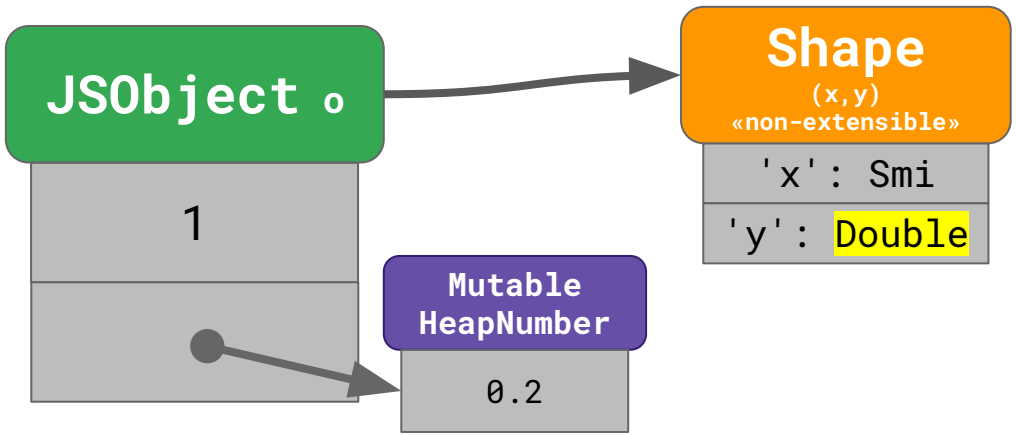
@bmeurer && @mathias

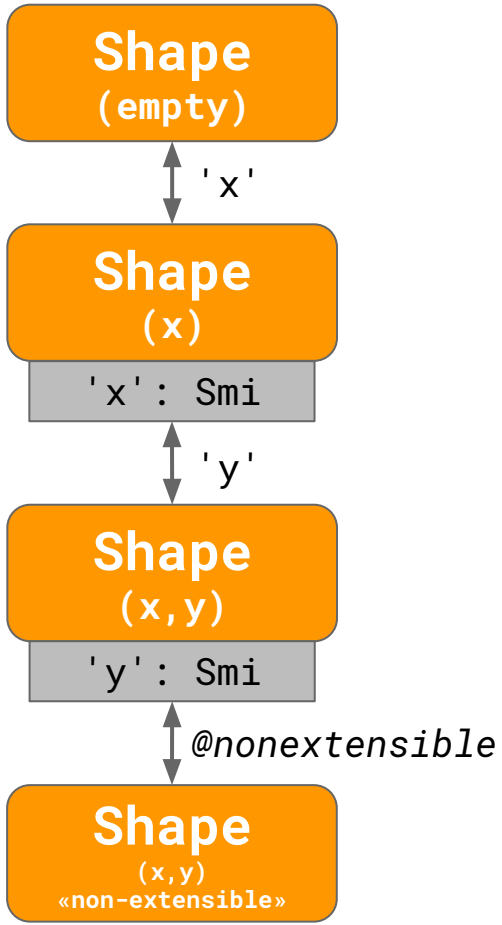
```
o = { x: 1, y: 2 };
Object.preventExtensions(o);
o.y = 0.2;
```

(ノ'□')ノ ⌒ ┻━┻

@bmeurer && @mathias

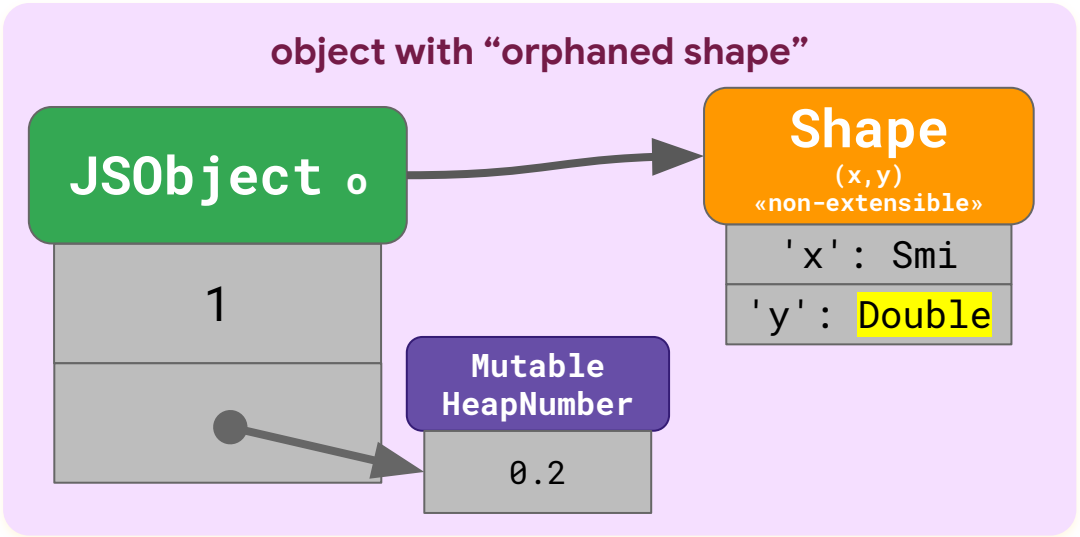
```
o = { x: 1, y: 2 };
Object.preventExtensions(o);
o.y = 0.2;
```

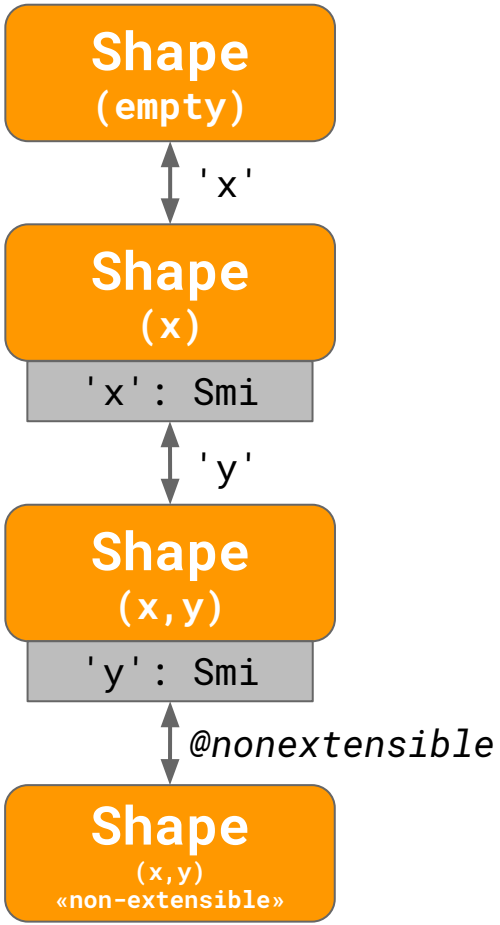




@bmeurer && @mathias

```
o = { x: 1, y: 2 };
Object.preventExtensions(o);
o.y = 0.2;
```





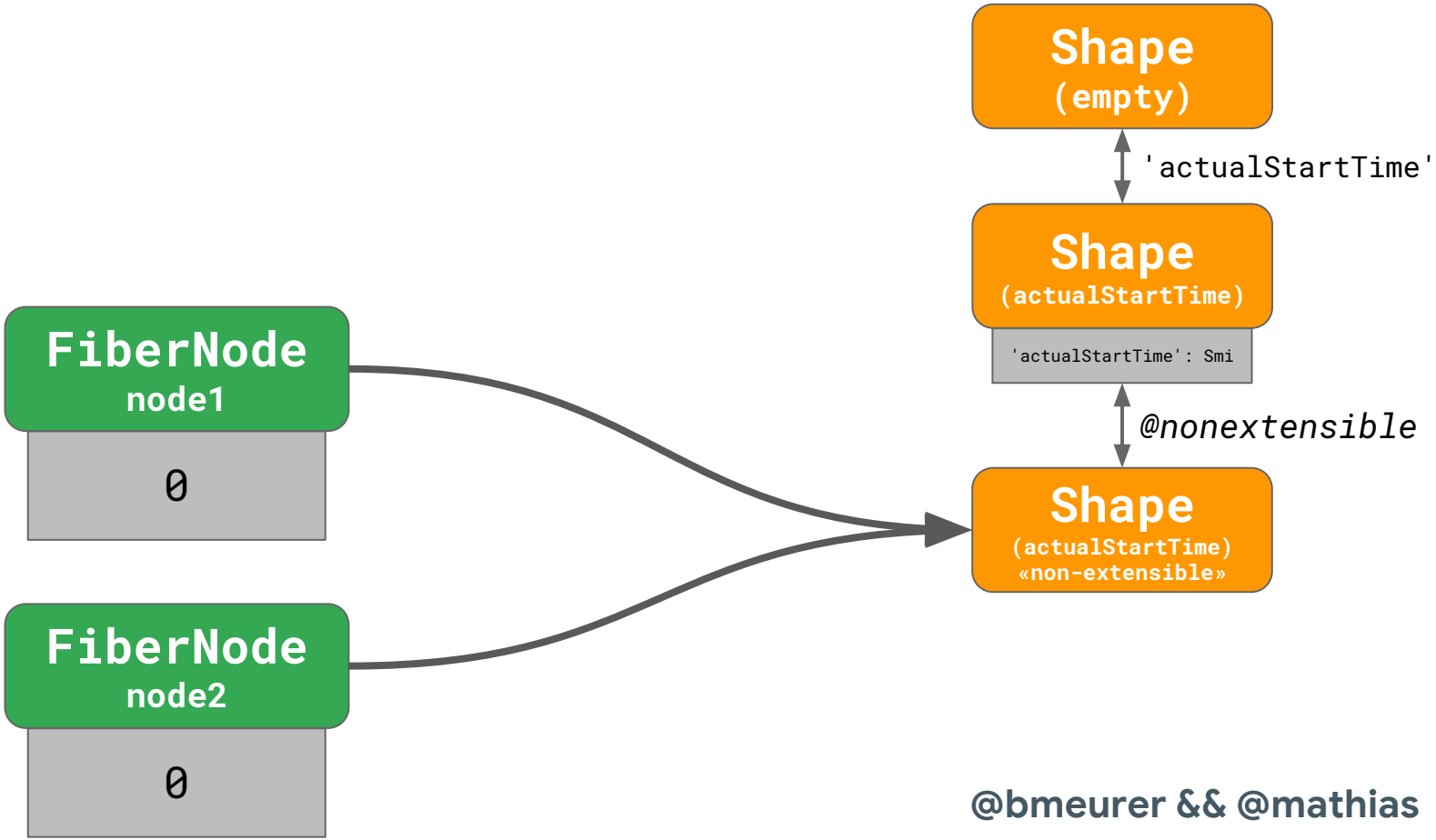
@bmeurer && @mathias

```
class FiberNode {
  constructor() {
    this.actualStartTime = 0;
    Object.preventExtensions(this);
  }
}

const node1 = new FiberNode();
const node2 = new FiberNode();
```

```
class FiberNode {
  constructor() {
    this.actualStartTime = 0;
    Object.preventExtensions(this);
  }
}

const node1 = new FiberNode();
const node2 = new FiberNode();
```

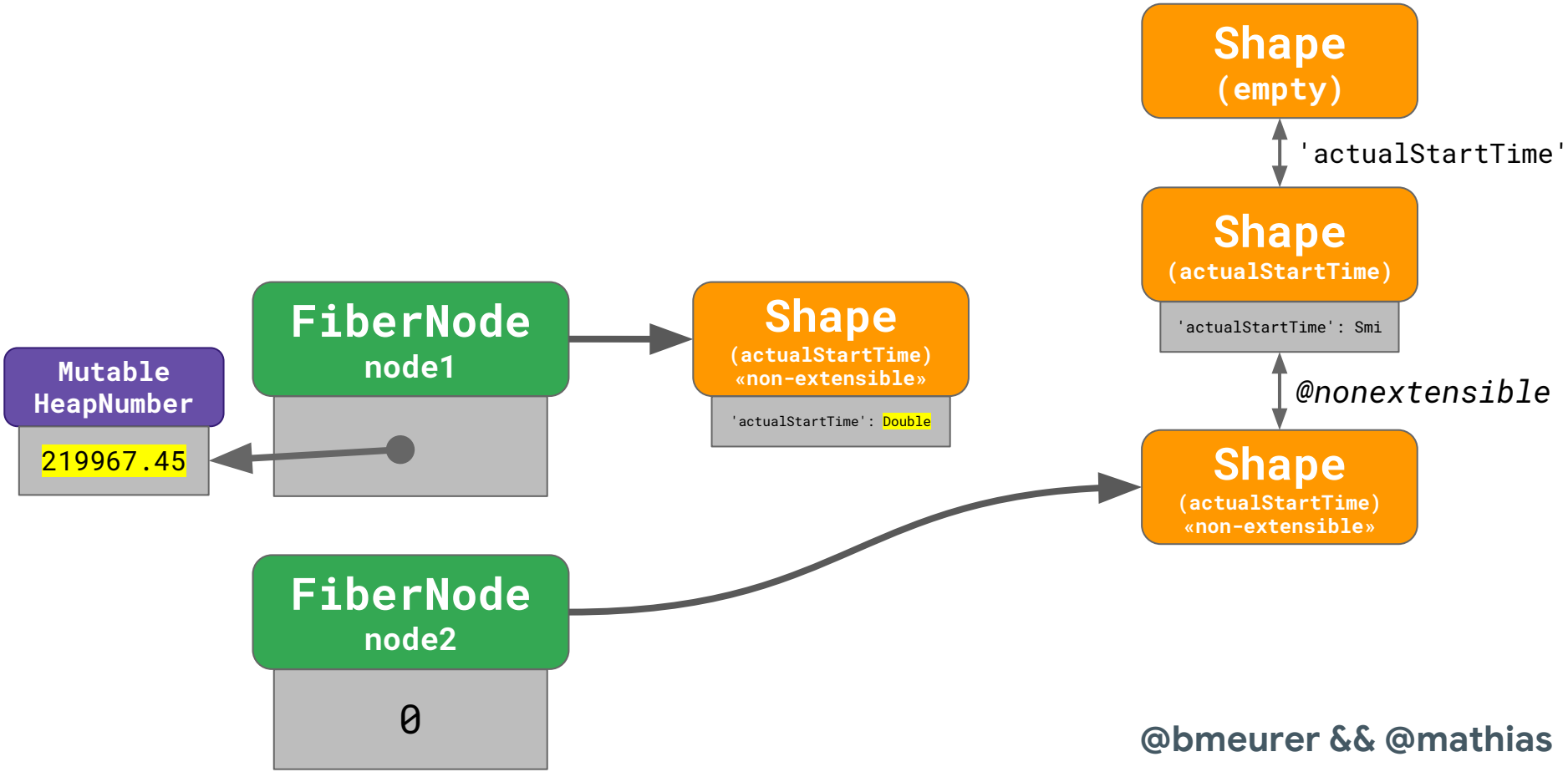
```
// Later on...
node1.actualStartTime =
  performance.now();

// Even later on...
node2.actualStartTime =
  performance.now();
```

@bmeurer && @mathias



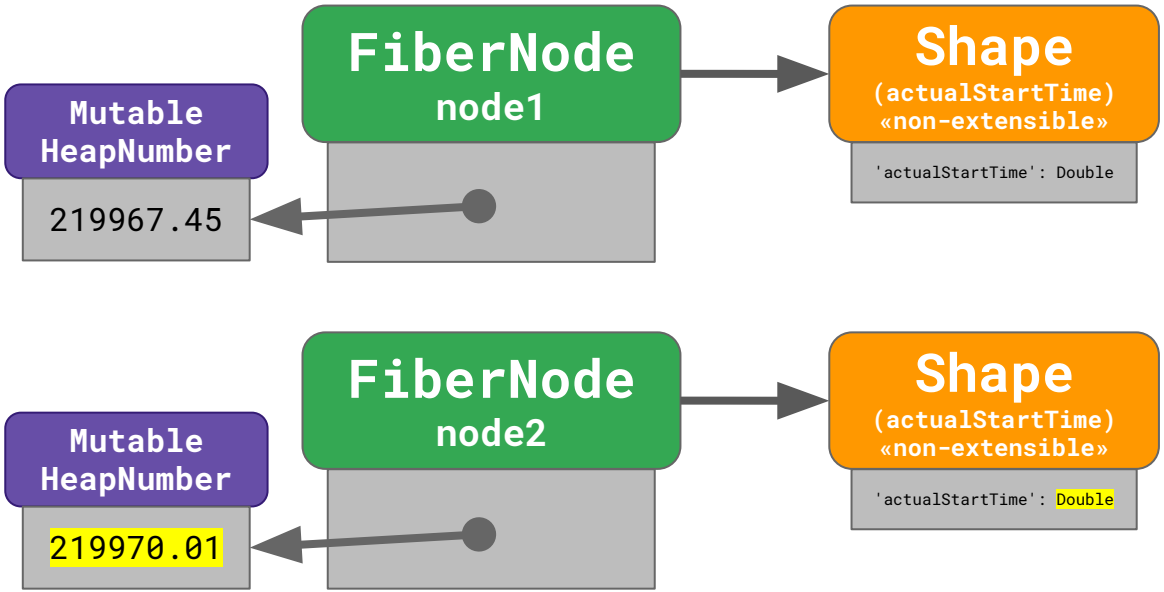
@bmeurer && @mathias

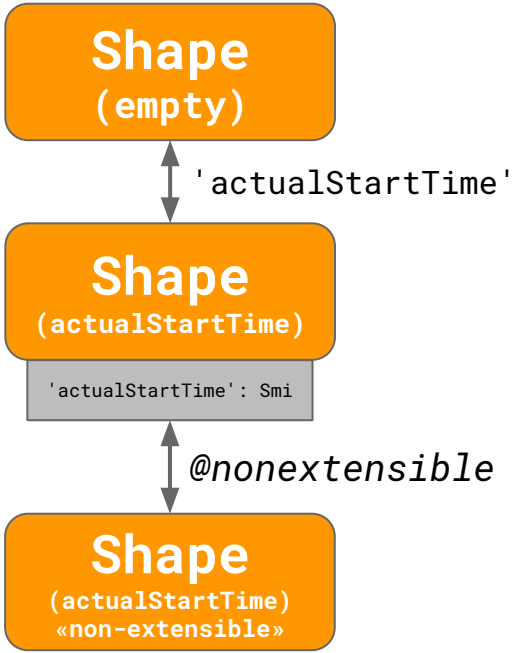
```
node1.actualStartTime = performance.now();
```



@bmeurer && @mathias

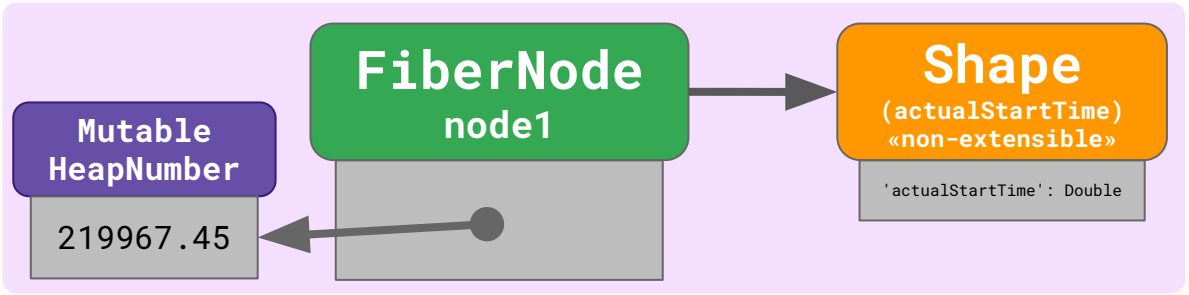
```
node1.actualStartTime = performance.now();
node2.actualStartTime = performance.now();
```

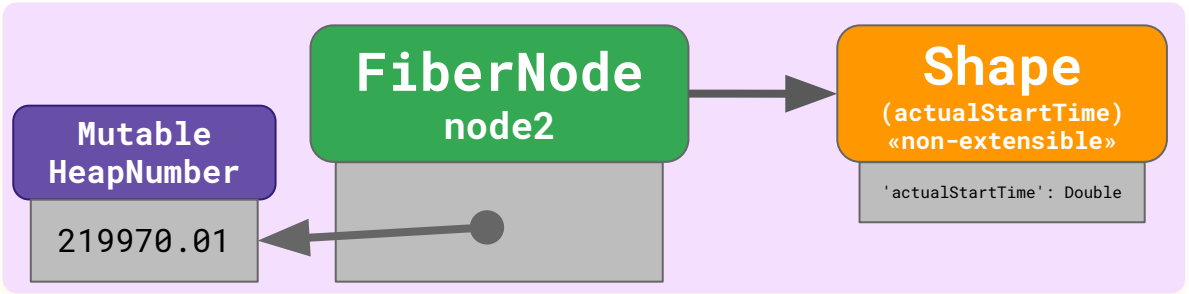


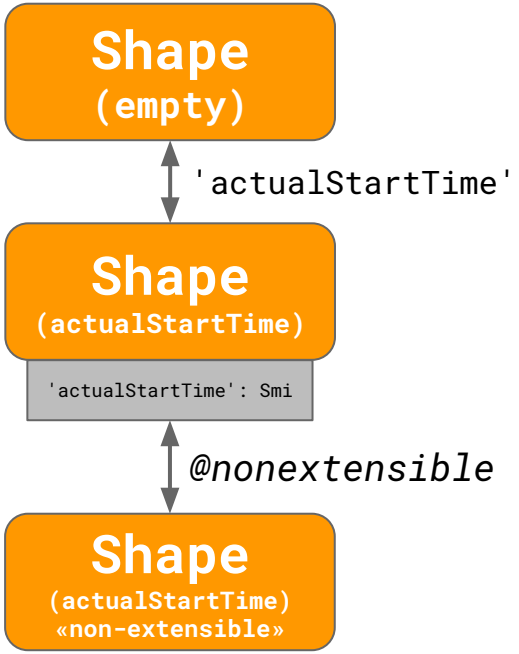


@bmeurer && @mathias

```
node1.actualStartTime = performance.now();
node2.actualStartTime = performance.now();
```







@bmeurer && @mathias

☆ Merged as 06ba822 | 1442640: Map update for integrity level transitions.

Updated Feb 01
Owner Jaroslav Sevcik
Uploader Commit Bot
Committer Commit Bot
Assignee
Reviewers Commit Bot
Igor Sheludko
ADD REVIEWER
CC v8-reviews@googlegroups.com
ADD CC
Repo v8/v8
Branch master
Parent cbeeb86
Topic No topic
Hashtags

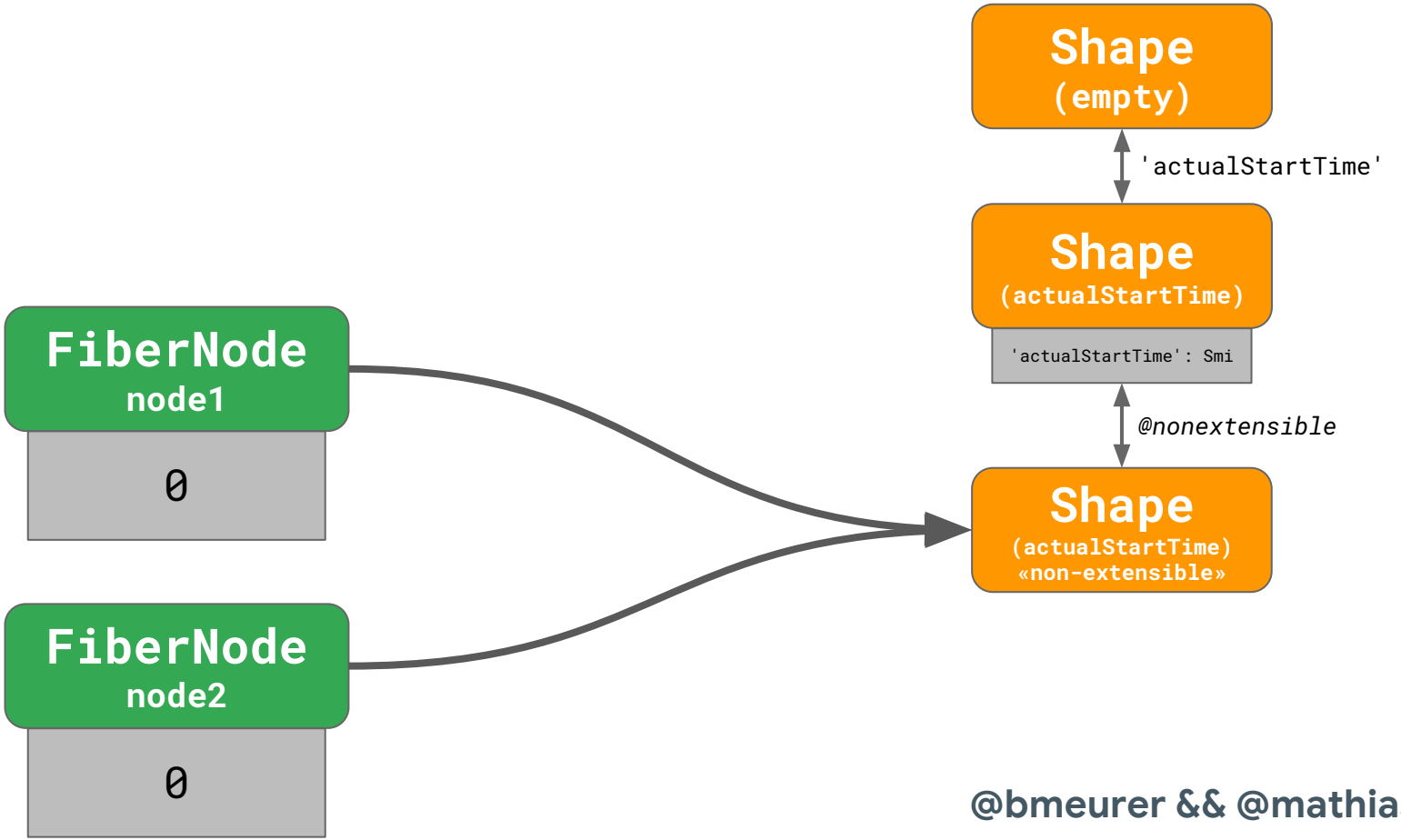
Code-Review +1 Igor Sheludko

Other labels

REPLY

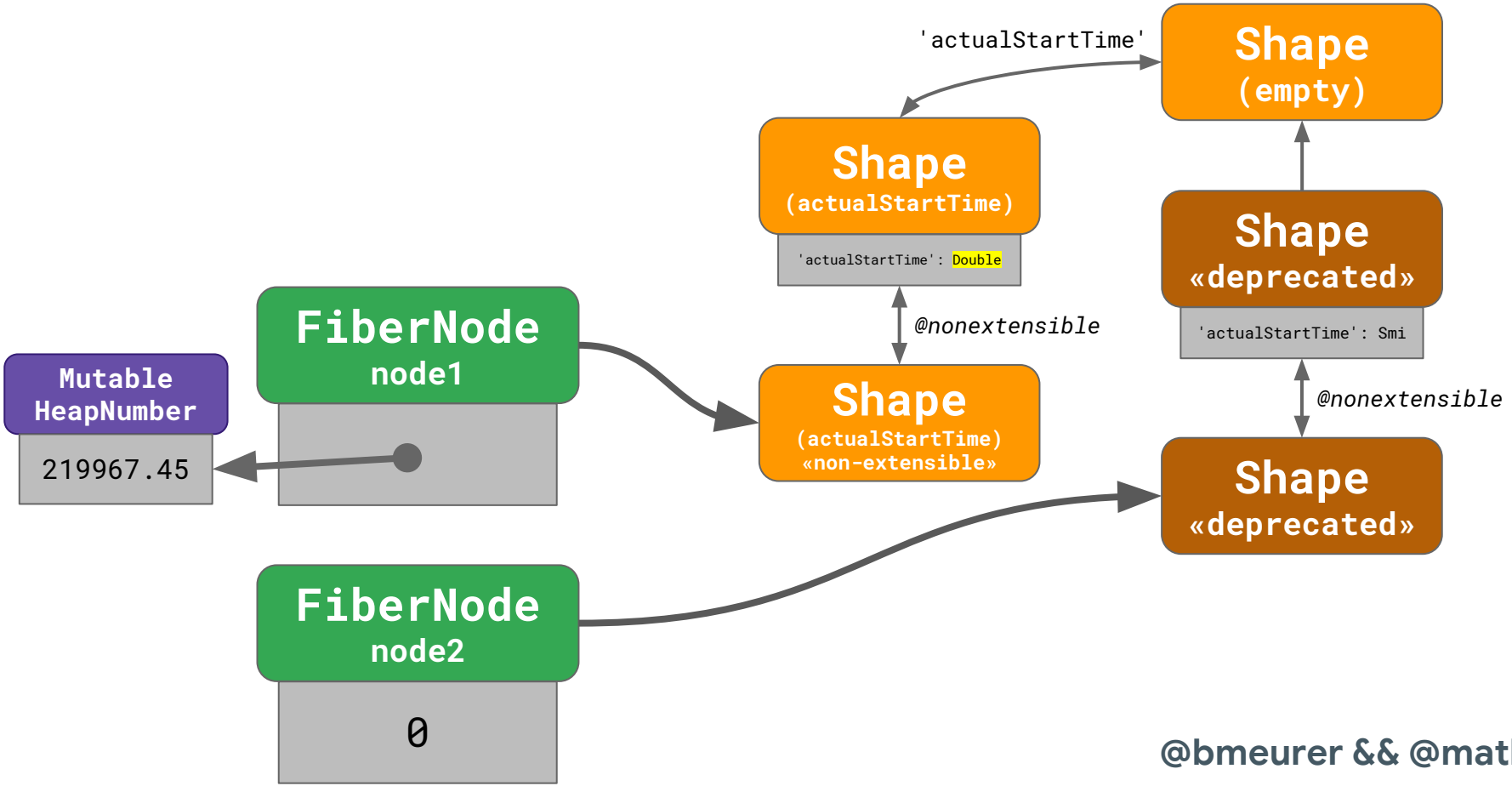
```
Map update for integrity level transitions.

This adds support for integrity level transitions (preventExtensions,
seal and freeze) to MapUpdater and Map::TryUpdate.

In both cases, we first try to detect whether there were integrity
level transitions in the transition tree to the old map and make note
of the most restrictive integrity transition and the map just before
the transition (integrity-source-map). Then we find an appropriate root
(based on integrity-source-map's elements kind) and replay the
transitions based on the integrity-source-map's descriptor
array. Finally, if we saw an integrity level transition in
the beginning, we will find-or-create that transition (on the
updated version of integrity-source-map).

For the following micro-benchmark, we get about 10x speedup.

```
function C() {
  this.x = 1;
  Object.seal(this);
  this.x = 0.1;
}

const start = Date.now();
for (let i = 0; i < 1e7; i++) {
```

▼ SHOW MORE

Tree is open (Automatic: (/·ω·)/)

chromium-review.googlesource.com/1442640
```



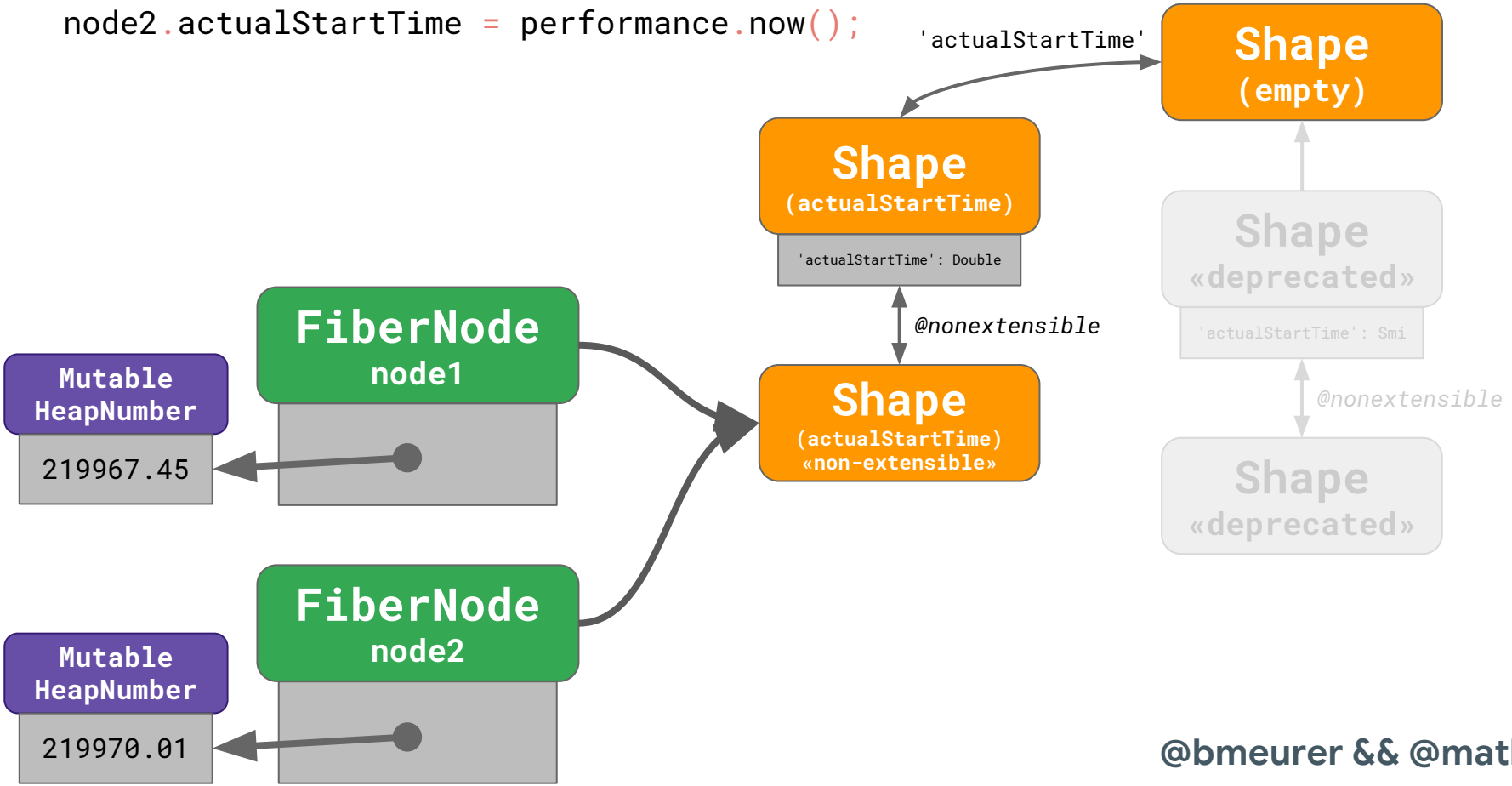
@bmeurer && @mathias

```
node1.actualStartTime = performance.now();
```



@bmeurer && @mathias

```
node1.actualStartTime = performance.now();
node2.actualStartTime = performance.now();
```



@bmeurer && @mathias

performance cliff

Prevent a ~~v8 deopt~~ when profiling #14383

Merged bvaughn merged 1 commit into `facebook:master` from `bvaughn:prevent-v8-profiler-deopt` on 3 Dec 2018

Conversation 7 · Commits 1 · Checks 0 · Files changed 1

Changes from all commits · File filter... · Jump to... · +17 −0 · Diff settings · Review changes

17 packages/react-reconciler/src/ReactFiber.js — Show comments · Copy path · View file

```
@@ -254,6 +254,23 @@ function FiberNode(
254 254     this.alternate = null;
255 255
256 256     if (enableProfilerTimer) {
257     +     // Note: The following is done to avoid a v8 deopt.
258     +     //
259     +     // It is important to initialize the fields below with doubles.
260     +     // Otherwise Fibers will deopt and end up having separate shapes when
261     +     // doubles are later assigned to fields that initially contained smis.
262     +     // This is a bug in v8 having something to do with Object.preventExtension().
263     +     //
264     +     // Learn more about this deopt here:
265     +     // https://github.com/facebook/react/issues/14365
266     +     // https://bugs.chromium.org/p/v8/issues/detail?id=8538
267     +     this.actualDuration = Number.NaN;
268     +     this.actualStartTime = Number.NaN;
269     +     this.selfBaseDuration = Number.NaN;
270     +     this.treeBaseDuration = Number.NaN;
271     +
272     +     // It's okay to replace the initial doubles with smis after initialization.
273     +     // This simplifies other profiler code and doesn't trigger the deopt.
257 274       this.actualDuration = 0;
258 275       this.actualStartTime = -1;
259 276       this.selfBaseDuration = 0;
260 277       this.treeBaseDuration = 0;
261 278     }
264 281     this._debugID = debugCounter++;
265 282     this._debugSource = null;
```

facebook/react#14383

```
// Introduce fields with `Double` representation
this.actualDuration = Number.MIN_VALUE;
this.actualStartTime = Number.MIN_VALUE;
this.selfBaseDuration = Number.MIN_VALUE;
this.treeBaseDuration = Number.MIN_VALUE;

// Actual field initialization
this.actualDuration = 0;
this.actualStartTime = -1;
this.selfBaseDuration = 0;
this.treeBaseDuration = 0;
```

@mathias && @bmeurer

```
class Point {
  x = null;
  y = null;
}

const p = new Point();
p.x = 0.1;
p.y = 402;
```

In-place field representation changes

Attention: Shared Google-externally

Authors: jarin@, bmeurer@, tebbi@

Last Updated: 2019-02-20

TL;DR V8 uses a technique called field representation tracking to optimize storage and access for fields based on the representation of values seen so far. Currently, changes between field representations always require map transitions, even if the representations are compatible. This document describes a design to make those changes in-place whenever possible.

Short Link bit.ly/v8-in-place-field-representation-changes

Bugs v8:8749, v8:8865

Motivation

The problem was first spotted when investigating `Speedometer2/ElmJS-TodoMVC` performance (v8:8749), where the browser spends **20% of time** migrating maps from [illegible] to [illegible]

We illustrate the problem on the following example:

bit.ly/v8-in-place-field-representation-changes

Initialize fields with realistic values

JavaScript engine trade-offs:

- *generating code quickly* vs. *generating quick code*

JavaScript engine trade-offs:

- *generating code quickly* vs. *generating quick code*
- *optimization level* vs. *memory usage*

JavaScript objects:

- string-keyed dictionaries

JavaScript objects:

- string-keyed dictionaries
- optimizations via *Shapes*

JavaScript types:

- primitive vs. object types

JavaScript types:

- primitive vs. object types
- types vs. value representations

Performance advice:

- always initialize objects in the same way

Performance advice:

- always initialize objects in the same way
- initialize fields with sensible values

Thank you!

[**@bmeurer**, **@mathias**].join(**@v8js**)

```
const object = {
  x: 42,
  y: 4.2,
};

// Later
object.x += 10;
object.y += 1;
object.x = object.y;
```