Typhoon

JEREMY FETIVEAU

ATTACKING TURBOFAN

TyphoonCon 2019 - Seoul

WHO AM I?

- digital nomad*
- independent security research
- I contribute to doar-e.github.io
  - Open to everyone!

@__x86

doar-e.github.io

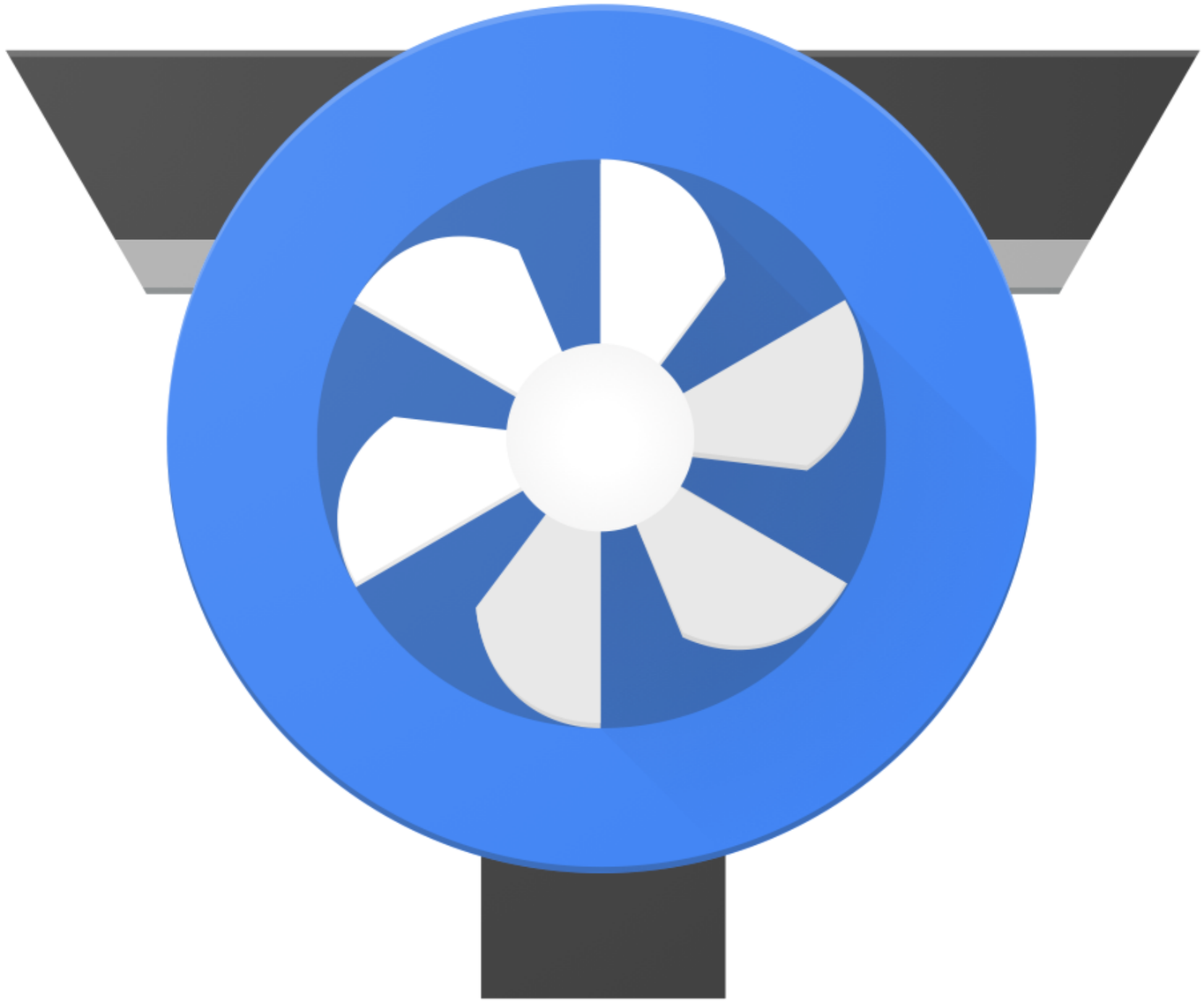
* french, as you can tell from my accent ;-)

WHAT IS THIS TALK ABOUT?

- Google Chrome
- V8 javascript engine
- TurboFan optimizing compiler

SUMMARY

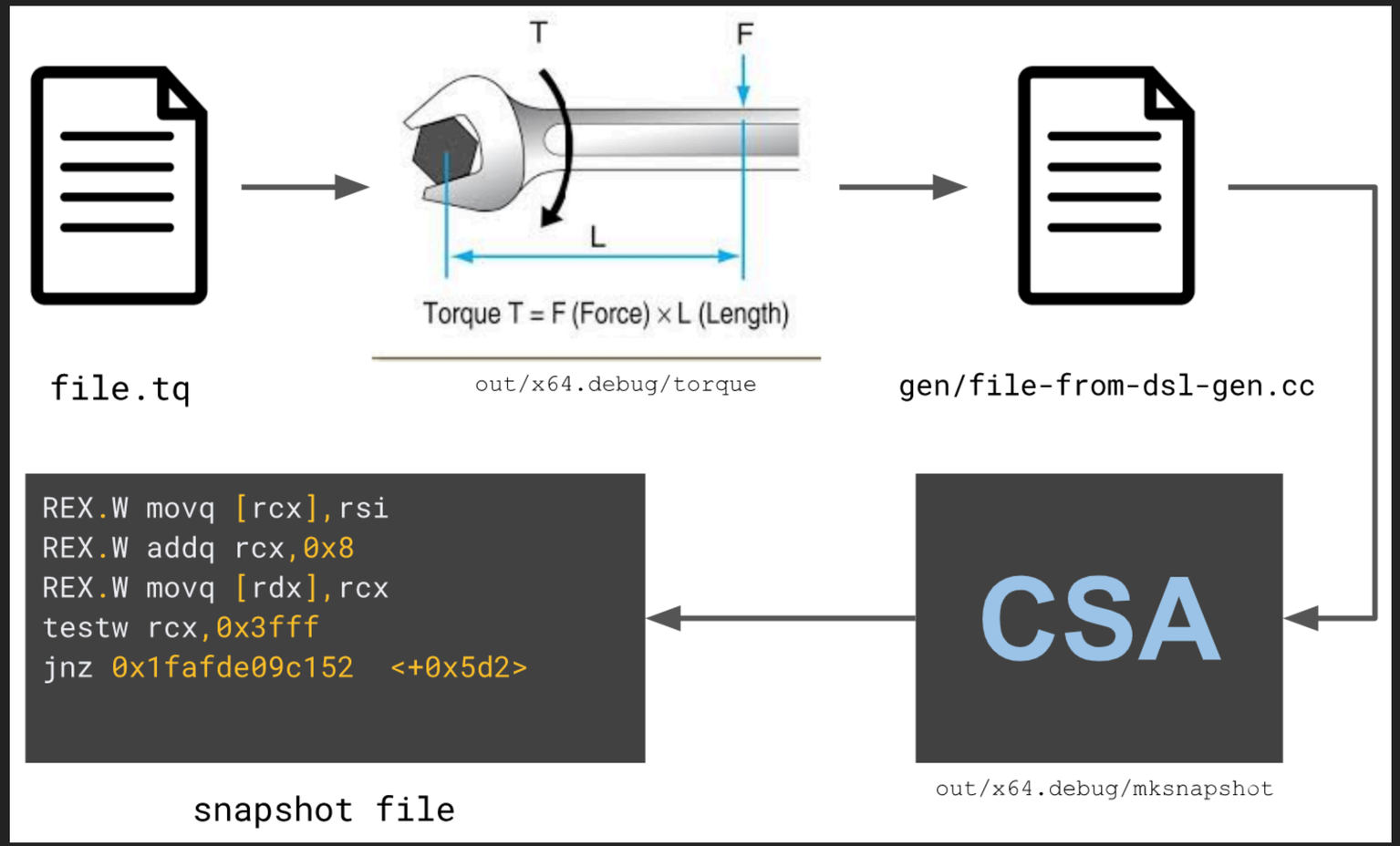
- Internals
- Typer bugs
- Incorrect assumptions on maps
- Exploitation



INTERNALS

BUILTINS

- Predefined
  - Array.prototype.concat
- Several kind
  - CPP
  - Code Stub Assembler
  - Torque
- Pre-optimized code in cache



https://v8.dev/docs/torque

MEMORY REPRESENTATION

```
d8> let a = new Array(1,2,3,4)
d8> %DebugPrint(a)


DebugPrint: [JSArray]
 - map: <Map(PACKED_SMI_ELEMENTS)> [FastProperties]
 - prototype: <JSArray[0]>
 - elements: <FixedArray[4]> [PACKED_SMI_ELEMENTS]
 - length: 4
 - properties: <FixedArray[0]> { #length }
 - elements: <FixedArray[4]> {
           0: 1
           1: 2
           2: 3
           3: 4
 }
```

MEMORY REPRESENTATION

```
d8> let a = new Array(1,2,3,4)
d8> %DebugPrint(a)


DebugPrint: [JSArray]
 - map: <Map(PACKED_SMI_ELEMENTS)> [FastProperties]
 - prototype: <JSArray[0]>
 - elements: <FixedArray[4]> [PACKED_SMI_ELEMENTS]
 - length: 4
 - properties: <FixedArray[0]> { #length }
 - elements: <FixedArray[4]> {
           0: 1
           1: 2
           2: 3
           3: 4
 }
```

The **map** describes the object structure.

Also known as :

- Hidden class
- Type (chakra)
- Structure (JavaScriptCore)
- Shape (SpiderMonkey)

MEMORY REPRESENTATION

```
d8> let a = new Array(1,2,3,4)
d8> %DebugPrint(a)


DebugPrint: [JSArray]
 - map: <Map(PACKED_SMI_ELEMENTS)> [FastProperties]
 - prototype: <JSArray[0]>
 - elements: <FixedArray[4]> [PACKED_SMI_ELEMENTS]
 - length: 4
 - properties: <FixedArray[0]> { #length }
 - elements: <FixedArray[4]> {
           0: 1
           1: 2
           2: 3
           3: 4
 }
```

The **map** describes the object structure.
Also known as :

- Hidden class
- Type (chakra)
- Structure (JavaScriptCore)
- Shape (SpiderMonkey)

The **ElementsKinds** tells us about the types of elements.

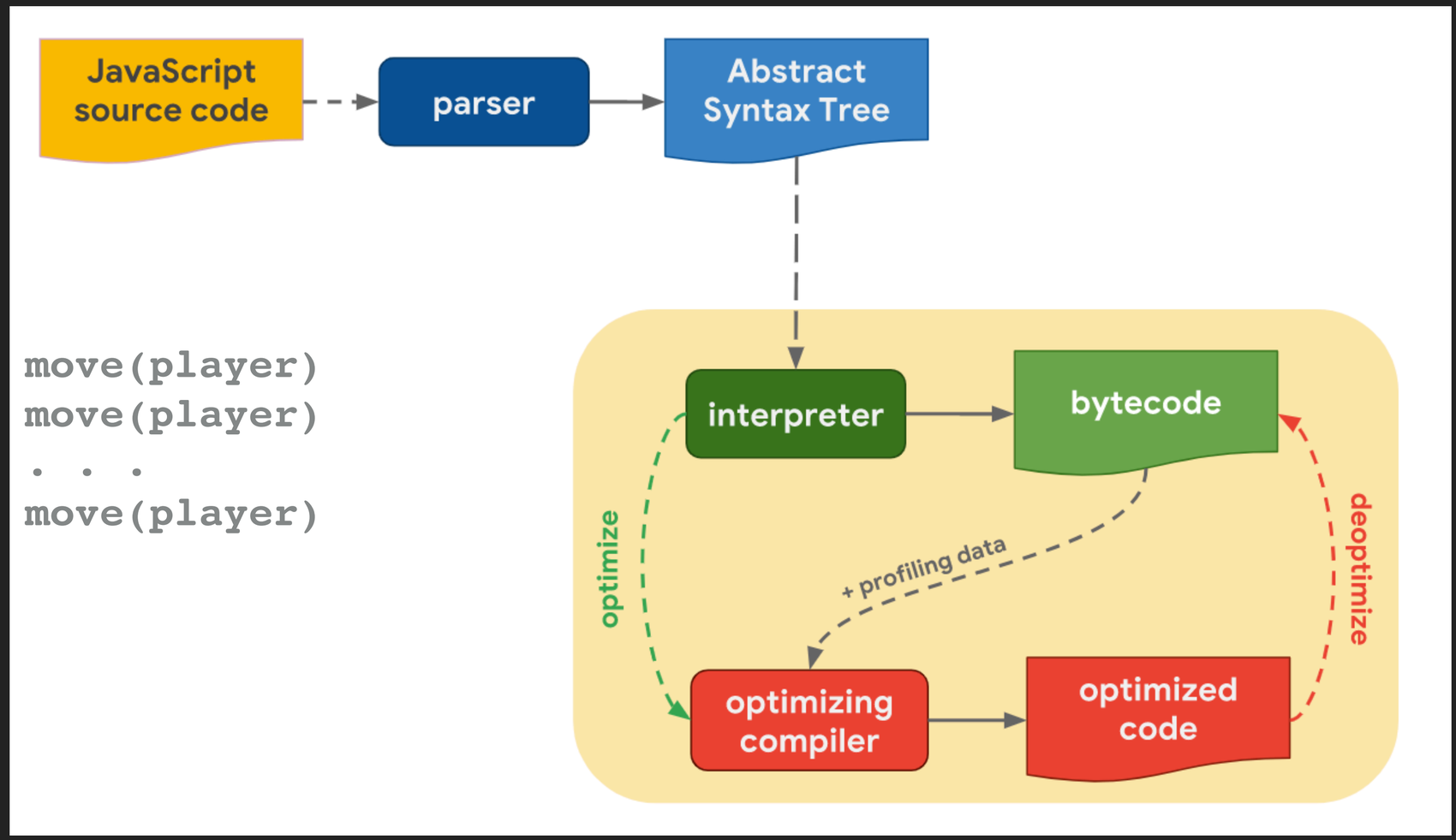
MEMORY REPRESENTATION

JSArray	ElementsKind
[1,2,3]	PACKED_SMI_ELEMENTS
[1,,,,,,2,3]	HOLEY_SMI_ELEMENTS
[1.1,1.2]	PACKED_DOUBLE_ELEMENTS
[{},1.1,2]	PACKED_ELEMENTS

TURBOFAN, IGNITION, LIFTOFF

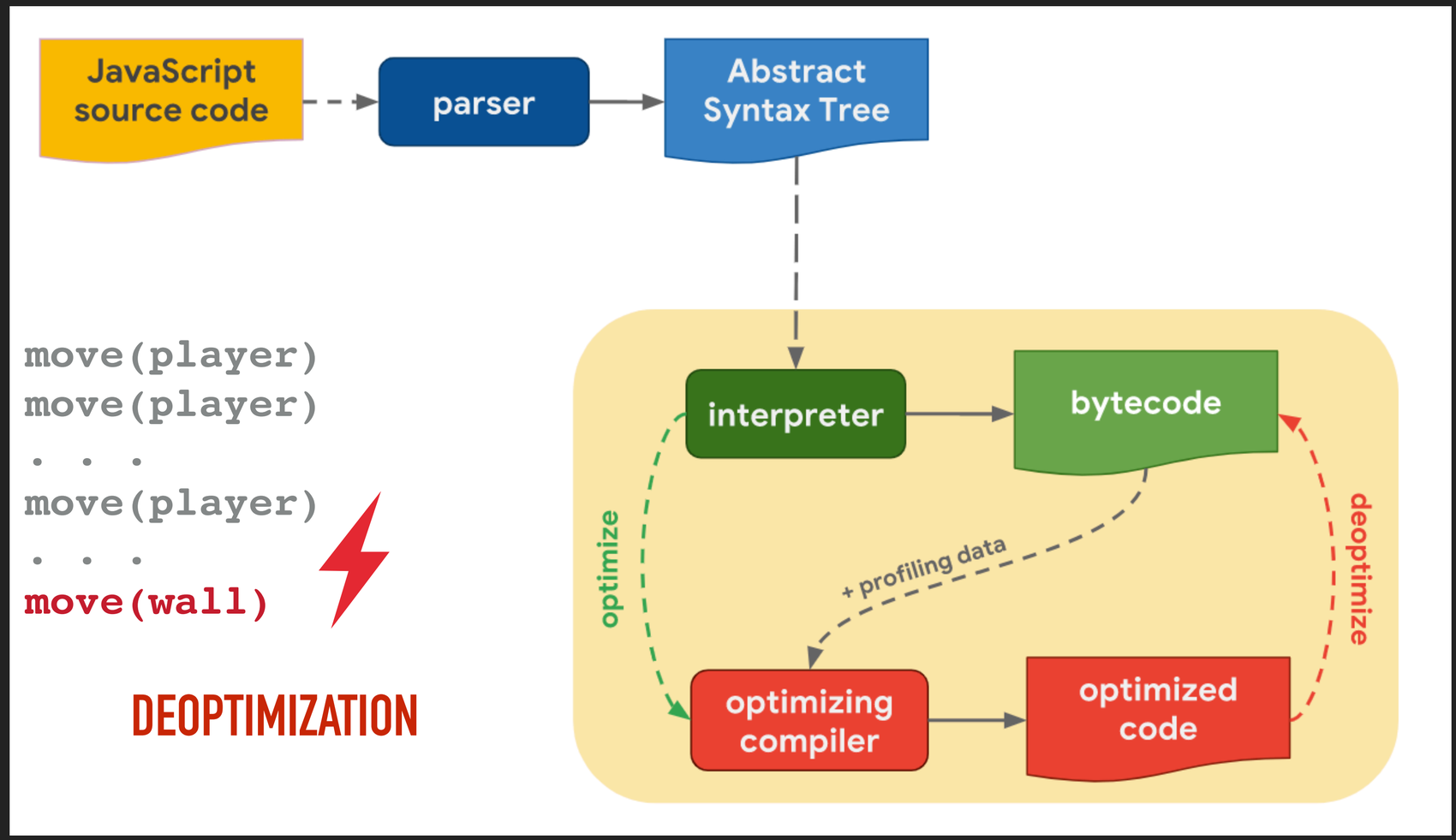
- TurboFan builtins (pre-optimized)
- Ignition bytecode for JavaScript
  - Optimized later by TurboFan
- WebAssembly code
  - Uses the Liftoff base compiler

SPECULATIVE OPTIMIZATIONS



https://mathiasbynens.be/notes/shapes-ics

SPECULATIVE OPTIMIZATIONS



https://mathiasbynens.be/notes/shapes-ics

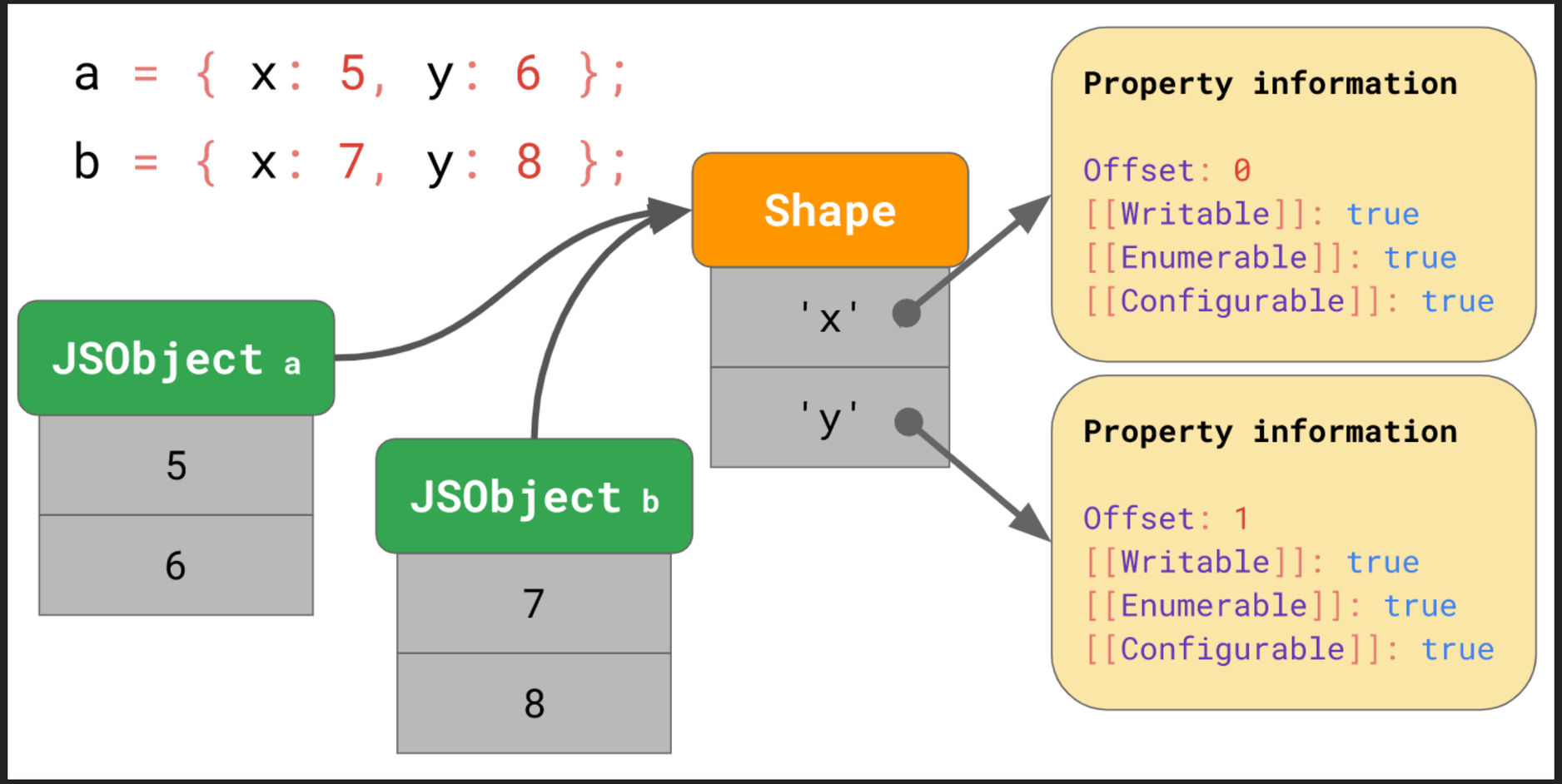
IGNITION BYTECODE

```
let inc = (obj) => (obj.x + 1)
```

```
StackCheck
LdaNamedProperty a0, [0], [1]
AddSmi [1], [0]
Return
```

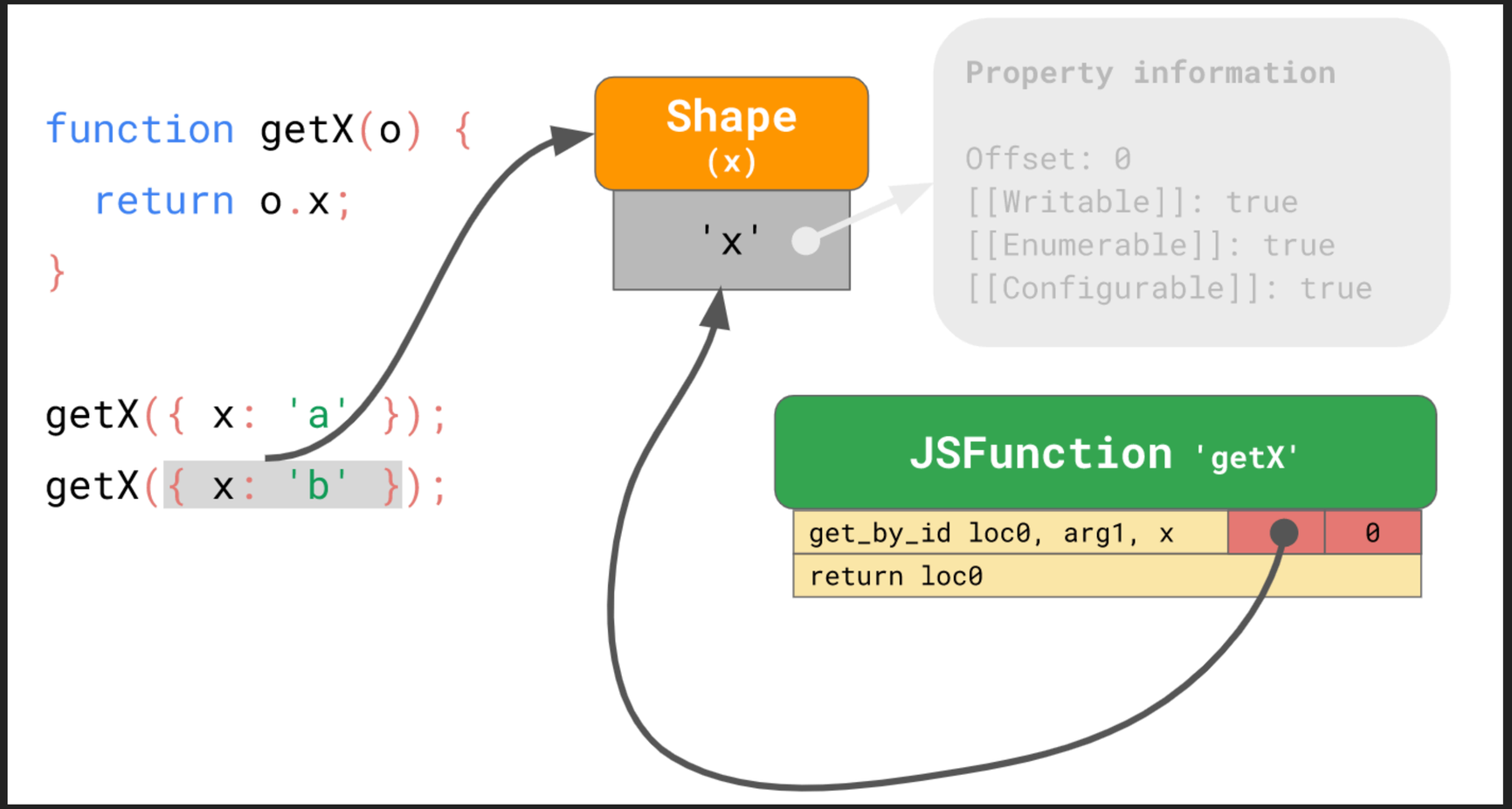
```
Constant pool (size = 1)
 - length: 1
 0: <String[#1]: x>
```

SHAPES



https://mathiasbynens.be/notes/prototypes

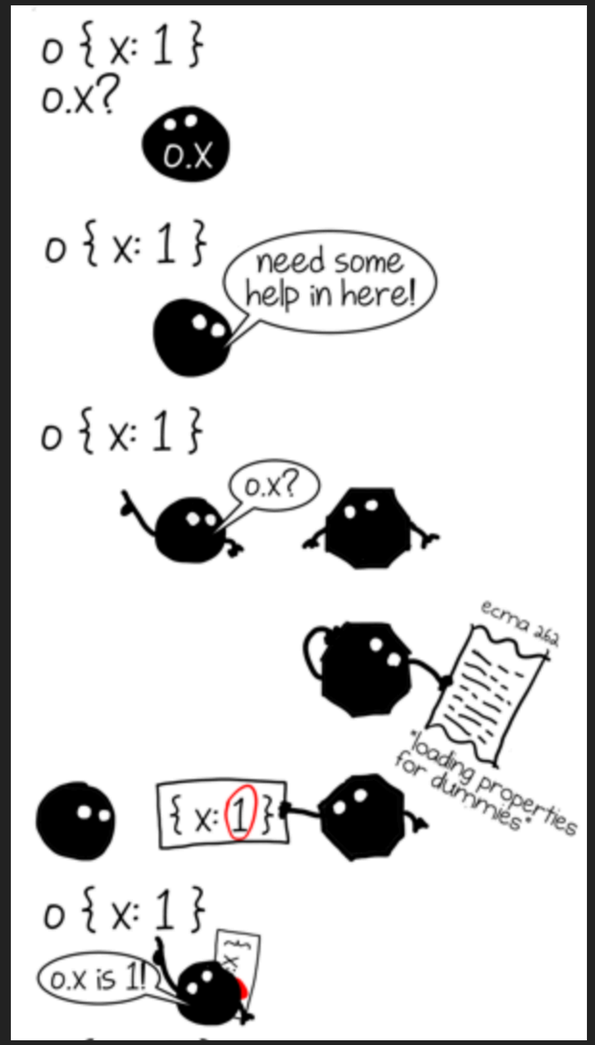
INLINE CACHES



Inline caching is remembering the result of a previous similar lookup.

https://mathiasbynens.be/notes/prototypes

INLINE CACHES

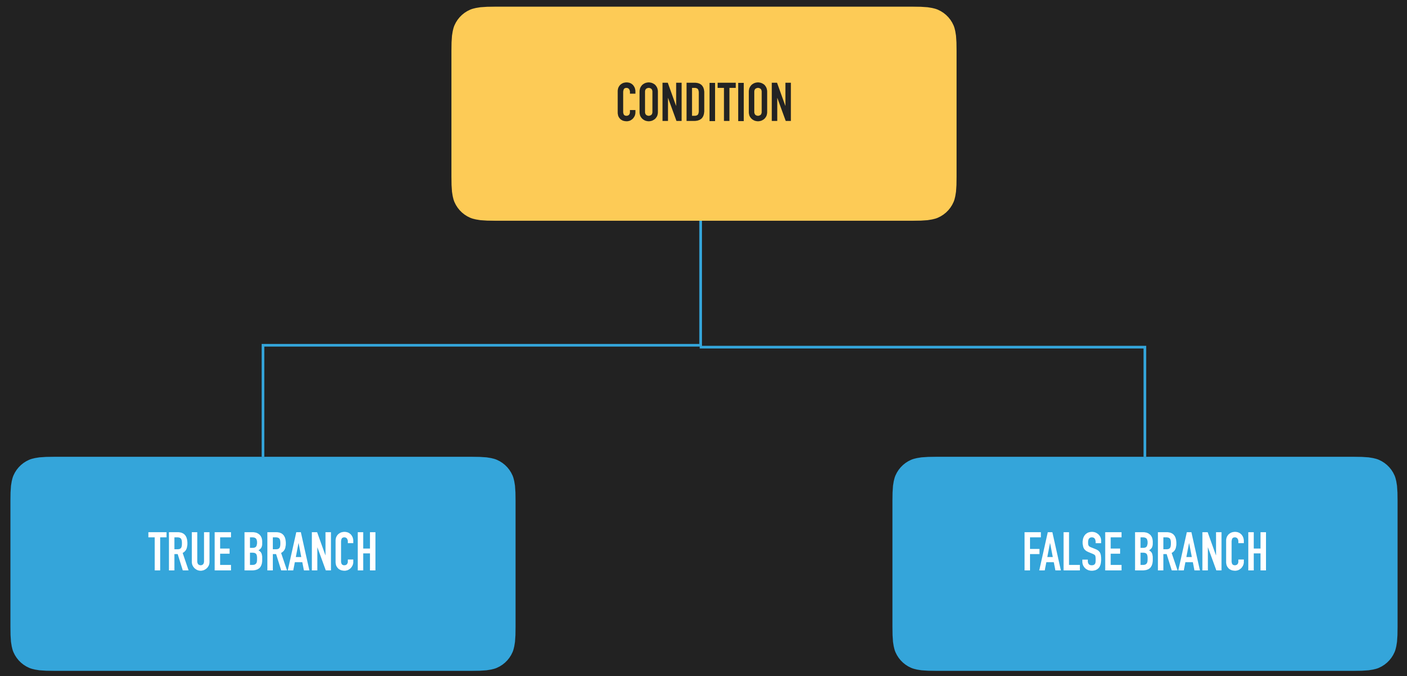


https://mrale.ph/blog/2015/01/11/whats-up-with-monomorphism.html

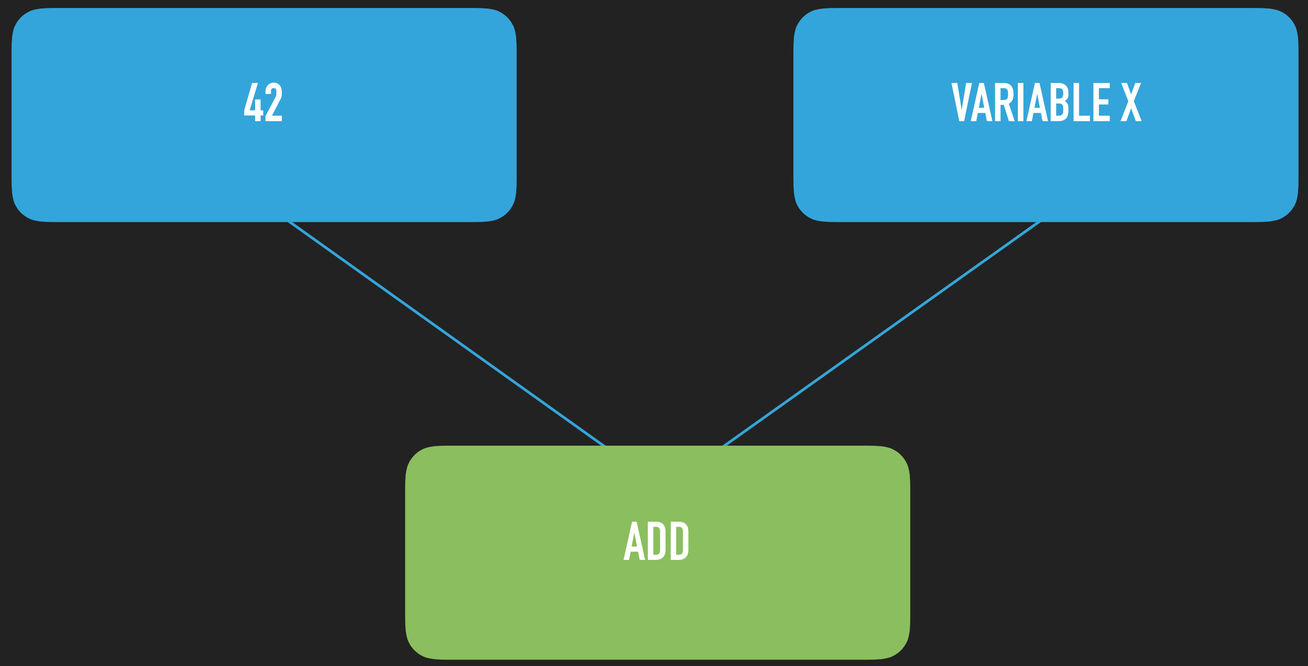
A SEA OF NODES

- Control-flow edges
- Data-flow edges
- Effect-flow edges

CONTROL EDGES

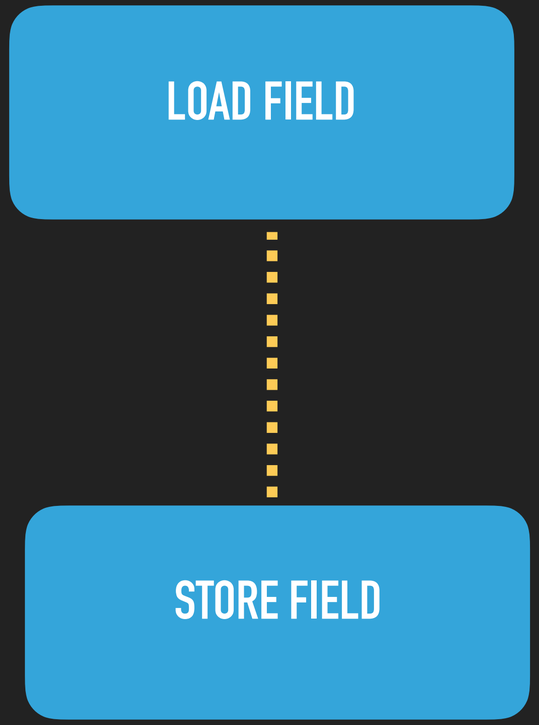


VALUE EDGES



EFFECT EDGES

- What happens if
  - No control dependency
  - No value dependency



Effect edges order operations reading and writing states

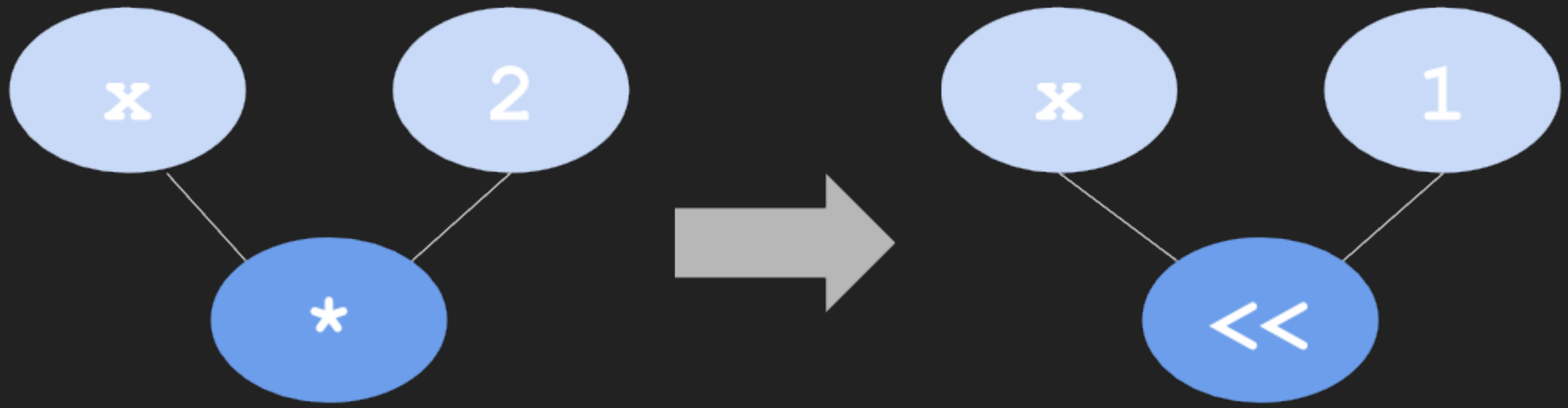
TURBOFAN'S PIPELINE

- Optimisations phases
  - Reducers
    - Work on nodes
- We can observe that with a tool called Turbolizer
  - d8 file.js --trace-turbo
  - tools/turbolizer/
- src/compiler/pipeline.cc

OPTIMIZATIONS

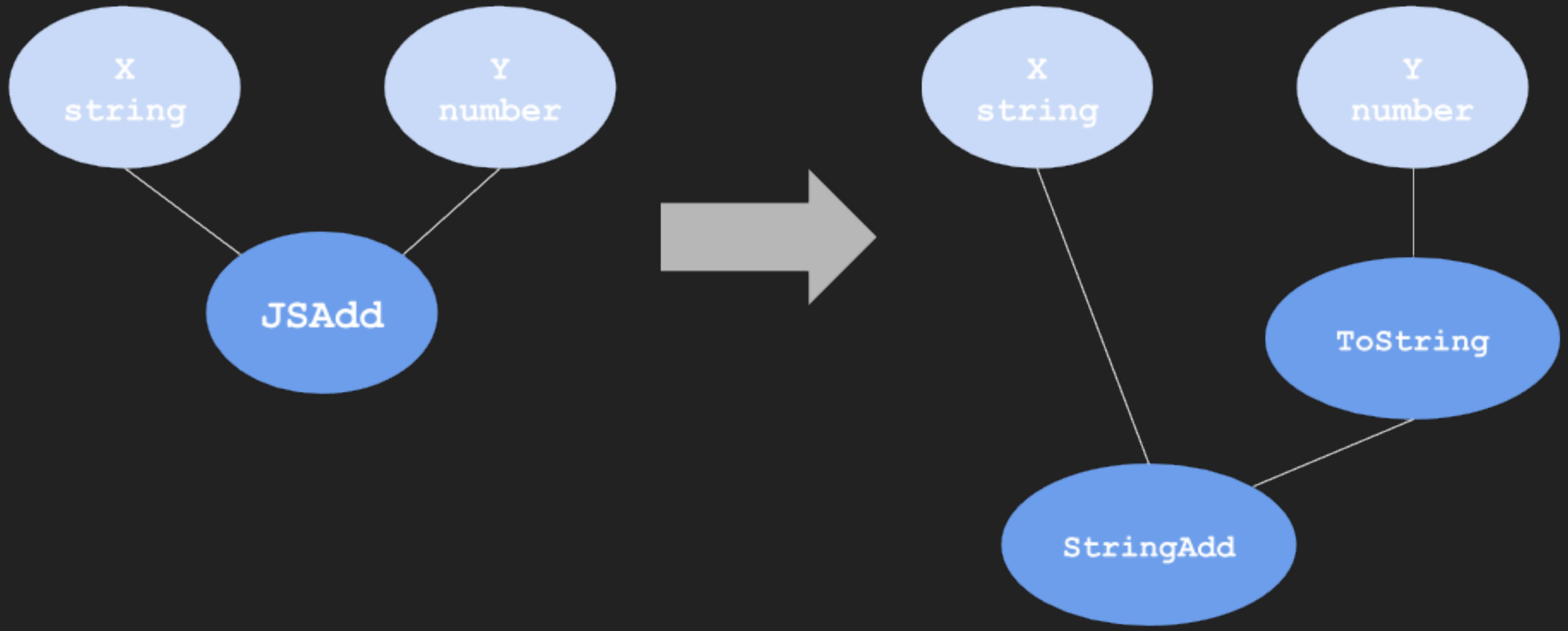
- Classical optimizations in reducers
  - Strength reduction
  - Constant folding
  - Type and range analysis
  - Typed lowering

REDUCTIONS- STRENGTH REDUCTION



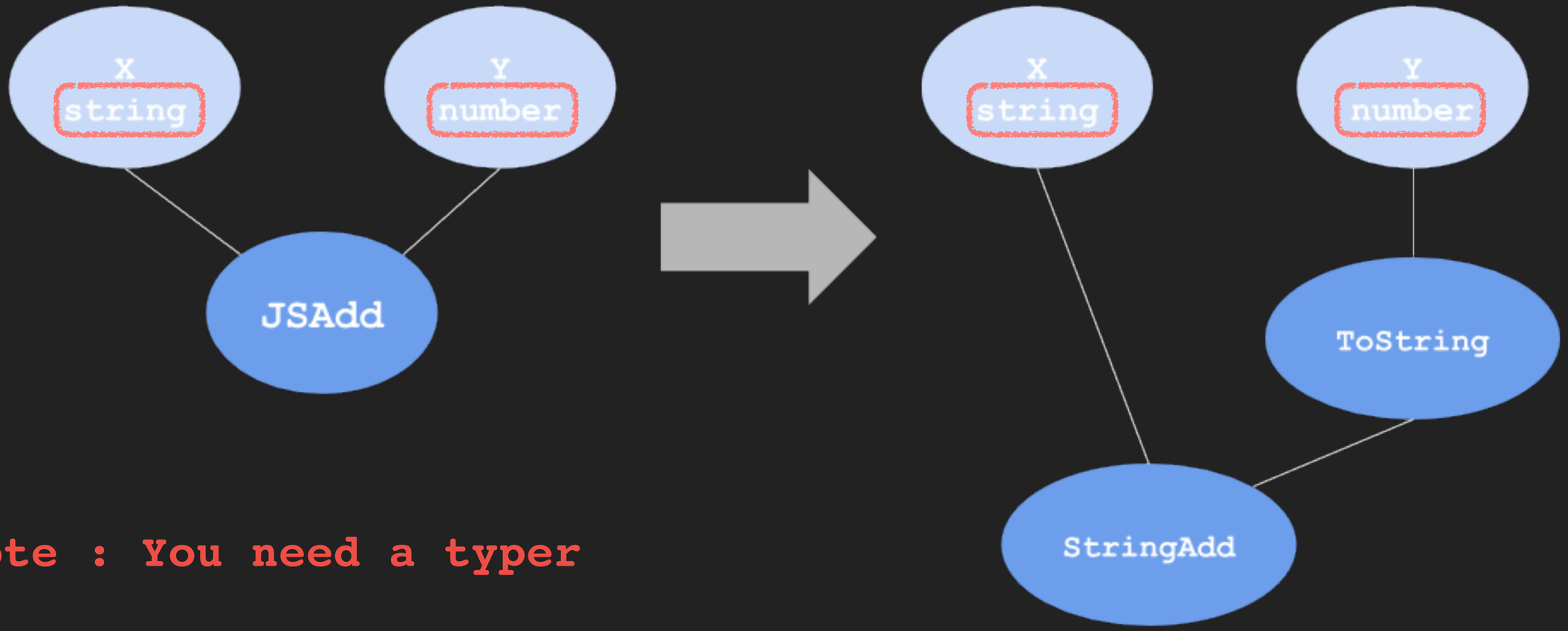
Recommended reading : TurboFan JIT Design, Ben Titzer

TYPED LOWERING



Recommended reading : TurboFan JIT Design, Ben Titzer

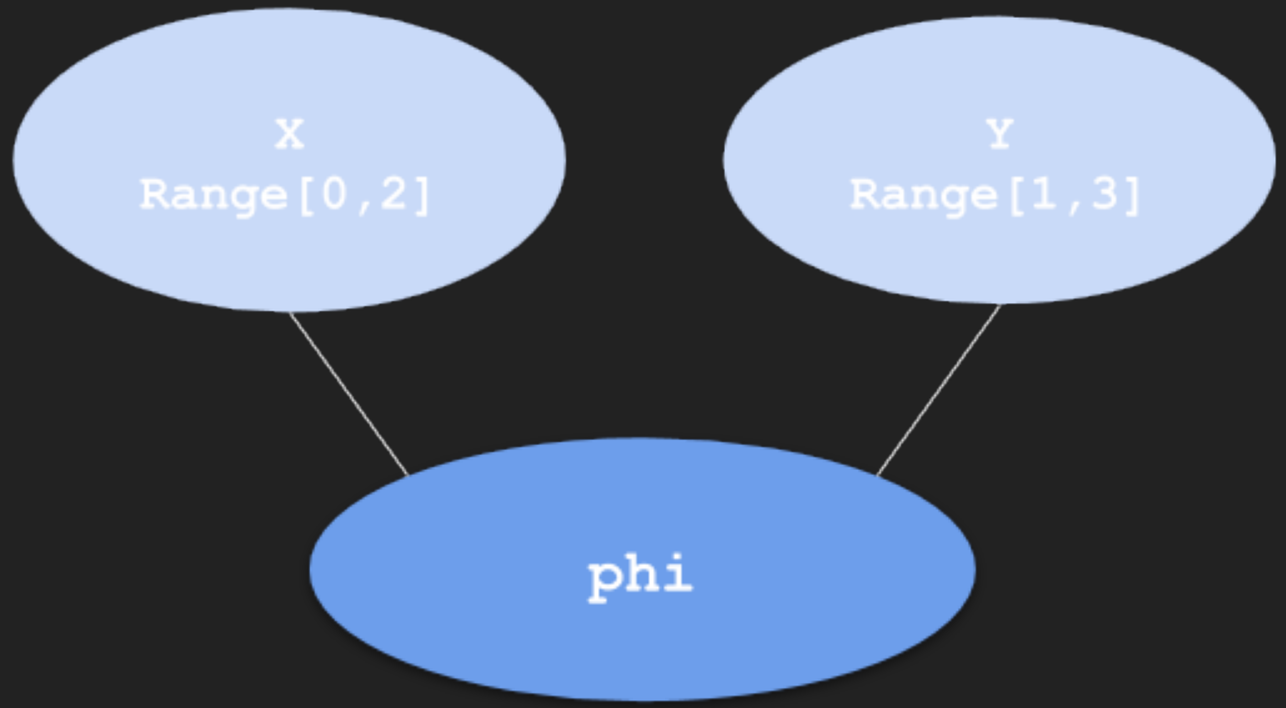
TYPED LOWERING



Note : You need a typer

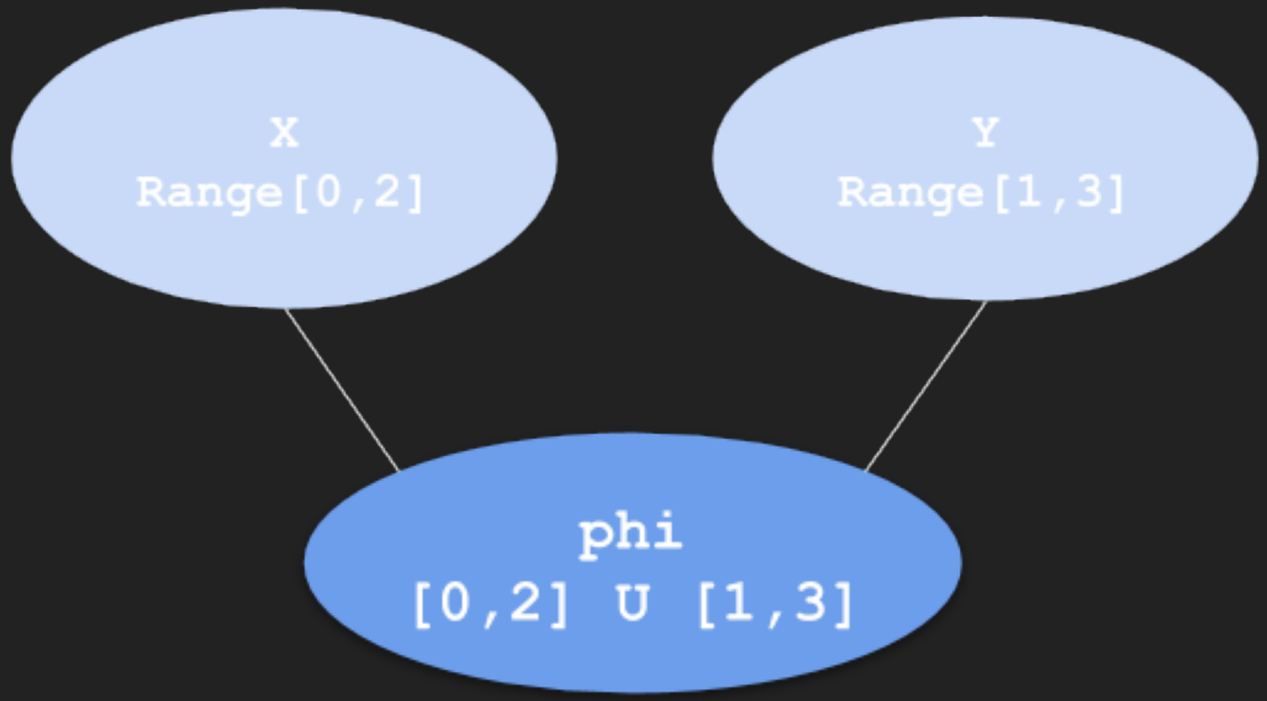
Recommended reading : TurboFan JIT Design, Ben Titzer

RANGE ANALYSIS



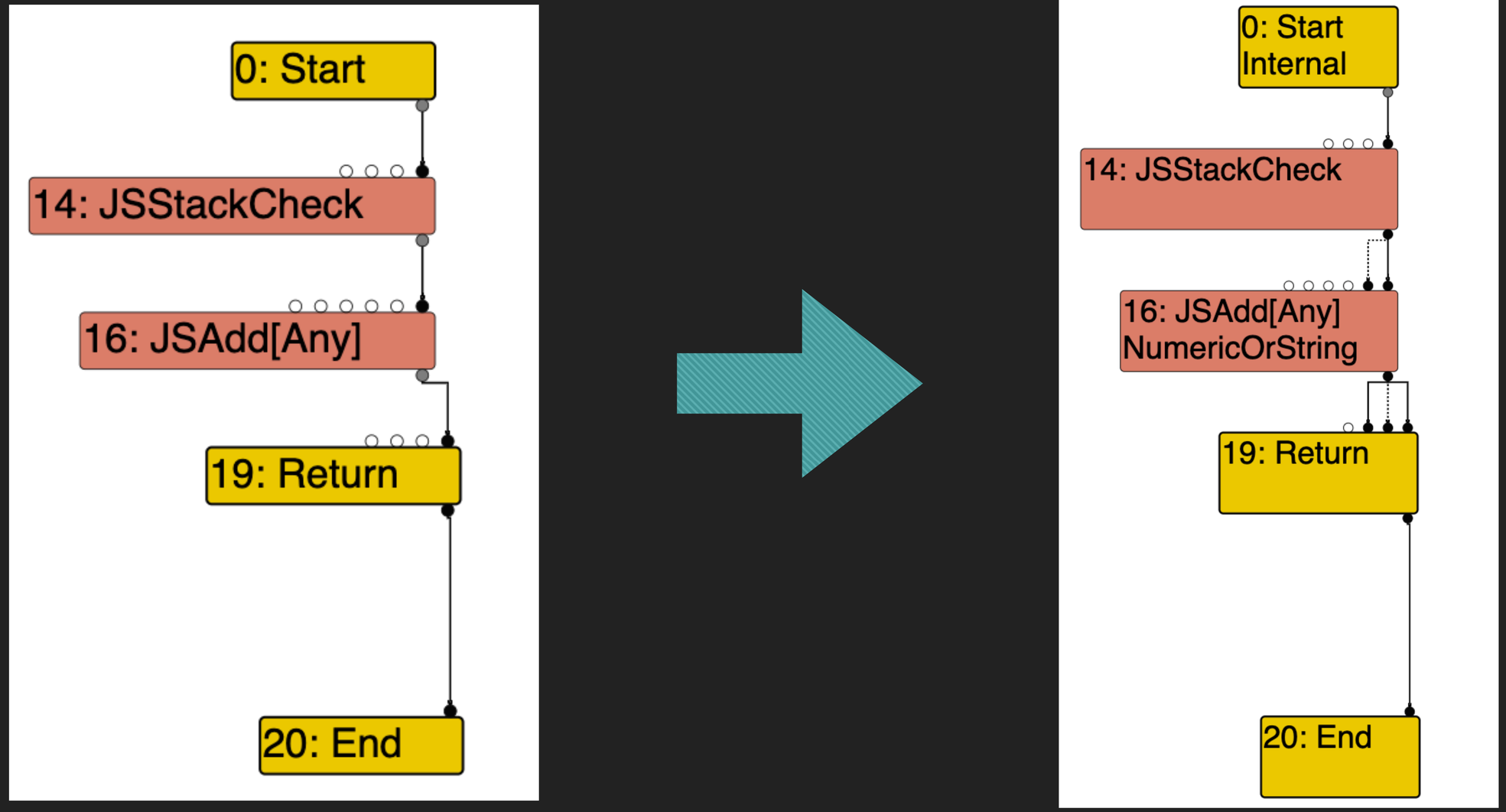
Recommended reading : TurboFan JIT Design, Ben Titzer

RANGE ANALYSIS

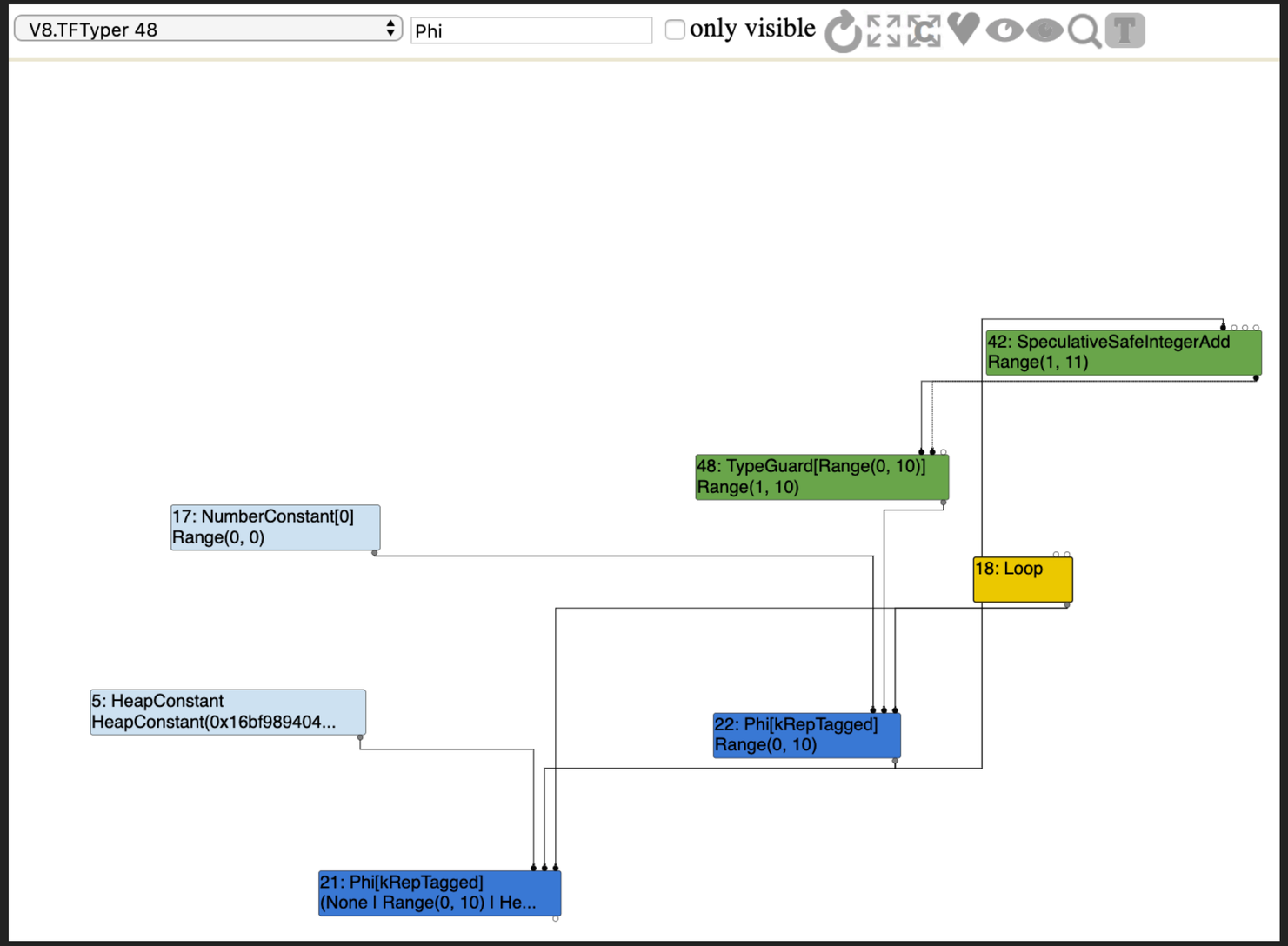


Recommended reading : TurboFan JIT Design, Ben Titzer

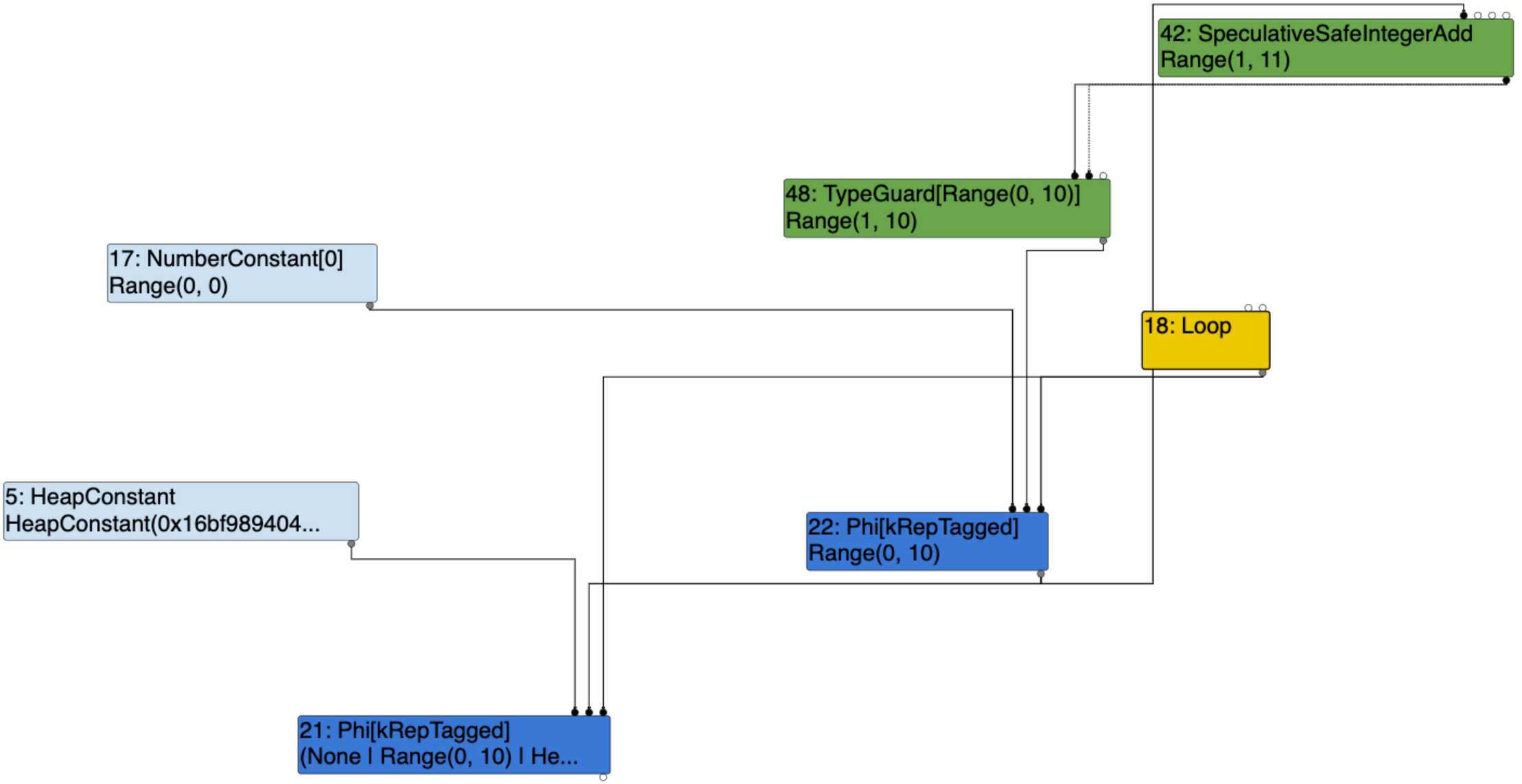
TURBOLIZER – BYTECODE GRAPH BUILDER VS TYPER PHASE



TURBOLIZER – TYPING A PHI NODE



V8.TFTyper 48 | Phi | only visible

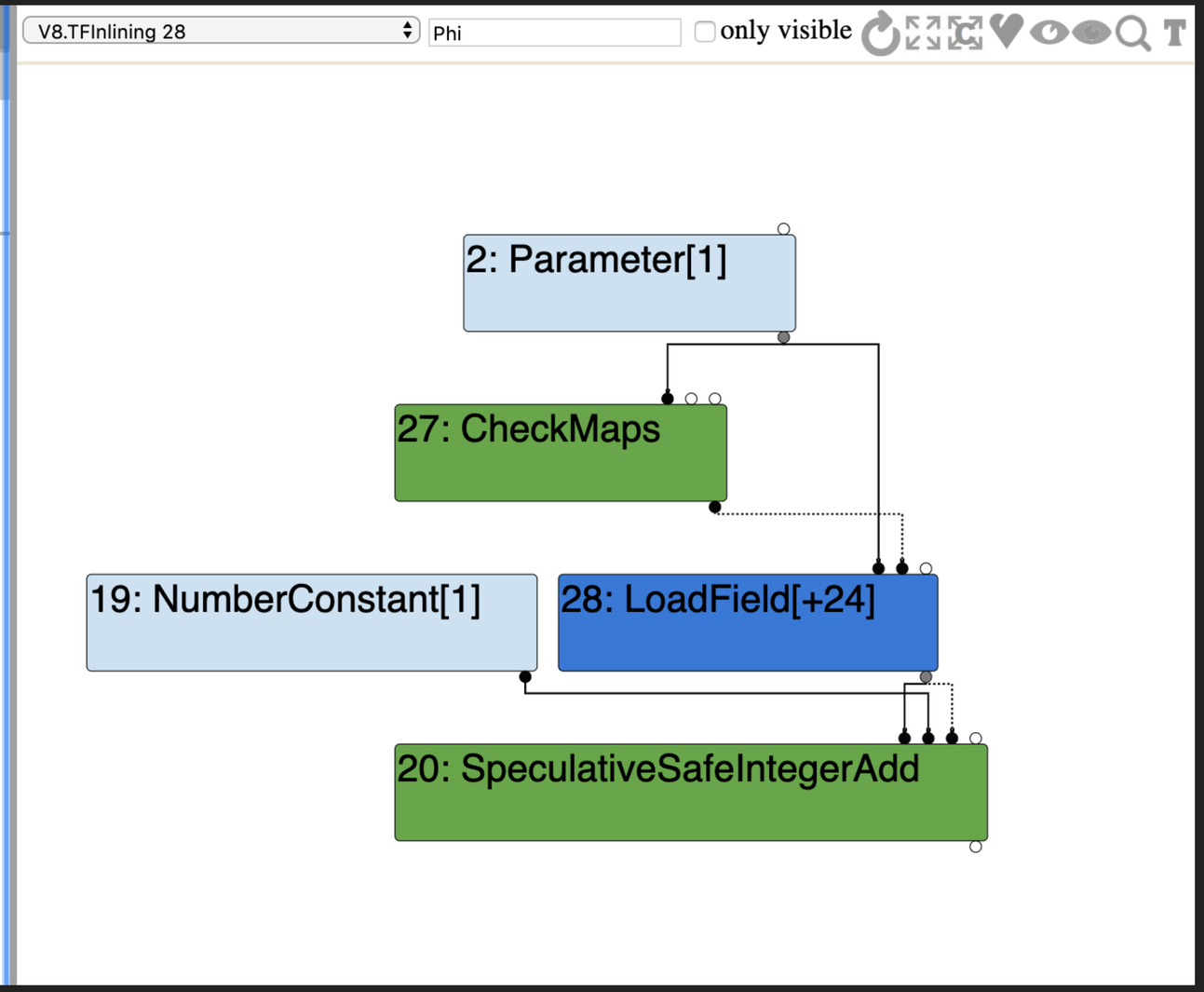


CHECKMAPS

Info Source +

add.js:inc main function

```
1 (o) {
2       return o.x + 1;
3 }
```

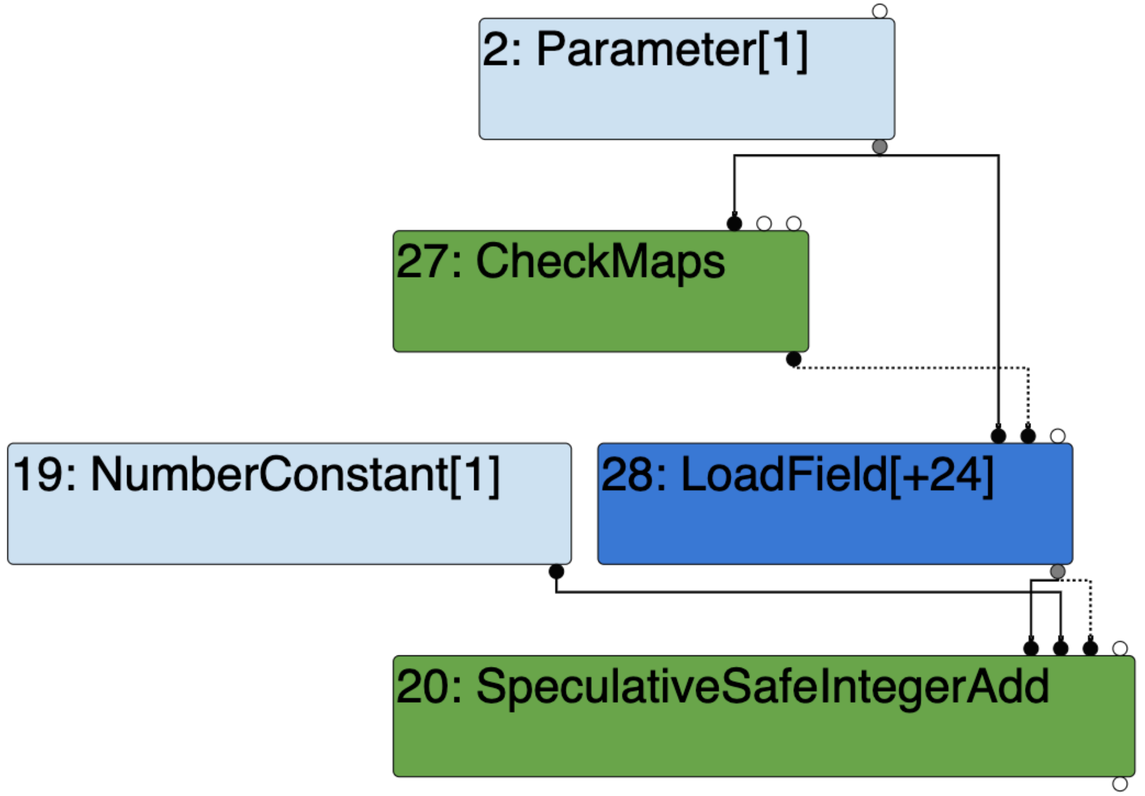


Info | Source | +

add.js:inc — **main function**

```
1 (o) {
2       return o.x + 1;
3 }
```

V8.TFInlining 28 | Phi | only visible



EXAMPLE 1 - MONOMORPHIC INLINE CACHE

```
1 (o) {
2       return o.x + 1;
3 }
```

```
var obj1 = { x : 1 };
var obj2 = { x : 2 };
var obj3 = { x : 3, y : 4 };
inc(obj1);
inc(obj2);
%OptimizeFunctionOnNextCall(inc);
```

```
- slot #1 LoadProperty MONOMORPHIC {
    [1]: [weak] 0x36733204a5e9 <Map(HOLEY_ELEMENTS)>
    [2]: 836
  }
```

EXAMPLE 1 - MONOMORPHIC INLINE CACHE

```
1 (o) {
2       return o.x + 1;
3 }
```

```
var obj1 = { x : 1 };
var obj2 = { x : 2 };
var obj3 = { x : 3, y : 4 };
inc(obj1);
inc(obj2);
%OptimizeFunctionOnNextCall(inc);
inc(obj3);
```

CHECKMAPS BAILS OUT TO DEOPTIMIZATION!

EXAMPLE 1 - MONOMORPHIC INLINE CACHE

```
3f  REX.W movq rcx,0x38af6664a5e9    ;; object: 0x38af6664a5e9 <Map(HOLEY_ELEMENTS)>
49  REX.W cmpq [rdx-0x1],rcx
4d  jnz 0x9daeb842e5f  <+0x9f>
```

```
9f  REX.W movq r13,0x1          ;; debug: deopt position, script offset '28'
                                   ;; debug: deopt position, inlining id '-1'
                                   ;; debug: deopt reason 'wrong map'
                                   ;; debug: deopt index 1
a6  call 0x9daeb882040          ;; eager deoptimization bailout
```

EXAMPLE 1 - MONOMORPHIC INLINE CACHE

```
$ d8 add.js  --trace-deopt
[deoptimizing (DEOPT eager): begin 0x3554bc6dfa41 <JSFunction inc (sfi = 0x3554bc6df6e1)> (opt #0) @1, FP to SP delta: 24, caller sp: 0x7ffee3bbced0]
;;; deoptimize at <add.js:2:11>, wrong map
  reading input frame inc => bytecode_offset=0, args=2, height=1, retval=0(#0); inputs:
      0: 0x3554bc6dfa41 ;  [fp -  16]  0x3554bc6dfa41 <JSFunction inc (sfi = 0x3554bc6df6e1)>
      1: 0x35541eb01599 ;  [fp +  24]  0x35541eb01599 <JSGlobal Object>
      2: 0x35541eb0bba1 ; rdx 0x35541eb0bba1 <Object map = 0x3554ad30a689>
      3: 0x3554bc6c1851 ;  [fp -  24]  0x3554bc6c1851 <NativeContext[249]>
      4: 0x35543d1c0e19 ; (literal  2) 0x35543d1c0e19 <Odd Oddball: optimized_out>
  translating interpreted frame inc => bytecode_offset=0, height=8
    0x7ffee3bbcec8: [top +  64] <- 0x35541eb01599 <JSGlobal Object> ;  stack parameter (input #1)
    0x7ffee3bbcec0: [top +  56] <- 0x35541eb0bba1 <Object map = 0x3554ad30a689> ;  stack parameter (input #2)
    -------------------------
    0x7ffee3bbceb8: [top +  48] <- 0x00010c94e6a4 ;  caller's pc
    0x7ffee3bbceb0: [top +  40] <- 0x7ffee3bbcf10 ;  caller's fp
    0x7ffee3bbcea8: [top +  32] <- 0x3554bc6c1851 <NativeContext[249]> ;  context (input #3)
    0x7ffee3bbcea0: [top +  24] <- 0x3554bc6dfa41 <JSFunction inc (sfi = 0x3554bc6df6e1)> ;  function (input #0)
    0x7ffee3bbce98: [top +  16] <- 0x3554bc6dfbd9 <BytecodeArray[9]> ;  bytecode array
    0x7ffee3bbce90: [top +   8] <- 0x003500000000 <Smi 53> ;  bytecode offset
    -------------------------
    0x7ffee3bbce88: [top +   0] <- 0x35543d1c0e19 <Odd Oddball: optimized_out> ;  accumulator (input #4)
[deoptimizing (eager): end 0x3554bc6dfa41 <JSFunction inc (sfi = 0x3554bc6df6e1)> @1 => node=0, pc=0x00010c94ea80, caller sp=0x7ffee3bbced0, took 0.182 ms]
```

EXAMPLE 2 - POLYMORPHIC INLINE CACHE

```
1 (o) {
2       return o.x + 1;
3 }
```

```
var obj1 = { x : 1 };
var obj2 = { x : 2 };
var obj3 = { x : 3, y : 4 };
inc(obj1);
inc(obj2);
inc(obj3);
%OptimizeFunctionOnNextCall(inc);
inc(obj3);
```

```
- slot #1 LoadProperty POLYMORPHIC {
    [1]: 0x209f8b14bc69 <Other heap object (WEAK_FIXED_ARRAY_TYPE)>
    [2]: 0x209f62284a39 <Symbol: (uninitialized_symbol)>
  }
```

EXAMPLE 2 - POLYMORPHIC INLINE CACHE

```
67  REX.W movq rdi,0x6829eeca5e9     ;; object: 0x06829eeca5e9 <Map(HOLEY_ELEMENTS)>
71  REX.W cmpq rdi,rcx
74  jz 0x2608320c2e4d  <+0x8d>
7a  REX.W movq rdi,0x6829eeca689     ;; object: 0x06829eeca689 <Map(HOLEY_ELEMENTS)>
84  REX.W cmpq rdi,rcx
87  jnz 0x2608320c2ebd  <+0xfd>
```

```
fd  REX.W movq r13,0x1       ;; debug: deopt position, script offset '28'
                                        ;; debug: deopt position, inlining id '-1'
                                        ;; debug: deopt reason 'wrong map'
                                        ;; debug: deopt index 1
104 call 0x260832102040      ;; eager deoptimization bailout
```

TYPER BUGS

OVERVIEW

- Typer
- Operation Typer
- Type cache

STRING LAST INDEX

- Bug found by tsuro

```
Type::Range(-1.0, String::kMaxLength - 1, t->zone());
```

```
Type::Range(-1.0, String::kMaxLength, t->zone());
```

EXPM1 - TYPER - OPERATION TYPER

- Bug found by tsuro,
- Nice write-ups by Andrea Biondo and Jay Bosamiya
- Present in both typer and operation typer
  - Typer for JSCall to builtin Math.expm1
  - Operation typer for nodes Expm1

https://abiondo.me/2019/01/02/exploiting-math-expm1-v8/

https://www.jaybosamiya.com/blog/2019/01/02/krautflare/

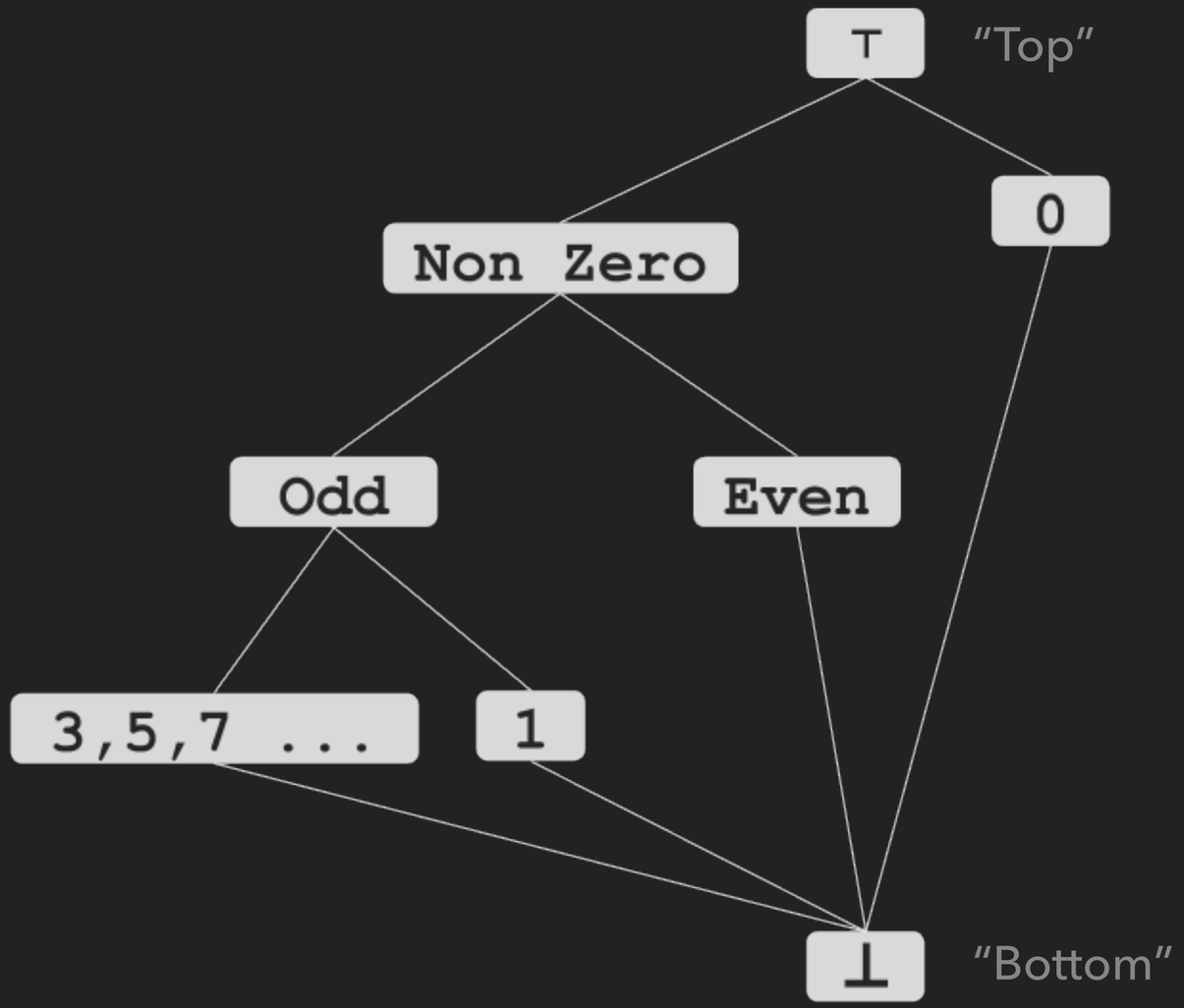
EXPM1 - TYPER - OPERATION TYPER

```
Type::Union(Type::PlainNumber(), Type::NaN(), t->zone())
```

```
Type::Number()
```

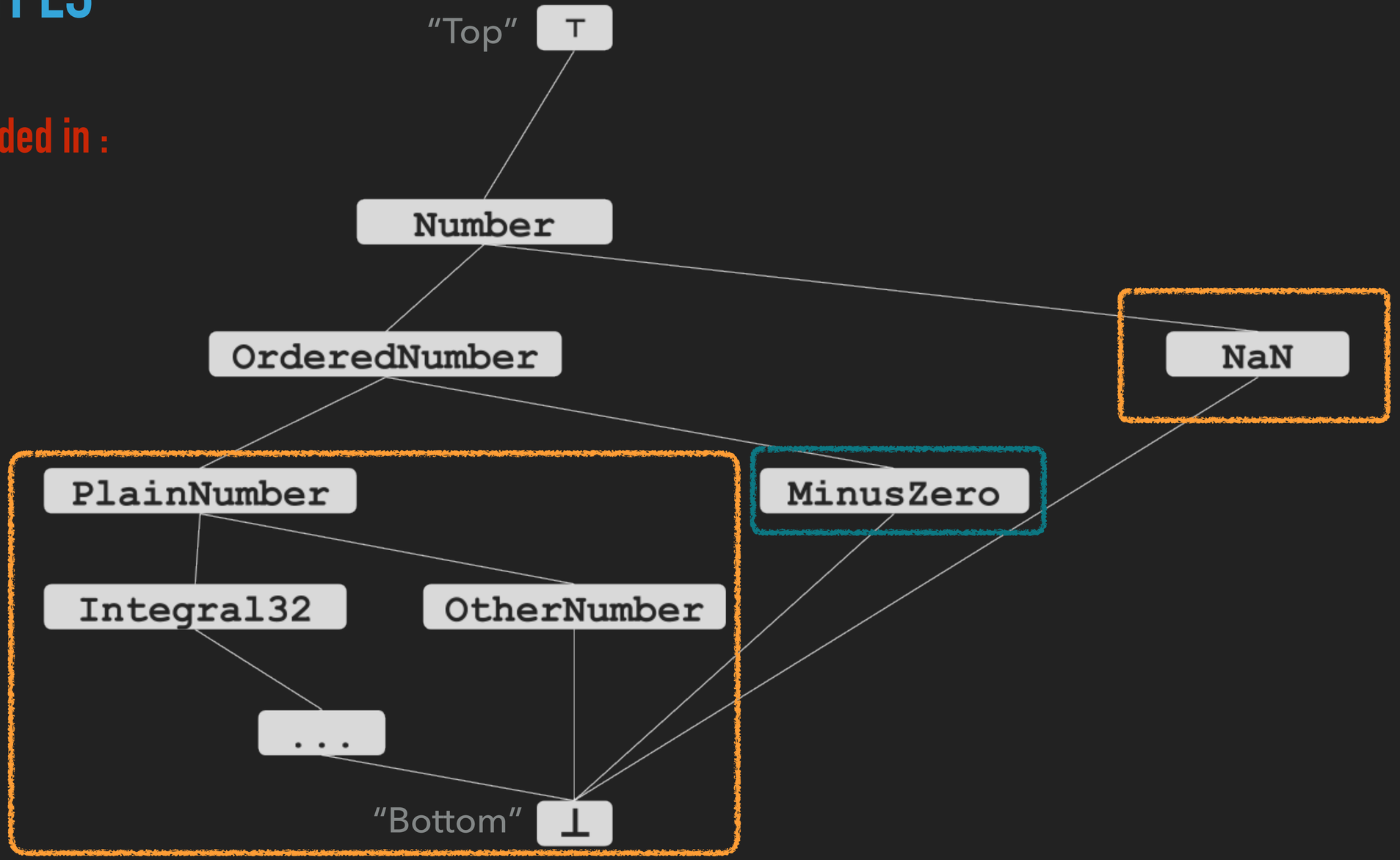
```
d8> Object.is(-0, Math.expm1(-0))
true
```

A LATTICE OF TYPES – WHAT IS A LATTICE?



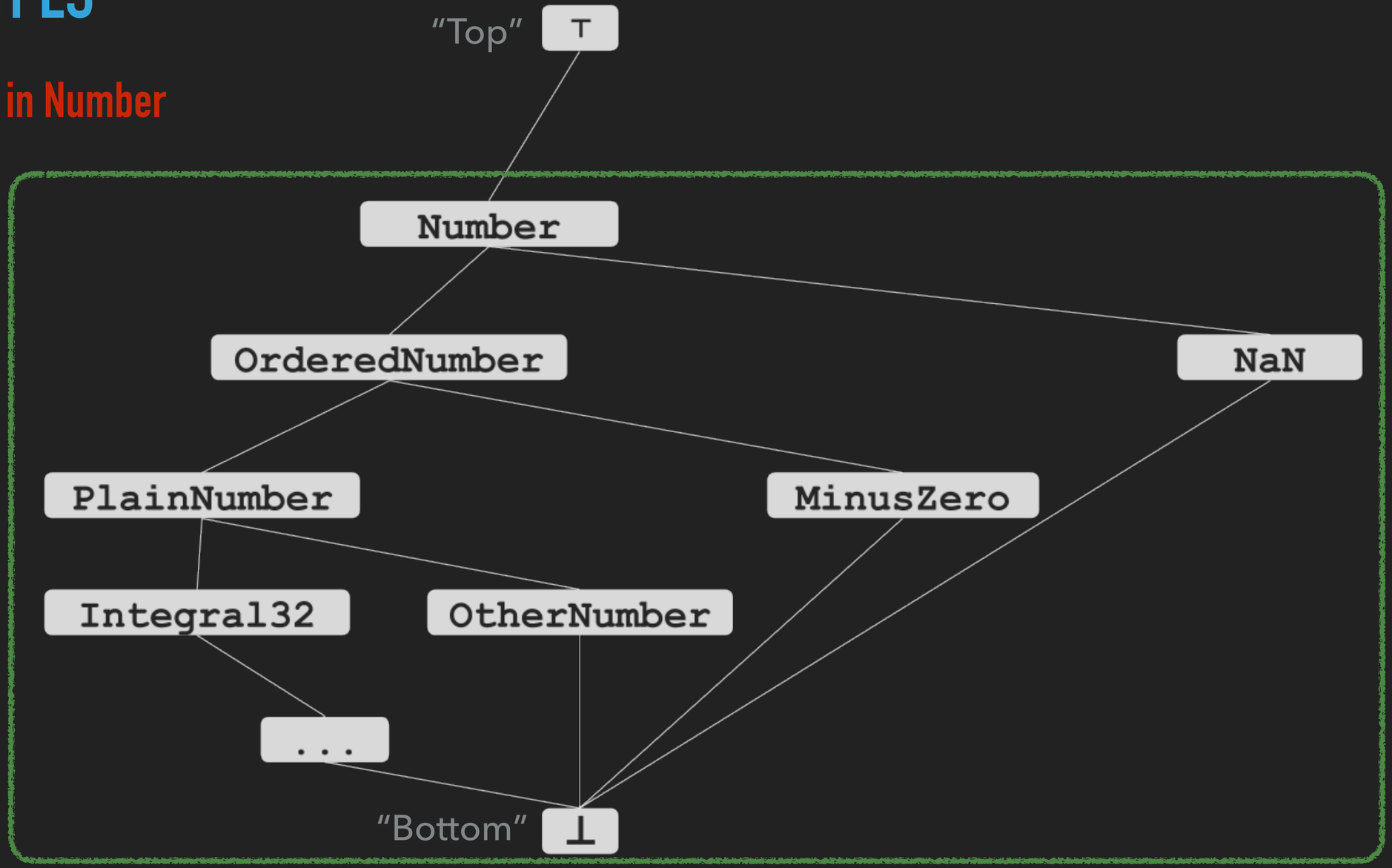
A LATTICE OF TYPES

MinusZero is not included in :
- PlainNumber
- NaN



A LATTICE OF TYPES

MinusZero is included in Number



SPEAKING OF WHICH – WHAT ABOUT TOP AND BOTTOM?

- Top is Type::Any
  - Type::Any maybe any type
- Bottom is Type::None
  - Type::None is any type

SPEAKING OF WHICH - WHAT ABOUT TOP AND BOTTOM?

```
none_type.Is(Type::Foo())
```

TAUTOLOGY!

```
Typer::Visitor::ComparisonOutcome Typer::Visitor::NumberCompareTyper(Type lhs,
                                                                     Type rhs,
                                                                     Typer* t) {
  DCHECK(lhs.Is(Type::Number()));
  DCHECK(rhs.Is(Type::Number()));

  // if (lhs.IsNone() || rhs.IsNone()) return {};

  // Shortcut for NaNs.
  if (lhs.Is(Type::NaN()) || rhs.Is(Type::NaN())) return kComparisonUndefined;
```

SPEAKING OF WHICH - WHAT ABOUT TOP AND BOTTOM?

```
none_type.Is(Type::Foo())
```

TAUTOLOGY!

```
Type Typer::Visitor::JSEqualTyper(Type lhs, Type rhs, Typer* t) {
  // if (lhs.IsNone() || rhs.IsNone()) return Type::None();
  if (lhs.Is(Type::NaN()) || rhs.Is(Type::NaN())) return t->singleton_false_;
  // […]
}
```

SPEAKING OF WHICH - WHAT ABOUT TOP AND BOTTOM?

```
any_type.Maybe(Type::Foo())
```

TAUTOLOGY!

```
Type Typer::Visitor::JSAddTyper(Type lhs, Type rhs, Typer* t) {
  lhs = ToPrimitive(lhs, t);
  rhs = ToPrimitive(rhs, t);
  if (lhs.Maybe(Type::String()) || rhs.Maybe(Type::String())) {
    if (lhs.Is(Type::String()) || rhs.Is(Type::String())) {
      return Type::String();
    } else {
      return Type::NumericOrString();
    }
  }
```

OsrValues are typed as Any

SPECULATIVE SAFE INTEGER SUBTRACTIONS / ADDITIONS

- Bug found by Jay Bosamiya

```
commit 82271defd67347c146a3f71d3aad313f00658b48
Author: Tobias Tebbi <tebbi@chromium.org>
Date:   Thu Nov 16 17:30:23 2017 +0100

    [turbofan] fix typing and lowering of SpeculativeSafeInteger{Add,Subtract}

    Bug:
    Change-Id: Ibd7c17b4ace25237c3d35466280aff27c44016f0
    Reviewed-on: https://chromium-review.googlesource.com/774461
    Reviewed-by: Jaroslav Sevcik <jarin@chromium.org>
    Commit-Queue: Tobias Tebbi <tebbi@chromium.org>
    Cr-Commit-Position: refs/heads/master@{#49427}
```

SPECULATIVE SAFE INTEGER SUBTRACTIONS / ADDITIONS

- Bug found by Jay Bosamiya

```
Type::Intersect(result, cache_.kSafeInteger, zone())
```

```
Type::Intersect(result, cache_.kSafeIntegerOrMinusZero, zone())
```

NUMBER MAX/MIN

```
commit fa54dff255ad747e27c85a7b7ccbf98849401904
Author: Benedikt Meurer <bmeurer@chromium.org>
Date:   Sun Sep 2 21:19:37 2018 +0200

    [turbofan] Add missing -0 support for NumberMax/NumberMin typing.

    The typing rules for NumberMax and NumberMin didn't properly deal with
    -0 up until now, leading to suboptimal typing, i.e. for a simple case
    like

      Math.max(Math.round(x), 1)

    TurboFan was unable to figure out that the result is definitely going
    to be a positive integer in the range [1,inf] or NaN (assuming that
    NumberOrOddball feedback is used for the value x).
```

ASIDE

- Other places impacted by incorrect handling of -0
- Incorrect truncation of **null** in representation change
  - `-0 == null (false)` truncated to `-0 == 0 (true)`
- `Inexistent representation of MinusZero`
  - `Float64 representation`
  - `Tagged representation`

PROMISE CATCH / FINALLY

```
commit 1bfb02471e76958fb5ff454bfbb7a3534495e18a
Author: Benedikt Meurer <bmeurer@chromium.org>
Date:   Mon Nov 26 13:24:46 2018 +0100

    [turbofan] Fix types of Promise#catch() and Promise#finally().

    We cannot assign a meaningful type to Promise#catch() or
    Promise#finally(), since they both return whatever the invocation of
    'then' on the receiver returns, and that is monkeypatchable by arbitrary
    user JavaScript.
```

```
--- a/src/compiler/typer.cc
+++ b/src/compiler/typer.cc
@@ -1644,10 +1644,6 @@ Type Typer::Visitor::JSCallTyper(Type fun, Typer* t) {

     case BuiltinFunctionId::kPromiseAll:
       return Type::Receiver();
-    case BuiltinFunctionId::kPromisePrototypeCatch:
-      return Type::Receiver();
-    case BuiltinFunctionId::kPromisePrototypeFinally:
-      return Type::Receiver();
```

CHROME TIANFU CUP

Chaouki Bekrar @cBekrar · 1 févr.

Look at this beautiful Chrome RCE fixed yesterday in Chrome v72 and used/reported by Qihoo 360 Vulcan Team at Tianfu Cup. Exploitation can be straightforward, fast, and fully reliable using JIT.

PoC: chromium-review.googlesource.com/c/v8/v8/+/1363…

```
commit 8e4588915ba7a9d9d744075781cea114d49f0c7b
Author: Peter Marshall <petermarshall@chromium.org>
Date:   Fri Nov 30 13:21:16 2018 +0100

    [turbofan] Relax range for arguments object length

    Bug: chromium:906043
    Change-Id: I3a397447be186eff7e6b2ab25341718b6c0d205d
    Reviewed-on: https://chromium-review.googlesource.com/c/1356507
    Commit-Queue: Peter Marshall <petermarshall@chromium.org>
    Reviewed-by: Jaroslav Sevcik <jarin@chromium.org>
    Cr-Commit-Position: refs/heads/master@{#57965}
```

CHROME TIANFU CUP

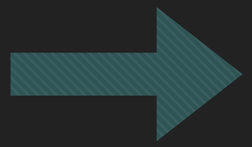
```
TypeCache   : type of ArgumentsLength nodes
```

```
Type::Range(0.0, Code::kMaxArguments, zone())
```

```
CreateRange(0.0, FixedArray::kMaxLength)
```

CHROME TIANFU CUP

```
static const int kMaxArguments = (1 << kArgumentsBits) - 2;
static const int kArgumentsBits = 16;
```



```
kMaxArguments = (1 << 16) - 2 = 0x10000 - 2
```

CHROME TIANFU CUP

- TurboFan type cache for ArgumentsLength nodes
  - (1 << 16) - 2
- What if we use a spread operator in an array to pass arguments?

```
array.length = (1 << 16)
funct(…array)
```

TURBOFAN

(1 << 16) - 2

ACTUAL EXECUTION

(1 << 16)

CHROME TIANFU CUP

```
let fun = () => {
  let x = arguments.length;
  a = new Array(0x10)
  a[0] = 1.1;
  a[(x >> 16) * 0x10] = 42.42;
}
```

TURBOFAN

`a[0]`

ACTUAL EXECUTION

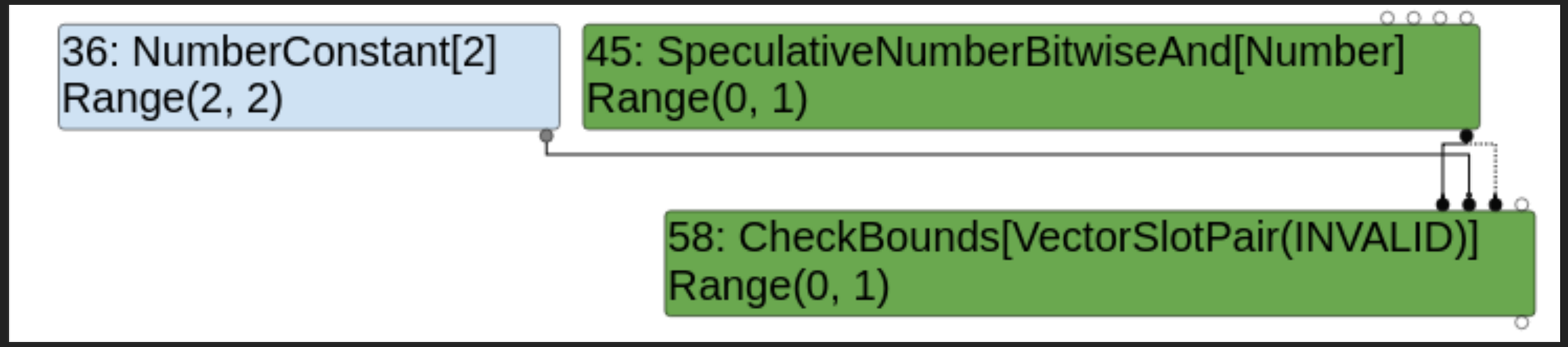
`a[0x10]`

EXPLOITATION

TYPER BUGS – ABUSING BCE

- Incorrect typing
- Propagated to other reductions
- What about BCE during simplified lowering phase?

BCE - SIMPLIFIED LOWERING




```
if ((index_type.Min() >= 0.0 &&
     index_type.Max() < length_type.Min()
    ))

    DeferReplacement(node, node->InputAt(0));
```

BCE REMOVED - ABORTING BOUND CHECKS

Issue 8806: Harden turbofan's bounds check against typer bugs

Reported by jarin@google.com on Thu, Feb 7, 2019, 3:52 PM GMT+1

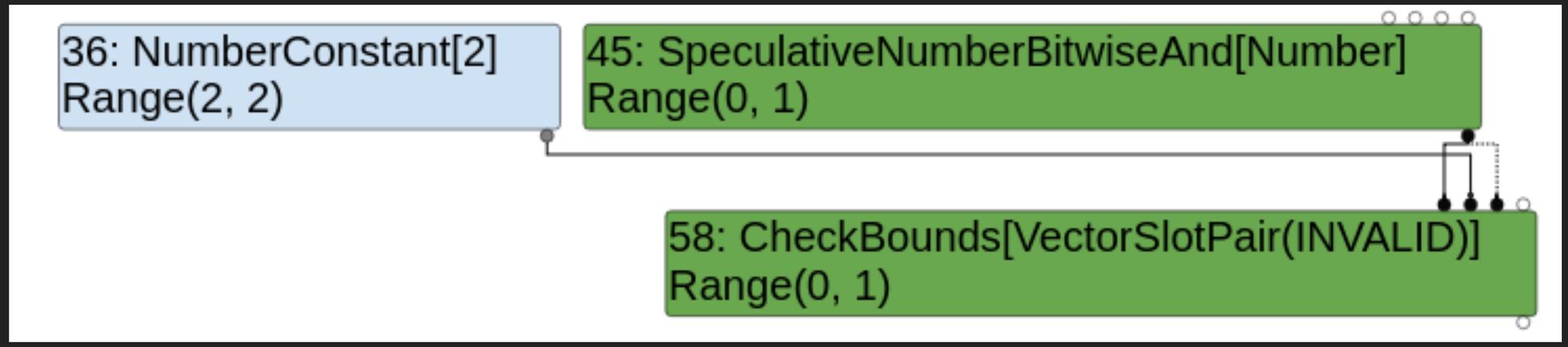
Rather than eliminating bounds checks, just abort if they do not pass.

```
commit 7bb6dc0e06fa158df508bc8997f0fce4e33512a5
Author: Jaroslav Sevcik <jarin@chromium.org>
Date:   Fri Feb 8 16:26:18 2019 +0100

    [turbofan] Introduce aborting bounds checks.

    Instead of eliminating bounds checks based on types, we introduce
    an aborting bounds check that crashes rather than deopts.
```

BCE REMOVED - ABORTING BOUND CHECKS



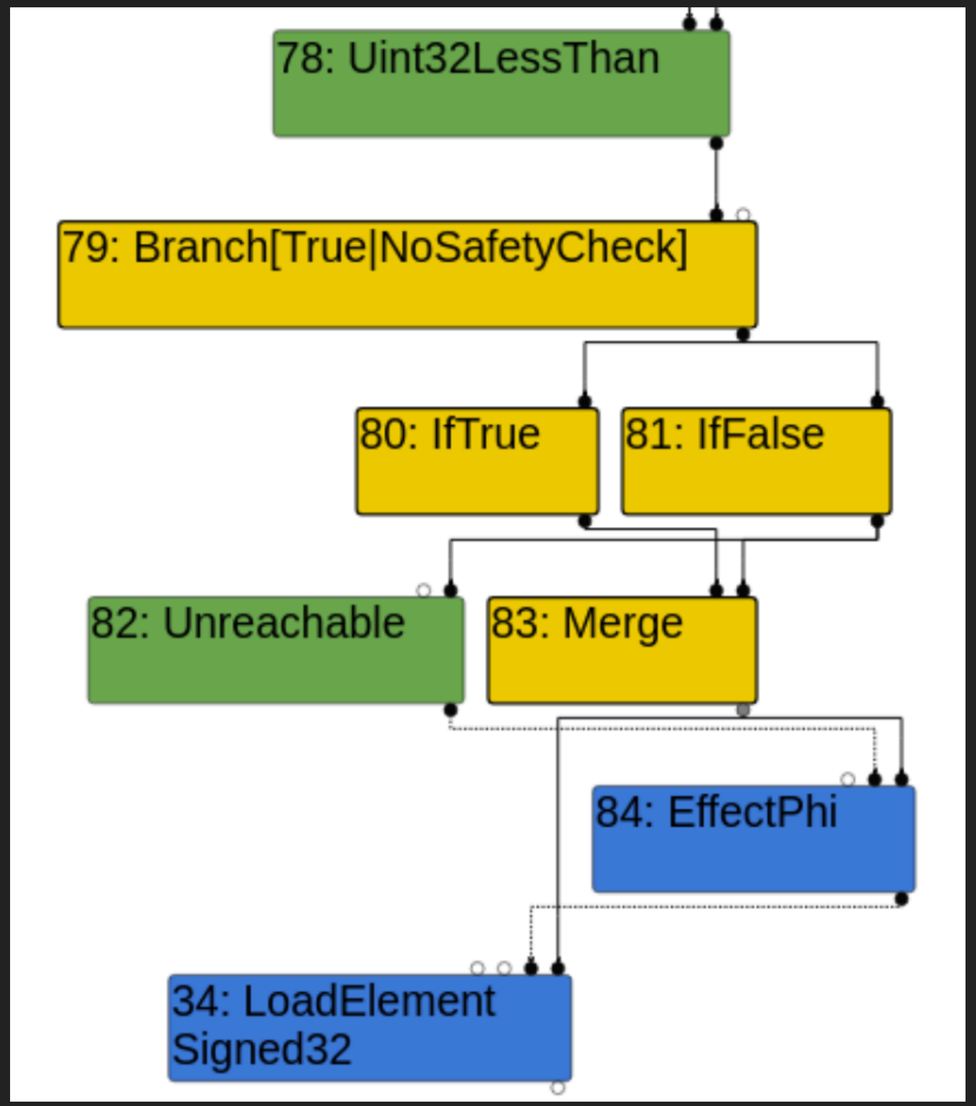

```
if ((index_type.Min() >= 0.0 &&
     index_type.Max() < length_type.Min()
    ))
```

```
mode = CheckBoundsParameters::kAbortOnOutOfBounds;
```

LOAD_IGNORE_OUT_OF_BOUNDS

```
CheckBounds(index, Smi::kMaxValue)
                +
NumberLessThan(index, length)
```

What if we propagate a bad typing to NumberLessThan?



https://doar-e.github.io/blog/2019/05/09/circumventing-chromes-hardening-of-typer-bugs/

TYPE NARROWING, CONSTANT FOLDING AND DEAD CODE ELIMINATION

- Type narrowing reducer
  - NumberLessThan : can be **typed as true or false**
- Constant folding reducer
  - A node typed as **true or false will be replaced by a HeapConstant**
- Dead code elimination reducer
  - Will remove the branch that is believed not to be taken
- TLDR : bound checks have been optimised out

https://doar-e.github.io/blog/2019/05/09/circumventing-chromes-hardening-of-typer-bugs/

MAPS, ELEMENTS KIND AND EFFECT CHAINS*

* I was lacking inspiration for this title :-)

JSCALL REDUCER AND UNRELIABLE MAPS

- Recent blogpost by tsuro from Project Zero

```
commit e80082bf549aa26d6e30f114a23a05df9c510849
Author: Georg Neis <neis@chromium.org>
Date:   Thu Mar 21 10:55:18 2019 +0100

    [turbofan] Add missing map checks in a reducer

    ReduceArrayIndexOfIncludes didn't account for kUnreliableReceiverMaps.
    Will think about a more robust mechanism for this.
```

https://googleprojectzero.blogspot.com/2019/05/trashing-flow-of-data.html

FIXING BY ADDING A CHECKMAPS

JSCall reducer of ArrayIndexOf / ArrayIncludes during inlining phase

```
+  // If we have unreliable maps, we need a map check.
+  if (result == NodeProperties::kUnreliableReceiverMaps) {
+    effect =
+        graph()->NewNode(simplified()->CheckMaps(CheckMapsFlag::kNone,
+                                                 receiver_maps, p.feedback()),
+                         receiver, effect, control);
+  }
+
```

1. INFERRING THE RECEIVER MAPS

```
ZoneHandleSet<Map> receiver_maps;
NodeProperties::InferReceiverMapsResult result =
    NodeProperties::InferReceiverMaps(broker(), receiver, effect,
                                      &receiver_maps);
if (result == NodeProperties::kNoReceiverMaps) return NoChange();
```

2. COMPUTING THE ELEMENTS KIND

```
ElementsKind kind;
if (!CanInlineArrayIteratingBuiltin(broker(), receiver_maps, &kind)) {
  return NoChange();
}
```

3. USING THE ELEMENTS KIND

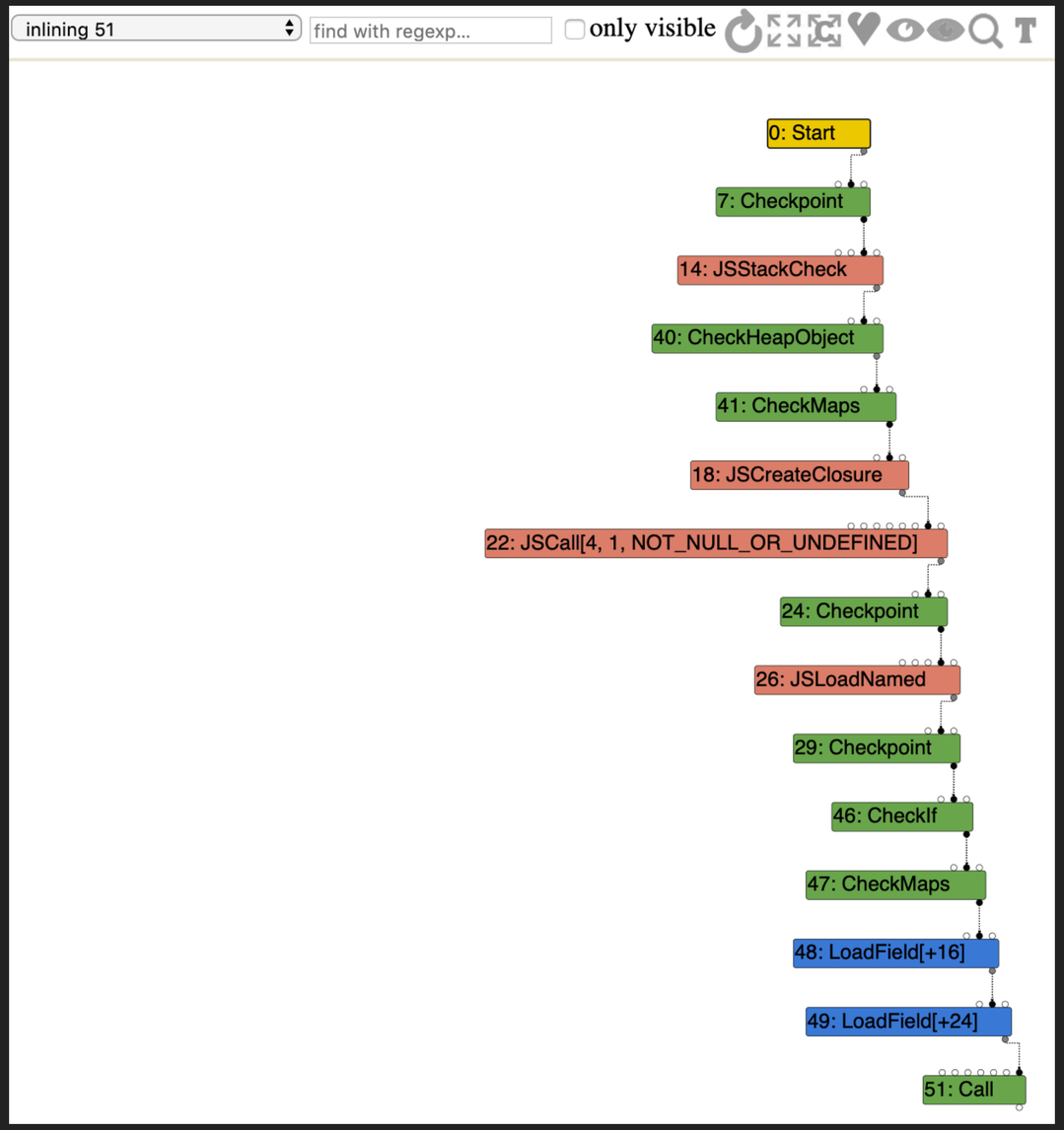
```
Callable const callable = search_variant == SearchVariant::kIndexOf
                              ? GetCallableForArrayIndexOf(kind, isolate())
                              : GetCallableForArrayIncludes(kind, isolate());
```

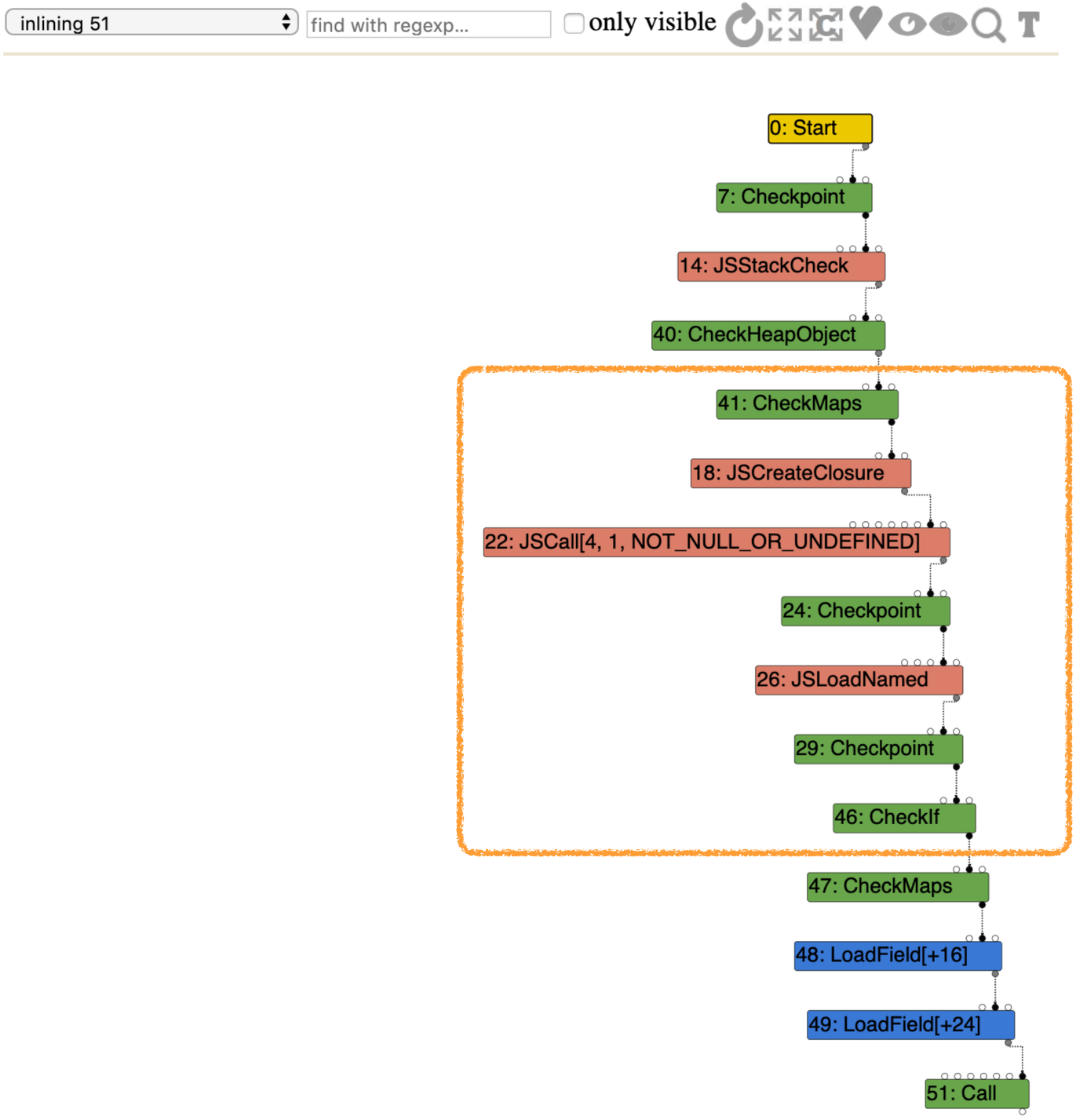
```
Node* length = effect = graph()->NewNode(
    simplified()->LoadField(AccessBuilder::ForJSArrayLength(kind)), receiver,
    effect, control);
```

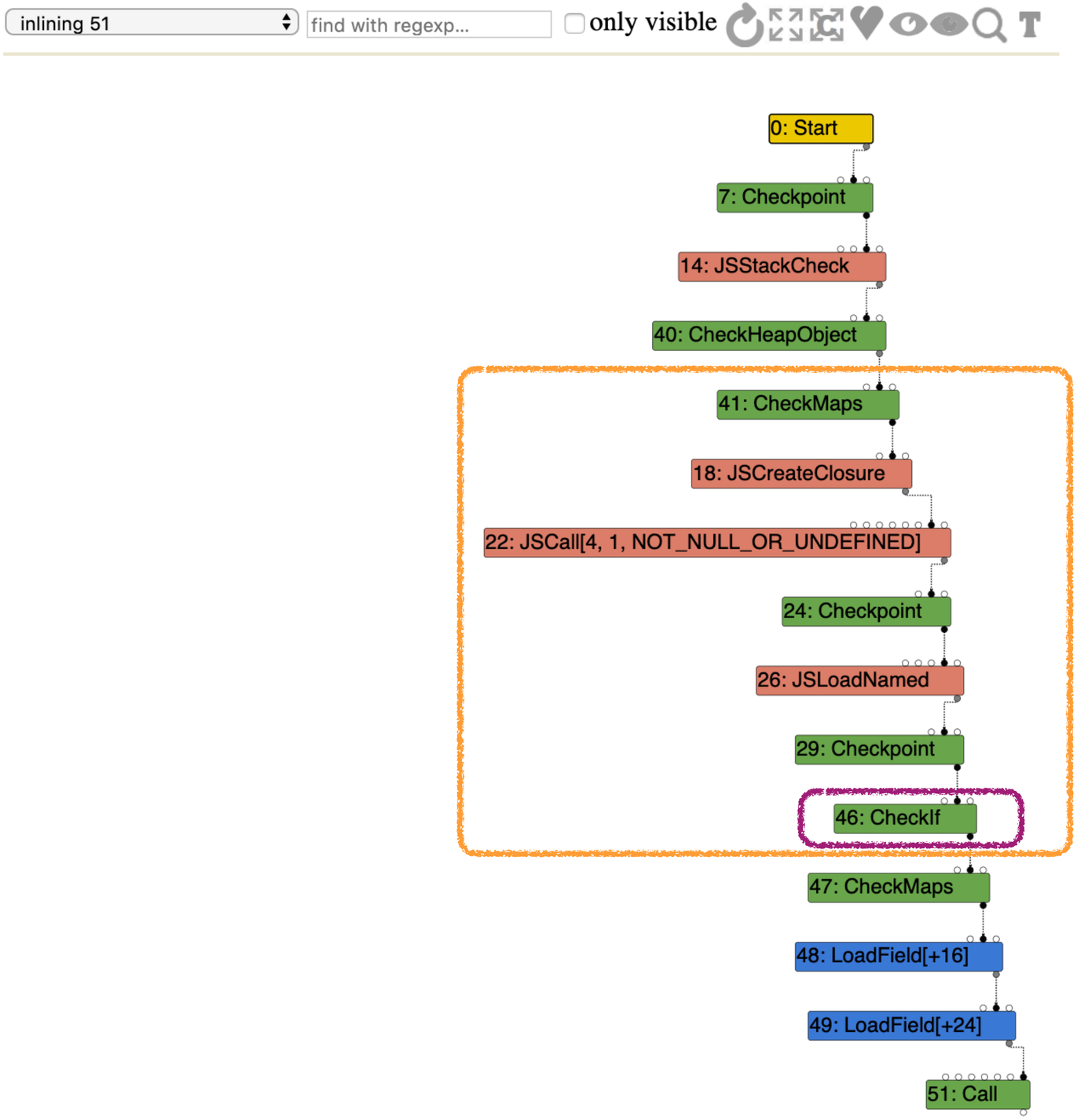
BUT HOW DO WE INFER THE RECEIVER MAPS?

- Walks up the effect chain
- Search for information concerning the map of the receiver
- 3 possible outcomes
    1. No map information has been found
    2. Receiver maps can be fully trusted
    3. Receiver maps may have changed

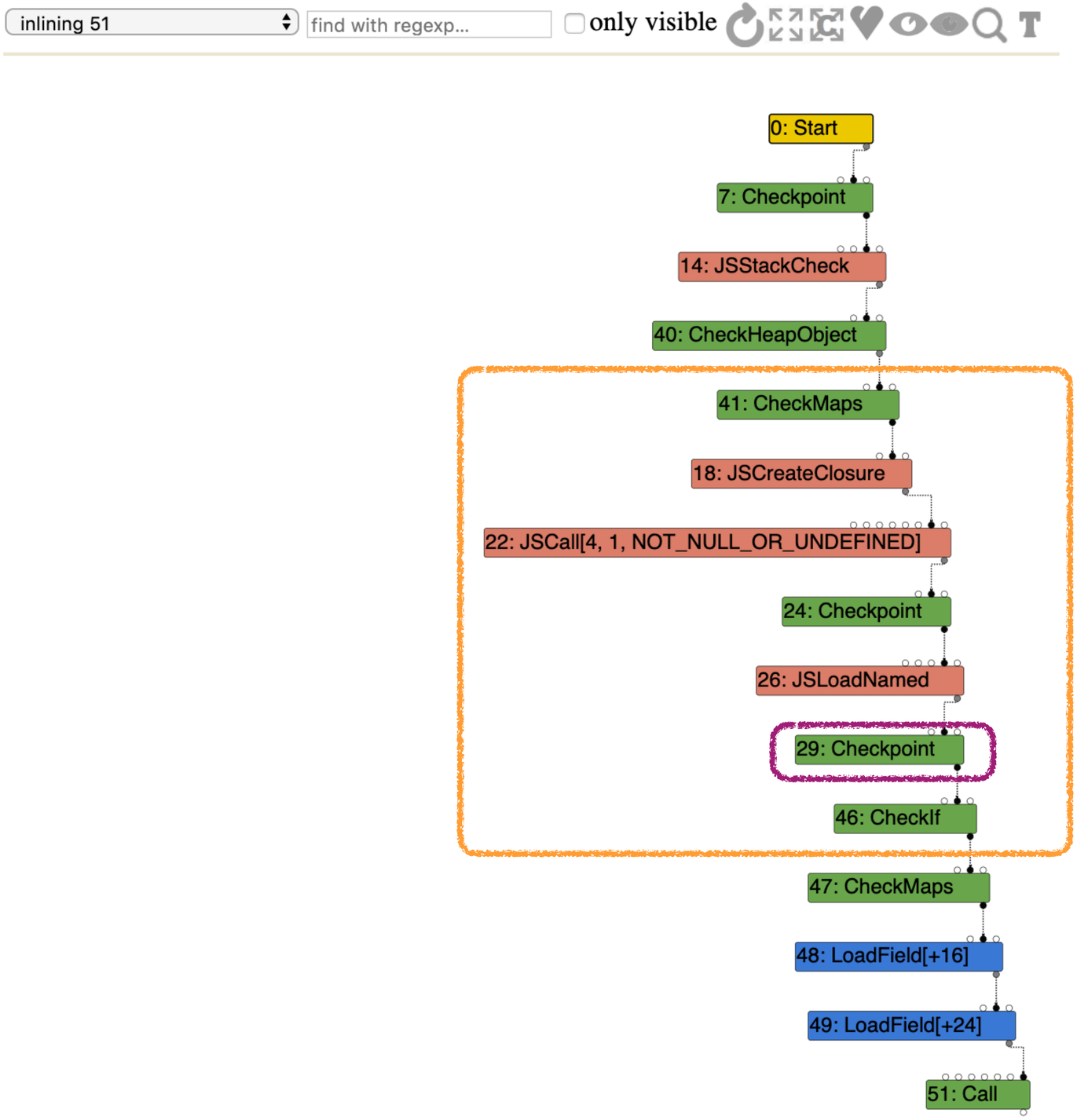
THE EFFECT CHAIN




inlining 51
find with regexp...
only visible
0: Start
7: Checkpoint
14: JSStackCheck
40: CheckHeapObject
41: CheckMaps
18: JSCreateClosure
22: JSCall[4, 1, NOT_NULL_OR_UNDEFINED]
24: Checkpoint
26: JSLoadNamed
29: Checkpoint
46: CheckIf
47: CheckMaps
48: LoadField[+16]
49: LoadField[+24]
51: Call

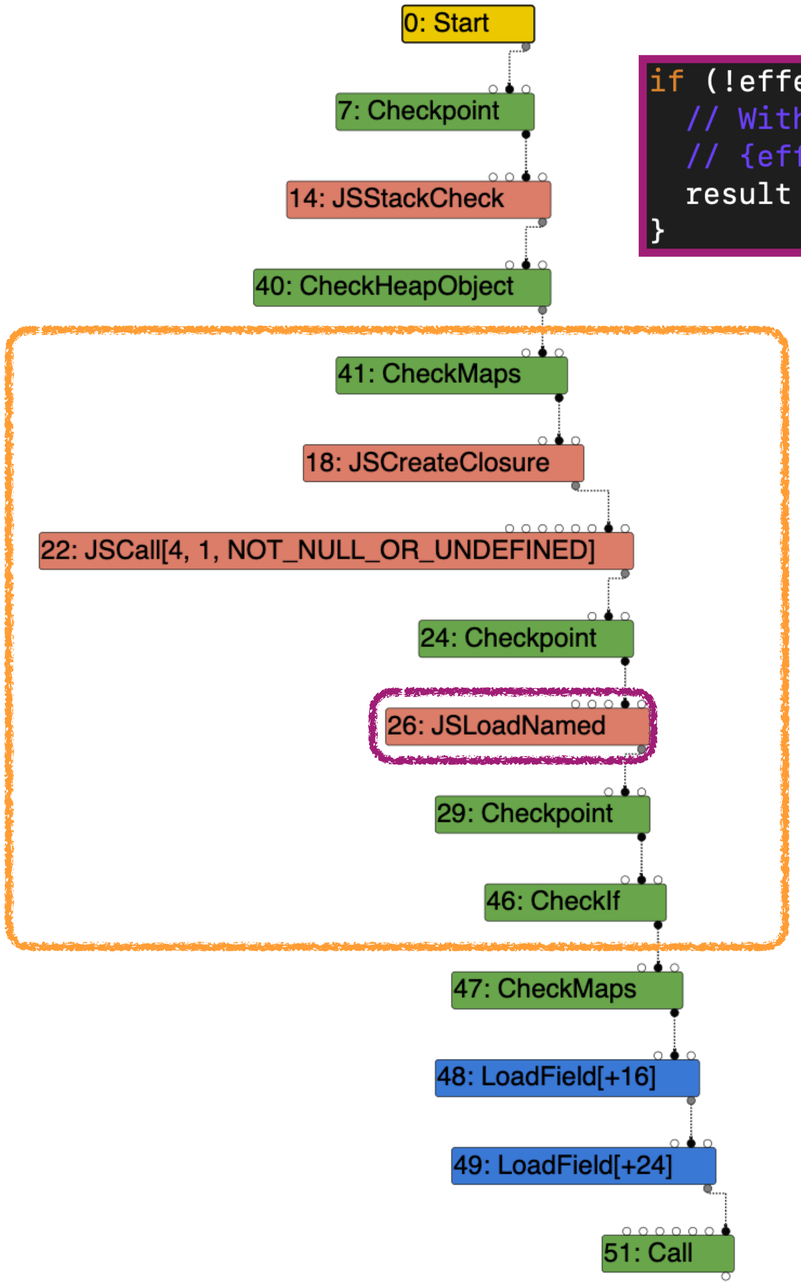


kReliableReceiverMaps



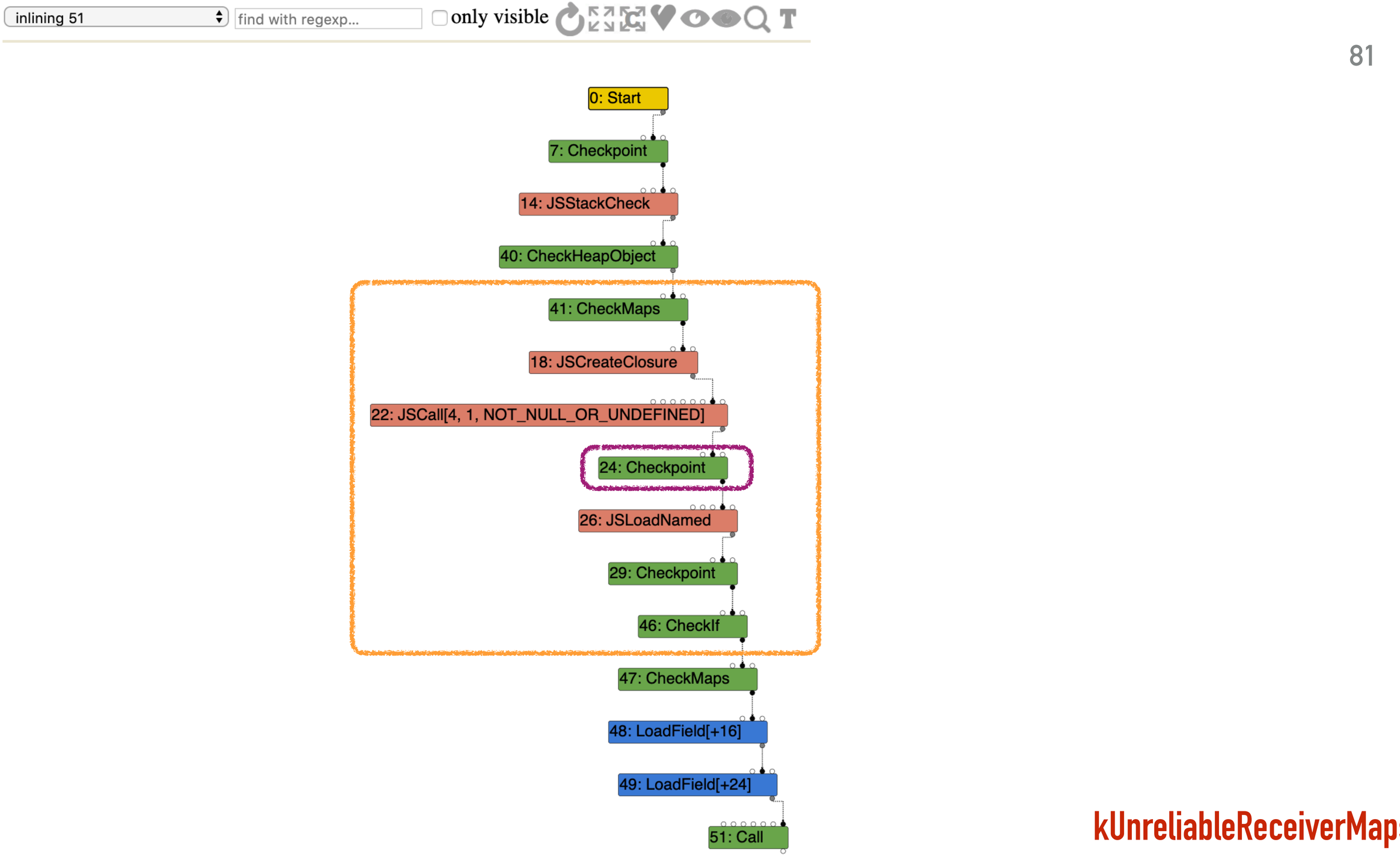
kReliableReceiverMaps

inlining 51 | find with regexp... | only visible



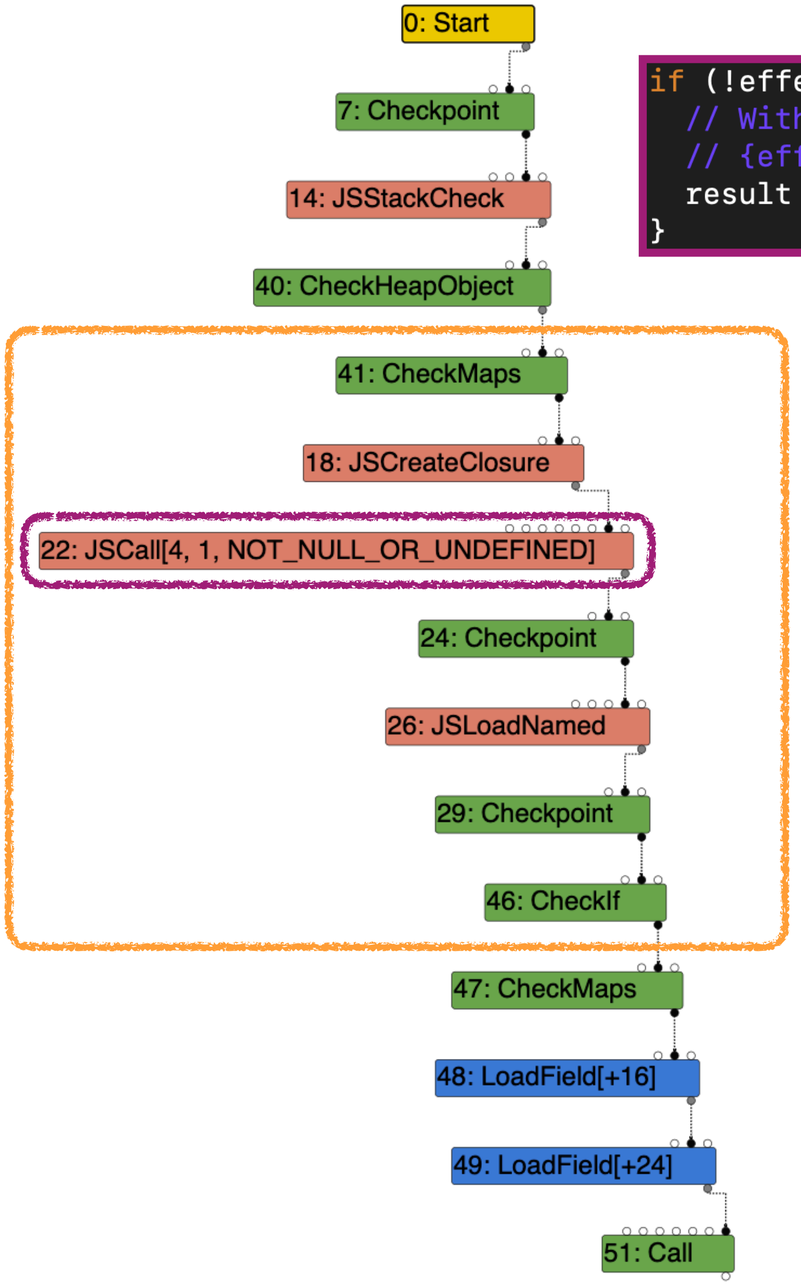

```
if (!effect->op()->HasProperty(Operator::kNoWrite)) {
  // Without alias/escape analysis we cannot tell whether this
  // {effect} affects {receiver} or not.
  result = kUnreliableReceiverMaps;
}
```

kUnreliableReceiverMaps



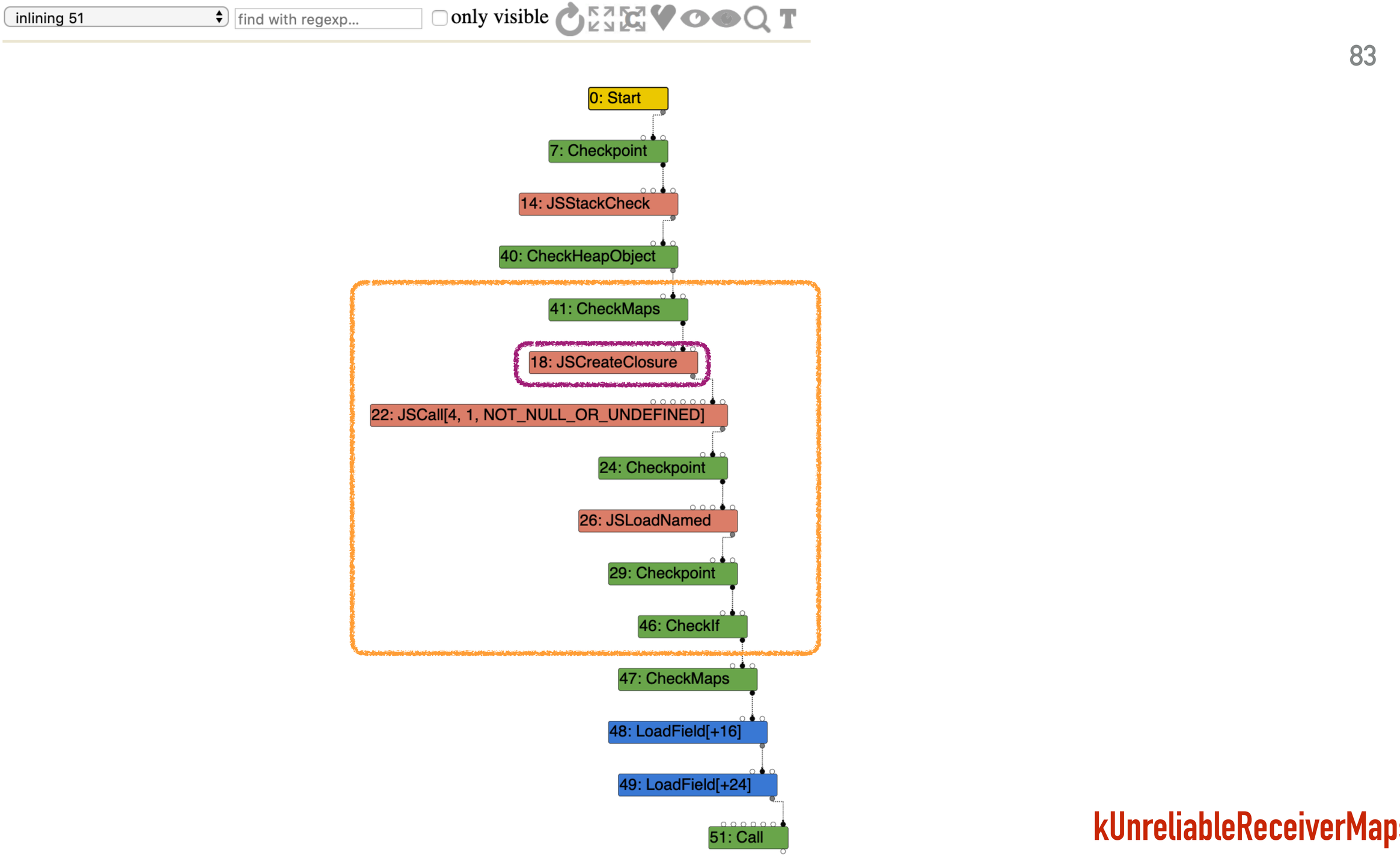
kUnreliableReceiverMaps

inlining 51 | find with regexp... | only visible

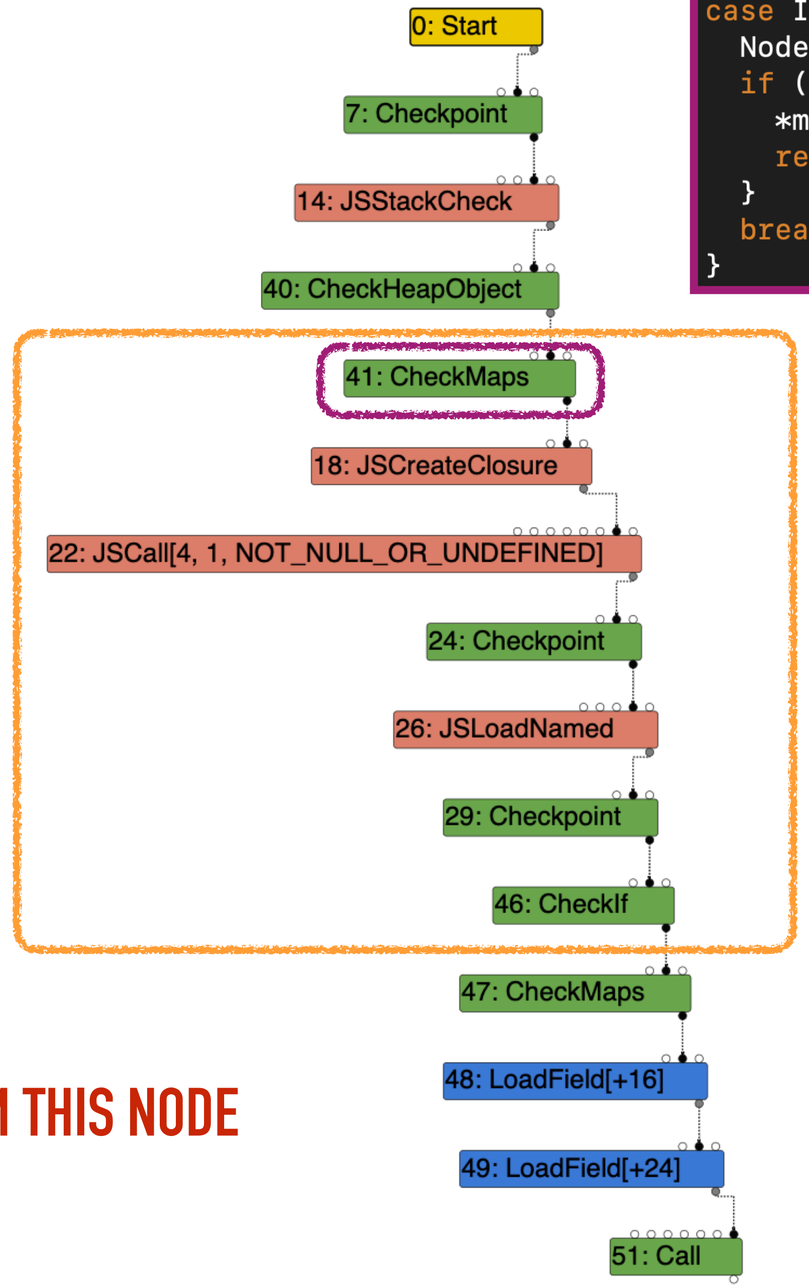



```
if (!effect->op()->HasProperty(Operator::kNoWrite)) {
  // Without alias/escape analysis we cannot tell whether this
  // {effect} affects {receiver} or not.
  result = kUnreliableReceiverMaps;
}
```

kUnreliableReceiverMaps



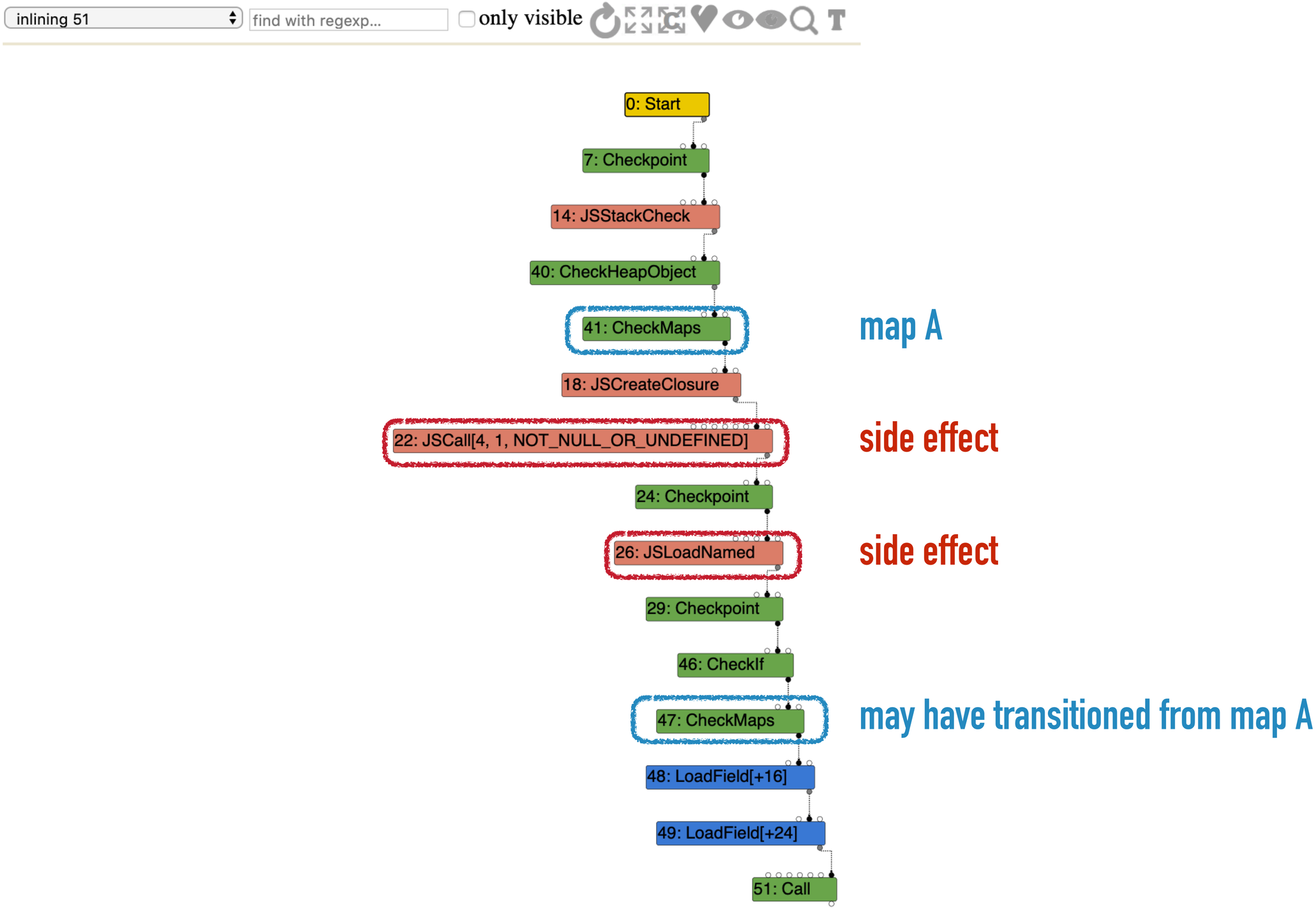
kUnreliableReceiverMaps

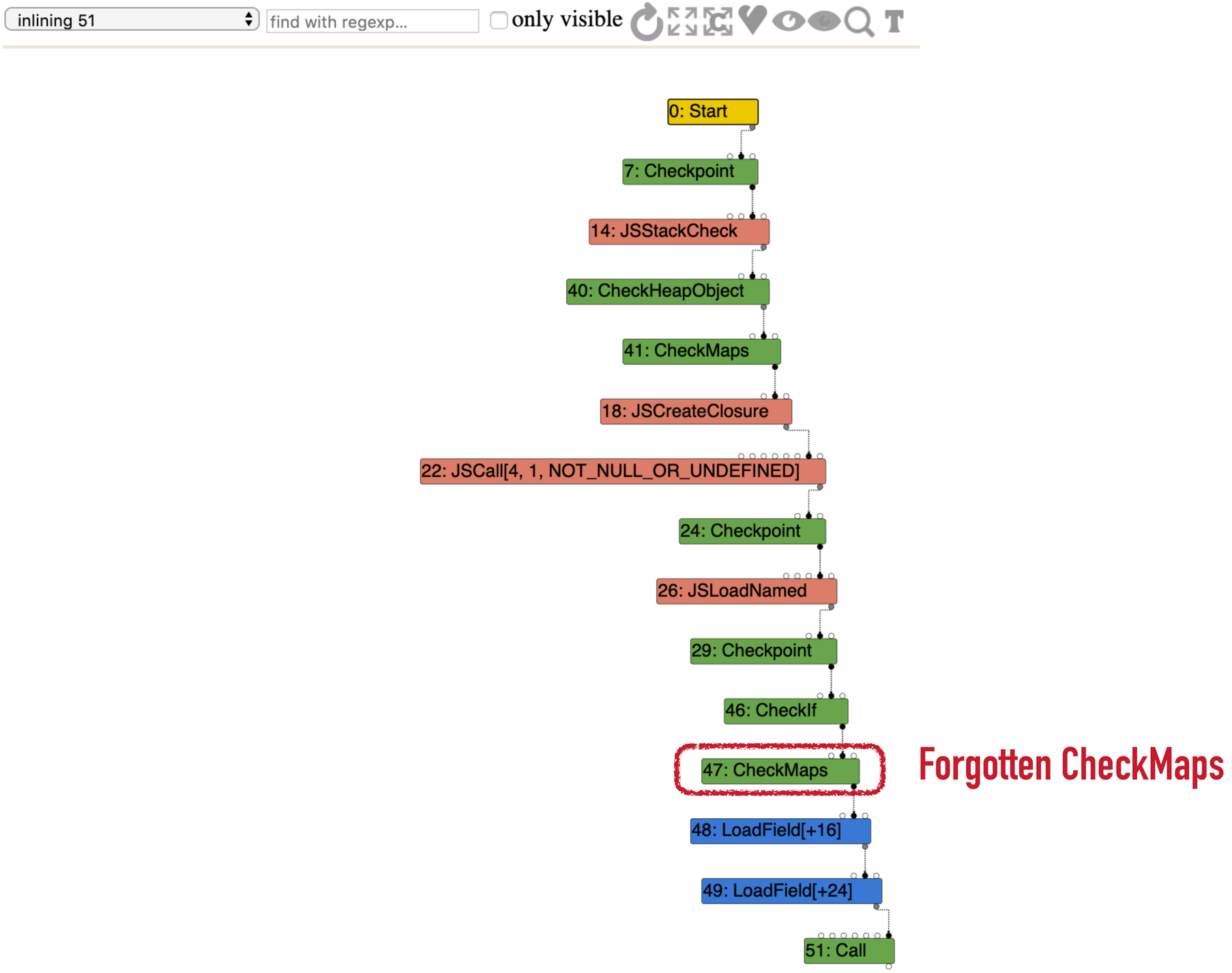



```
case IrOpcode::kCheckMaps: {
  Node* const object = GetValueInput(effect, 0);
  if (IsSame(receiver, object)) {
    *maps_return = CheckMapsParametersOf(effect->op()).maps();
    return result;
  }
  break;
}
```

WE INFER THE RECEIVER MAPS FROM THIS NODE

kUnreliableReceiverMaps


inlining 51
find with regexp...
only visible
0: Start
7: Checkpoint
14: JSStackCheck
40: CheckHeapObject
41: CheckMaps
18: JSCreateClosure
22: JSCall[4, 1, NOT_NULL_OR_UNDEFINED]
24: Checkpoint
26: JSLoadNamed
29: Checkpoint
46: CheckIf
47: CheckMaps
48: LoadField[+16]
49: LoadField[+24]
51: Call
map A
side effect
side effect
may have transitioned from map A


inlining 51
find with regexp...
only visible
0: Start
7: Checkpoint
14: JSStackCheck
40: CheckHeapObject
41: CheckMaps
18: JSCreateClosure
22: JSCall[4, 1, NOT_NULL_OR_UNDEFINED]
24: Checkpoint
26: JSLoadNamed
29: Checkpoint
46: CheckIf
47: CheckMaps
48: LoadField[+16]
49: LoadField[+24]
51: Call
Forgotten CheckMaps

THE REDUCER NEVER CHECKS FOR UNRELIABLE MAPS

1. Infer receiver maps by walking up the effect chain
    1. If the set is empty : do nothing
    2. Forget to check if the set is unreliable
        1. Map informations may not be true anymore because of side effects
2. Compute the union of all the elements kinds of the receiver maps
3. Create nodes assuming this (potentially incorrect) elements kind

JSCALL REDUCER AND INCORRECT ELEMENTS KIND

- Blogpost by István Kurucsai from Exodus Intelligence

```
commit 96de5eeba9b461a2d405dcfa448901c9582f3f07
Author: Mike Stanton <mvstanton@chromium.org>
Date:   Mon Mar 18 12:49:52 2019 +0100

    [TurboFan] Array.prototype.map wrong ElementsKind for output array.
```

https://blog.exodusintel.com/2019/04/03/a-window-of-opportunity/

FIXING BY DEOPTIMIZING WHEN THE ARRAY CONTAINS DICTIONARY ELEMENTS

```
+  // If the array length >= kMaxFastArrayLength, then CreateArray
+  // will create a dictionary. We should deopt in this case, and make sure
+  // not to attempt inlining again.
+  original_length = effect = graph()->NewNode(
+      simplified()->CheckBounds(p.feedback()), original_length,
+      jsgraph()->Constant(JSArray::kMaxFastArrayLength), effect, control);
+
```

INFERRING RECEIVER MAPS AND ELEMENTS KIND

```
ZoneHandleSet<Map> receiver_maps;
NodeProperties::InferReceiverMapsResult result =
    NodeProperties::InferReceiverMaps(broker(), receiver, effect,
                                      &receiver_maps);
if (result == NodeProperties::kNoReceiverMaps) return NoChange();

ElementsKind kind;
if (!CanInlineArrayIteratingBuiltin(broker(), receiver_maps, &kind)) {
  return NoChange();
}
```

OBSERVE THE CORRECT HANDLING OF UNRELIABLE RECEIVER MAPS

```
// If we have unreliable maps, we need a map check.
if (result == NodeProperties::kUnreliableReceiverMaps) {
  effect =
      graph()->NewNode(simplified()->CheckMaps(CheckMapsFlag::kNone,
                                               receiver_maps, p.feedback()),
                       receiver, effect, control);
}
```

BUT THIS CAN CHANGE THE ELEMENTS KIND

```
Node* a = control = effect = graph()->NewNode(
    javascript()->CreateArray(1, MaybeHandle<AllocationSite>()),
    array_constructor, array_constructor, original_length, context,
    outer_frame_state, effect, control);
```

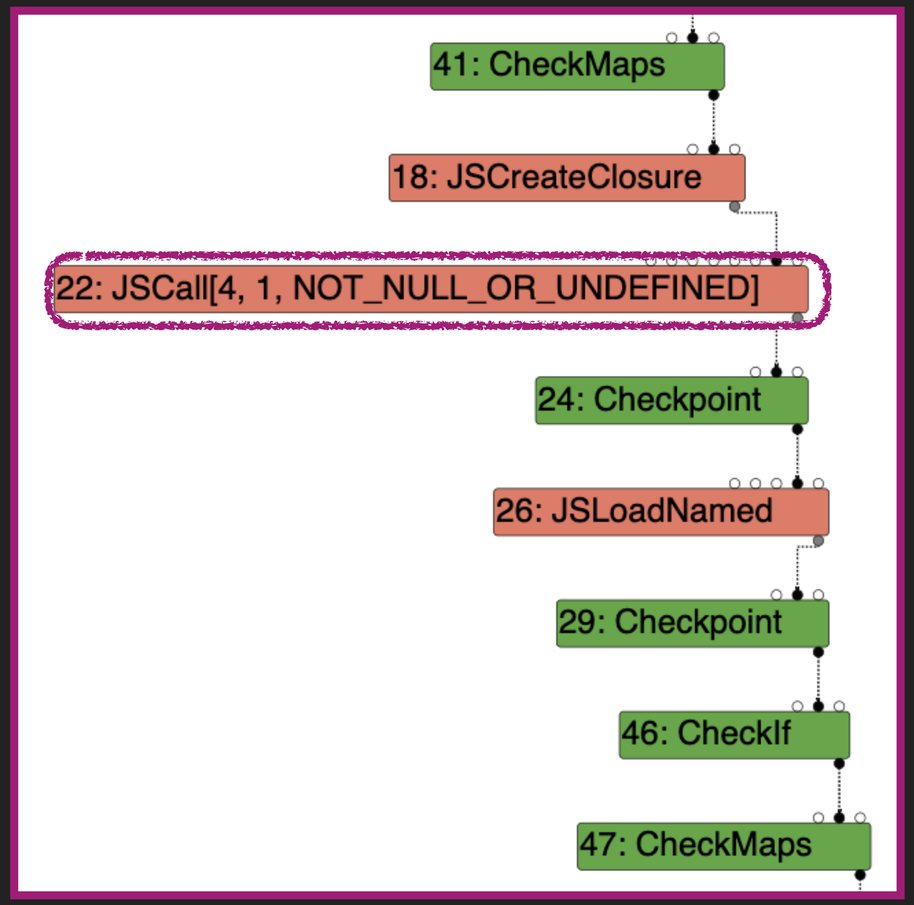
THE ELEMENTS KIND MAY HAVE CHANGED

1. Infer the receiver maps
2. Compute the ElementsKind from it
3. Make a new CreateArray node
    1. Can change the ElementsKind
4. Inline the rest of code, assuming a specific (incorrect) ElementsKind

HOW ARE THOSE BUGS INTERESTING?

- ElementsKind confusion
- PACKED vs DICTIONARY

REMEMBER THIS?




```
if (!effect->op()->HasProperty(Operator::kNoWrite)) {
  // Without alias/escape analysis we cannot tell whether this
  // {effect} affects {receiver} or not.
  result = kUnreliableReceiverMaps;
}
```

How does TurboFan knows that JSCall has side effects?

Because it is ***not*** **kNoWrite**!

CREATEOBJECT

- Bug found by 5aelo
- Awesome phrack article
- CreateObject
  - kNoWrite (means “side effect free”)
  - But actually does have side effects

http://phrack.org/papers/jit_exploitation.html

HOW IS THIS INTERESTING

- Same primitive : ElementsKind confusion

1. Check map A of object O
2. Trigger a side effect with CreateObject so that O.map goes from A to A1
3. Forget to check map A (and thus deoptimize because of type A1)
4. Use object O as if it was of map A while it is actually A1

MISSING KILLMAPS IN LOAD ELIMINATION

- Bug found by lokihardt
- What does “killing maps” means?

Killing maps of an object means ***removing all the knowledge about an object’s maps***

MISSING KILLMAPS IN LOAD ELIMINATIION

```
commit 6b30393536c44943076739e2eaf00f0a34c2d2c9
Author: Jaroslav Sevcik <jarin@chromium.org>
Date:   Fri Jan 5 10:37:45 2018 +0100

    [turbofan] Kill transition-kind source map in load elimination.
```

MISSING KILLMAPS IN LOAD ELIMINATIION

```
--- a/src/compiler/load-elimination.cc
+++ b/src/compiler/load-elimination.cc
@@ -863,6 +863,8 @@ Reduction LoadElimination::ReduceTransitionElementsKind(Node* node) {
     if (object_maps.contains(ZoneHandleSet<Map>(source_map))) {
       object_maps.remove(source_map, zone());
       object_maps.insert(target_map, zone());
+      AliasStateInfo alias_info(state, object, source_map);
+      state = state->KillMaps(alias_info, zone());
       state = state->SetMaps(object, object_maps, zone());
     }
   } else {
@@ -887,6 +889,7 @@ Reduction LoadElimination::ReduceTransitionAndStoreElement(Node* node) {
   if (state->LookupMaps(object, &object_maps)) {
     object_maps.insert(double_map, zone());
     object_maps.insert(fast_map, zone());
+    state = state->KillMaps(object, zone());
     state = state->SetMaps(object, object_maps, zone());
   }
```

HOW IS THIS INTERESTING

- Unexpected transition
- Again, ElementsKind confusion

EXPLOITATION

ELEMENTS KIND CONFUSION

- Dictionary vs packed
- Packed double vs Packed elements
  - Mis-interpret a float as a (tagged) object pointer

THE OBJECTIVE – CONTROL AN ARRAY BUFFER

- Corrupt / fake an ArrayBuffer
  - Underlying memory is fully controlled
  - Length
  - Backing store pointer

HOW?

- OOB R on PACKED_ELEMENTS
  - Leak data
    - Leak object pointers
    - Leak backing store pointer of an ArrayBuffer
  - Get a fake object
    - Read a float as an object

HOW?

- OOB W with PACKED_SMI_ELEMENTS
  - Place an array with PACKED_DOUBLE_ELEMENTS after the object
  - Modify its length
- OOB W with the PACKED_DOUBLE_ELEMENTS
  - Modify an ArrayBuffer
- Or corrupt a PACKED_ELEMENTS array and read a pointer to a crafted object

CODE EXECUTION FROM ARBITRARY WRITE

- JSFunction used to contain a pointer to RWX code
- You can still get a pointer to RWX memory with a WASM module!

RWX MEMORY FROM WASM

```
d8> load("sample_wasm.js")
d8> %DumpObjects(global_test,10)
----- [ JS_FUNCTION_TYPE : 0x38 ] -----
0x00002fac7911ed10    0x00001024ebc84191    MAP_TYPE
0x00002fac7911ed18    0x00000cdfc0080c19    FIXED_ARRAY_TYPE
0x00002fac7911ed20    0x00000cdfc0080c19    FIXED_ARRAY_TYPE
0x00002fac7911ed28    0x00002fac7911ecd9    SHARED_FUNCTION_INFO_TYPE
0x00002fac7911ed30    0x00002fac79101741    NATIVE_CONTEXT_TYPE
0x00002fac7911ed38    0x00000d1caca00691    FEEDBACK_CELL_TYPE
0x00002fac7911ed40    0x00002dc28a002001    CODE_TYPE
```

https://github.com/JeremyFetiveau/debugging-tools/tree/master/v8_doare-helpers

RWX MEMORY FROM WASM

```
d8> %DumpObjects(0x00002fac7911ecd9,11)
----- [ SHARED_FUNCTION_INFO_TYPE : 0x38 ] -----
0x00002fac7911ecd8    0x00000cdfc0080989    MAP_TYPE
0x00002fac7911ece0    0x00002fac7911ecb1    WASM_EXPORTED_FUNCTION_DATA_TYPE
0x00002fac7911ece8    0x00000cdfc00842c1    ONE_BYTE_INTERNALIZED_STRING_TYPE
0x00002fac7911ecf0    0x00000cdfc0082ad1    FEEDBACK_METADATA_TYPE
0x00002fac7911ecf8    0x00000cdfc00804c9    ODDBALL_TYPE
0x00002fac7911ed00    0x000000000000004f
0x00002fac7911ed08    0x000000000000ff00
```

https://github.com/JeremyFetiveau/debugging-tools/tree/master/v8_doare-helpers

RWX MEMORY FROM WASM

```
d8> %DumpObjects(0x00002fac7911ecb1,11)
----- [ WASM_EXPORTED_FUNCTION_DATA_TYPE : 0x28 ] -----
0x00002fac7911ecb0    0x00000cdfc00857a9    MAP_TYPE
0x00002fac7911ecb8    0x00002dc28a002001    CODE_TYPE
0x00002fac7911ecc0    0x00002fac7911eb29    WASM_INSTANCE_TYPE
0x00002fac7911ecc8    0x0000000000000000
0x00002fac7911ecd0    0x0000000100000000
```

https://github.com/JeremyFetiveau/debugging-tools/tree/master/v8_doare-helpers

RWX MEMORY FROM WASM

```
d8> %DumpObjects(0x00002fac7911eb29,41)
----- [ WASM_INSTANCE_TYPE : 0x118 : REFERENCES RWX MEMORY] ——

0x00002fac7911ec20    0x0000087e7c50a000    JumpTableStart [RWX]
```

https://github.com/JeremyFetiveau/debugging-tools/tree/master/v8_doare-helpers

- TurboFan is a very interesting target
- Bug classes are quite specific to JIT engines

- Shout-out to Axel ‘0vercl0k’ Souchet, Bruno Keith and Georgi Geshev
- Special thanks to Noam & Ido from SSD for the opportunity to speak
- Kudos to the V8 team for their amazing work

THANK YOU

@__x86

doar-e.github.io