JavaScript Engines - how do they even?

JSConfEU
May 2017, Berlin

https://www.youtube.com/watch?v=p-iiEDtpy6I

JavaScript Engines - how do they even?

Dr. Franziska Hinkelmann
Google

@fhinkel

```
})(ts.Comparison || (ts.Comparison = {}));
var Comparison = ts.Comparison;
/**
 * Iterates through 'array' by index and performs the callback on each element of array until the callback
 * returns a truthy value, then returns that value.
 * If no such value is found, the callback is applied to each element of array and undefined is returned.
 */
function forEach(array, callback) {
    if (array) {
        for (var i = 0, len = array.length; i < len; i++) {
            var result = callback(array[i], i);
            if (result) {
                return result;
            }
        }
    }
    return undefined;
}
ts.forEach = forEach;
/**
 * Iterates through `array` by index and performs the callback on each element of array until the callback
 * returns a falsey value, then returns false.
 * If no such value is found, the callback is applied to each element of array and `true` is returned.
 */
function every(array, callback) {
    if (array) {
        for (var i = 0, len = array.length; i < len; i++) {
            if (!callback(array[i], i)) {
                return false;
            }
        }
    }
    return true;
}
ts.every = every;
/** Works like Array.prototype.find, returning `undefined` if no element satisfying the p
function find(array, predicate) {
    for (var i = 0, len = array.length; i < len; i++) {
        var value = array[i];
        if (predicate(value, i)) {
            return value;
        }
    }
    return undefined;
}
ts.find = find;
/**
 * Returns the first truthy result of `callback`, or else fails.
 * This is like `forEach`, but never returns undefined.
 */
function findMap(array, callback) {
    for (var i = 0, len = array.length; i < len; i++) {
        var result = callback(array[i], i);
        if (result) {
            return result;
        }
    }
    Debug.fail();
}
ts.findMap = findMap;
function contains(array, value) {
    if (array) {
        for (var _i = 0, array_1 = array; _i < array_1.length; _i++) {
            var v = array_1[_i];
            if (v === value) {
                return true;
            }
        }
```

JS Engine

Elements Console Sources Network Timeline

top Preserve log

```
Move disk 1 from initial stack to temporary stack.
Move disk 2 from initial stack to final stack.
Move disk 1 from temporary stack to final stack.
Move disk 3 from initial stack to temporary stack.
Move disk 1 from final stack to initial stack.
Move disk 2 from final stack to temporary stack.
Move disk 1 from initial stack to temporary stack.
Move disk 2 from initial stack to final stack.
Move disk 1 from temporary stack to final stack.
Move disk 2 from temporary stack to initial stack.
Move disk 1 from final stack to initial stack.
Move disk 3 from temporary stack to final stack.
Move disk 1 from initial stack to temporary stack.
Move disk 2 from initial stack to final stack.
Move disk 1 from temporary stack to final stack.
```

- Browser: Chakra, JavaScriptCore, Spidermonkey, V8
- Node.js: Chakra, V8, SpiderNode
- Electron: V8
- IoT: Duktape, JerryScript

EcmaScript Standard defined by TC39

ECMAScript® 2018 Language

Secure https://tc39.github.io/ecma262/#sec-async-functions-abstract-operations-async-function-await

6. Assert: When we return here, *asyncContext* has already been removed from the execution context stack and *runningContext* is the currently running execution context.
7. Assert: *result* is a normal completion with a value of **undefined**. The possible sources of completion values AsyncFunctionAwait or, if the async function doesn't await anything, the step 3.g above.
8. Return.

25.5.5.3 AsyncFunctionAwait (*value*)

1. Let *asyncContext* be the running execution context.
2. Let *promiseCapability* be ! NewPromiseCapability(%Promise%).
3. Let *resolveResult* be ! Call(*promiseCapability*.[[Resolve]], **undefined**, « *value* »).
4. Let *onFulfilled* be a new built-in function object as defined in AsyncFunction Awaited Fulfilled.
5. Let *onRejected* be a new built-in function object as defined in AsyncFunction Awaited Rejected.
6. Set *onFulfilled*.[[AsyncContext]] to *asyncContext*.
7. Set *onRejected*.[[AsyncContext]] to *asyncContext*.
8. Let *throwawayCapability* be ! NewPromiseCapability(%Promise%).
9. Set *throwawayCapability*.[[Promise]].[[PromiseIsHandled]] to **true**.
10. Perform ! PerformPromiseThen(*promiseCapability*.[[Promise]], *onFulfilled*, *onRejected*, *throwawayCapability*).
11. Remove *asyncContext* from the execution context stack and restore the execution context that is at the top of execution context stack as the running execution context.
12. Set the code evaluation state of *asyncContext* such that when evaluation is resumed with a Completion *resumptionValue* the following steps will be performed:
    a. Return *resumptionValue*.
13. Return.

NOTE The return value of this abstract operation is unused. The interesting return is that of *resumptionV*

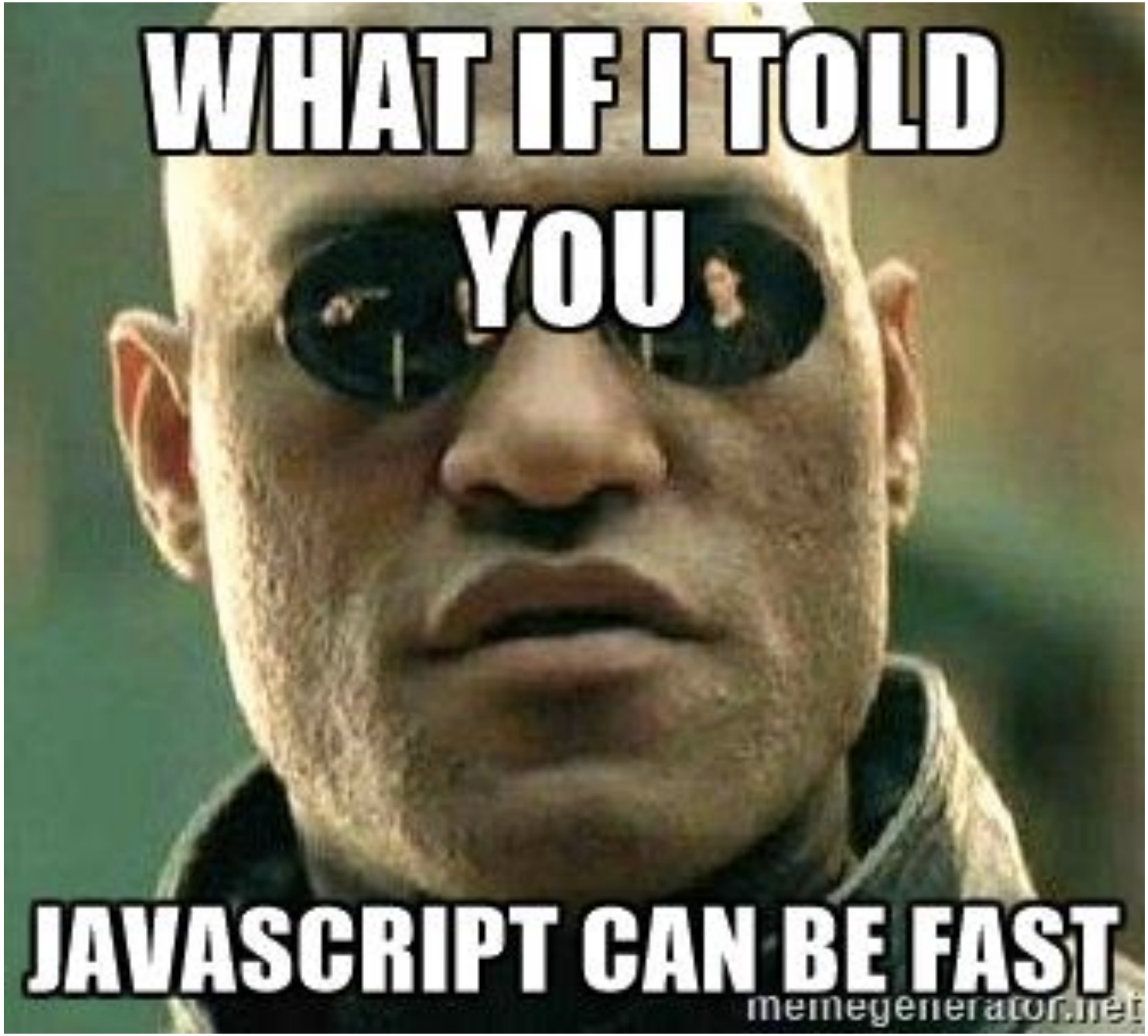
JavaScript	C++
`var x = 17;`	`int x = 17;`

JS is dynamically typed.

C++ is statically typed.

Type of an object

```
var obj = {
    x: 1,
    y: 1
};

delete obj.x;

obj.z = 1;

obj.hasOwnProperty
```


WHAT IF I TOLD YOU
JAVASCRIPT CAN BE FAST
memegenerator.net

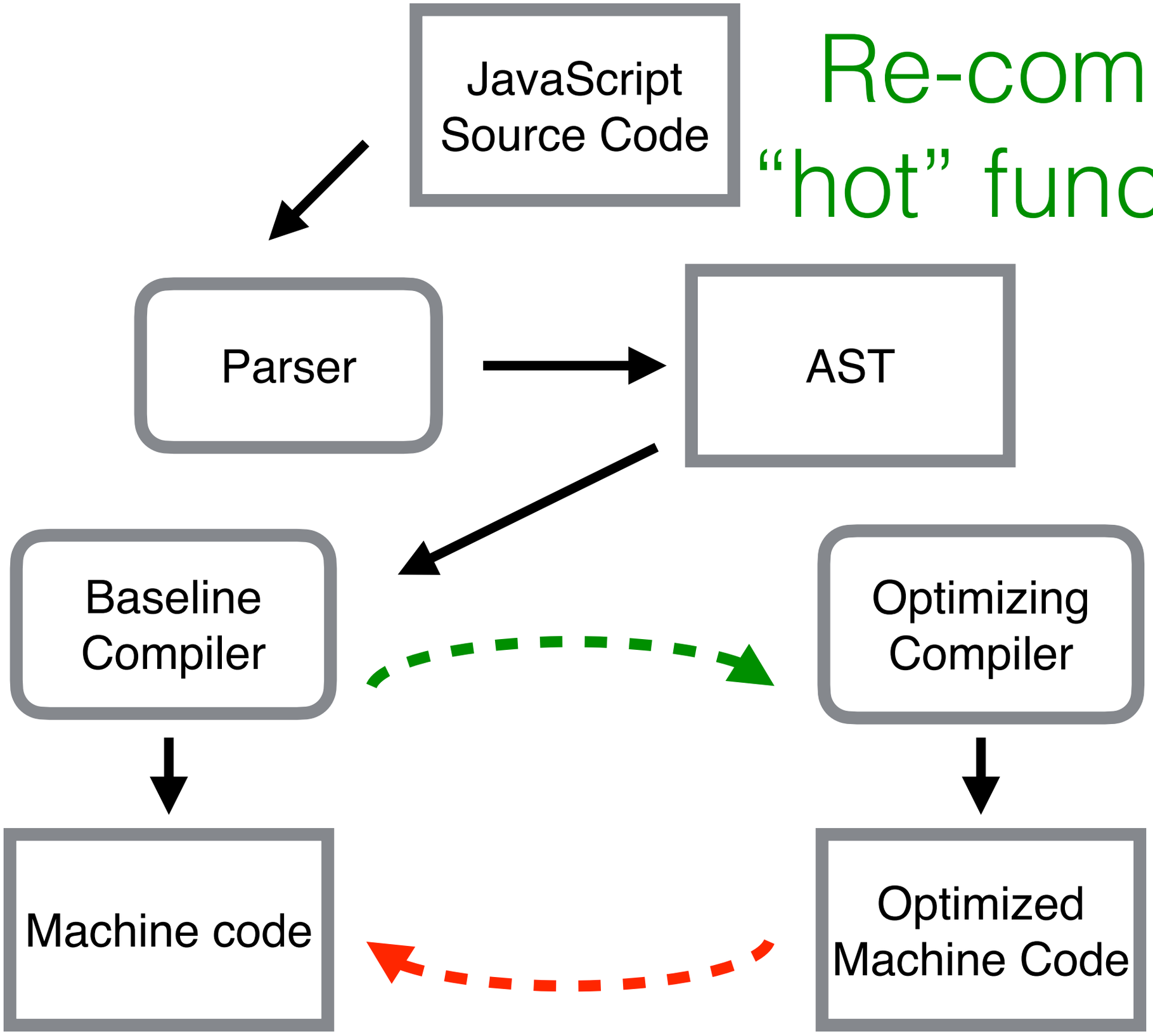
Just In Time (JIT) Compilation

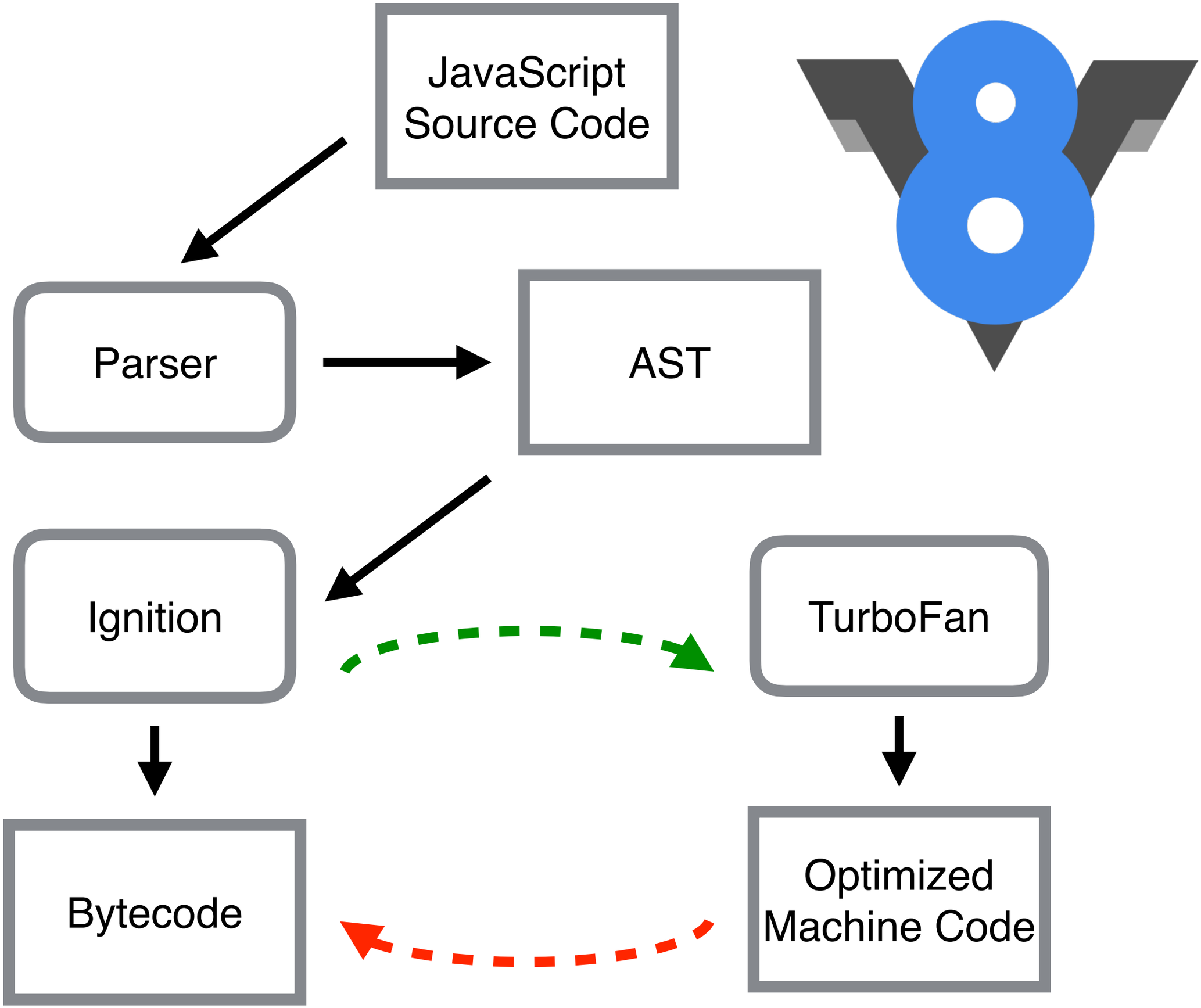
Generate machine code during runtime, not **ahead of time** (AOT).

Optimizing compiler

Re-compile “hot” functions with type information from previous execution 🔥

De-optimize if the type has changed


JavaScript Source Code
Re-compile "hot" functions
Parser
AST
Baseline Compiler
Optimizing Compiler
Machine code
Optimized Machine Code

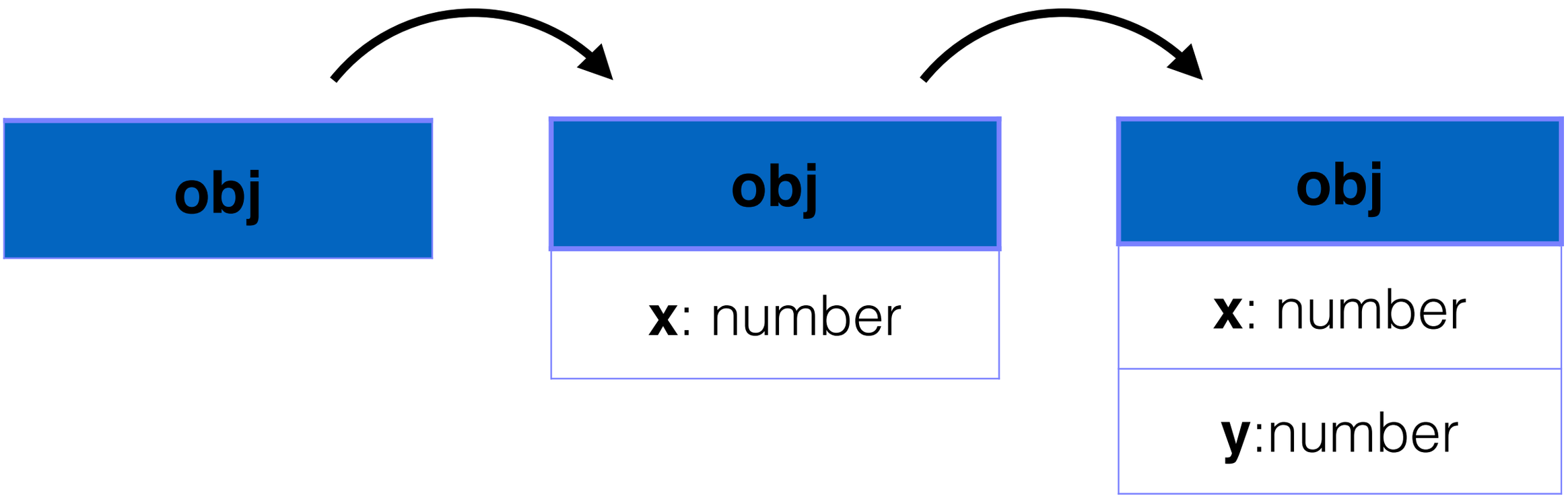

JavaScript
Source Code
Parser
AST
Ignition
TurboFan
Bytecode
Optimized
Machine Code

Optimizing compiler uses previously seen type information - don’t change types!

```
function load(obj) {
  return obj.x
}
```

Object types internally

```
var obj = {
  x: 1,
  y: 4
};
```

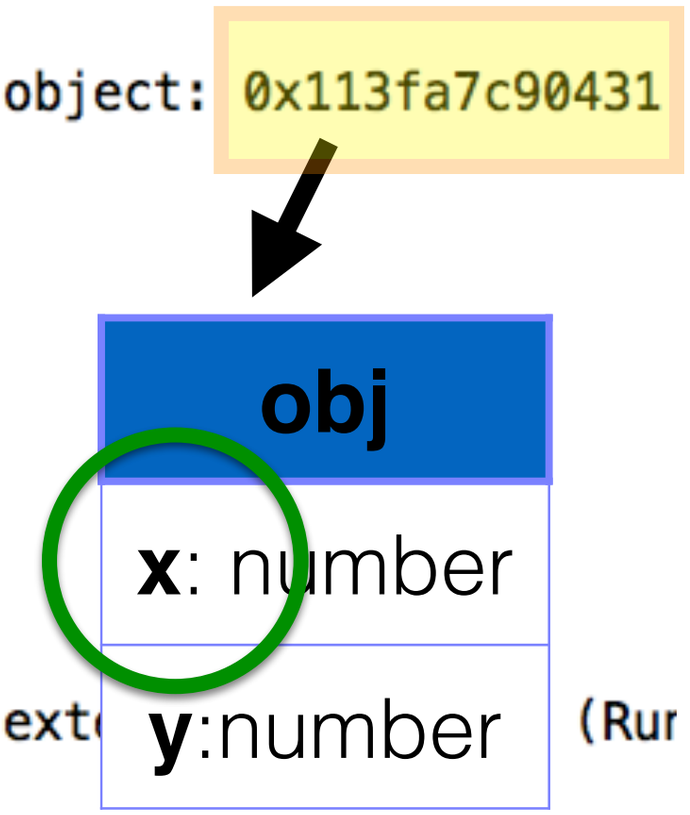


```
function load(obj) {
  return obj.x;
}

load({x:4, y:7});
load({x:2, y:9});
load({x:1, y:3});
load({x:6, y:1});
load({x:3, y:8});
```

obj
x: number
y:number

Set up stack and enter function

```
-- B3 start (deconstruct frame) --
13  488b4510          REX.W movq rax,[rbp+0x10]
17  a801              test al,0x1
19  0f8436000000      jz 0x2233d07042b5  <+0x55>
1f  48bb3104c9a73f110000 REX.W movq rbx,0x113fa7c90431    ;; object: 0x113fa7c90431
29  483958ff          REX.W cmpq [rax-0x1],rbx
2d  0f8527000000      jnz 0x2233d07042ba  <+0x5a>
33  488b4017          REX.W movq rax,[rax+0x17]
37  488be5            REX.W movq rsp,rbp
3a  5d                pop rbp
3b  c21000            ret 0x10
-- B4 start (no frame) --
-- B1 start (deferred) --
-- <mono.js:1:14> --
3e  48bb500a2aa4237f0000 REX.W movq rbx,0x7f23a42a0a50    ;; ext
48  33c0              xorl rax,rax
4a  488b75f8          REX.W movq rsi,[rbp-0x8]
4e  e84d02d8ff        call 0x2233d0484500     ;; code: STUB, CEntryStub, minor: 8
53  ebbe              jmp 0x2233d0704273  <+0x13>
55  e846fdbfff        call 0x2233d0304000     ;; deoptimization bailout 0
5a  e84bfdbfff        call 0x2233d030400a     ;; deoptimization bailout 1
5f  90                nop
```



```
function load(obj) {
  return obj.x;
}

load({x:4, a:7});
load({x:2, b:9});
load({x:1, c:3});
load({x:6, d:1});
```

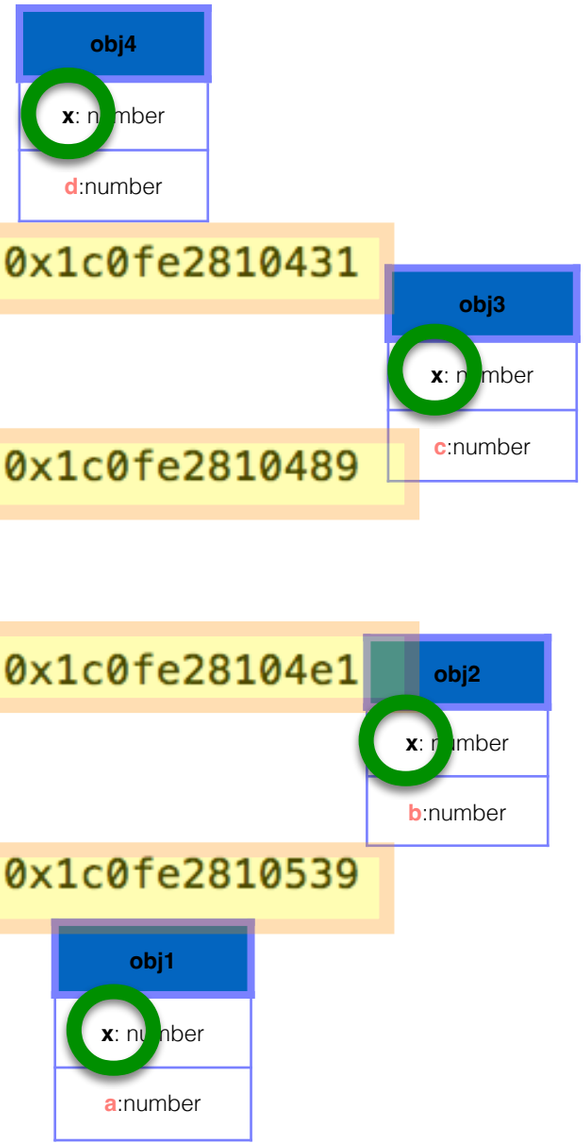
obj1
x: number
a:number

obj2
x: number
b:number

obj3
x: number
c:number

obj4
x: number
d:number

```
-- B3 start --
13  488b4510          REX.W movq rax,[rbp+0x10]
17  a801              test al,0x1
19  0f8472000000      jz 0x3b5172e842f1  <+0x91>
1f  488b58ff          REX.W movq rbx,[rax-0x1]
23  48ba310481e20f1c0000 REX.W movq rdx,0x1c0fe2810431    ;; object: 0x1c0fe2810431
2d  483bd3            REX.W cmpq rdx,rbx
30  0f8439000000      jz 0x3b5172e842cf  <+0x6f>
-- B4 start --
36  48ba890481e20f1c0000 REX.W movq rdx,0x1c0fe2810489    ;; object: 0x1c0fe2810489
40  483bd3            REX.W cmpq rdx,rbx
43  0f8426000000      jz 0x3b5172e842cf  <+0x6f>
-- B5 start --
49  48bae10481e20f1c0000 REX.W movq rdx,0x1c0fe28104e1    ;; object: 0x1c0fe28104e1
53  483bd3            REX.W cmpq rdx,rbx
56  0f8413000000      jz 0x3b5172e842cf  <+0x6f>
-- B6 start --
5c  48ba390581e20f1c0000 REX.W movq rdx,0x1c0fe2810539    ;; object: 0x1c0fe2810539
66  483bd3            REX.W cmpq rdx,rbx
69  0f8527000000      jnz 0x3b5172e842f6  <+0x96>
-- B7 start --
-- B8 start --
-- B9 start --
-- B10 start (deconstruct frame) --
6f  488b4017          REX.W movq rax,[rax+0x17]
73  488be5            REX.W movq rsp,rbp
76  5d                pop rbp
```




```
81  eb82              jmp 0x3b5172e84273  <+0x13>
91  e80afdbfff        call 0x3b5172a84000     ;; deoptimization bailout 0
96  e80ffdbfff        call 0x3b5172a8400a     ;; deoptimization bailout 1
9b  90                nop
```

More than 4 types

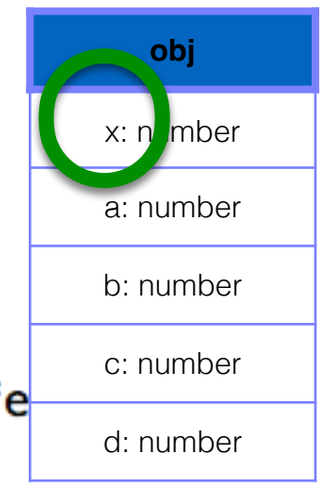
```
80d0000 REX.W movq rcx,0xda88fb8cea9     ;; object: 0xda88fb8cea9 <String[1]: x>
3000000 REX.W movq rax,0x300000000
  REX.W movq rdx,[rbp+0x10]
  REX.W movq rsi,[rbp-0x8]
  call 0x3beb682a4ee0  (LoadICTrampoline)    ;; code: LOAD_IC
  REX.W movq rsp,rbp
  pop rbp
  ret 0x10
```

```
function load(obj) {
  return obj.x
}

load({x: 4, a: 7, b: undefined, c: undefined, d: undefined})
load({x: 4, a: undefined , b: 9, c: undefined, d: undefined})
load({x: 4, a: undefined , b: undefined, c: 3, d: undefined})
load({x: 4, a: undefined , b: undefined, c: undefined, d: 1})
```

Always construct the same type of objects!

obj
x: number
a: number
b: number
c: number
d: number

```
-- B3 start (deconstruct frame) --
13  488b4510        REX.W movq rax,[rbp+0x10]
17  a801            test al,0x1
19  0f8436000000    jz 0x2c1aa92042b5  <+0x55>
1f  48bba908e94b70020000 REX.W movq rbx,0x2704be908a9    ;; object: 0x2704be908a9
29  483958ff        REX.W cmpq [rax-0x1],rbx
2d  0f8527000000    jnz 0x2c1aa92042ba  <+0x5a>
33  488b4017        REX.W movq rax,[rax+0x17]
37  488be5          REX.W movq rsp,rbp
3a  5d              pop rbp
3b  c21000          ret 0x10
-- B4 start (no frame) --
-- B1 start (deferred) --
-- <mono-long.js:1:14> --
3e  48bb503a220c2a7f0000 REX.W movq rbx,0x7f2a0c223a50    ;; external refe
48  33c0            xorl rax,rax
4a  488b75f8        REX.W movq rsi,[rbp-0x8]
4e  e84d02d8ff      call 0x2c1aa8f84500     ;; code: STUB, CEntryStub, minor: 8
53  ebbe            jmp 0x2c1aa9204273  <+0x13>
55  e846fdbfff      call 0x2c1aa8e04000     ;; deoptimization bailout 0
5a  e84bfdbfff      call 0x2c1aa8e0400a     ;; deoptimization bailout 1
```



Computed names in object literal definitions

ES5

```
function foo() {
  let o = {};
  o[x] = 1;
  return o;
}
```

ES6

```
function foo() {
  return {[x]: 1}
}
```

Reminder

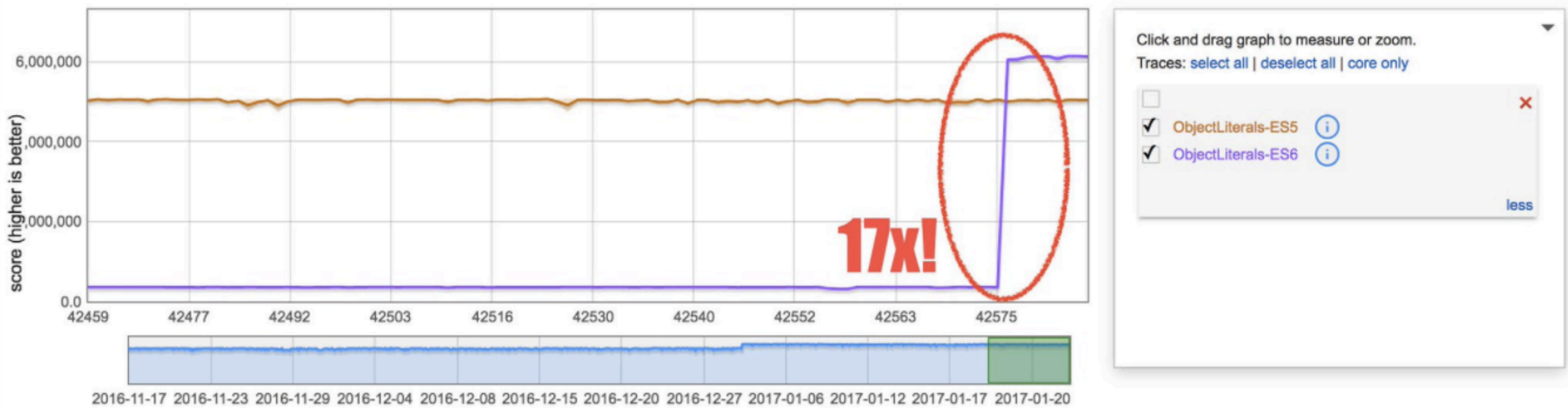
```
var x = 'foo'

obj.x = 1;
obj[x] = 1;    // o.foo
```

Remember the key and use fast path to create the object.

```
function foo() {
  return {[x]: 1}
}
```

Benedikt Meurer
@bmeurer
Following

internal.client.v8/x64/v8 / SixSpeed / Computed property names in object literals /



Benedikt Meurer @bmeurer · Jan 22
Another great ES2015 improvement: @fhinkel just boosted performance of computed property names by 17x! Now faster than naive ES5! 🎉🔥 pic.twitter.com/DesB9kzA4q

2 135 286

Munich
benediktmeurer.de
Joined May 2011

2 135 286

Try with Node.js or Chrome

- `–print-opt-code`: code generated by optimizing compiler
- `–print-bytecode`: bytecode generated by interpreter
- `–trace-ic`: different object types a call sites encounters
- `–trace-opt` and `–trace-deopt`: which functions are (de)optimized

Write code that looks like statically typed.

@fhinkel

franzih@google.com

More resources

- v8project.blogspot.com
- benediktmeurer.de
- http://ripsawridge.github.io/
- https://medium.com/@tverwaes