Google

What’s inside of TurboFan?

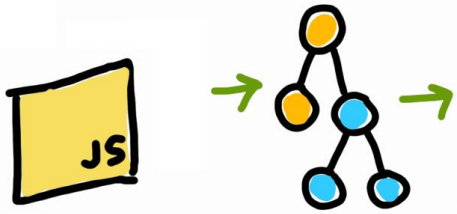
Benedikt Meurer
@bmeurer

Overview

1. How does V8 work?
2. What is TurboFan?
3. The optimizing compiler in detail
4. Performance result

How

Works



We take your JS

Parse it

turn that into an Abstract Syntax Tree

IGNITION

05 3c
fe ba

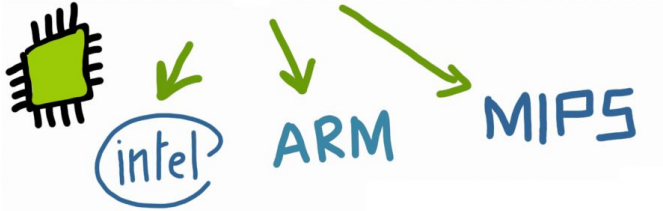
Generate bytecode

Get feedback

for speculative optimisations

TURBOFAN

Optimize & Compile it

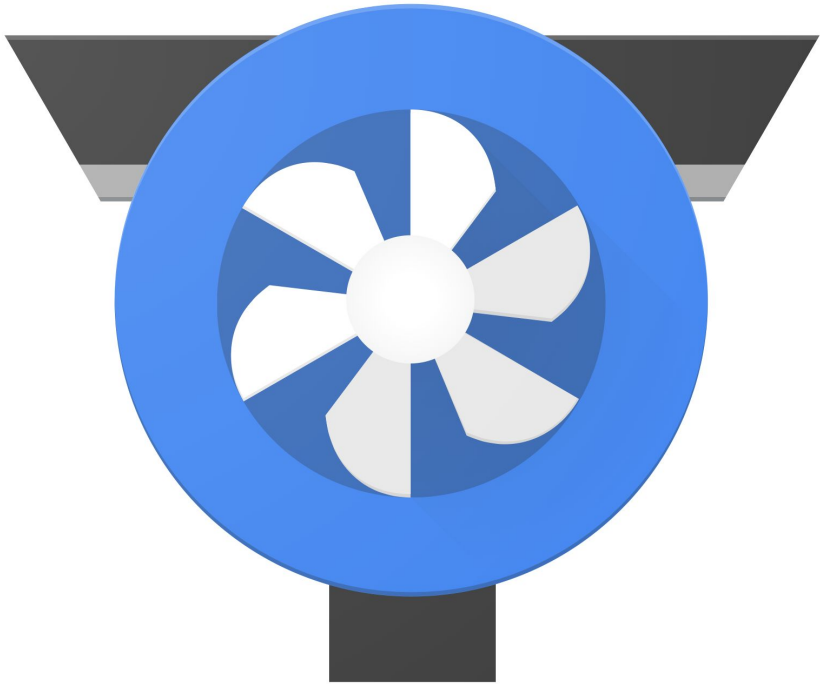


then run your optimized code!

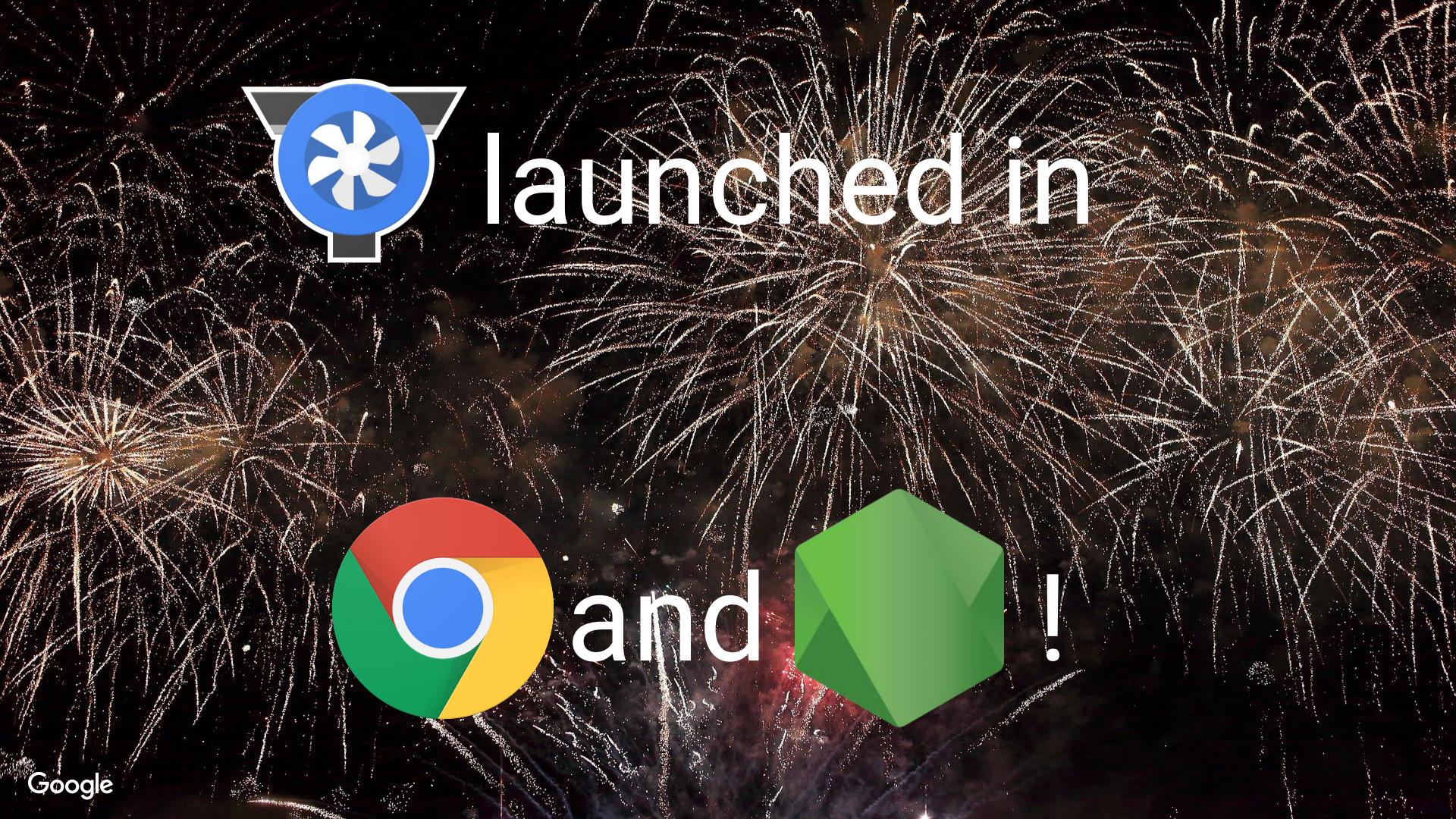
Google

By @addyosmani

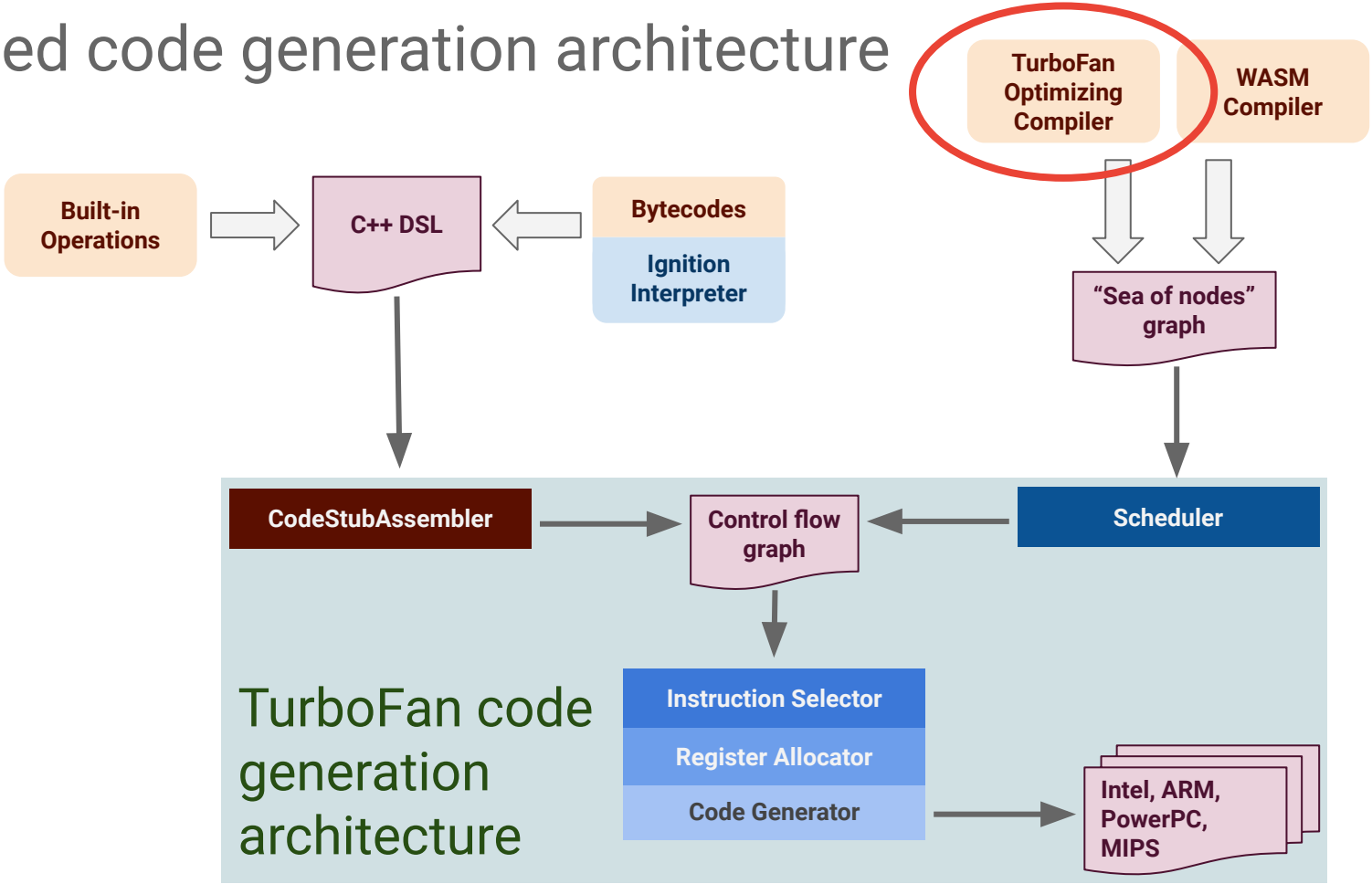
What is TurboFan?



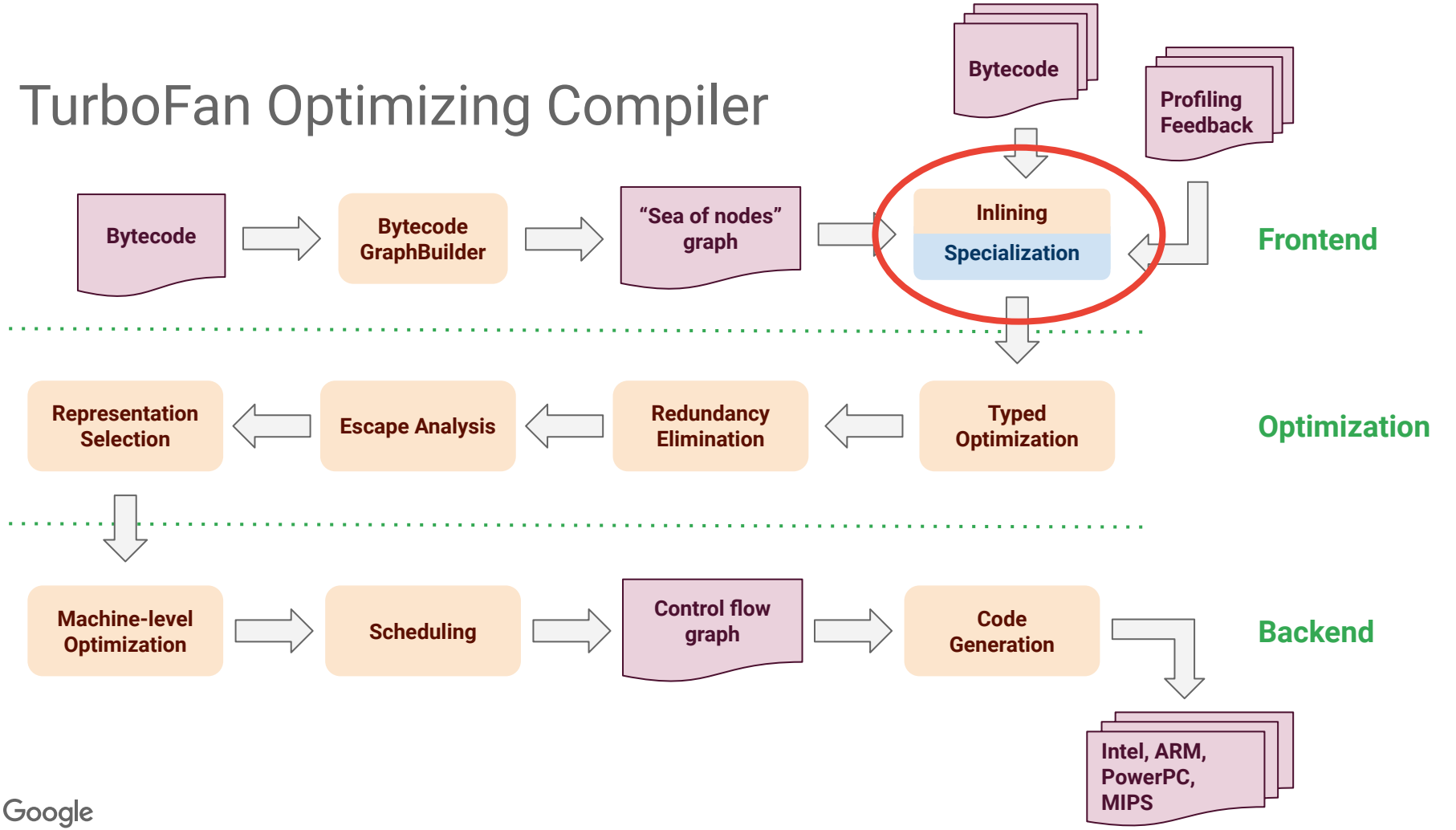
- Modern code generation architecture for growing number of platforms (8+)
- Replacement for the aging Crankshaft compiler
- Optimizing compiler with full ES2015+ language support
- Predictable and consistent performance profile


launched in
and
!
Google

Unified code generation architecture



TurboFan Optimizing Compiler



Inlining

Inlining

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

function three() {
  return add(1, 2);
}
```

```
function three_add_inlined() {
  var x = 1;
  var y = 2;
  var add_return_value = x + y;
  return add_return_value;
}
```

Constant-folding

```
function three_add_const_folded() {
  return 3;
}
```

Inlining

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

function three() {
  return add(1, 2);
}
```

```
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+0xdb0]
jna StackCheck
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
```

Inlining

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

function three() {
  return add(1, 2);
}
```

Prologue

```
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+0xdb0]
jna StackCheck
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
```

Inlining

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

function three() {
  return add(1, 2);
}
```

Prologue

```
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+0xdb0]
jna StackCheck
```

Check x is a small integer

```
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
```

Inlining

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

function three() {
  return add(1, 2);
}
```

Prologue

```
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+0xdb0]
jna StackCheck
```

Check x is a small integer

```
movq rax,[rbp+0x18]
test al,0x1
jnz Deoptimize
```

Check y is a small integer

```
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
```

Inlining

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

function three() {
  return add(1, 2);
}
```

Prologue

```
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+0xdb0]
jna StackCheck
```

Check x is a small integer

```
movq rax,[rbp+0x18]
test al,0x1
jnz Deoptimize
```

Check y is a small integer

```
movq rbx,[rbp+0x10]
testb rbx,0x1
jnz Deoptimize
```

Convert y from tagged representation to word32

```
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
```

Inlining

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

function three() {
  return add(1, 2);
}
```

Prologue

```
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+0xdb0]
jna StackCheck
```

Check x is a small integer

```
movq rax,[rbp+0x18]
test al,0x1
jnz Deoptimize
```

Check y is a small integer

```
movq rbx,[rbp+0x10]
testb rbx,0x1
jnz Deoptimize
```

Convert y from tagged representation to word32

```
movq rdx,rbx
shrq rdx, 32
```

Convert x from tagged representation to word32

```
movq rcx,rax
shrq rcx, 32
addl rdx,rcx
jo Deoptimize
shlq rdx, 32
movq rax,rdx
movq rsp,rbp
pop rbp
ret 0x18
```

Inlining

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

function three() {
  return add(1, 2);
}
```

Prologue

```
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+0xdb0]
jna StackCheck
```

Check x is a small integer

```
movq rax,[rbp+0x18]
test al,0x1
jnz Deoptimize
```

Check y is a small integer

```
movq rbx,[rbp+0x10]
testb rbx,0x1
jnz Deoptimize
```

Convert y from tagged representation to word32

```
movq rdx,rbx
shrq rdx, 32
```

Convert x from tagged representation to word32

```
movq rcx,rax
shrq rcx, 32
```

Add x and y (incl. overflow check)

```
addl rdx,rcx
jo Deoptimize
```

```
shlq rdx, 32
movq rax,rdx
movq rsp,rbp
pop rbp
ret 0x18
```

Inlining

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

function three() {
  return add(1, 2);
}
```

Prologue	`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+0xdb0]`
	`jna StackCheck`

Check x is a small integer	`movq rax,[rbp+0x18]`
	`test al,0x1`
	`jnz Deoptimize`

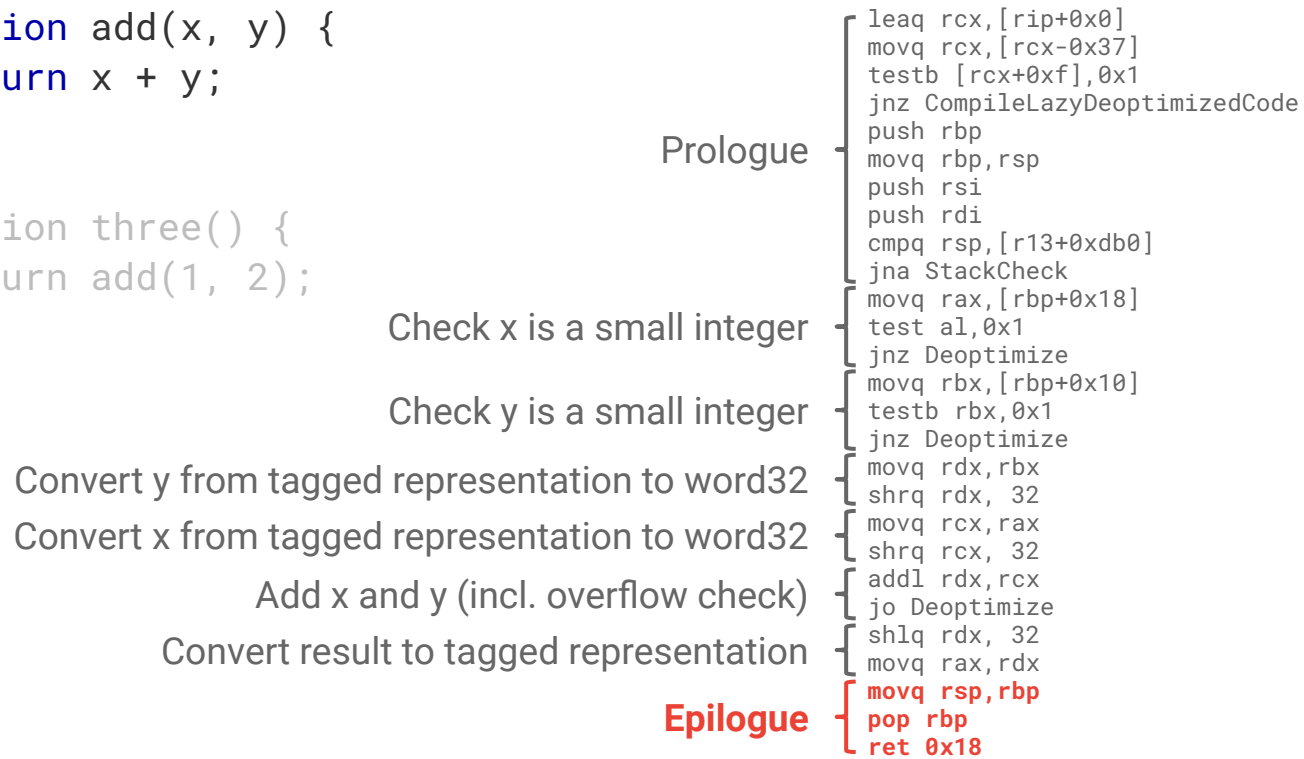
Check y is a small integer	`movq rbx,[rbp+0x10]`
	`testb rbx,0x1`
	`jnz Deoptimize`

Convert y from tagged representation to word32	`movq rdx,rbx` `shrq rdx, 32`
Convert x from tagged representation to word32	`movq rcx,rax` `shrq rcx, 32`
Add x and y (incl. overflow check)	`addl rdx,rcx` `jo Deoptimize`
Convert result to tagged representation	**`shlq rdx, 32`** **`movq rax,rdx`**

	`movq rsp,rbp`
	`pop rbp`
	`ret 0x18`

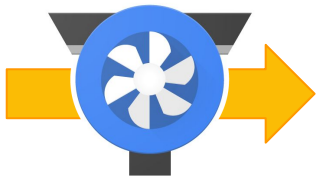
Inlining

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

function three() {
  return add(1, 2);
}
```



Inlining

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

function three() {
  return add(1, 2);
}
```



```
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+0xdb0]
jna StackCheck
movq rdi,<JSFunction add>
movq rsi,[rdi+0x1f]
movq rax,<JSGlobal Object>
push rax
movq rax,0x100000000
push rax
movq rax,0x200000000
push rax
movq rdx,[r13-0x60]
movl rax,0x2
movq rcx,[rdi+0x2f]
addq rcx,0x5f
call rcx
movq rsp,rbp
pop rbp
ret 0x8
```

Inlining

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

function three() {
  return add(1, 2);
}
```

Prologue

```
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+0xdb0]
jna StackCheck
movq rdi,<JSFunction add>
movq rsi,[rdi+0x1f]
movq rax,<JSGlobal Object>
push rax
movq rax,0x100000000
push rax
movq rax,0x200000000
push rax
movq rdx,[r13-0x60]
movl rax,0x2
movq rcx,[rdi+0x2f]
addq rcx,0x5f
call rcx
movq rsp,rbp
pop rbp
ret 0x8
```

Inlining

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

function three() {
  return add(1, 2);
}
```

Prologue

```
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+0xdb0]
jna StackCheck
```

Load call target add

```
movq rdi,<JSFunction add>
movq rsi,[rdi+0x1f]
movq rax,<JSGlobal Object>
push rax
movq rax,0x100000000
push rax
movq rax,0x200000000
push rax
movq rdx,[r13-0x60]
movl rax,0x2
movq rcx,[rdi+0x2f]
addq rcx,0x5f
call rcx
movq rsp,rbp
pop rbp
ret 0x8
```

Inlining

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

function three() {
  return add(1, 2);
}
```

Prologue

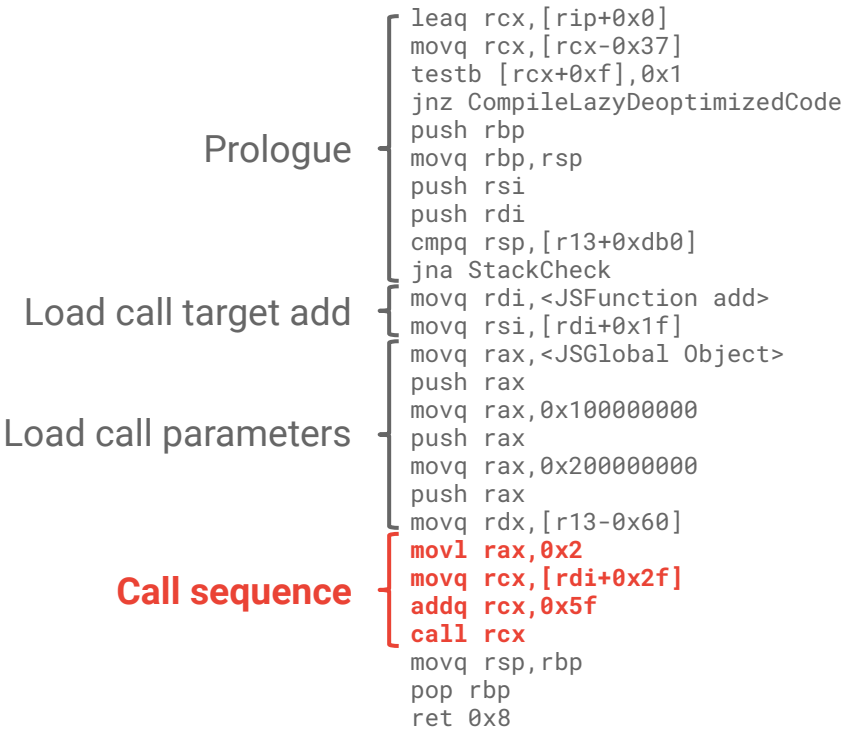
Load call target add

Load call parameters

```
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+0xdb0]
jna StackCheck
movq rdi,<JSFunction add>
movq rsi,[rdi+0x1f]
movq rax,<JSGlobal Object>
push rax
movq rax,0x100000000
push rax
movq rax,0x200000000
push rax
movq rdx,[r13-0x60]
movl rax,0x2
movq rcx,[rdi+0x2f]
addq rcx,0x5f
call rcx
movq rsp,rbp
pop rbp
ret 0x8
```

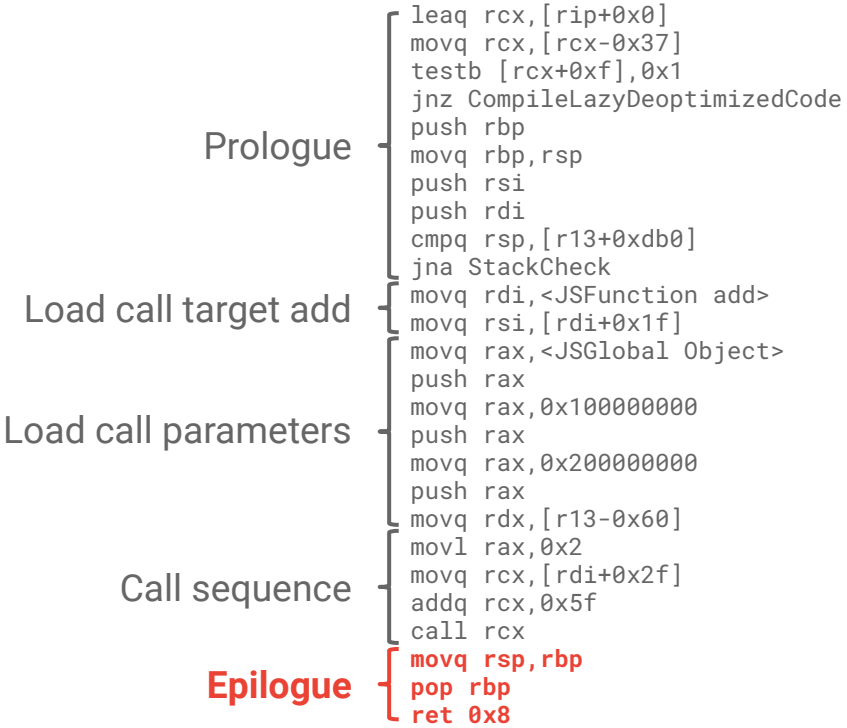
Inlining

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

function three() {
  return add(1, 2);
}
```



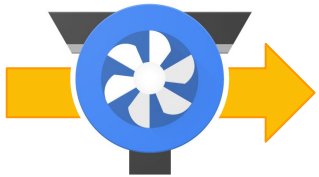
Inlining

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

function three() {
  return add(1, 2);
}
```



Inlining

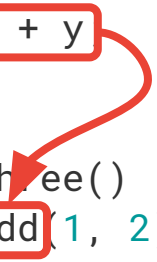
```
function add(x, y) {
  return x + y
}

function three() {
  return add(1, 2);
}
```



```
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+0xdb0]
jna StackCheck
movq rax,0x300000000
movq rsp,rbp
pop rbp
ret 0x8
```

Inlining

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

function three() {
  return add(1, 2);
}
```

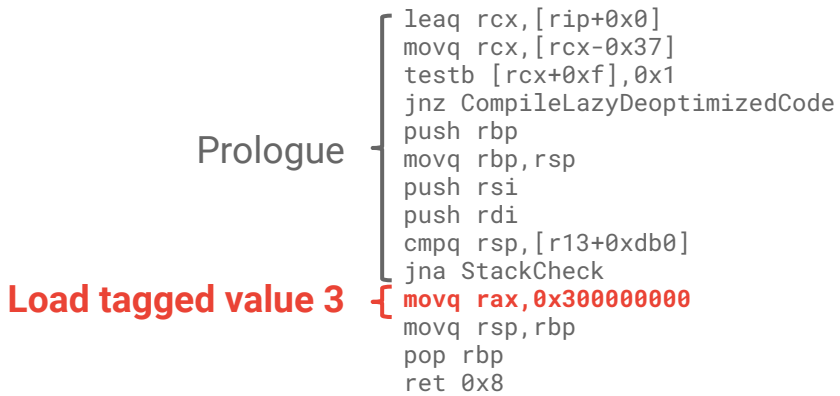


Prologue

```
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+0xdb0]
jna StackCheck
movq rax,0x300000000
movq rsp,rbp
pop rbp
ret 0x8
```

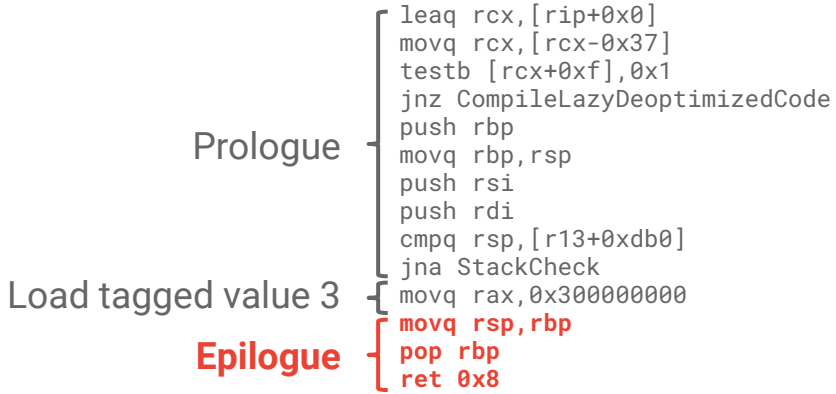
Inlining

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

function three() {
  return add(1, 2);
}
```



Inlining

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

function three() {
  return add(1, 2);
}
```



Inlining

```
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+0xdb0]
jna StackCheck
movq rdi,<JSFunction add>
movq rsi,[rdi+0x1f]
movq rax,<JSGlobal Object>
push rax
movq rax,0x100000000
push rax
movq rax,0x200000000
push rax
movq rdx,[r13-0x60]
movl rax,0x2
movq rcx,[rdi+0x2f]
addq rcx,0x5f
call rcx
movq rsp,rbp
pop rbp
ret 0x8
```

+

```
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+0xdb0]
jna StackCheck
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
```

VS

```
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+0xdb0]
jna StackCheck
movq rax,0x300000000
movq rsp,rbp
pop rbp
ret 0x8
```

Inlining

- Crucial to reduce small function overhead
  - Not just in JavaScript!
- Makes other optimizations more effective
  - Constant folding
  - Strength reduction
  - Redundancy elimination
  - Escape Analysis and Scalar Replacement of Aggregates

Builtin Inlining

Higher-order Array Builtins

```
function allPositive(a) {
  return a.every(x => x >= 0);
}


allPositive([-1, 0, 1]);
// => false
allPositive([1, 2, 3, 4]);
// => true
```

Higher-order Array Builtins

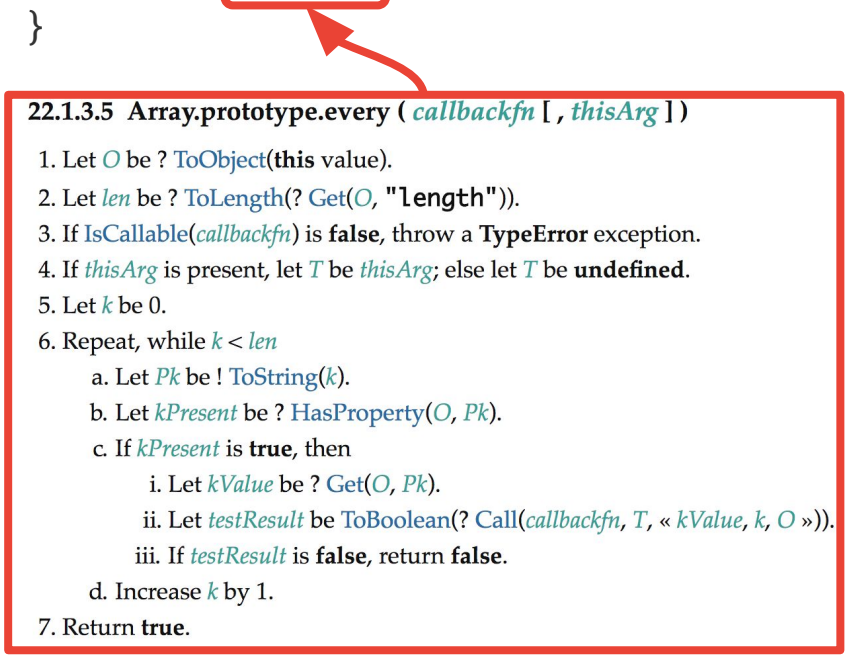
```
function allPositive(a) {
  return a.every(x => x >= 0);
}
```

VS

```
function allPositiveManual(a) {
  var l = a.length;
  for (var i = 0; i < l; ++i) {
    if (a[i] < 0) {
      return false;
    }
  }
  return true;
}
```

Higher-order Array Builtins

```
function allPositive(a) {
  return a.every(x => x >= 0);
}
```

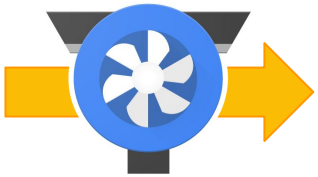


22.1.3.5 Array.prototype.every (*callbackfn* [, *thisArg*])

1. Let *O* be ? ToObject(**this** value).
2. Let *len* be ? ToLength(? Get(*O*, "length")).
3. If IsCallable(*callbackfn*) is **false**, throw a **TypeError** exception.
4. If *thisArg* is present, let *T* be *thisArg*; else let *T* be **undefined**.
5. Let *k* be 0.
6. Repeat, while *k* < *len*
    a. Let *Pk* be ! ToString(*k*).
    b. Let *kPresent* be ? HasProperty(*O*, *Pk*).
    c. If *kPresent* is **true**, then
        i. Let *kValue* be ? Get(*O*, *Pk*).
        ii. Let *testResult* be ToBoolean(? Call(*callbackfn*, *T*, « *kValue*, *k*, *O* »)).
        iii. If *testResult* is **false**, return **false**.
    d. Increase *k* by 1.
7. Return **true**.

Higher-order Array Builtins

```
function allPositive(a) {
  return a.every(x => x >= 0);
}
```



```
function allPositive_every(a) {
  var c = x => x >= 0;
  var l = a.length;
  if (typeof c !== "function") {
    throw new TypeError();
  }
  for (var i = 0; i < l; ++i) {
    if (i in a) {
      if (!c(a[i])) {
        return false;
      }
    }
  }
  return true;
}
```

22.1.3.5 Array.prototype.every (*callbackfn* [, *thisArg*])

1. Let *O* be ? ToObject(**this** value).
2. Let *len* be ? ToLength(? Get(*O*, "length")).
3. If IsCallable(*callbackfn*) is **false**, throw a **TypeError** exception.
4. If *thisArg* is present, let *T* be *thisArg*; else let *T* be **undefined**.
5. Let *k* be 0.
6. Repeat, while *k* < *len*
   a. Let *Pk* be ! ToString(*k*).
   b. Let *kPresent* be ? HasProperty(*O*, *Pk*).
   c. If *kPresent* is **true**, then
      i. Let *kValue* be ? Get(*O*, *Pk*).
      ii. Let *testResult* be ToBoolean(? Call(*callbackfn*, *T*, « *kValue*, *k*, *O* »)).
      iii. If *testResult* is **false**, return **false**.
   d. Increase *k* by 1.
7. Return **true**.

Higher-order Array Builtins

```
function allPositive_every(a) {
  var c = x => x >= 0;
  var l = a.length;
  if (typeof c !== "function") {
    throw new TypeError();
  }
  for (var i = 0; i < l; ++i) {
    if (i in a) {
      if (!c(a[i])) {
        return false;
      }
    }
  }
  return true;
}
```

Higher-order Array Builtins

```
function allPositive_every(a) {
  var c = x => x >= 0;
  var l = a.length;
  if (typeof c !== "function") {
    throw new TypeError();
  }
  for (var i = 0; i < l; ++i) {
    if (i in a) {
      if (!c(a[i])) {
        return false;
      }
    }
  }
  return true;
}
```

- Callable check is redundant

Higher-order Array Builtins

```
function allPositive_every(a) {
  var c = x => x >= 0;
  var l = a.length;
  if (typeof c !== "function") {
    throw new TypeError();
  }
  for (var i = 0; i < l; ++i) {
    if (i in a) {
      if (!c(a[i])) {
        return false;
      }
    }
  }
  return true;
}
```

- Callable check is redundant
- Assuming that a is non-holey (learned from the `a.every` property access) and not mutated (speculated on), the `i in a` check is redundant

Higher-order Array Builtins

```
function allPositive_every(a) {
  var c = x => x >= 0;
  var l = a.length;
  if (typeof c !== "function") {
    throw new TypeError();
  }
  for (var i = 0; i < l; ++i) {
    if (i in a)
      if (!(a[i] >= 0)) {
        return false;
      }
    }
  }
  return true;
}
```

- Callable check is redundant
- Assuming that a is non-holey (learned from the `a.every` property access) and not mutated (speculated on), the `i in a` check is redundant
- Regular inlining can inline c at it's callsite

Higher-order Array Builtins

```
function allPositive_every(a) {
  var c = x => x >= 0;
  var l = a.length;
  if (typeof c !== "function") {
    throw new TypeError();
  }
  for (var i = 0; i < l; ++i) {
    if (i in a) {
      if (!(a[i] >= 0)) {
        return false;
      }
    }
  }
  return true;
}
```

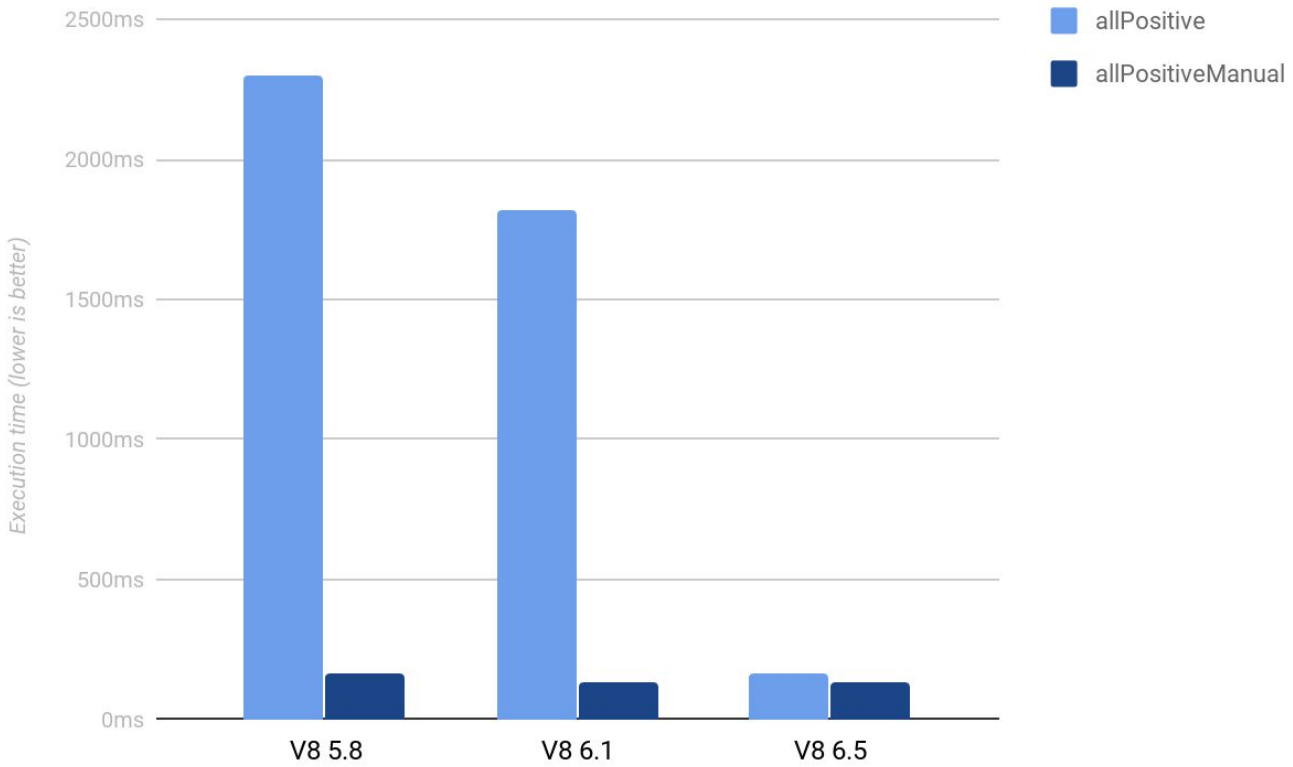
- Callable check is redundant
- Assuming that a is non-holey (learned from the `a.every` property access) and not mutated (speculated on), the `i in a` check is redundant
- Regular inlining can inline c at it's callsite
- Escape analysis and scalar replacement eliminate the closure allocation

Higher-order Array Builtins

```
function allPositive(a) {
  return a.every(x => x >= 0);
}
```

```
function allPositive_every(a) {
  var l = a.length;
  for (var i = 0; i < l; ++i) {
    if (!(a[i] >= 0)) {
      return false;
    }
  }
  return true;
}
```

Higher-order Array Builtins - Performance

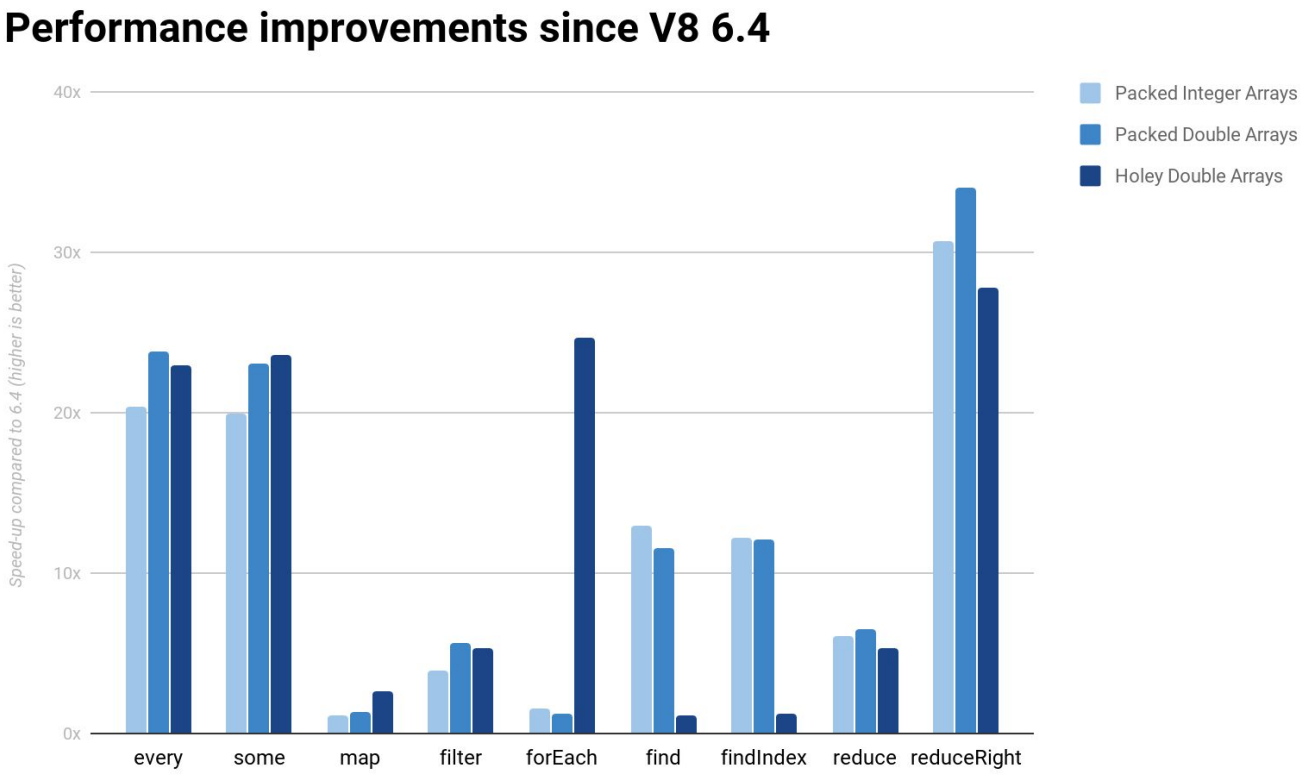


Higher-order Array Builtins - Performance

Full support planned for Chrome 66 / Node 10

- `Array.prototype.map`
- `Array.prototype.filter`
- `Array.prototype.every`
- `Array.prototype.some`
- `Array.prototype.reduce`
- `Array.prototype.reduceRight`
- `Array.prototype.forEach`
- `Array.prototype.find`
- `Array.prototype.findIndex`

Higher-order Array Builtins - Performance



Not just `Array` builtins...

Function builtins

```
function dispatch(f, ...args) {
  return f.apply(this, args);
}
```

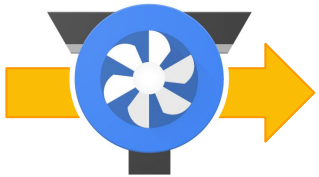
`args` are *forwarded* to `f`

19.2.3.1 Function.prototype.apply (*thisArg*, *argArray*)

1. If IsCallable(*func*) is **false**, throw a **TypeError** exception.
2. If *argArray* is **undefined** or **null**, then
    a. Perform PrepareForTailCall().
    b. Return ? Call(*func*, *thisArg*).
3. Let *argList* be ? CreateListFromArrayLike(*argArray*).
4. Perform PrepareForTailCall().
5. Return ? Call(*func*, *thisArg*, *argList*).

Function builtins

```
function dispatch(f, ...args) {
  return f.apply(this, args);
}
```



```
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+0xe60]
jna StackCheck
movq rbx,[rbp+0x10]
testb rbx,0x1
jz Deoptimize
movq rax,<JSFunction hidden class>
cmpq [rbx-0x1],rax
jnz Deoptimize
movq rdx,[rbp+0x18]
push rdx
xorl rax,rax
movl rcx,0x1
movq rdi,rbx
movq rsi,[rbp-0x8]
call CallForwardVarargs
movq rsp,rbp
pop rbp
ret 0x10
```

Function builtins

```
function dispatch(f, ...args) {
  return f.apply(this, args);
}
```

Prologue

```
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+0xe60]
jna StackCheck
movq rbx,[rbp+0x10]
testb rbx,0x1
jz Deoptimize
movq rax,<JSFunction hidden class>
cmpq [rbx-0x1],rax
jnz Deoptimize
movq rdx,[rbp+0x18]
push rdx
xorl rax,rax
movl rcx,0x1
movq rdi,rbx
movq rsi,[rbp-0x8]
call CallForwardVarargs
movq rsp,rbp
pop rbp
ret 0x10
```

Function builtins

```
function dispatch(f, ...args) {
  return f.apply(this, args);
}
```

Prologue

```
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+0xe60]
jna StackCheck
```

Check f is not a small integer

```
movq rbx,[rbp+0x10]
testb rbx,0x1
jz Deoptimize
```

```
movq rax,<JSFunction hidden class>
cmpq [rbx-0x1],rax
jnz Deoptimize
movq rdx,[rbp+0x18]
push rdx
xorl rax,rax
movl rcx,0x1
movq rdi,rbx
movq rsi,[rbp-0x8]
call CallForwardVarargs
movq rsp,rbp
pop rbp
ret 0x10
```

Function builtins

```
function dispatch(f, ...args) {
  return f.apply(this, args);
}
```

Prologue

```
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+0xe60]
jna StackCheck
```

Check f is not a small integer

```
movq rbx,[rbp+0x10]
testb rbx,0x1
jz Deoptimize
```

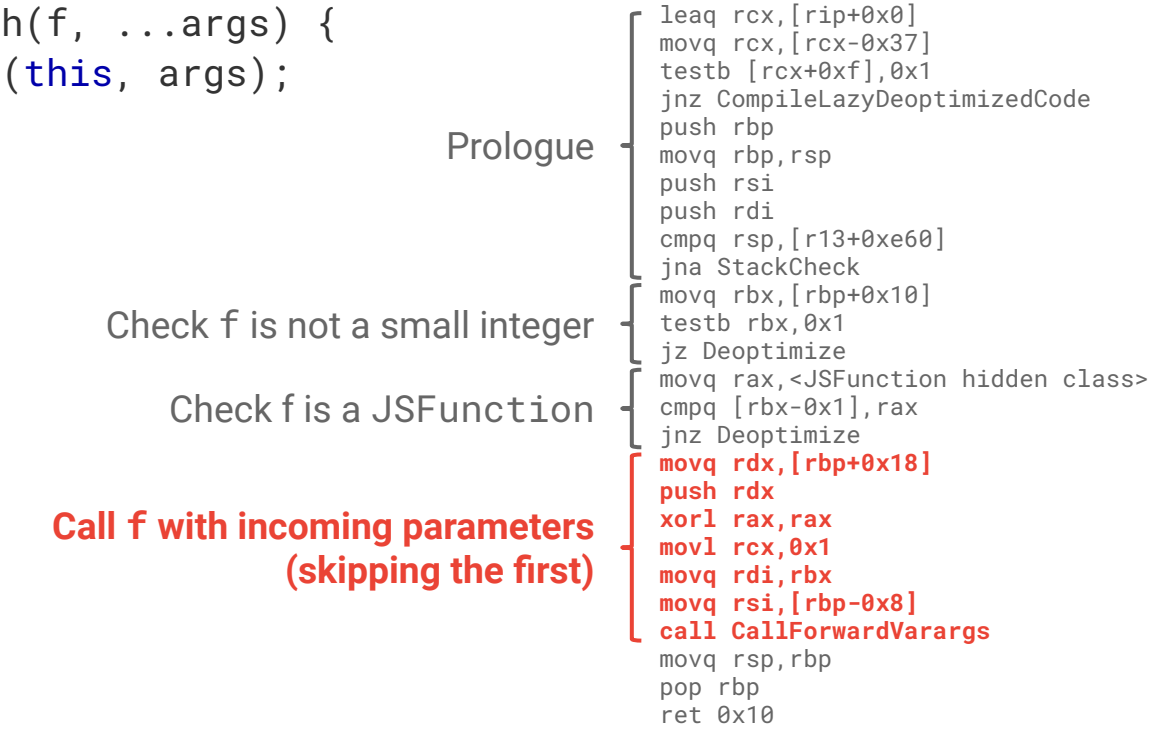
Check f is a JSFunction

```
movq rax,<JSFunction hidden class>
cmpq [rbx-0x1],rax
jnz Deoptimize
```

```
movq rdx,[rbp+0x18]
push rdx
xorl rax,rax
movl rcx,0x1
movq rdi,rbx
movq rsi,[rbp-0x8]
call CallForwardVarargs
movq rsp,rbp
pop rbp
ret 0x10
```

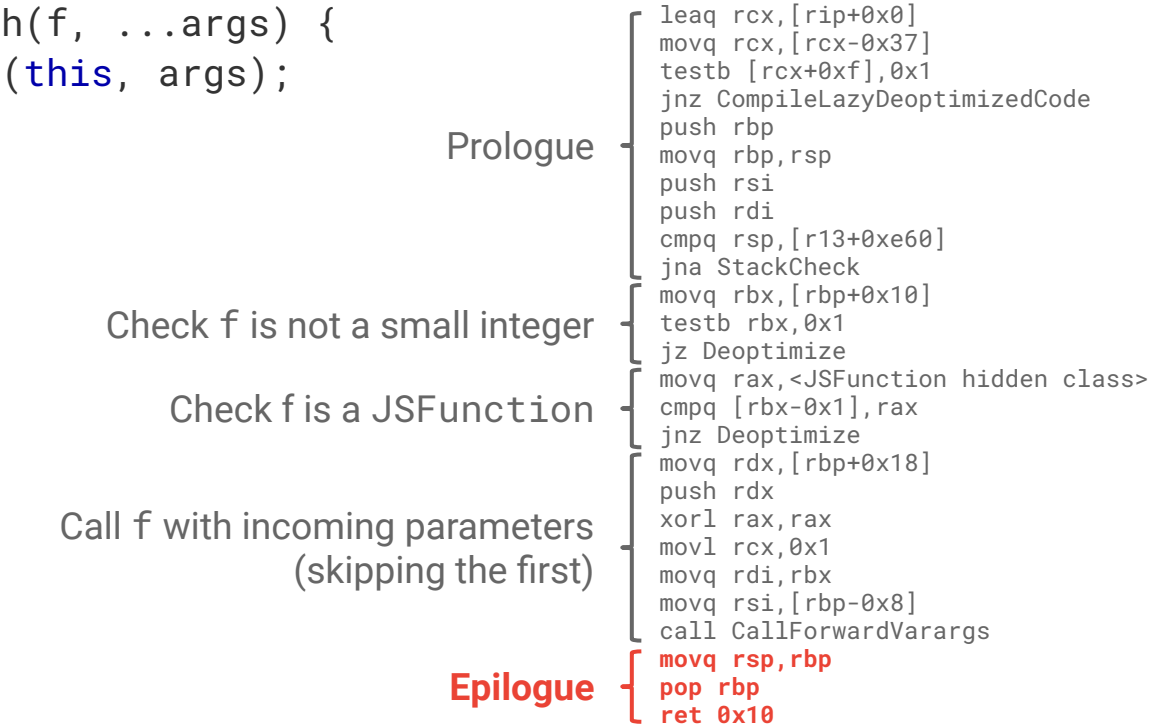
Function builtins

```
function dispatch(f, ...args) {
  return f.apply(this, args);
}
```



Function builtins

```
function dispatch(f, ...args) {
  return f.apply(this, args);
}
```



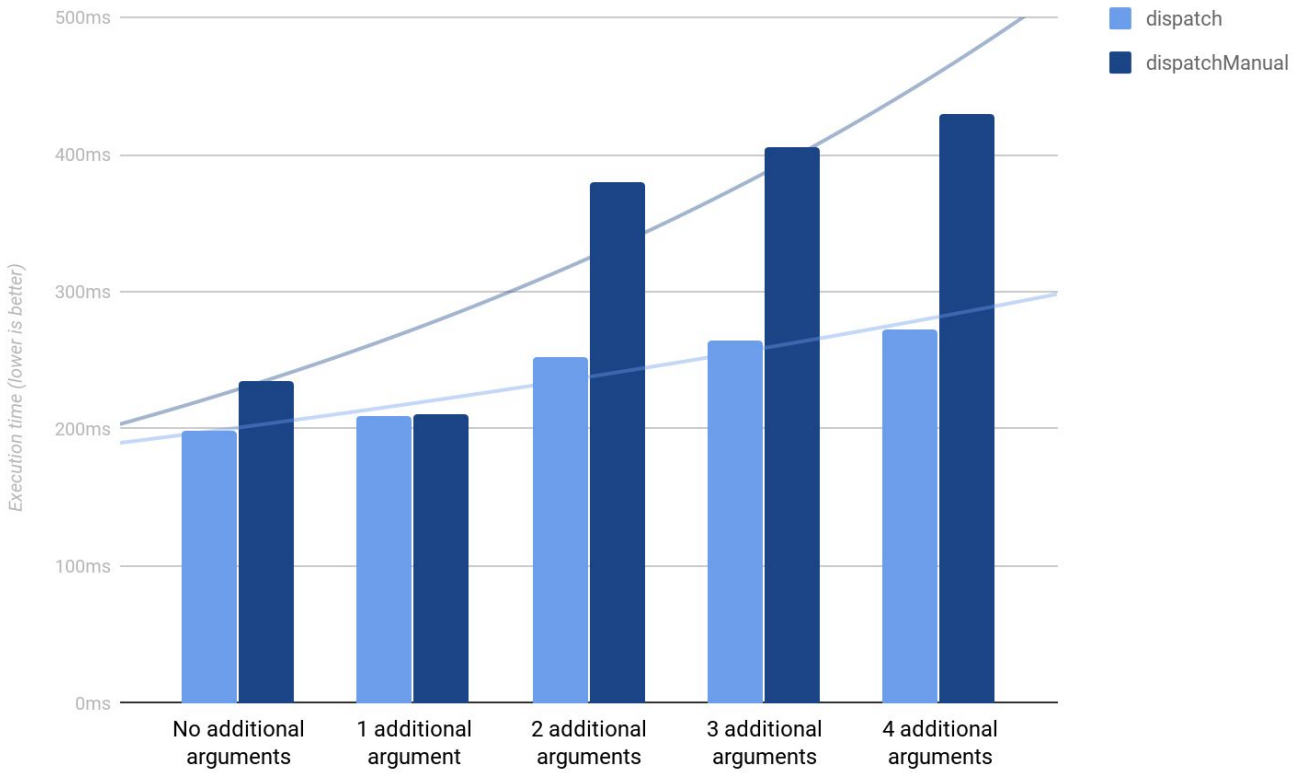
Function builtins

```
function dispatch(f, ...args) {
  return f.apply(this, args);
}
```

VS

```
function dispatchManual() {
  var l = arguments.length;
  var f = arguments[0];
  switch (l) {
    case 1: return f.call(this);
    case 2: return f.call(this,
                          arguments[1]);
    default: {
      var a = new Array(l - 1);
      for (var i = 1; i < l; ++i)
        a[i - 1] = arguments[i];
      return f.apply(this, a);
    }
  }
}
```

Function builtins



Builtin Inlining

- Makes idiomatic JavaScript performant
  - Crucial for functional-style programming
- Again makes other optimizations more effective
  - Inlining
  - Strength reduction
  - Redundancy elimination
  - Escape Analysis and Scalar Replacement


Predictable
Performance

Performance cliffs in Crankshaft

```
(function good() {
  const start = Date.now();
  for (var i = 0; i < 1e8; i++) {}
  console.log(Date.now() - start);
})();
```

Runs ~80ms

```
(function bad() {
  const start = Date.now();
  for (var i = 0; i < 1e8; i++) {}
  console.log(Date.now() - start);
  const whatever = 1;
})();
```

Runs ~230ms

~3x slowdown with Crankshaft

v8 / v8

mirrored from https://chromium.googlesource.com/v8/v8.git

Watch 708 | Star 7,270 | Fork 1,603

Code | Pull requests 0 | Projects 0 | Wiki | Insights

[crankshaft] Remove Crankshaft.

Browse files

```
R=danno@chromium.org
BUG=v8:6408

Change-Id: I6613557e474f415293feb164a30c15485d81ff2c
Reviewed-on: https://chromium-review.googlesource.com/547717
Reviewed-by: Daniel Clifford <danno@chromium.org>
Commit-Queue: Michael Starzinger <mstarzinger@chromium.org>
Cr-Commit-Position: refs/heads/master@{#46212}
```

master | 6.4.272 ... 6.1.279

Michael Starzinger committed with Commit Bot on Jun 26 — 1 parent f030838 — commit c751e79ec382a7240b6bd2987b29c1677394920b

Showing 134 changed files with 2 additions and 130,380 deletions.

Unified | Split

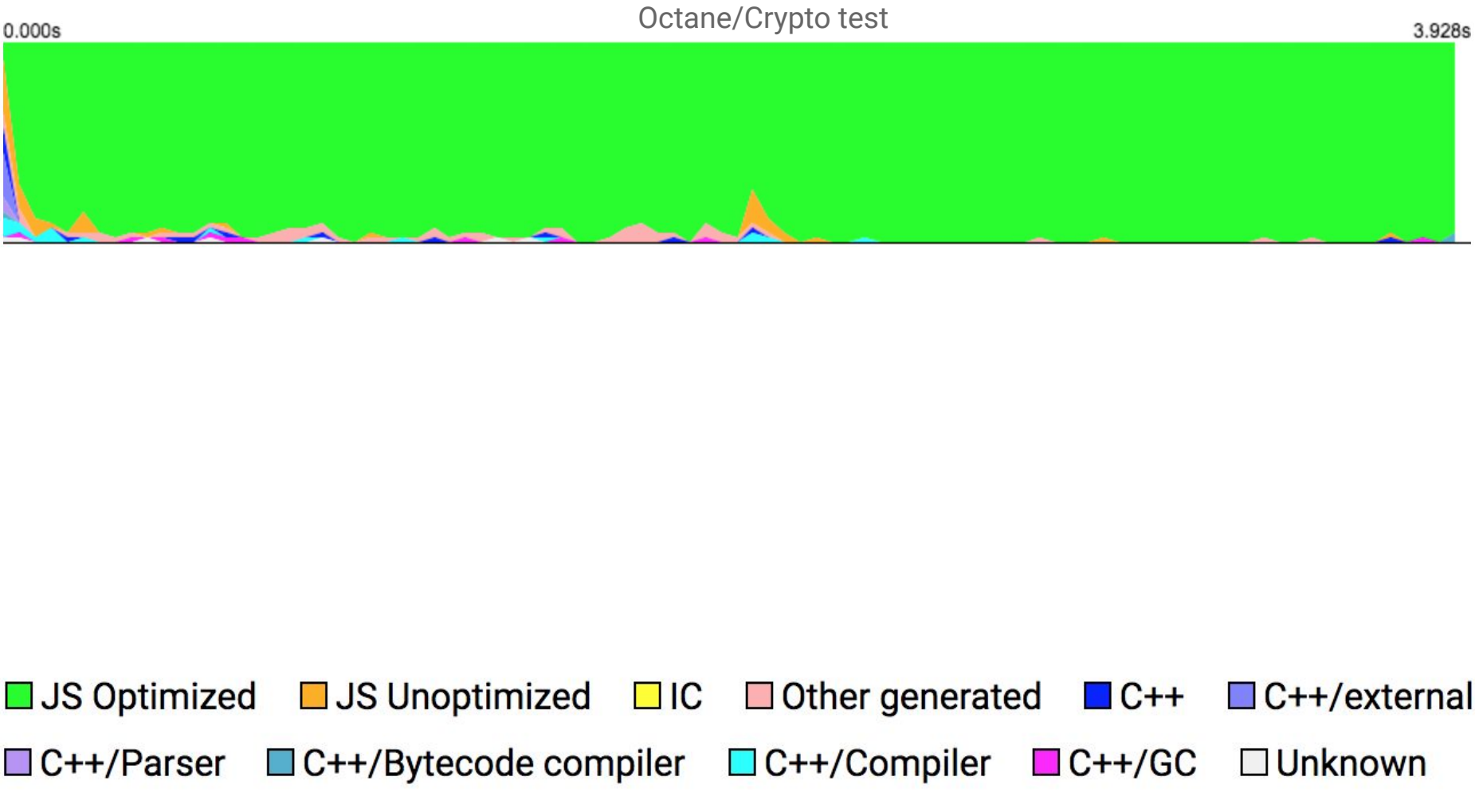
Google

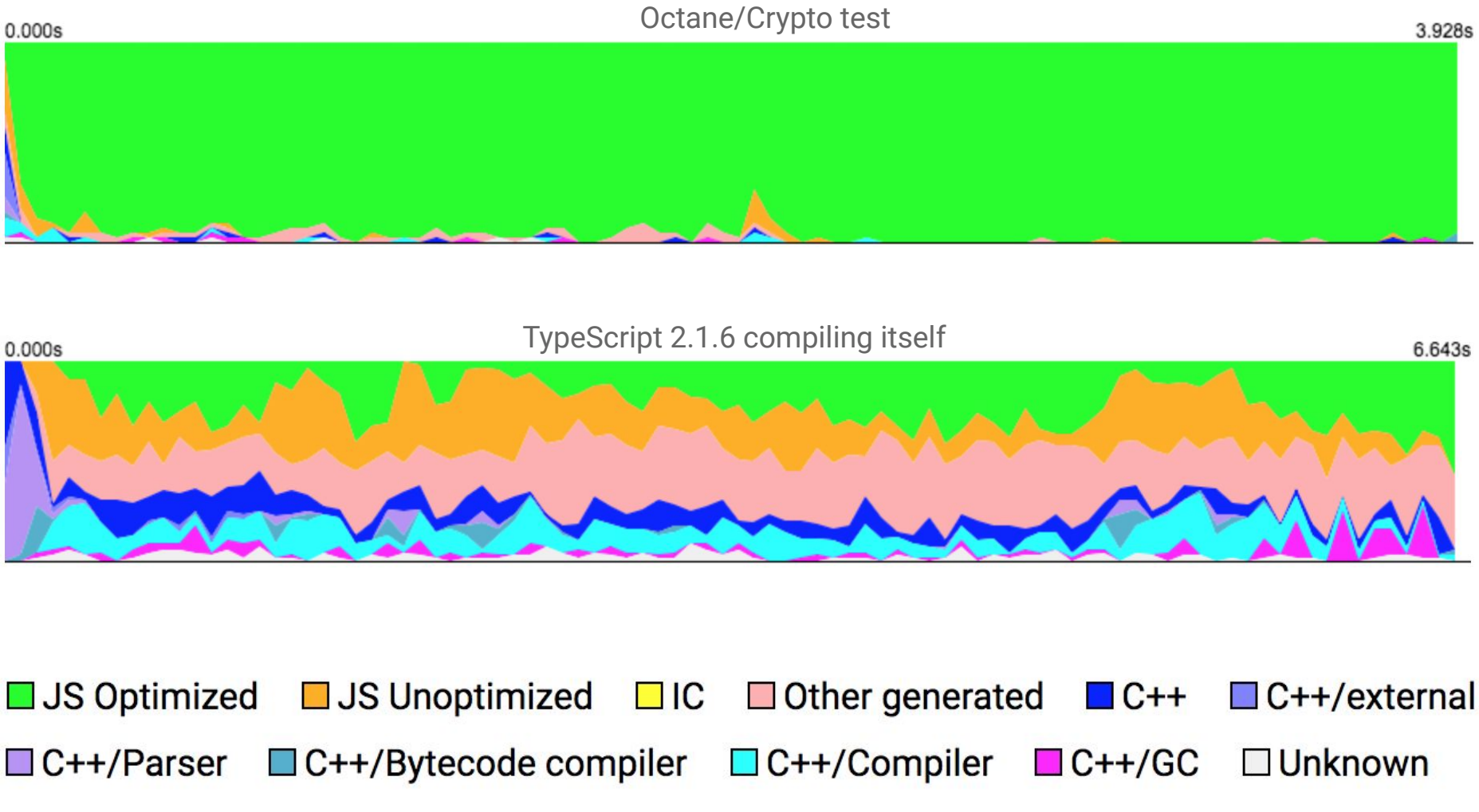
Google

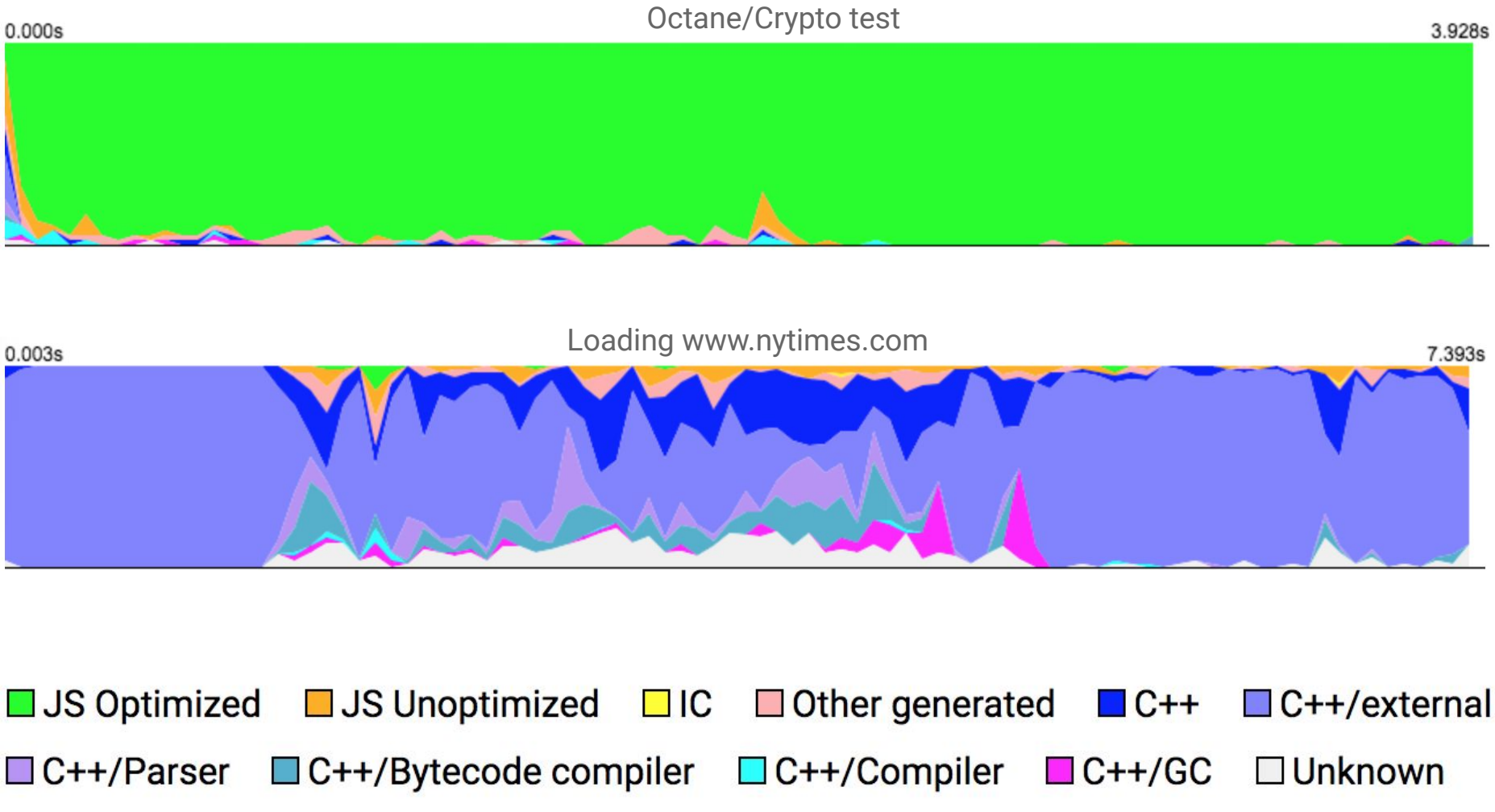
Optimization Killers

- Generators and **async** functions
- **for-** nd destructuring
- **try**-**catch** a **try**-**finally**
- Compound **let** or **co** gnment
- Object literals that tain **__p to__**, or **get** or **set** declarations.
- **debugger** **with** statements
- Li calls to **eval**( )
- …


It’s not just
optimized code...

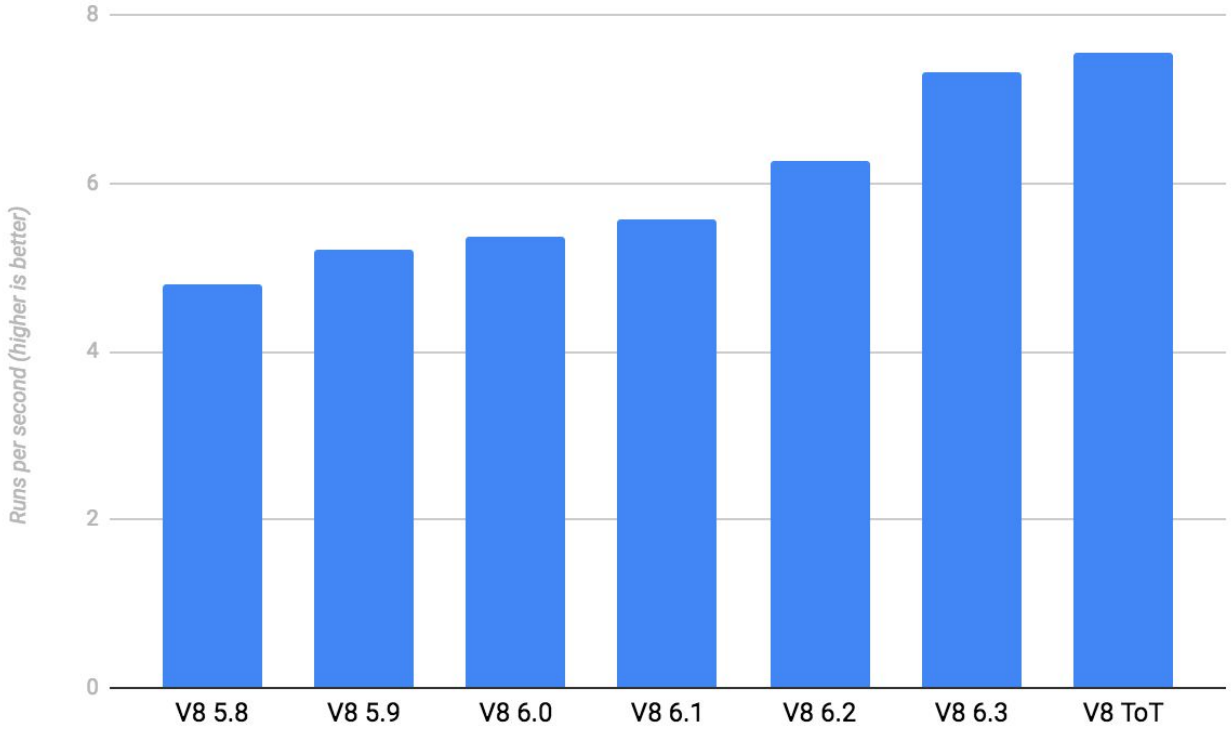

Octane/Crypto test
0.000s
3.928s
JS Optimized
JS Unoptimized
IC
Other generated
C++
C++/external
C++/Parser
C++/Bytecode compiler
C++/Compiler
C++/GC
Unknown


Octane/Crypto test
0.000s
3.928s
TypeScript 2.1.6 compiling itself
0.000s
6.643s
JS Optimized
JS Unoptimized
IC
Other generated
C++
C++/external
C++/Parser
C++/Bytecode compiler
C++/Compiler
C++/GC
Unknown


Octane/Crypto test
0.000s
3.928s
Loading www.nytimes.com
0.003s
7.393s
JS Optimized
JS Unoptimized
IC
Other generated
C++
C++/external
C++/Parser
C++/Bytecode compiler
C++/Compiler
C++/GC
Unknown

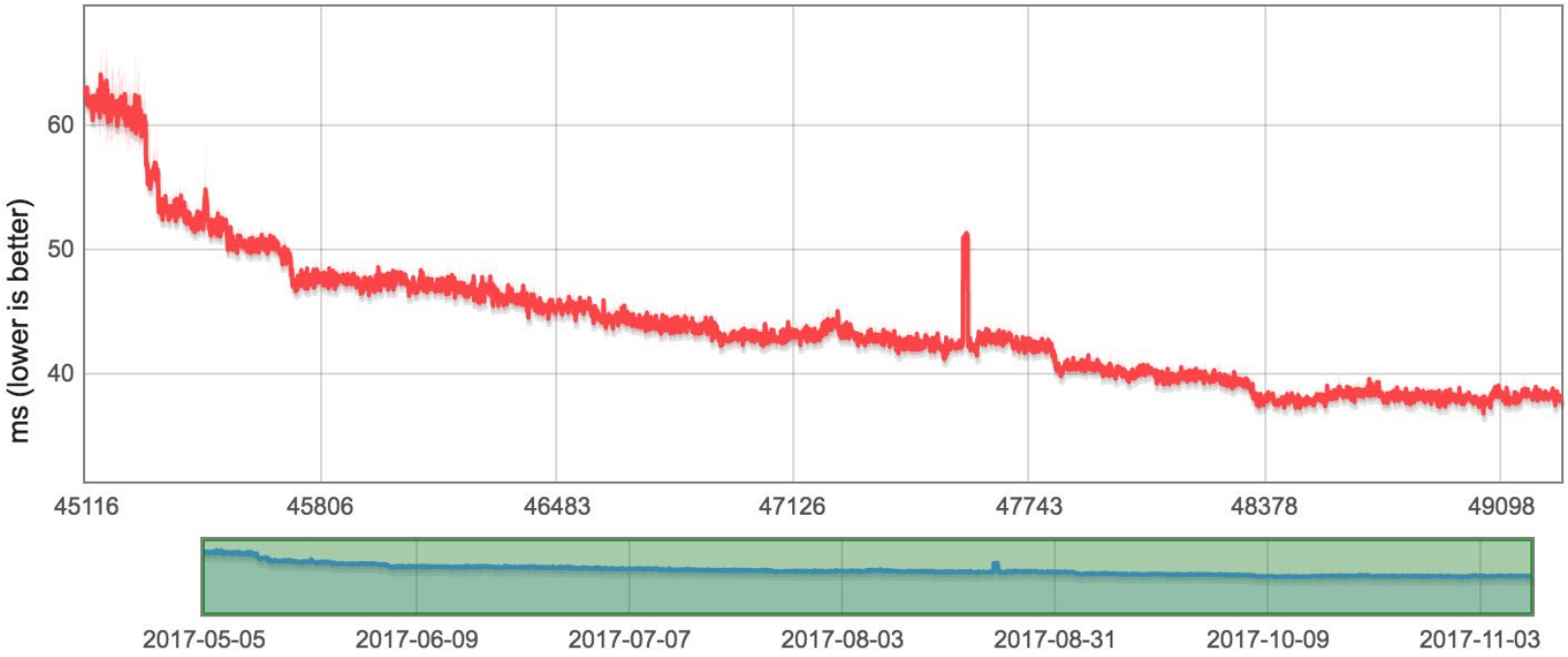
Performance Results

60% faster web developer tools



v8project.blogspot.com/2017/11/web-tooling-benchmark.html

1.7× faster on ARES-6



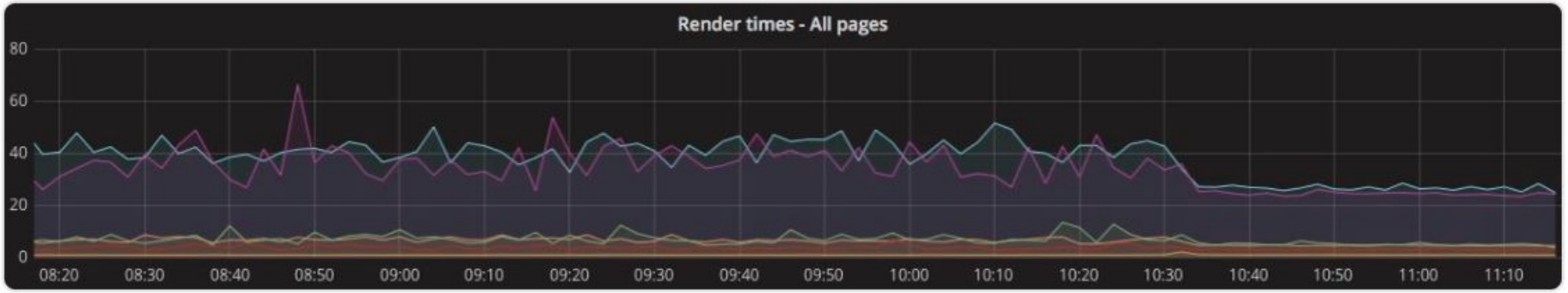
browserbench.org/ARES-6

Google

Alexander Pope

@popeindustries

Here’s a pretty picture (React SSR render).
Can you guess when we deployed Node 8.3?



twitter.com/popeindustries/status/895575424406867968

What’s coming in 2018?

- Better asynchronous performance
- BigInt, class features, async iteration
- Reducing memory footprint

Follow us!

@v8js

Google

Benedikt Meurer
@bmeurer

Thanks!