V8、JavaScript 的现在与未来

justjavac(迷渡)

React从入门到精通

掌握当下最热门的前端利器

王沛
eBay 资深技术专家

你将获得

1. 全面学习 React 常用技术栈
2. 深入理解 React 设计模式
3. 常见场景下的编程实战指南
4. 掌握用 React 开发大型项目的能力

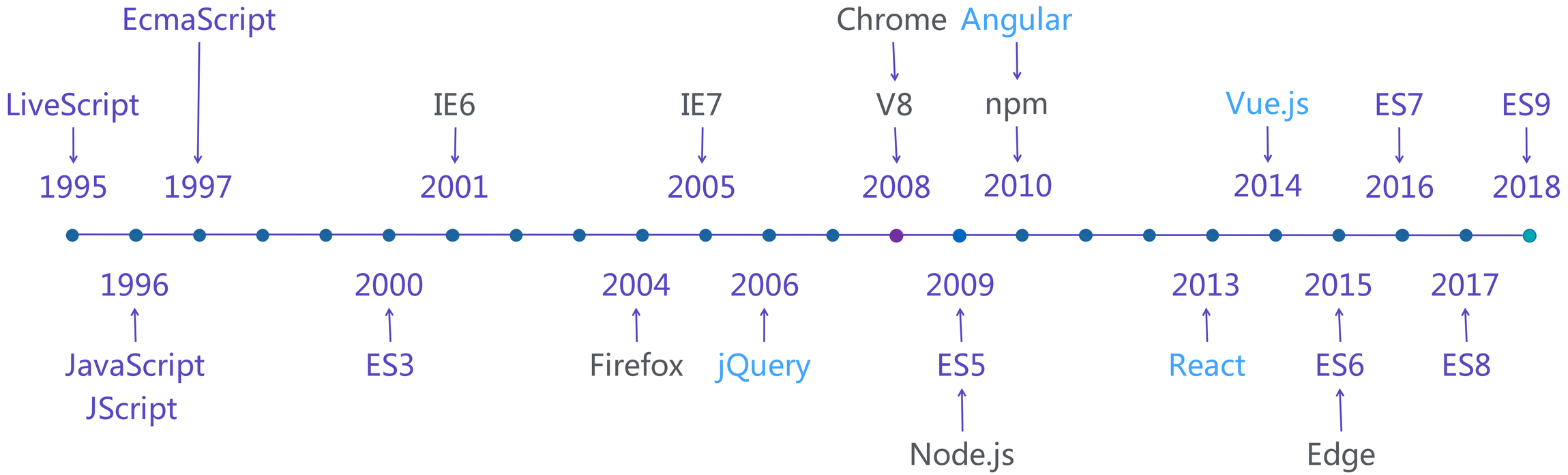
立即扫码，免费试看

TABLE OF

CONTENTS 大纲

- History of JavaScript
- V8 引擎针对 JavaScript 的性能优化
- TC39 的最新提案以及应用场景
- 编写高性能 JavaScript 代码

history of JavaScript

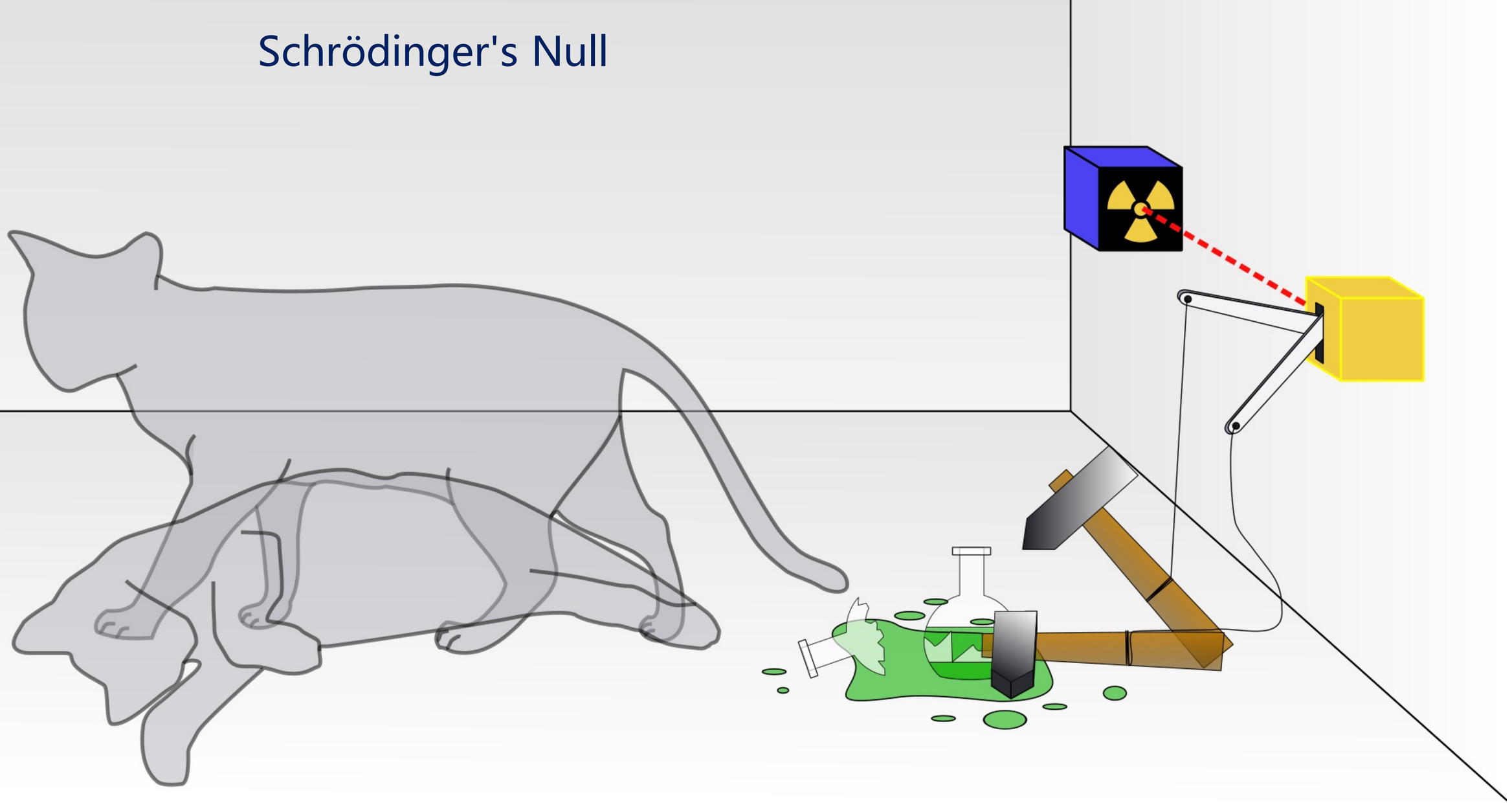



I created JavaScript in
10 days

bug or feature?

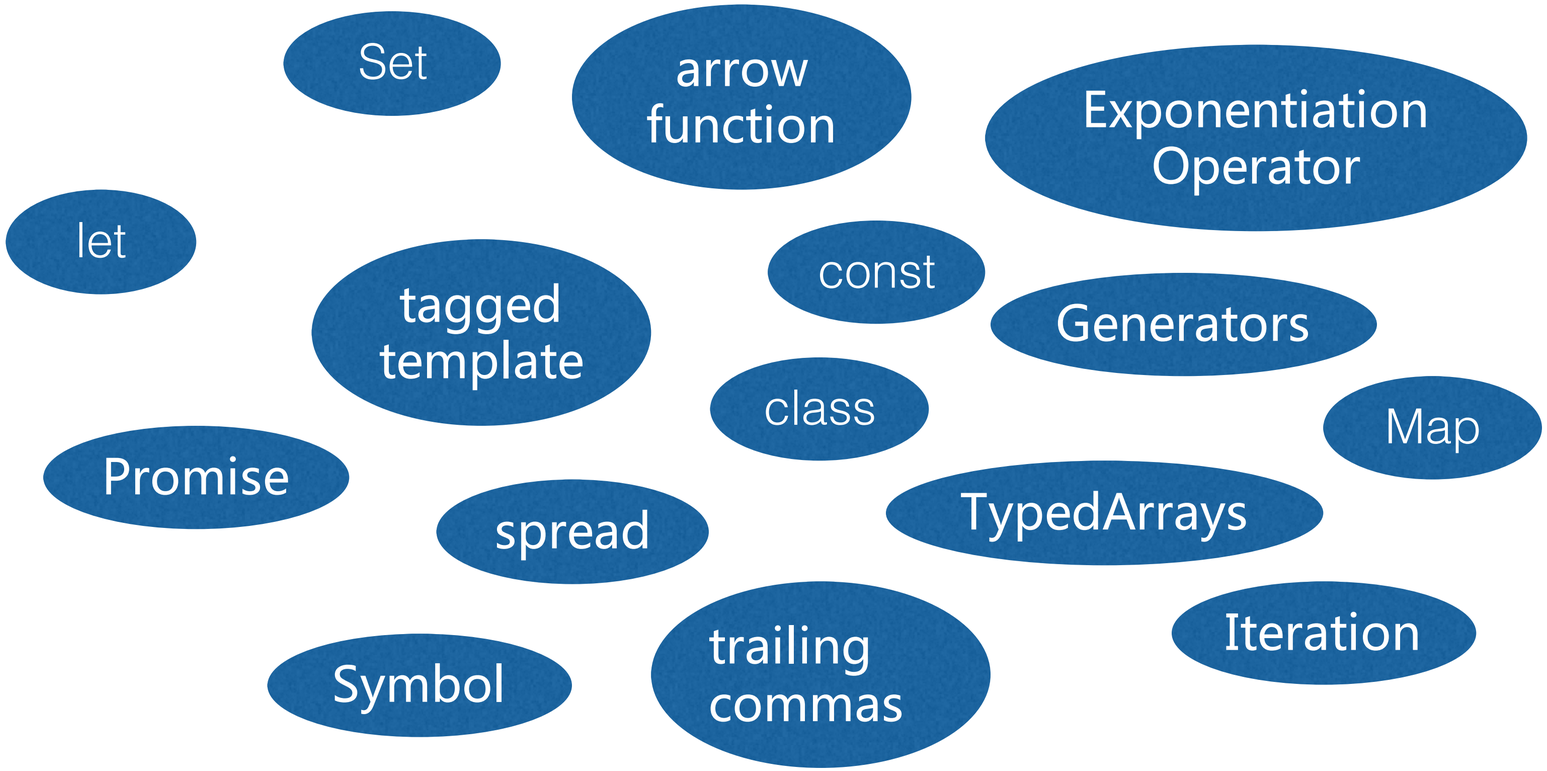
- `typeof null === 'object';  // WTF?`
- `null instanceof Object === false;  // WTF??`
- `null instanceof null; // TypeError: Right-hand side of 'instanceof' is not an object WTF???`

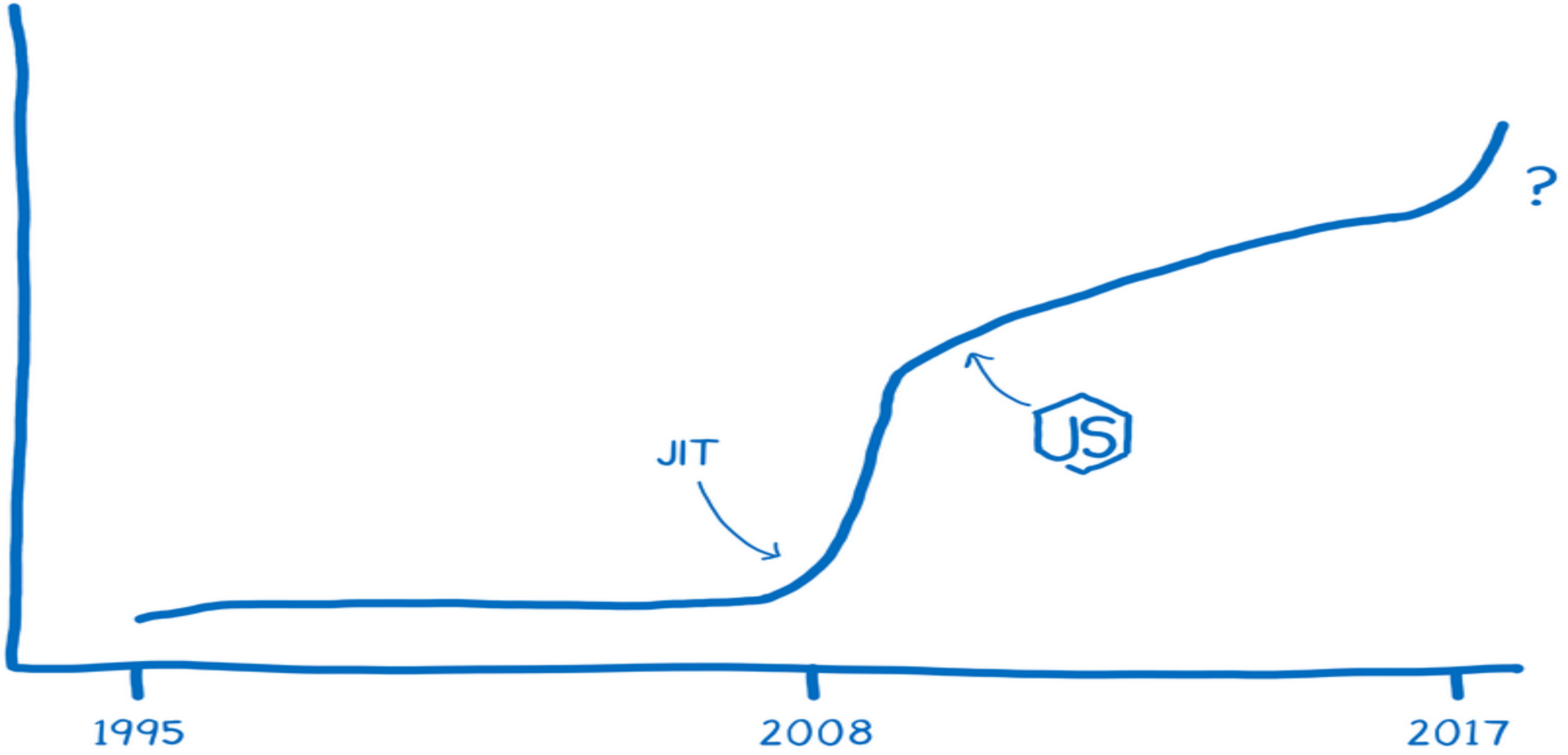
Schrödinger's Null



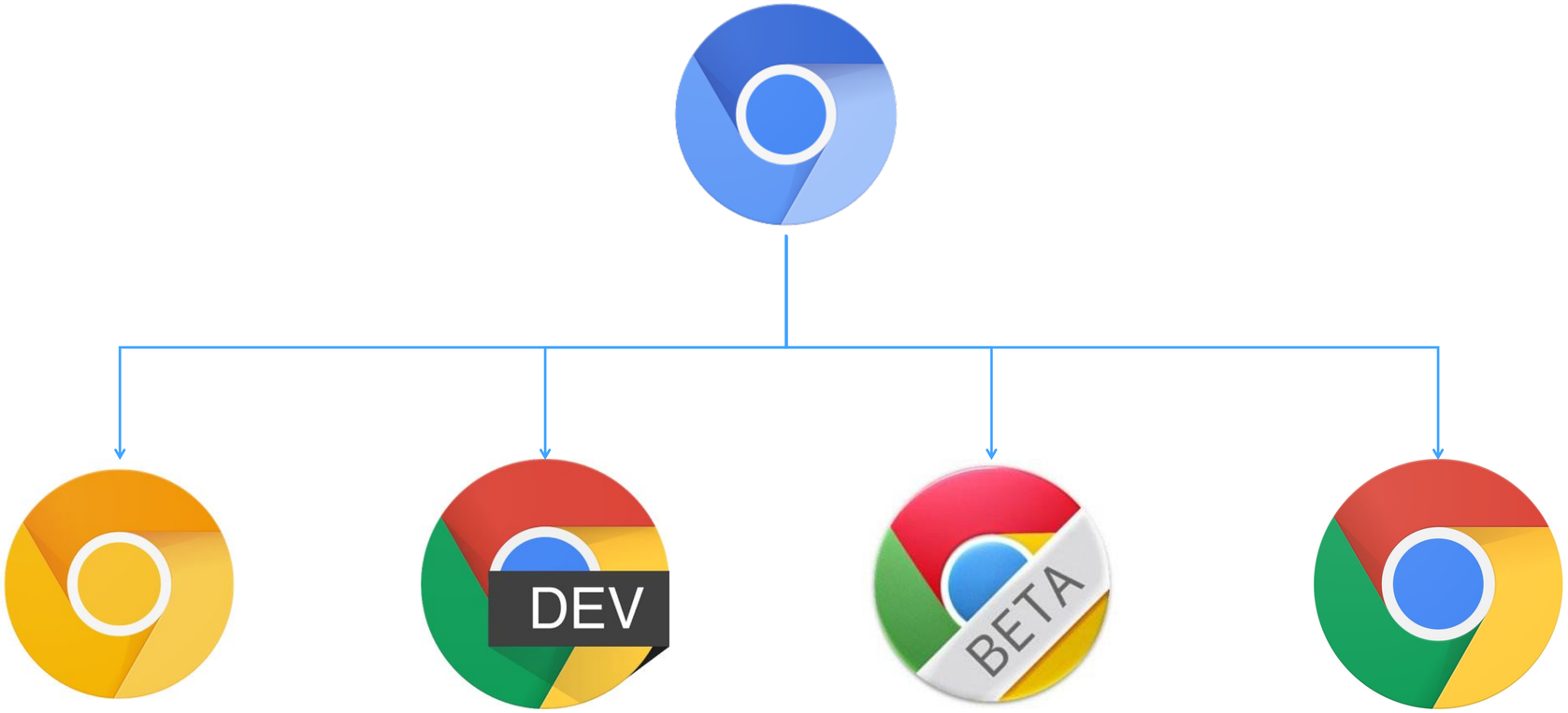
图片来源：wikipedia.org

ES201x


Set
arrow function
Exponentiation Operator
let
const
tagged template
Generators
class
Map
Promise
spread
TypedArrays
Iteration
Symbol
trailing commas


JIT
JS
?
1995
2008
2017

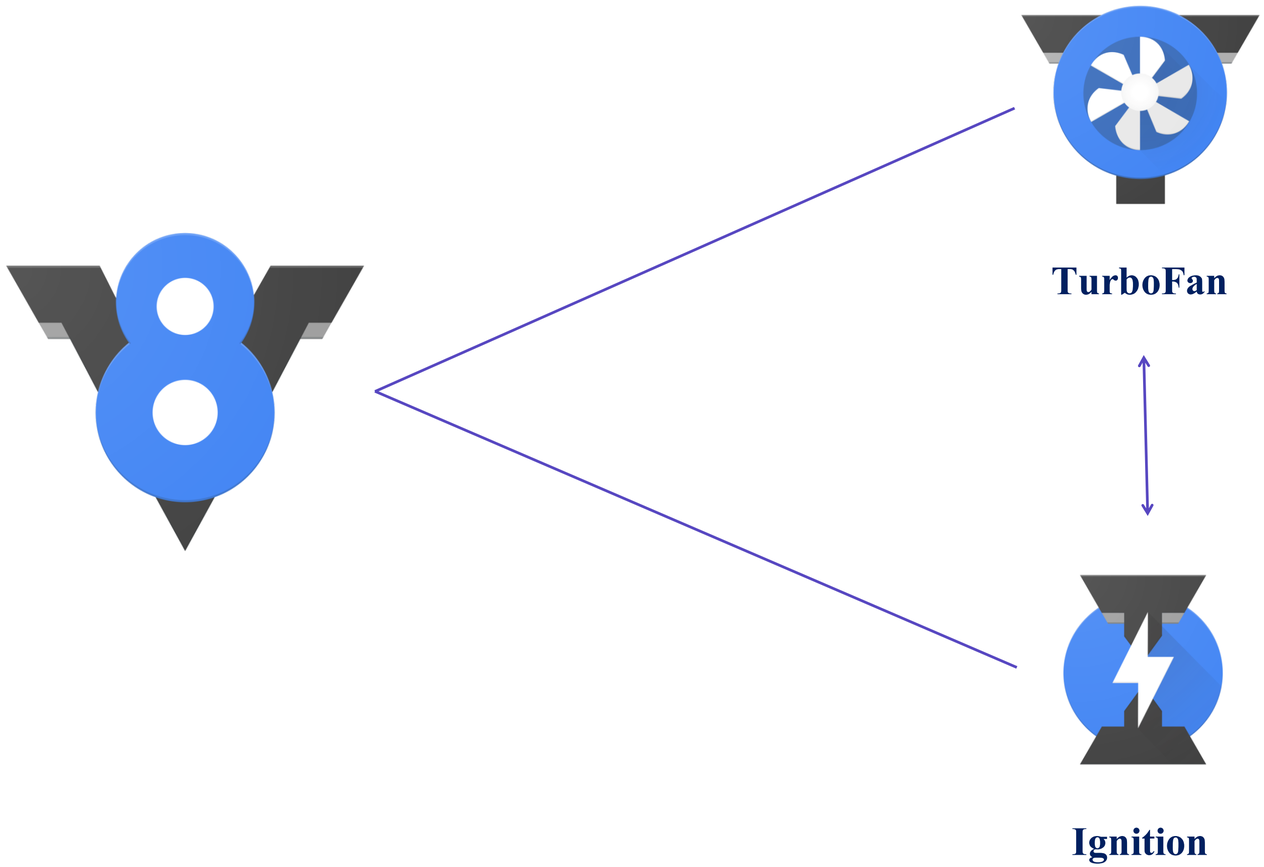
V8 Release Process



每天　每周　每6周　每6周

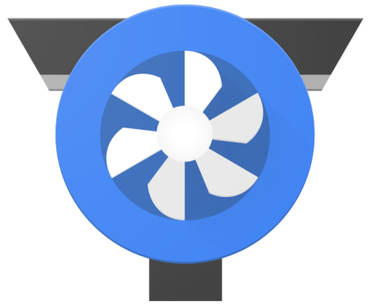
V8 引擎针对 JavaScript 的性能优化

- V8 概览
- 快速的算术运算
- 数组的高阶函数
- 内联编译
- 逃逸分析


TurboFan
Ignition

Optimization Killers

- 生成器和 **async** 函数
- **for-of** 和数组解构
- **try-catch** 和 **try-finally**
- 复合 **let** 或 **const** 赋值
- 包含 **__proto__**、**get**、**set** 的对象声明
- **debugger** 或 **with** 语句
- **eval()**
- ...

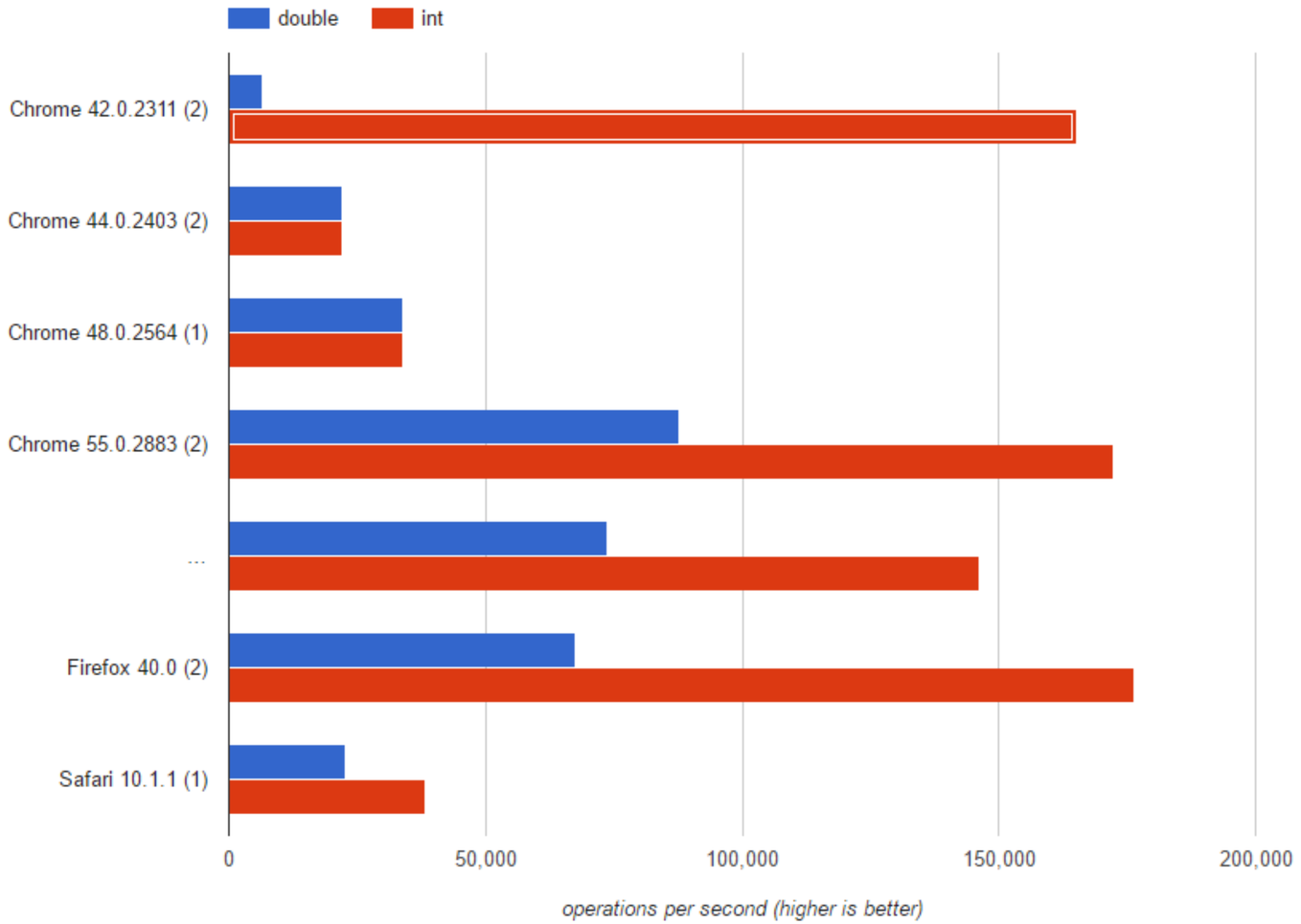


TurboFan

快速的算术运算

- `const i = 1;` `// SmallInteger (Smi)`
- `const d = 1.1;` `// HeapNumber (Double)`
- `const o = {};` `// JSObject(Heap)`

SMI vs Double



代码: a+b

- mov eax, a
- mov ebx, b
- call RuntimeAdd ×

代码: a+b

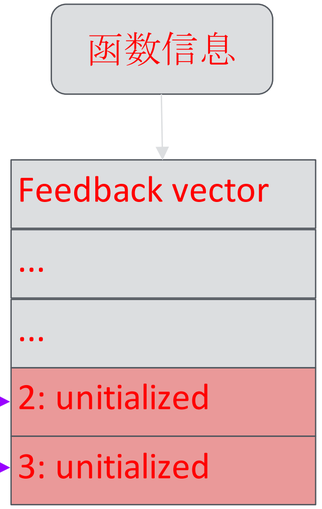
- mov eax, a
- mov ebx, b
- ~~call RuntimeAdd~~ ×
- add eax, ebx ✓

Type feedback

二元运算符可以收集函数的 type feedback 信息.

```
function f(a, b){
    var c = a + a;
    return b + c;
}
```

```
StackCheck
Ldar a0
Add a0, [2]
Star r0
Ldar r0
Add a1, [3]
Return
```

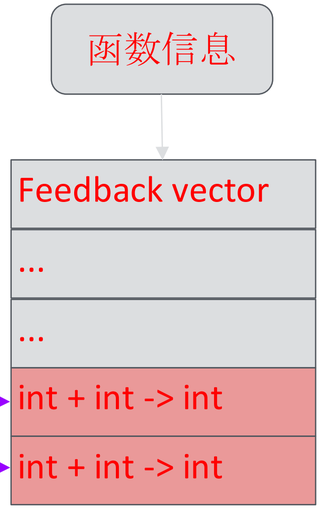


Type feedback

二元运算符可以收集函数的 type feedback 信息.

```
function f(a, b){
    var c = a + a;
    return b + c;
}

f(1, 2);
```

```
StackCheck
Ldar a0
Add a0, [2]
Star r0
Ldar r0
Add a1, [3]
Return
```

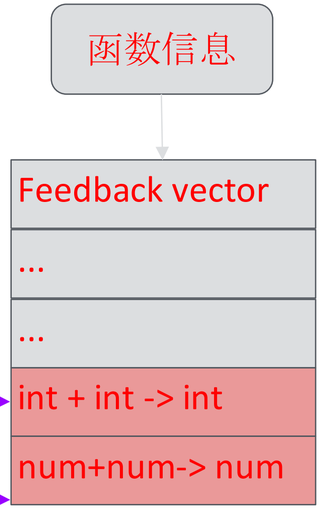


Type feedback

二元运算符可以收集函数的 type feedback 信息.

```
function f(a, b){
    var c = a + a;
    return b + c;
}

f(1, 2);
f(1, 0.1)
```

```
StackCheck
Ldar a0
Add a0, [1]
Star r0
Ldar r0
Add a1, [1.1]
Return
```

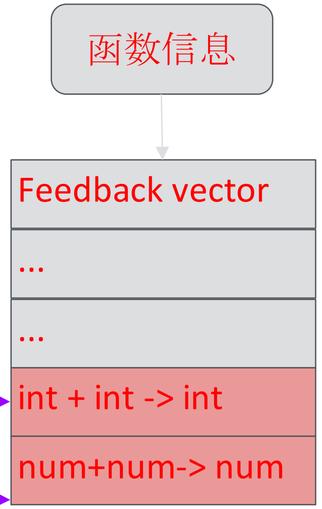


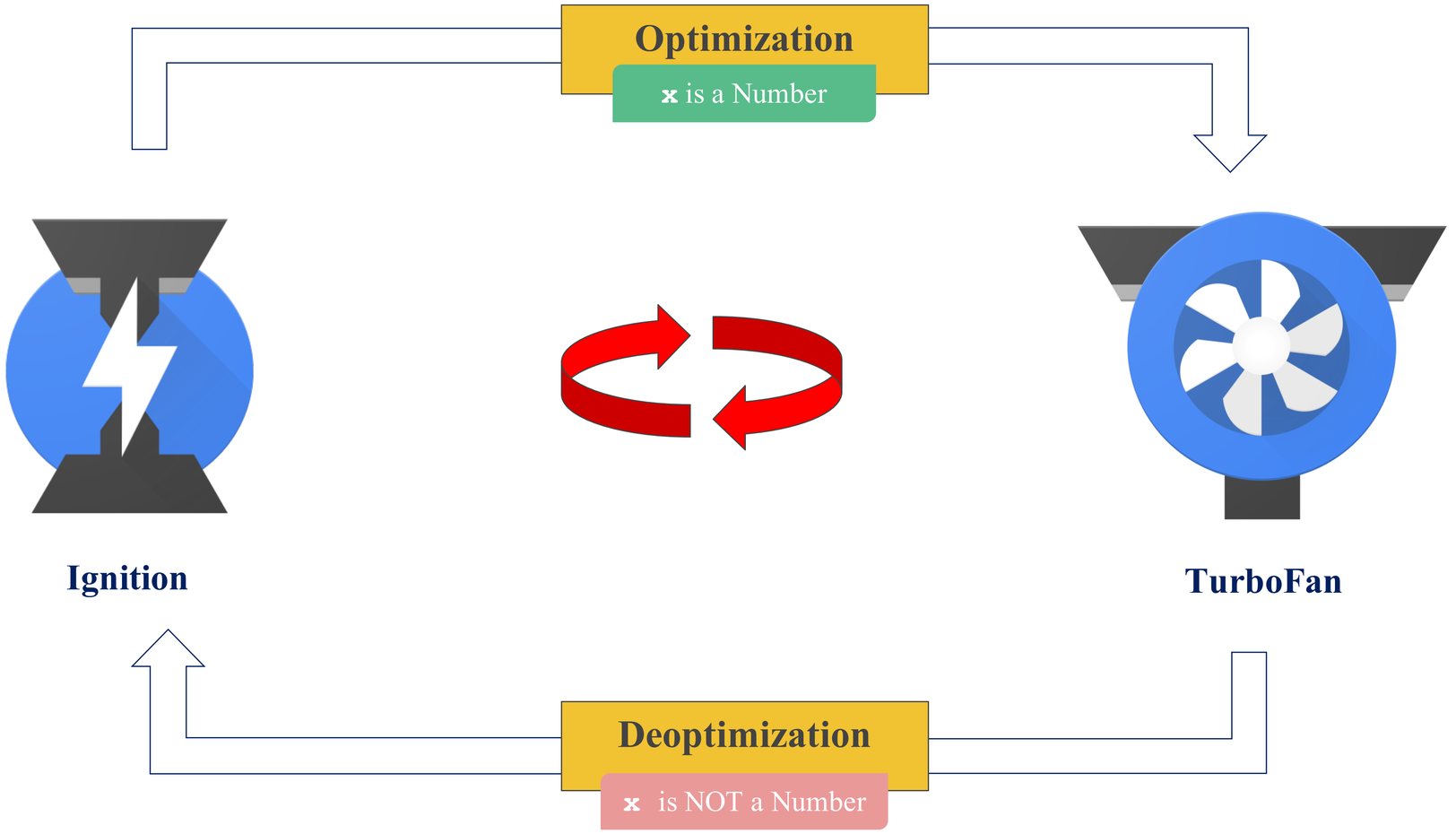
Type feedback

二元运算符可以收集函数的 type feedback 信息.

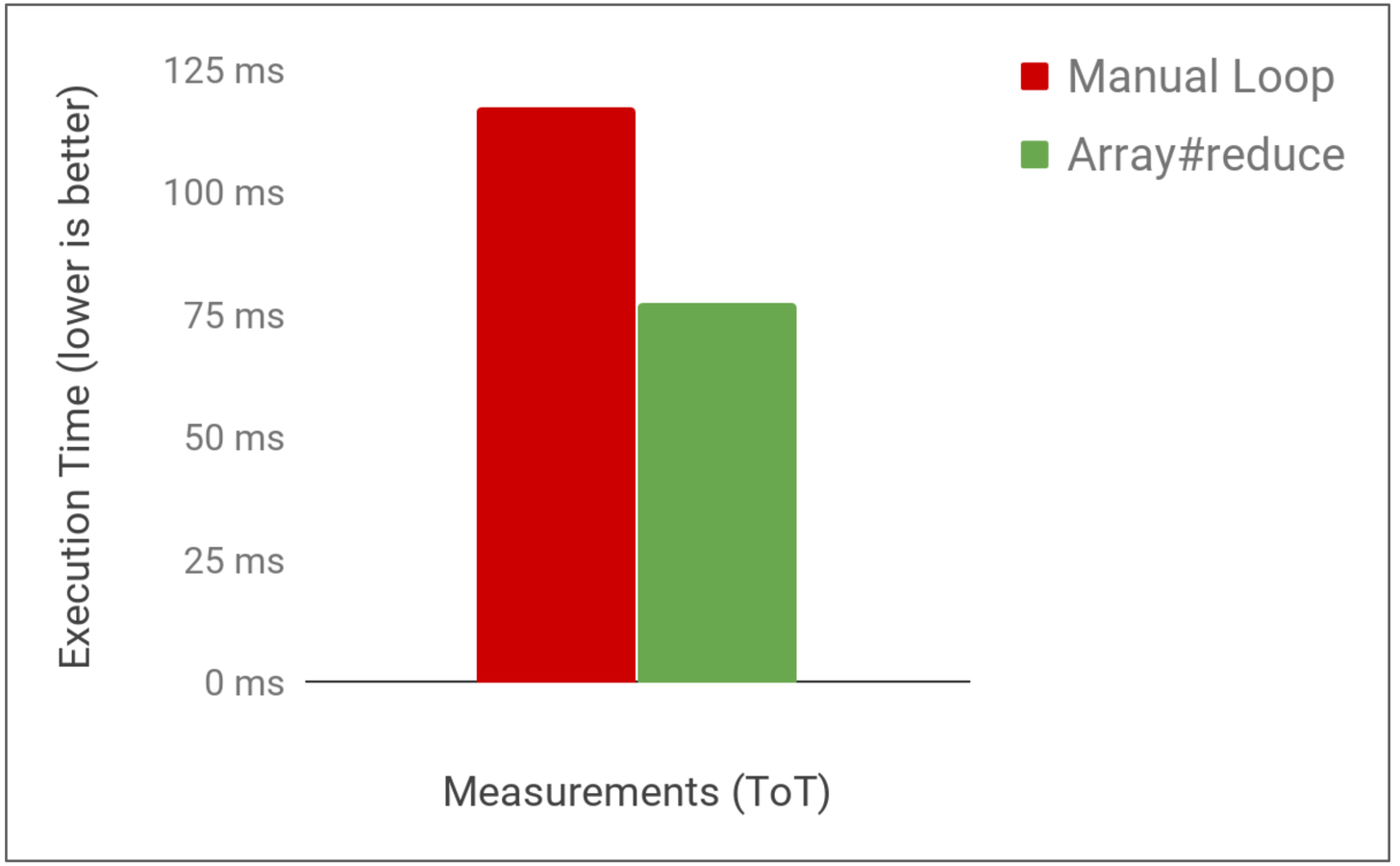
```
function f(a, b){
    var c = a + a;
    return b + c;
}

f(1, 2);
f(1, 0.1) // Deopt!!!
```

```
StackCheck
Ldar a0
Add a0, [1]
Star r0
Ldar r0
Add a1, [1.1]
Return
```




Optimization
x is a Number
Ignition
TurboFan
Deoptimization
x is NOT a Number

数组的高阶函数


125 ms
100 ms
75 ms
50 ms
25 ms
0 ms
Execution Time (lower is better)
Manual Loop
Array#reduce
Measurements (ToT)

```
const array = [1, 2, 3];
// elements kind: PACKED_SMI_ELEMENTS
array.push(4.56);
// elements kind: PACKED_DOUBLE_ELEMENTS
array.push('x');
// elements kind: PACKED_ELEMENTS
```

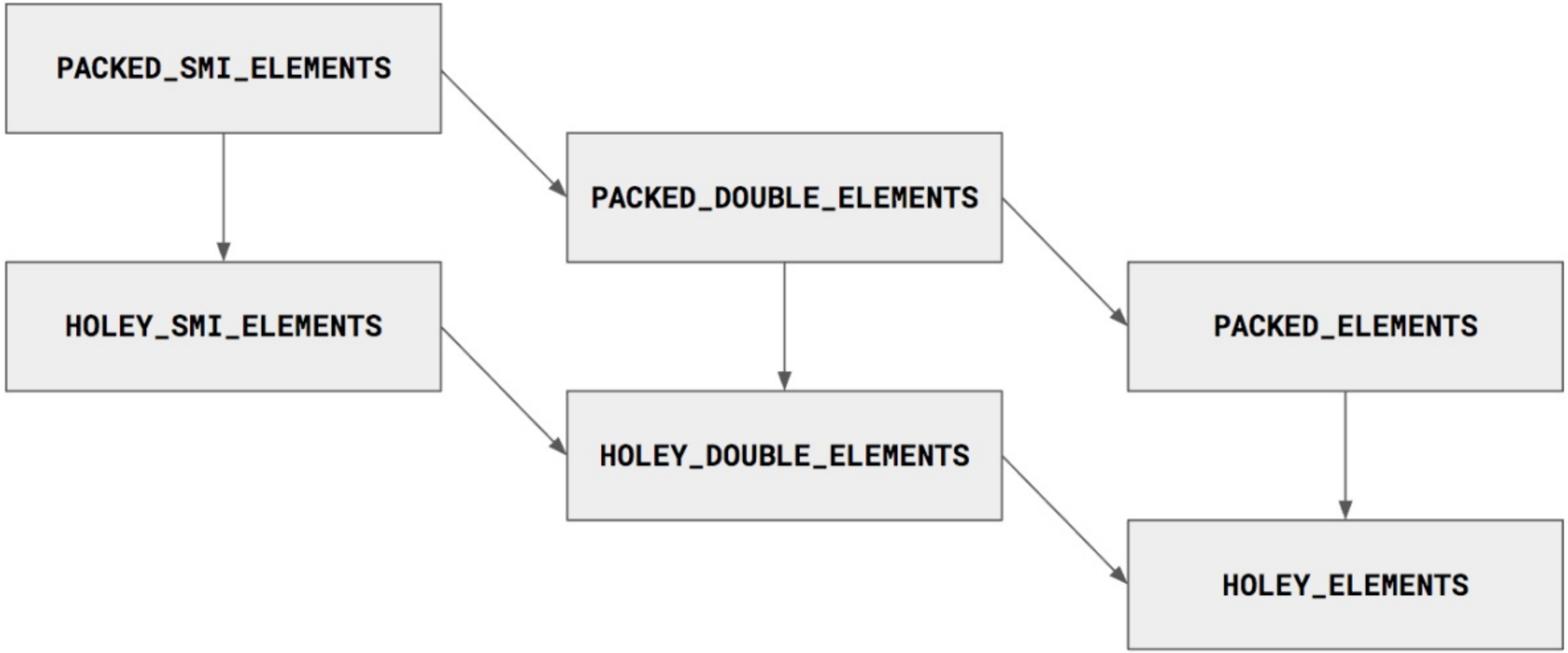
- `array.length; // 5`

index	0	1	2	3	4
value	1	2	3	4.56	“x”

- `array.length; // 5`
- `array[9] = 1;`
- `// array[5] ~ array[8]`
- `// elements kind: HOLEY_ELEMENTS`

index	0	1	2	3	4	5	6	7	8	9
value	1	2	3	4.56	“x”					1

- `array[8]; // ???`
- `8 >= 0 && 8 < array.length; // bounds check`
- `hasOwnProperty(array, '8');`
- `hasOwnProperty(Array.prototype, '8');`
- `hasOwnProperty(Object.prototype, '8');`


PACKED_SMI_ELEMENTS
PACKED_DOUBLE_ELEMENTS
PACKED_ELEMENTS
HOLEY_SMI_ELEMENTS
HOLEY_DOUBLE_ELEMENTS
HOLEY_ELEMENTS

```
const array = [1, 2, 3];
```

对数组的元素进行累加

b 的初始值为 0

```
array.reduce((b, x) => b + x, 0)

= ((0 + 1) + 2) + 3

= 6
```

```
function foo(A) {

  return A.reduce((b, x) => b + x, 0);

}
```

```
function foo_inlined(A) {
  const f = (b, x) => b + x;
  const len = A.length;
  if (typeof f !== "function") {
    throw new TypeError();
  }
  let b = 0;
  for (let i = 0; i < len; ++i) {
    if (i in A) { b = f(b, A[i]); }
  }
  return b;
}
```

```
function Array_reduce_inline(A) {
  const f = (b, x) => b + x;
  if (typeof f !== "function") {
    throw new TypeError();
  }
  let b = 0;
  for (let i = 0; i < A.length; ++i) {
    if (i in A) { b = f(b, A[i]); }
  }
  return b;
}
```

PACKED_SMI_ELEMENTS

```
function Array_reduce_inline(A) {
  const f = (b, x) => b + x;
  if (typeof f !== "function") {
    throw new TypeError();
  }
  let b = 0;
  for (let i = 0; i < A.length; ++i) {
    if (i in A) { b = f(b, A[i]); }
  }
  return b;
}
```

PACKED_SMI_ELEMENTS

```
function Array_reduce_inline(A) {
  const f = (b, x) => b + x;
  if (typeof f !== "function") {
    throw new TypeError();
  }
  let b = 0;
  for (let i = 0; i < A.length; ++i) {
    if (i in A) { b = b + A[i]; }
  }
  return b;
}
```

PACKED_SMI_ELEMENTS

```
function Array_reduce_inline(A) {
```

PACKED_SMI_ELEMENTS

```
  let b = 0;
  for (let i = 0; i < A.length; ++i) {
    if (i in A) { b = b + A[i]; }


  }
  return b;
}
```

```
function Array_reduce_inline(A) {
  if (A not PACKED_SMI_ELEMENTS) Deopt;
  let b = 0;
  for (let i = 0; i < A.length; ++i) {
    if (i in A) { b = b + A[i]; }
    if (A not PACKED_SMI_ELEMENTS) Deopt;
  }
  return b;
}
```

PACKED_SMI_ELEMENTS

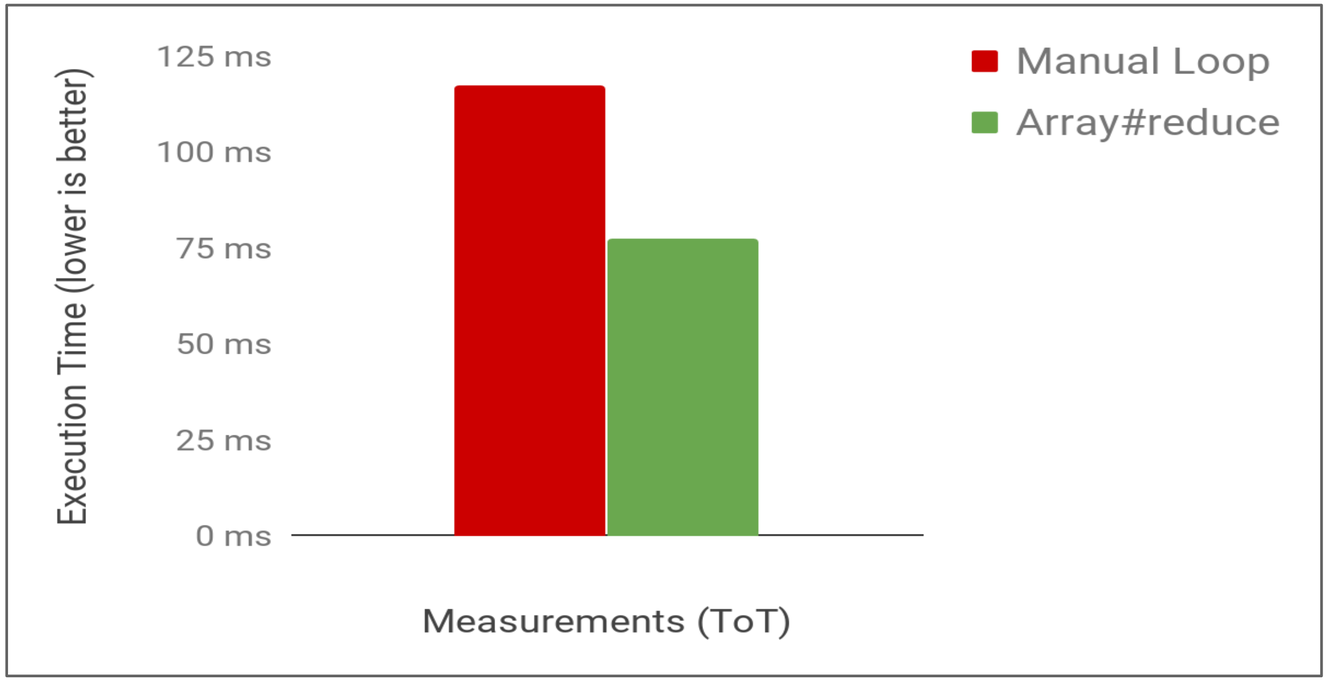
```
function Array_reduce_inline(A) {   PACKED_SMI_ELEMENTS
  if (A not PACKED_SMI_ELEMENTS) Deopt;
  let b = 0;
  for (let i = 0; i < A.length; ++i) {
    if (i in A) { b = b + A[i]; }
    if (A not PACKED_SMI_ELEMENTS) Deopt;
  }
  return b;
}
```

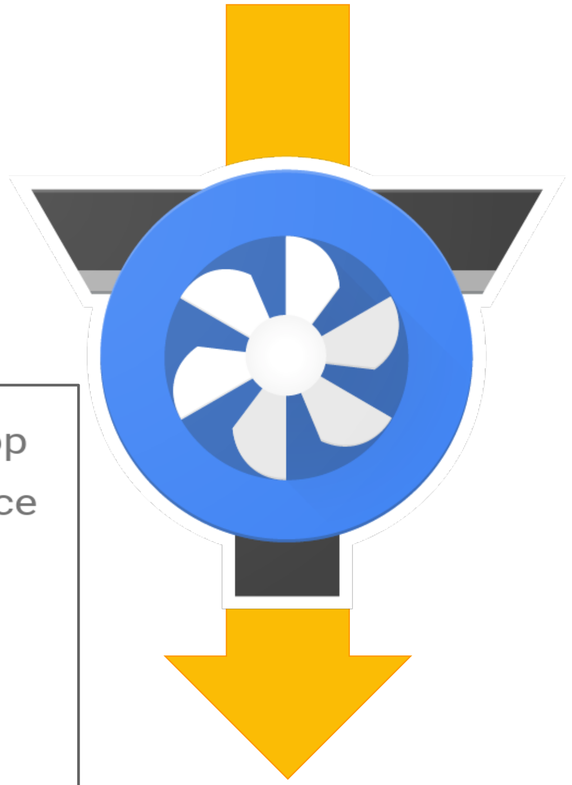
```
function Array_reduce_inline(A) {
  if (A not PACKED_SMI_ELEMENTS) Deopt;
  let b = 0;
  for (let i = 0; i < A.length; ++i) {
    if (i in A) { b = b + A[i]; }
    if (A not PACKED_SMI_ELEMENTS) Deopt;
  }
  return b;
}
```

PACKED_SMI_ELEMENTS

```
function Array_reduce_inline(A) {
  if (A not PACKED_SMI_ELEMENTS) Deopt;
  let b = 0;
  for (let i = 0; i < A.length; ++i) {
    b = b + A[i];
  }
  return b;
}
```

PACKED_SMI_ELEMENTS





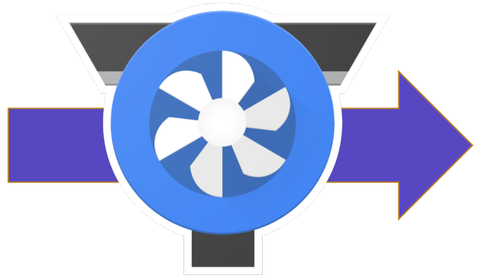
Inlining

```
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;
}
```

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

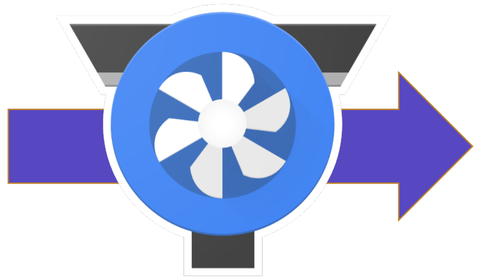
```
function add(x, y) {
  return x + y;
}

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
```

```
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
```

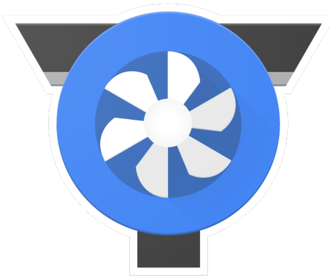
```
leaq rcx,[rip+0x0]
movq rcx,[rcx-0x37]
testb [rcx+0xf],0x1
jnz CompileLazyDeoptimizedCode
push rbp
movq rbp,rsp
push rsi
push rdi
cmpq rsp,[r13+0xdb0]
jna StackCheck
movq rax,[rbp+0x18]
test al,0x1
jnz Deoptimize
movq rbx,[rbp+0x10]
testb rbx,0x1
jnz Deoptimize
movq rdx,rbx
shrq rdx, 32
movq rcx,rax
shrq rcx, 32
addl rdx,rcx
jo Deoptimize
shlq rdx, 32
movq rax,rdx
movq rsp,rbp
pop rbp
ret 0x18
```

+

```
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,0x300000000
movq rsp,rbp
pop rbp
ret 0x8
```



escape analysis

```
function foo(a) {
  return abs({x:a, y:a});
}
```

```
return Math.sqrt(a*a + a*a);
```

```
function abs(v) {
  return Math.sqrt(v.x*v.x + v.y*v.y);
}
```

How ???

Step 1: 内联

```
function foo(a) {
  let v = {x:a, y:a};
  return abs(v);
}

function abs(v) {
  return Math.sqrt(v.x*v.x + v.y*v.y);
}
```

```
function foo(a) {
  let v = {x:a, y:a};
  return Math.sqrt(v.x*v.x + v.y*v.y);
}
```

Step 2: 替换属性访问操作

```
function diagonal(a) {
  let v = {x:a, y:a};
  return Math.sqrt(v.x*v.x + v.y*v.y);
}
```

```
function diagonal(a) {
  let v = {x:a, y:a};
  return Math.sqrt(a*a + a*a);
}
```

Step 3: 删除未使用的变量

```
function diagonal(a) {
  let v = {x:a, y:a};
  return Math.sqrt(v.x*v.x + v.y*v.y);
}
```

```
function diagonal(a) {

  return Math.sqrt(a*a + a*a);
}
```

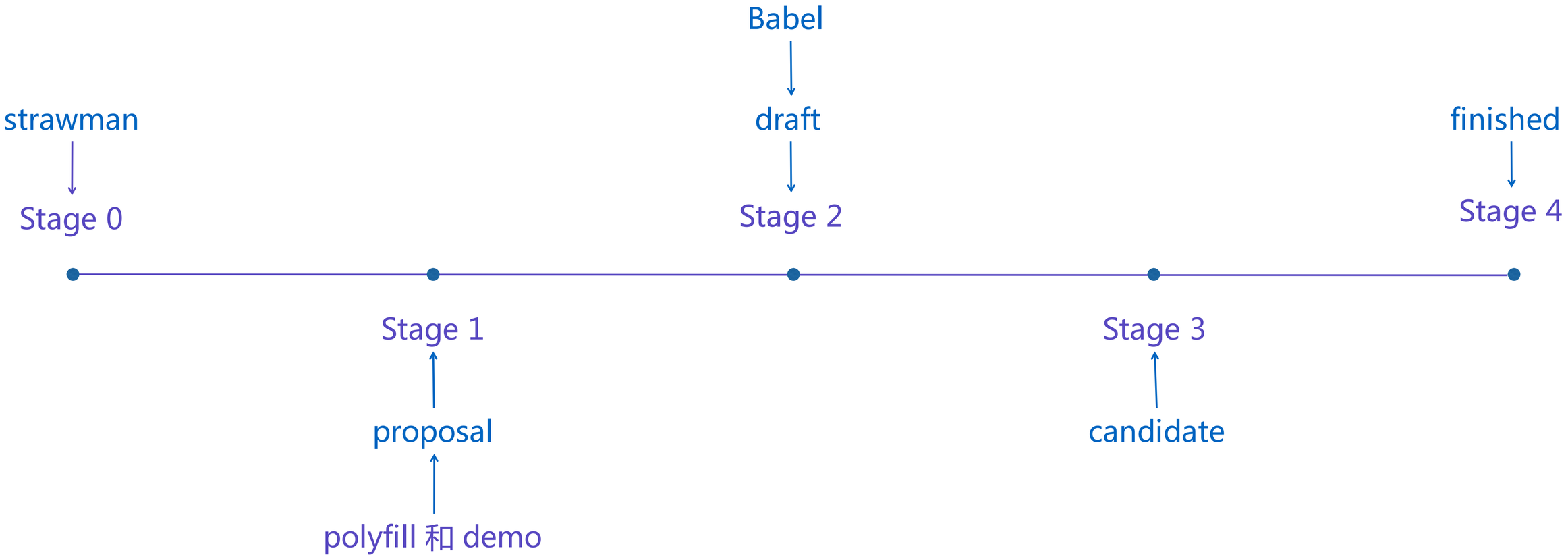
TC39 的最新提案以及应用场景

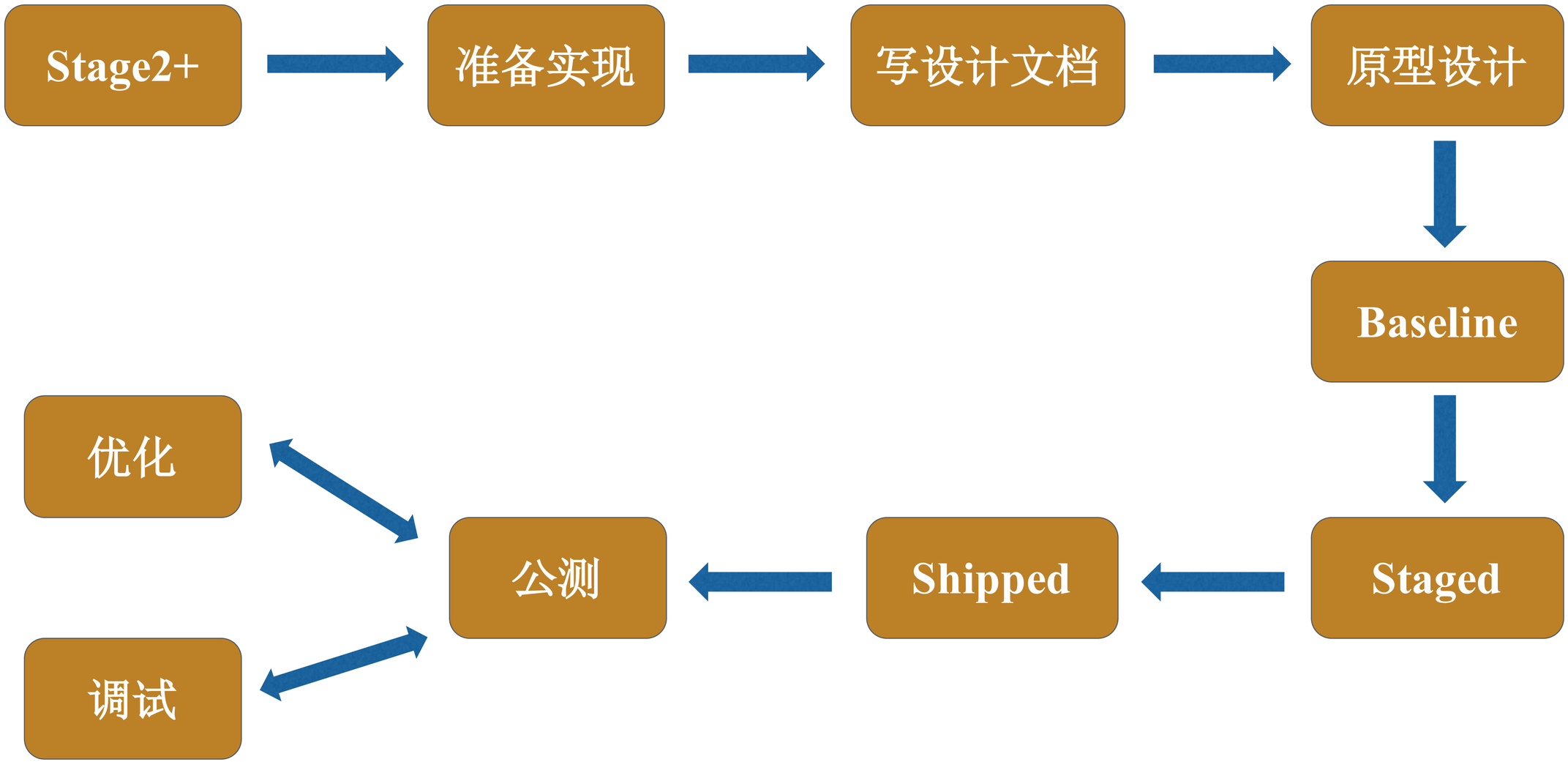
- TC39 process
- BigInt
- Pipeline operator
- Pattern Matching

TC39 process

- 共包含 5 个阶段
- 使用 HTML 的超集进行格式化
- 基于 GitHub pull requests

Stage


Babel
strawman
draft
finished
Stage 0
Stage 2
Stage 4
Stage 1
Stage 3
proposal
candidate
polyfill 和 demo

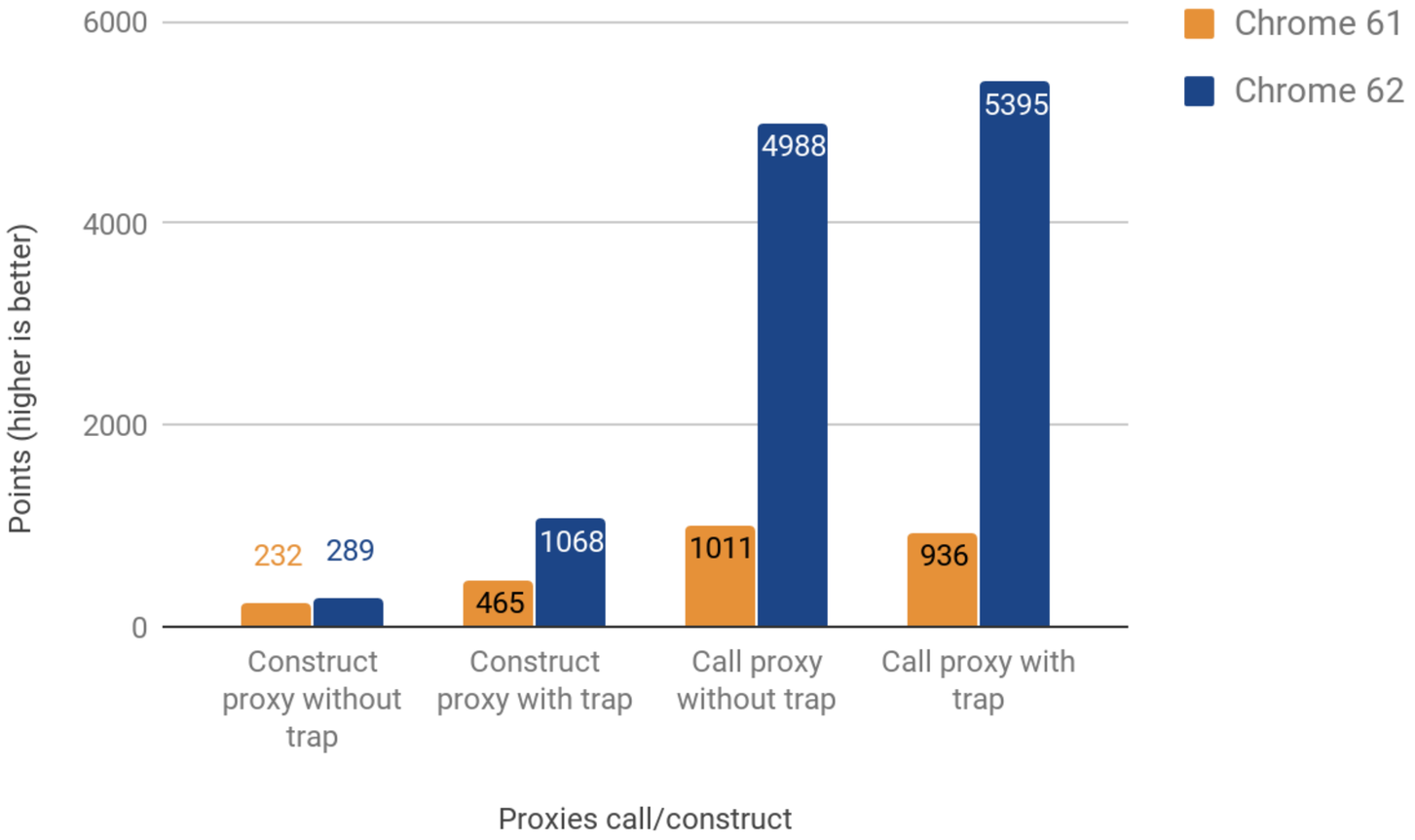

Stage2+
准备实现
写设计文档
原型设计
Baseline
Staged
Shipped
公测
优化
调试


栗子

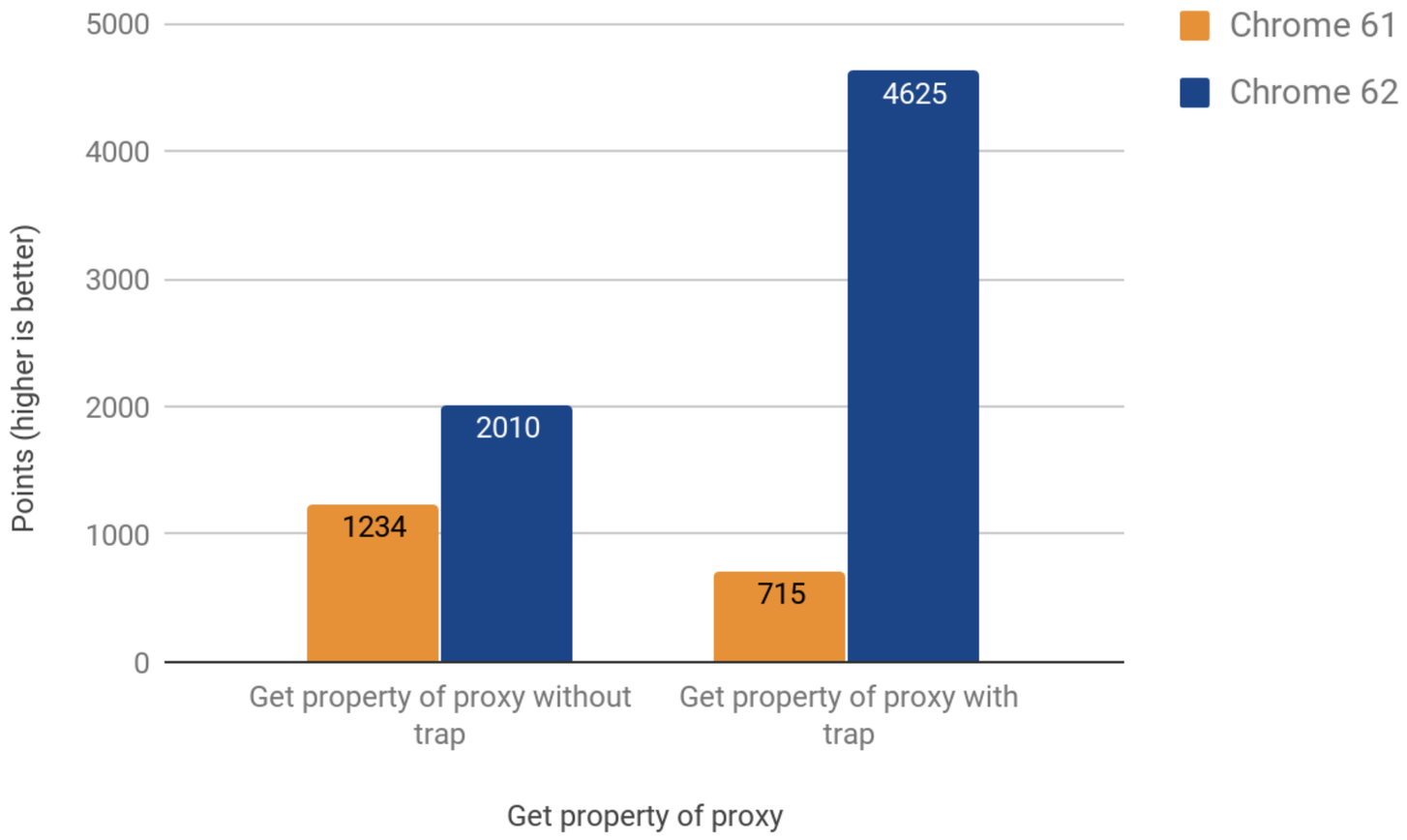
Proxy

- Chrome 49 开始支持 Proxy
- Chrome 62 改进了 Proxy 的性能

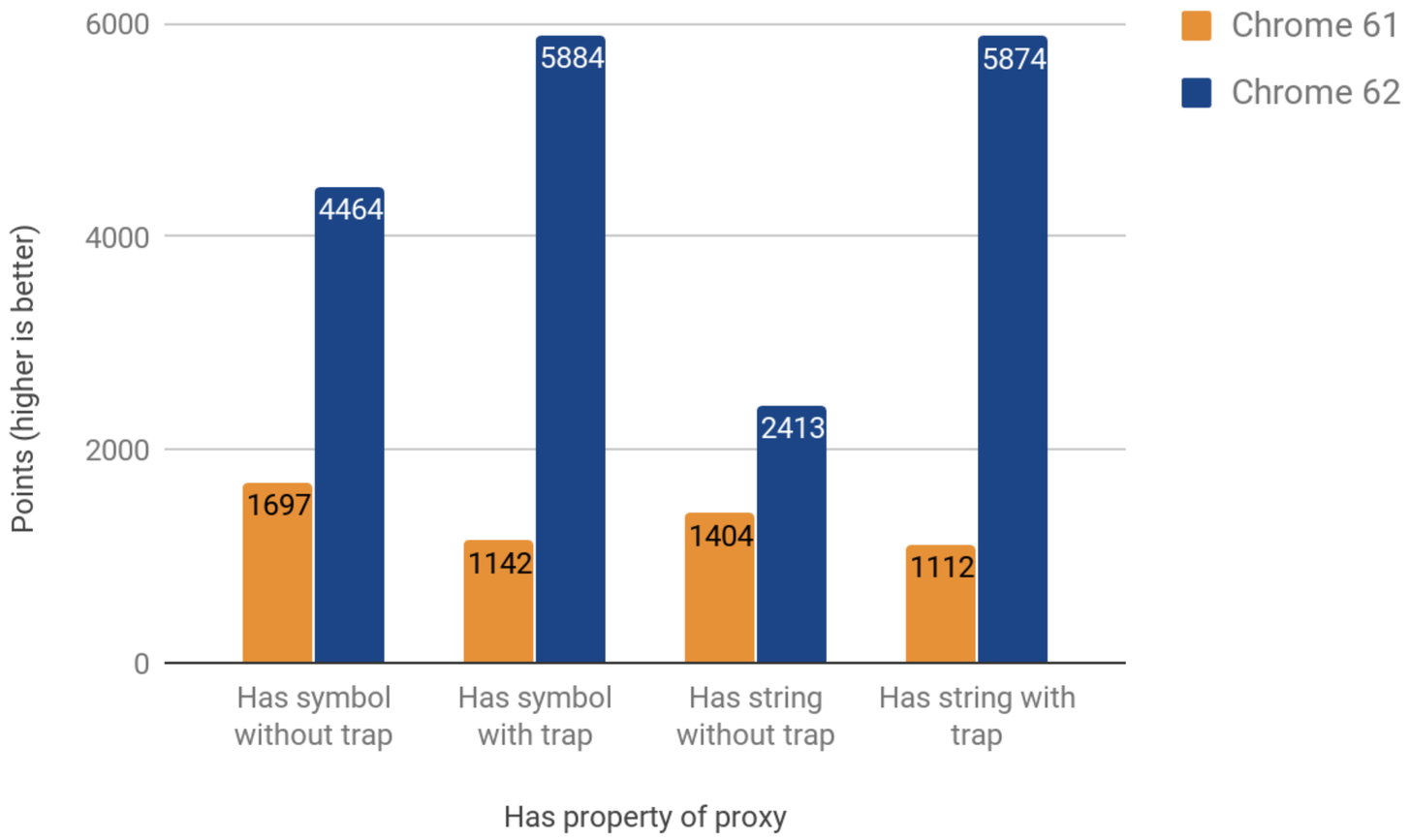
Proxy 构造函数


6000
4000
2000
0
Points (higher is better)
Chrome 61
Chrome 62
232
289
465
1068
1011
4988
936
5395
Construct proxy without trap
Construct proxy with trap
Call proxy without trap
Call proxy with trap
Proxies call/construct

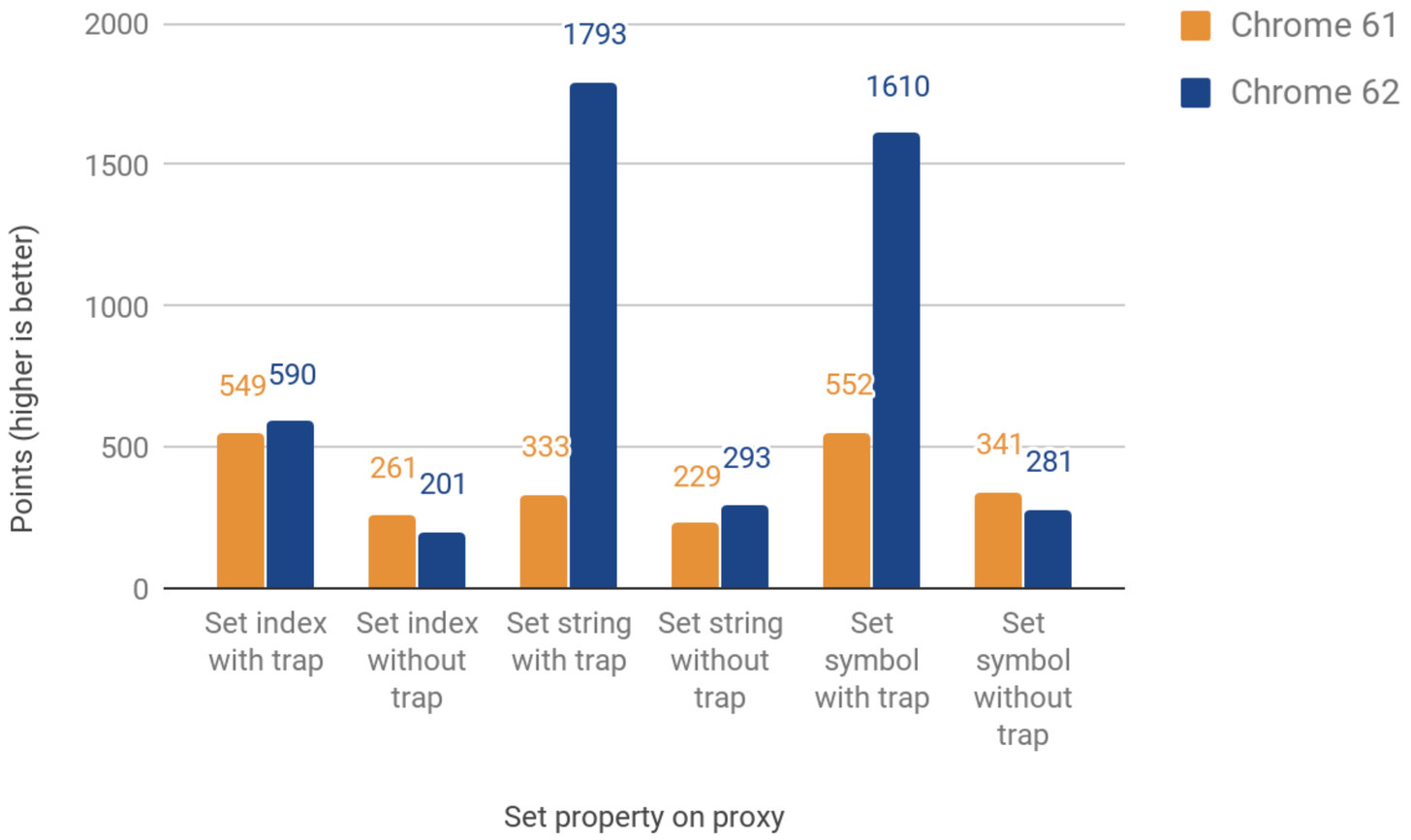
Proxy 获取属性值


5000
4000
3000
2000
1000
0
Points (higher is better)
Chrome 61
Chrome 62
1234
2010
715
4625
Get property of proxy without trap
Get property of proxy with trap
Get property of proxy

Proxy 判断属性值


Chrome 61
Chrome 62
Points (higher is better)
6000
4000
2000
0
1697
4464
1142
5884
1404
2413
1112
5874
Has symbol without trap
Has symbol with trap
Has string without trap
Has string with trap
Has property of proxy

Proxy 设置属性值

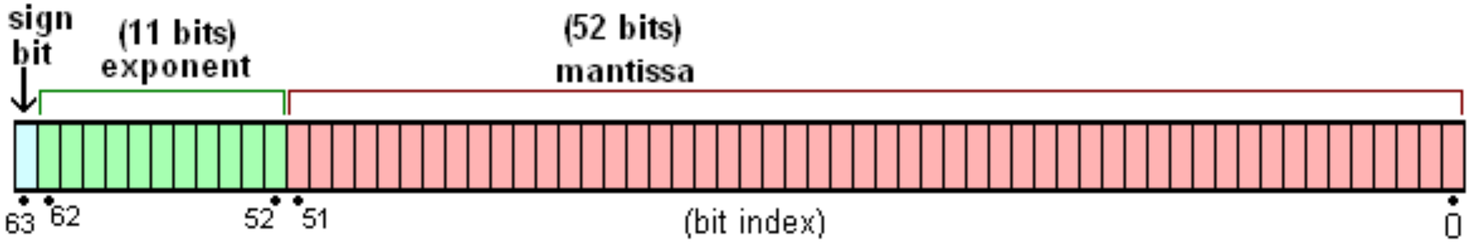

Chrome 61
Chrome 62
Points (higher is better)
2000
1500
1000
500
0
549
590
261
201
333
1793
229
293
552
1610
341
281
Set index with trap
Set index without trap
Set string with trap
Set string without trap
Set symbol with trap
Set symbol without trap
Set property on proxy

Proxy 性能提升

24% ~ 546%

BigInt

$$(-1)^{\text{sign}}(1.b_{51}b_{50}\ldots b_0)_2 \times 2^{e-1023}$$



```
const max = Number.MAX_SAFE_INTEGER;
// → 9,007,199,254,740,991 (= 2**53-1)
```

```
const max = Number.MAX_SAFE_INTEGER;
// → 9,007,199,254,740,991 (= 2**53-1)
max + 1;
// → 9,007,199,254,740,992 (= 2**53)
```

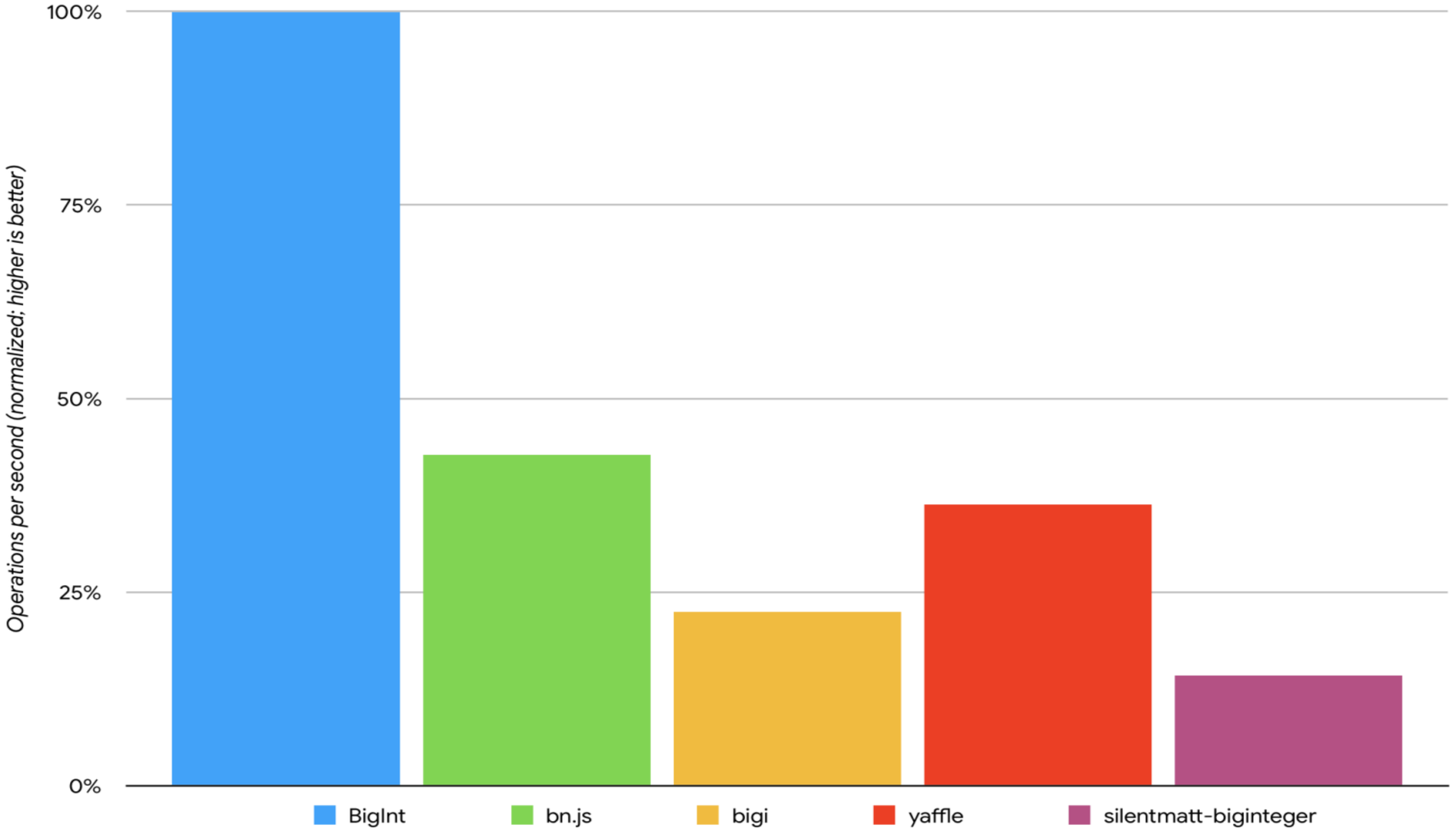
```
const max = Number.MAX_SAFE_INTEGER;
// → 9,007,199,254,740,991 (= 2**53-1)
max + 1;
// → 9,007,199,254,740,992 (= 2**53)
max + 2;
// → 9,007,199,254,740,992 (= 2**53)
```

```
const max = Number.MAX_SAFE_INTEGER;
// → 9,007,199,254,740,991 (= 2**53-1)
max + 1;
// → 9,007,199,254,740,992 (= 2**53)
max + 2;
// → 9,007,199,254,740,992 (= 2**53)
max + 3;
// → 9,007,199,254,740,994 (= 2**53+2)
```

```
const max = Number.MAX_SAFE_INTEGER;
// → 9,007,199,254,740,991 (= 2**53-1)
max + 1;
// → 9,007,199,254,740,992 (= 2**53)
max + 2;
// → 9,007,199,254,740,992 (= 2**53)
max + 3;
// → 9,007,199,254,740,994 (= 2**53+2)
BigInt(max) + 2n;
// → 9,007,199,254,740,993n (= 2**53+1)
```

```
1234567890123456789 * 123;
// → 151851850485185200000

1234567890123456789n * 123n;
// → 151851850485185185047n
```

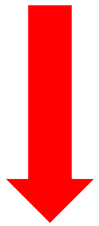
```
typeof 123;
// → 'number'

typeof 123n;
// → 'bigint'
```


100%
75%
50%
25%
0%
Operations per second (normalized; higher is better)
BigInt
bn.js
bigi
yaffle
silentmatt-biginteger

Pipeline operator

Stage-1

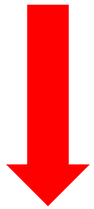
```
log(capitalize(double("hello")))
```



```
log(
  capitalize(
    double (
      "hello"
    )
  )
)
```

```
"hello"
  |> double
  |> capitalize
  |> log
```

```
new User.Message(
  capitalize(
    doubledSay(value, ', ')
  ) + '!'
)
```



```
value
  |> (x => doubleSay(x, ', '))
  |> capitalize
  |> (x => x + '!')
  |> (x => new User.Message(x))
```

Optional Chaining

Stage 1

```
// 获取用户 name 属性
// 可能抛出异常
const name = response.body.user.name;
```

```
// 获取用户 name 属性
// 可能抛出异常
const name = response.body.user.name;

// 对变量和属性进行检查
const name = response
    && response.body
```

```
// 获取用户 name 属性
// 可能抛出异常
const name = response.body.user.name;

// 对变量和属性进行检查
const name = response
    && response.body
    && response.body.user
```

```
// 获取用户 name 属性
// 可能抛出异常
const name = response.body.user.name;

// 对变量和属性进行检查
const name = response
    && response.body
    && response.body.user
    && response.body.user.name
```

```
// 获取用户 name 属性
// 可能抛出异常
const name = response.body.user.name;

// 对变量和属性进行检查
const name = response
    && response.body
    && response.body.user
    && response.body.user.name || 'justjavac';
```

With Optional Chaining

```
const name = response?.body?.user?.name || 'justjavac';
```

Pattern Matching

Stage 1

```
case (input) {
    when 1 -> ... // matches if `input` is 1
    when 'foo' -> ... // matches if `input` is 'foo'
    when false -> ... // matches if `input` is `false`
    when null -> ... // matches if `input` is `null`
    when {x: 1} -> ... // matches if `input` can do ToObject and `input.x` is 1
    when [1,2] -> ... // matches if `input` can do GetIterator, has exactly 2 items,
                      // and the items are 1, then 2.
    when x -> ... // always matches
    when /^foo/ -> ... // SyntaxError
    when x if (x.match(/^foo/)) -> ... // ok!
}
```

Redux reducers

```
function todoApp (state = initialState, action) {
  case (action) {
    when {type: 'set-visibility-filter', filter: visFilter} ->
      return {...state, visFilter}
    when {type: 'add-todo', text} ->
      return {...state, todos: [...state.todos, {text}]}
    when {type: 'toggle-todo', index} -> {
      return {
        ...state,
        todos: state.todos.map((todo, idx) => idx === index ? {...todo, done: !todo.done} : todo)
      }
    }
    when {} -> {} // ignore unknown actions
  }
}
```

编写高性能 JavaScript 代码

- 不要修改原型链
- 使用 Object.keys()
- 不要在遍历时修改元素

不要修改原型链

```
> const o = {a:1,b:2};
< undefined
```

不要修改原型链

```
> const o = {a:1,b:2};
< undefined
> console.time('fast');
  for (let i = 0; i < 1e7; ++i) {
    for (const x in o) {}
  };
  console.timeEnd('fast');
  fast: 59.775146484375ms
< undefined
```

不要修改原型链

```
> const o = {a:1,b:2};
< undefined
> console.time('fast');
  for (let i = 0; i < 1e7; ++i) {
    for (const x in o) {}
  };
  console.timeEnd('fast');
  fast: 59.775146484375ms
< undefined
> Object.prototype.c = 0;
< 0
```

不要修改原型链

```
> const o = {a:1,b:2};
< undefined
> console.time('fast');
  for (let i = 0; i < 1e7; ++i) {
    for (const x in o) {}
  };
  console.timeEnd('fast');
  fast: 59.775146484375ms
< undefined
> Object.prototype.c = 0;
< 0
> console.time('slow');
  for (let i = 0; i < 1e7; ++i) {
    for (const x in o) {}
  };
  console.timeEnd('slow');
  slow: 7266.184814453125ms
< undefined
```

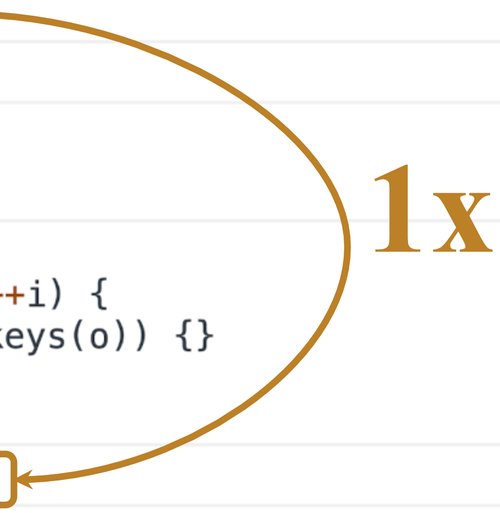
100x

Object.keys()

```
> const o = {a:1,b:2};
< undefined
> console.time('fast');
  for (let i = 0; i < 1e7; ++i) {
    for (const x of Object.keys(o)) {}
  };
  console.timeEnd('fast');
  fast: 246.964111328125ms
< undefined
```

Object.keys()

```
> const o = {a:1,b:2};
< undefined
> console.time('fast');
  for (let i = 0; i < 1e7; ++i) {
    for (const x of Object.keys(o)) {}
  };
  console.timeEnd('fast');
  fast: 246.964111328125ms
< undefined
> Object.prototype.c = 0;
< 0
> console.time('slow');
  for (let i = 0; i < 1e7; ++i) {
    for (const x of Object.keys(o)) {}
  };
  console.timeEnd('slow');
  slow: 241.490966796875ms
< undefined
```

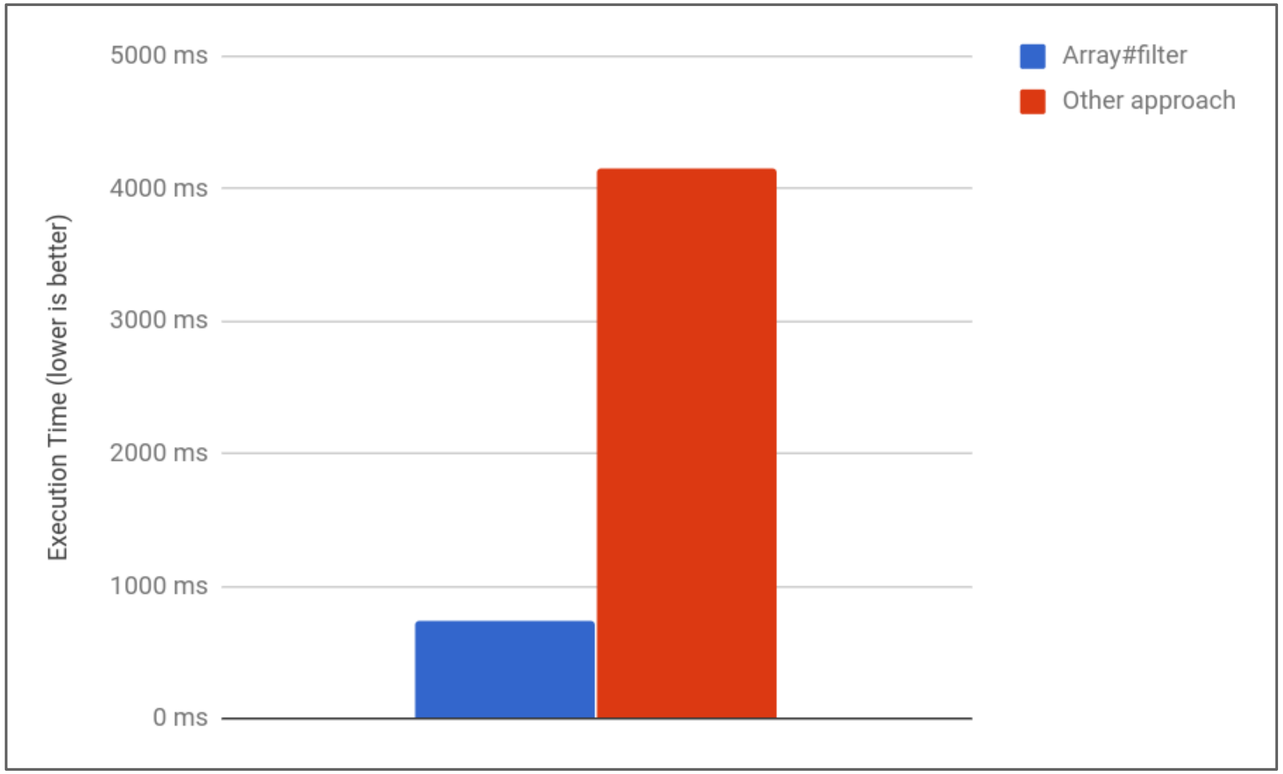


1x

不要在遍历时修改元素

```
A.filter((x, i) => x > 50);  // OK
```

```
A.forEach((x, i) => {

    if (x > 50)  A[i] = undefined;

});
```



JavaScript函数式编程

函数式编程都是垃圾

极客邦企业培训与咨询

帮助企业与技术人成长

精品课程

Excellent Course

- ☑《互联网大规模分布式架构设计与实践》
- ☑《基于大数据的企业运营与精准营销》
- ☑《大数据和人工智能在金融领域的应用》
- ☑《区块链应用与开发技术高级培训》
- ☑《通往卓越管理的阶梯》

扫码关注官方微信服务号
了解更多课程详细信息

Geekbang
极客邦科技

THANKS