ПРОИЗВОДИТЕЛЬНОСТЬ JAVASCRIPT ЧЕРЕЗ ПОДЗОРНУЮ ТРУБУ

ВЯЧЕСЛАВ ЕГОРОВ

Excelsior JET

V8

Dart VM

LuaJIT

Excelsior JET

V8

Dart VM

LuaJIT

ВСЁ СЛОЖНО

```
function find(val){
  function index(value) {
    return [1, 2, 3].indexOf(value);
  }

  return index(val);
}
```

```
function find(val){
  function index(value) {
    return [1, 2, 3].indexOf(value);
  }

  return index(val);
}
// => 5464 op/ms
```

```
var arr = [1, 2, 3];
function find(val){
  function index(value) {
    return arr.indexOf(value);
  }

  return index(val);
}
```

```
var arr = [1, 2, 3];
function find(val){
  function index(value) {
    return arr.indexOf(value);
  }

  return index(val);
}
// => 14084 op/ms
```

«аллокация массива — дорогая операция»

```
var makeArr = () => [1, 2, 3];
function find(val){
  function index(value) {
    return makeArr().indexOf(value);
  }

  return index(val);
}
```

```
var makeArr = () => [1, 2, 3];
function find(val){
  function index(value) {
    return makeArr().indexOf(value);
  }

  return index(val);
}
// => 12048 op/ms
```

```
var makeArr = () => [1, 2, 3];
function find(val){
  function index(value) {
    return makeArr().indexOf(value);
  }

  return index(val);
}
// => 12048 op/ms
```

```
$ perf record d8 --perf-basic-prof test.js
$ perf report
```

```
14.80% v8::internal::Factory::NewFunction
 9.98% v8::internal::Factory::NewFunctionFromShared
 8.74% *InnerArrayIndexOf native array.js:1019
 7.24% v8::internal::Factory::NewFunctionFromShared
 5.79% v8::internal::Runtime_NewClosure
 4.84% Builtin:ArgumentsAdaptorTrampoline
 4.44% v8::internal::Heap::AllocateRaw
 4.30% v8::internal::Factory::New
 3.74% v8::internal::SharedFunctionInfo::SearchOpti
 3.20% ~index test.js:2
```

```
14.80% v8::internal::Factory::NewFunction
 9.98% v8::internal::Factory::NewFunctionFromShared
 8.74% *InnerArrayIndexOf native array.js:1019
 7.24% v8::internal::Factory::NewFunctionFromShared
 5.79% v8::internal::Runtime_NewClosure
 4.84% Builtin:ArgumentsAdaptorTrampoline
 4.44% v8::internal::Heap::AllocateRaw
 4.30% v8::internal::Factory::New<v8::internal::JSF
 3.74% v8::internal::SharedFunctionInfo::SearchOpti
 3.20% ~index test.js:2
```

```
14.49% *InnerArrayIndexOf native array.js:1019
10.47% LoadIC:A load IC from the snapshot
 8.45% ~index b2.js:12
 8.05% Stub:FastNewClosureStub
 5.77% Builtin:ArgumentsAdaptorTrampoline
 5.23% *find2 b2.js:11
```

```
14.49% *InnerArrayIndexOf native array.js:1019
10.47% LoadIC:A load IC from the snapshot
 8.45% ~index b2.js:12
 8.05% Stub:FastNewClosureStub
 5.77% Builtin:ArgumentsAdaptorTrampoline
 5.23% *find2 b2.js:11
```

FastNewClosureStub
не умела создавать функции
содержащие литералы

`FastNewClosureStub` не умела создавать функции содержащие литералы

[сейчас уже умеет, пример из Февраля]

Strange JavaScript performance

6

2

When I was implementing ChaCha20 in JavaScript, I stumbled upon some strange behavior.

My first version was build like this (let's call it "Encapsulated Version"):

```
function quarterRound(x, a, b, c, d) {
    x[a] += x[b]; x[d] = ((x[d] ^ x[a]) << 16) | ((x[d] ^ x[a]) >>> 16);
    x[c] += x[d]; x[b] = ((x[b] ^ x[c]) << 12) | ((x[b] ^ x[c]) >>> 20);
    x[a] += x[b]; x[d] = ((x[d] ^ x[a]) <<  8) | ((x[d] ^ x[a]) >>> 24);
    x[c] += x[d]; x[b] = ((x[b] ^ x[c]) <<  7) | ((x[b] ^ x[c]) >>> 25);
}

function getBlock(buffer) {
    var x = new Uint32Array(16);

    for (var i = 16; i--;) x[i] = input[i];
    for (var i = 20; i > 0; i -= 2) {
        quarterRound(x, 0, 4, 8,12);
        quarterRound(x, 1, 5, 9,13);
        quarterRound(x, 2, 6,10,14);
        quarterRound(x, 3, 7,11,15);
        quarterRound(x, 0, 5,10,15);
        quarterRound(x, 1, 6,11,12);
        quarterRound(x, 2, 7, 8,13);
        quarterRound(x, 3, 4, 9,14);
    }
    for (i = 16; i--;) x[i] += input[i];
    for (i = 16; i--;) U32TO8_LE(buffer, 4 * i, x[i]);
    input[12]++;
    return buffer;
}
```

To reduce unnecessary function calls (with parameter overhead etc.) I removed the `quarterRound` -function and put it's contents inline (it's correct; I verified it against some test vectors):

```
function getBlock(buffer) {
    var x = new Uint32Array(16);
```

ChaCha20

```
function getBlock(buffer) {
  var x = new Uint32Array(16);

  for (var i = 16; i--;) x[i] = input[i];
  for (var i = 20; i > 0; i -= 2) {
    quarterRound(x, 0, 4, 8,12);
    quarterRound(x, 1, 5, 9,13);
    quarterRound(x, 2, 6,10,14);
    quarterRound(x, 3, 7,11,15);
    quarterRound(x, 0, 5,10,15);
    quarterRound(x, 1, 6,11,12);
    quarterRound(x, 2, 7, 8,13);
    quarterRound(x, 3, 4, 9,14);
  }
  for (i = 16; i--;) x[i] += input[i];
  for (i = 16; i--;) U32TO8_LE(buffer, 4 * i, x[i]);
  input[12]++;
  return buffer;
}
```

19MB/s

```
function getBlock(buffer) {
  var x = new Uint32Array(16);

  for (var i = 16; i--;) x[i] = input[i];
  for (var i = 20; i > 0; i -= 2) {
    quarterRound(x, 0, 4, 8,12);
    quarterRound(x, 1, 5, 9,13);
    quarterRound(x, 2, 6,10,14);
    quarterRound(x, 3, 7,11,15);
    quarterRound(x, 0, 5,10,15);
    quarterRound(x, 1, 6,11,12);
    quarterRound(x, 2, 7, 8,13);
    quarterRound(x, 3, 4, 9,14);
  }
  for (i = 16; i--;) x[i] += input[i];
  for (i = 16; i--;) U32TO8_LE(buffer, 4 * i, x[i]);
  input[12]++;
  return buffer;
}
```

```
function getBlock(buffer) {
  var x = new Uint32Array(16);

  for (var i = 16; i--;) x[i] = input[i];
  for (var i = 20; i > 0; i -= 2) {
    // quarterRound(x, 0, 4, 8,12);
    x[ 0] += x[ 4]; x[12] = ((x[12] ^ x[ 0]) << 16) | ((x[12] ^ x[ 0])
    x[ 8] += x[12]; x[ 4] = ((x[ 4] ^ x[ 8]) << 12) | ((x[ 4] ^ x[ 8])
    x[ 0] += x[ 4]; x[12] = ((x[12] ^ x[ 0]) <<  8) | ((x[12] ^ x[ 0])
    x[ 8] += x[12]; x[ 4] = ((x[ 4] ^ x[ 8]) <<  7) | ((x[ 4] ^ x[ 8])
    // ... и так далее ...
  }
  for (i = 16; i--;) x[i] += input[i];
  for (i = 16; i--;) U32TO8_LE(buffer, 4 * i, x[i]);
  input[12]++;
  return buffer;
}
```

2MB/s

«ИНЛАЙН ЖЕ ДОЛЖЕН УЛУЧШАТЬ ПРОИЗВОДИЛЬНОСТЬ!»

```
function U32TO8_LE(x, i, u) {
    x[i]   = u; u >>>= 8;
    x[i+1] = u; u >>>= 8;
    x[i+2] = u; u >>>= 8;
    x[i+3] = u;
}
```

```
function U32TO8_LE(x, i, u) {
    // ____     ___  ____   __________              _        ____   ___       ___
    // `MM'     `M' 6MMMMb\ `MMMMMMMMM             dM.      6MMMMb\ `MMb     dMM'
    //  MM       M 6M'    `  MM      \            ,MMb     6M'    `  MMM.   ,PMM
    //  MM       M MM        MM                   d'YM.    MM        M`Mb   d'MM
    //  MM       M YM.       MM    ,             ,P `Mb    YM.       M YM. ,P MM
    //  MM       M  YMMMMb   MMMMMMM             d'  YM.    YMMMMb   M `Mb d' MM
    //  MM       M      `Mb  MM    `            ,P   `Mb        `Mb  M  YM.P  MM
    //  MM       M       MM  MM                 d'    YM.        MM  M  `Mb'  MM
    //  YM       M       MM  MM                ,MMMMMMMMb        MM  M   YP   MM
    //   8b     d8 L    ,M9  MM      /         d'      YM. L    ,M9  M   `'   MM
    //    YMMMMM9  MYMMMM9  _MMMMMMMMM       _dM_     _dMM_MYMMMM9  _M_      _MM_
    //
    x[i]   = u; u >>>= 8;
    x[i+1] = u; u >>>= 8;
    x[i+2] = u; u >>>= 8;
    x[i+3] = u;
}
```

19MB/s

```
function U32TO8_LE(x, i, u) {
    // ____     ___  ____   __________              _        ____   ___       ___
    // `MM'     `M' 6MMMMb\ `MMMMMMMMM             dM.      6MMMMb\ `MMb     dMM'
    //  MM       M 6M'    `  MM      \            ,MMb     6M'    `  MMM.   ,PMM
    //  MM       M MM        MM                   d'YM.    MM        M`Mb   d'MM
    //  MM       M YM.       MM    ,             ,P `Mb    YM.       M YM. ,P MM
    //  MM       M  YMMMMb   MMMMMMM             d'  YM.    YMMMMb   M `Mb d' MM
    //  MM       M      `Mb  MM    `            ,P   `Mb        `Mb  M  YM.P  MM
    //  MM       M       MM  MM                 d'    YM.        MM  M  `Mb'  MM
    //  YM       M       MM  MM                ,MMMMMMMMb        MM  M   YP   MM
    //   8b     d8 L    ,M9  MM      /         d'      YM. L    ,M9  M   `'   MM
    //    YMMMMM9  MYMMMM9  _MMMMMMMMM       _dM_     _dMM_MYMMMM9  _M_      _MM_
    //
    // ^^^^^^ НЕ УДАЛЯТЬ! РАЗГОНЯЕТ КОД В 10 РАЗ. ^^^^^^
    x[i]   = u; u >>>= 8;
    x[i+1] = u; u >>>= 8;
    x[i+2] = u; u >>>= 8;
    x[i+3] = u;
}
```

говорим производительность
подразумеваем jsperf

говорим производительность
подразумеваем микробенчмарк

[jsperf.com сейчас мертвый]

Count the occurrences of a single character in a string.

Preparation code

```
<script>
  Benchmark.prototype.setup = function() {
    var matcher = /e/g;
    var testString = "We the People of the United States, in Order to form a more perfect Union, establish
Justice, insure domestic Tranquility, provide for the common defence,promote the general Welfare, and secure
the Blessings of Liberty to ourselves and our Posterity, do ordain and establish this Constitution for the
United States of America.";
  };
</script>
```

Test runner

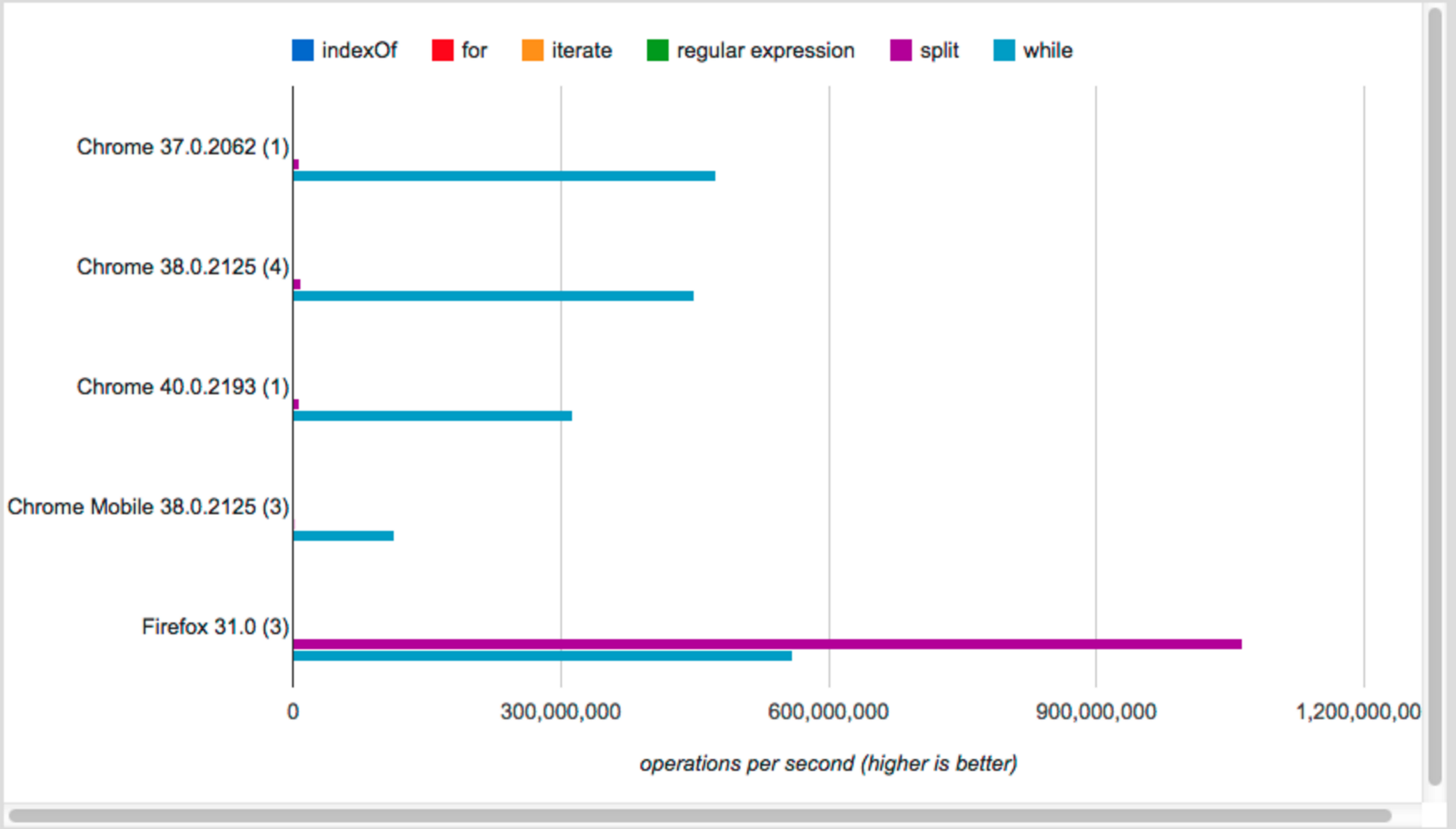
Ready to run. Run tests

Testing in Chrome 41.0.2224.3 on OS X 10.9.5		
Test		**Ops/sec**
regular expression	`testString.match(matcher).length;`	ready
split	`testString.split("e").length - 1;`	ready

iterate	`var count = 0;`
	`for (var i = 0; i < testString.length; i++) {`
	`  if (testString.charAt(i) === "e") {`
	`  count += 1;`
	`  }`
	`}`
		ready

while	`var count = 0;`
	`while (testString.length) {`
	`  if (testString.charAt(0) === "e") {`
	`  count += 1;`
	`  }`
	`  testString = testString.slice(1);`
	`}`
		ready

Browserscope



Browserscope thinks you are using **Chrome 41.0.2224**

```
// split
str.split("e").length - 1;

// while
var count = 0;
while (str.length) {
   if (str.charAt(0) === "e") {
      count += 1;
   }
   str = str.slice(1);
}
```

```
// split (на FF показывает больше 9000K OP/S!)
str.split("e").length - 1;

// while
var count = 0;
while (str.length) {
   if (str.charAt(0) === "e") {
      count += 1;
   }
   str = str.slice(1);
}
```

СОМНЕВАЙСЯ

ВО ВСЁМ

μбенчмарки для чайников

цена одной операции OP?

```
function benchmark() {
  var start = Date.now();
  /* OP */
  return (Date.now() - start);
}
```

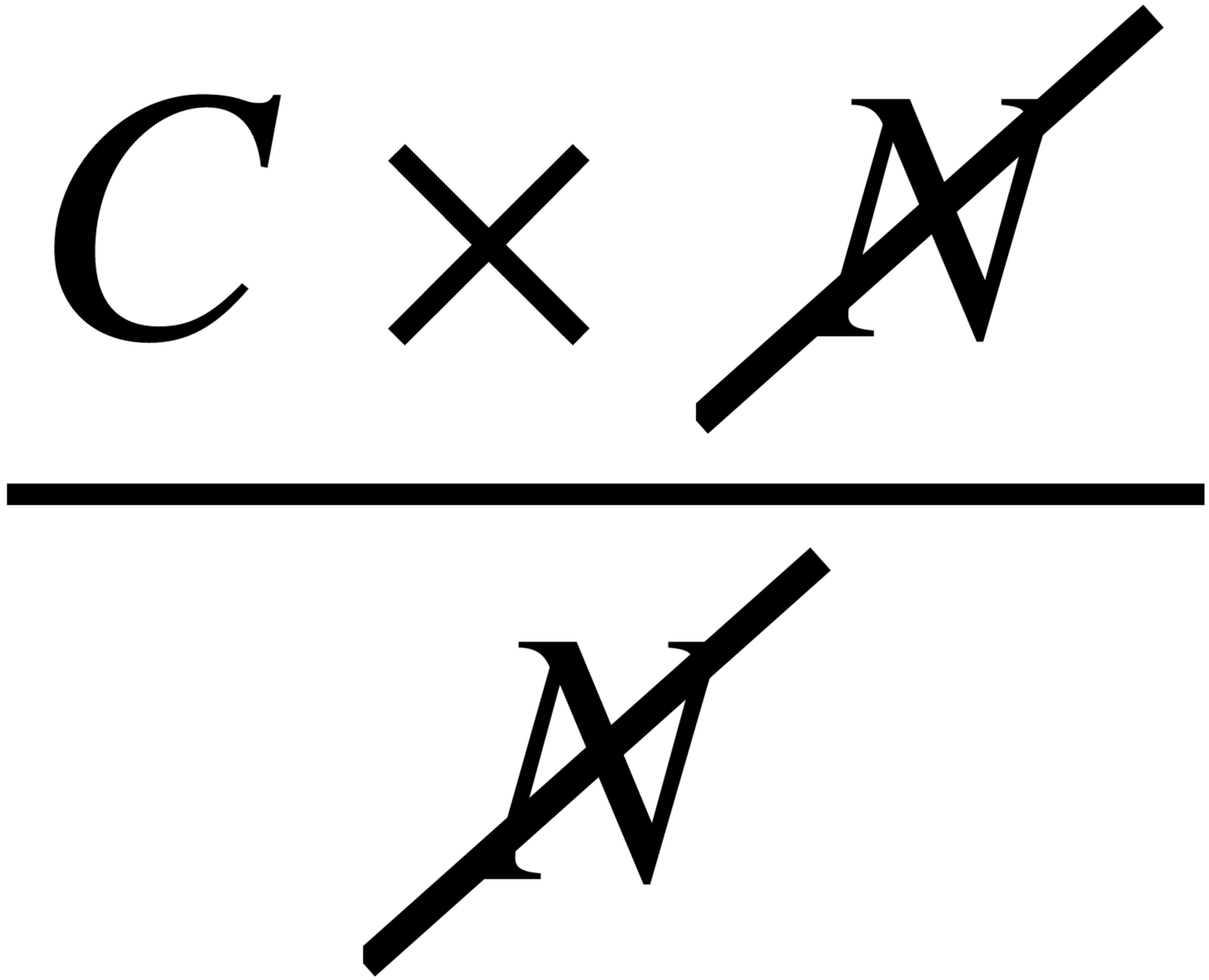
а что если ОР быстрее часов?

```
function benchmark(N) {
  var start = Date.now();
  for (var i = 0; i < N; i++) {
    /* OP */
  }
  return (Date.now() - start) / N;
}
```

C

$$C \times N$$

$$\frac{C \times N}{N}$$



```
while (str.length) {
  if (str.charAt(0) === "e") {
    count += 1;
  }
  str = str.slice(1);
}
```

```
function benchmark() {
  var str = "...";
  var start = Date.now();
  for (var i = 0; i < N; i++) {
    var count = 0;
    while (str.length) {
      if (str.charAt(0) === "e") {
        count += 1;
      }
      str = str.slice(1);
    }
  }
  return (Date.now() - start) / N;
}
```

```
function benchmark() {
  var str = "...";
  var start = Date.now();
  for (var i = 0; i < N; i++) {
    var count = 0;
    while (str.length) {
      if (str.charAt(0) === "e") {
        count += 1;
      }
      str = str.slice(1);
    }
  }  // здесь str == ""
  return (Date.now() - start) / N;
}
```

повторения начинаются со str === ""

$$C \times N$$

$$C \times 1 + 0 \times (N - 1)$$

$$\frac{C \times 1 + 0 \times (N - 1)}{N}$$

$$\frac{C}{N}$$

$$\frac{C}{N} \approx 0$$

неправильный бенчмарк

OP должен стоить одинаково при каждом повторении

Preparation code

```
<script>
  Benchmark.prototype.setup = function() {
    var i = '1561565', j,
    superParseInt = function(a) {
            return ~~parseInt(a, 10);
        };
  };
</script>
```

Test runner

Ready to run.

Run tests

Testing in Chrome 29.0.1547.76 on OS X 10.8.5		
Test		**Ops/sec**
Double Tilde	`j = ~~i;`	ready
ParseInt	`j = parseInt(i, 10);` `j = isNaN(j) ? 0 : j;`	ready
Including true	`j = i === true ? 1 : ~~parseInt(i, 10);`	ready
Slight change to parseInt	`j = ~~parseInt(i, 10);`	ready


Firefox 19.0 (1)
Firefox 21.0 (1)
Firefox 21.0 (1)
Double Tilde: 607,665,833 ops/sec

«~~ быстрее всех»

НЕТУШКИ

JITы умны

JITы оптимизируют

одновременно с исполнением

$$C \times N$$

$$C_u N_u + C_o N_o$$

[неоптимизированая + оптимизированная версия]

$$\sum C_i N_i$$

[множественные оптимизированные версии]

$$\sum C_i N_i + \frac{1}{\sqrt{2\pi}} \int C(\xi) e^{-\frac{\xi^2}{2}} d\xi$$

[математика становится слишком мудрённой]

JITы умны

хочется измерить что-то,
нужно *перехитрить*

Предупреждение: На слайдах оптимизации, выполняемые JITом, изображены как трансформации исходного кода. В реальности JITы оперируют на более сложных внутренних представлениях.

```
function benchmark() {
  var i = '1561565', j;
  var start = Date.now();
  for (var n = 0; n < N; n++) {
    j = ~~i;
  }
  return (Date.now() - start) / N;
}
```

```
function benchmark() {
  var j;
  var start = Date.now();
  for (var n = 0; n < N; n++) {
    j = ~~'1561565';
  }
  return (Date.now() - start) / N;
}
```

```
function benchmark() {
  var j;
  var start = Date.now();
  for (var n = 0; n < N; n++) {
    j = ~-1561566;
  }
  return (Date.now() - start) / N;
}
```

```
function benchmark() {
  var j;
  var start = Date.now();
  for (var n = 0; n < N; n++) {
    j = 1561565;
  }
  return (Date.now() - start) / N;
}
```

протяжка констант

constant propagation

«а я знаю как обхитрить компилятор!»

```
function benchmark() {
  var i = Date.now().toString(), j;
  //  ^ not a constant any more
  var start = Date.now();
  for (var n = 0; n < N; n++) {
    j = ~~i;
  }
  return (Date.now() - start) / N;
}
```

«НЕТУШКИ^2»

```
function benchmark() {
  var i = Date.now().toString(), j;
  var start = Date.now();
  for (var n = 0; n < N; n++) {
    j = ~~i;
    //  ^^^ loop invariant
  }
  return (Date.now() - start) / N;
}
```

```
function benchmark() {
  var i = Date.now().toString(), j;
  var start = Date.now();
  var j0 = ~~i;
  //  ^^^^^^^^ hoisted from the loop
  for (var n = 0; n < N; n++) {
    j = j0;
  }
  return (Date.now() - start) / N;
}
```

вынос инвариантов цикла

loop invariant code motion

```
function benchmark() {
  var i = Date.now().toString(), j;
  var start = Date.now();
  var j0 = ~~i;
  for (var n = 0; n < N; n++) {
    j = j0;
  }
  return (Date.now() - start) / N;
}
```

```
function benchmark() {
  var i = Date.now().toString(), j;
  var start = Date.now();
  var j0 = ~~i;
  for (var n = 0; n < N; n++) {
    j = j0;
  }
  return (Date.now() - start) / N;
}
```

```
function benchmark() {
  var i = Date.now().toString(), j;
  var start = Date.now();
  var j0 = ~~i;
  for (var n = 0; n < N; n++) {
    j = j0;
  }
  return (Date.now() - start) / N;
}
```

```
function benchmark() {
  var i = Date.now().toString(), j;
  var start = Date.now();
  var j0 = ~~i;
  for (var n = 0; n < N; n++) {
    j = j0;
  }
  return (Date.now() - start) / N;
}
```

```
function benchmark() {
  var start = Date.now();
  for (var n = 0; n < N; n++) {
    /* стрекотание сверчков */
  }
  return (Date.now() - start) / N;
}
```

```
function benchmark() {
  var start = Date.now();
  /*
   * стрекотание сверчков
   */
  return (Date.now() - start) / N;
}
```

удаление мертвого кода

dead code elimination

```
// Помните этот пример?
// split (на FF показывает больше 9000K OP/S!)
str.split("e").length - 1;
```

```
// Помните этот пример?
// split (на FF показывает больше 9000K OP/S!)
str.split("e").length - 1;
// FF просто видит, что это мертвый код
```

КОМПИЛЯТОР КУШАЕТ μБЕНЧМАРКИ НА ЗАВТРАК

КАК БЫ СДЕЛАТЬ ТАКОЙ,
ЧТОБЫ ОН ПОДАВИЛСЯ?

ЛУЧШЕ, КОНЕЧНО,
НИКАКИХ НЕ ПИСАТЬ,
НО...

2. никаких констант
3. никаких инвариантов
4. никакого мертвого кода

1. **проверка результатов**
2. никаких констант
3. никаких инвариантов
4. никакого мертвого кода

```
function benchmark() {


  var j;
  var start = Date.now();
  for (var n = 0; n < N; n++) {

    j = ~~i;
  }

  return (Date.now() - start) / N;
}
```

```
function benchmark() {
  var i = Date.now().toString(),
      i1 = Date.now().toString(), t;
  var j;
  var start = Date.now();
  for (var n = 0; n < N; n++) {

    j = ~~i;
  }

  return (Date.now() - start) / N;
}
```

```
function benchmark() {
  var i = Date.now().toString(),
      i1 = Date.now().toString(), t;
  var j;
  var start = Date.now();
  for (var n = 0; n < N; n++) {
    t = i; i = i1; i1 = t;
    j = ~~i;
  }

  return (Date.now() - start) / N;
}
```

```
function benchmark() {
  var i = Date.now().toString(),
      i1 = Date.now().toString(), t;
  var j;
  var start = Date.now();
  for (var n = 0; n < N; n++) {
    t = i; i = i1; i1 = t;
    j = ~~i;
  }
  if (i != j || i1 != j) throw "MMM?";
  return (Date.now() - start) / N;
}
```

неплохо

(возможно)

недостаточно

```
for (var n = 0; n < N; n++) {
  t = i; i = i1; i1 = t;
  j = ~~i;
}
```

```
for (var n = 0; n < (N / 2); n++) {
  t = i; i = i1; i1 = t;
  j = ~~i;
  t = i; i = i1; i1 = t;
  j = ~~i;
}
if ((N % 2) === 1) {
  t = i; i = i1; i1 = t;
  j = ~~i;
}
```

```
for (var n = 0; n < (N / 2); n++) {
  j = ~~i1;
  t = i1;
  j = ~~i;
}
```

```
for (var n = 0; n < (N / 2); n++) {
  j = ~~i1; /* dead */
  t = i1;   /* dead */
  j = ~~i;  /* invariant */
}
```

развертка циклов

loop unrolling

V8 не умеет

V8 не умеет

но я хочу вселить параною 😌

презумпция производительности

разумный код
разумно быстр

confirmation bias

склонность к подтверждению своей точки зрения

«цепочки прототипов медленные»

```
var obj =
  Object.create(
    Object.create(
      Object.create(
        Object.create(
          Object.create({prop: 10})))));
```

```
var obj =
  Object.create(
    Object.create(
      Object.create(
        Object.create(
          Object.create(
            Object.create({prop: 10}))))));
                  // дети любят LISP ^^^^^
                  // и Серёжа тоже
```

```
function doManyLookups() {
  var counter = 0;
  for(var i = 0; i < 1000; i++)
    for(var j = 0; j < 1000; j++)
      for(var k = 0; k < 1000; k++)
        counter += obj.prop;
  print('Всего: ' + counter);
}
```

```
function lookupAndCache() {
  var counter = 0;
  var value = obj.prop;
  for(var i = 0; i < 1000; i++)
    for(var j = 0; j < 1000; j++)
      for(var k = 0; k < 1000; k++)
        counter += value;
  print('Всего: ' + counter);
}
```

```
// Супер продвинутый бенчмарк фреймворк.
function measure(f) {
  var start = Date.now();
  f();
  var end = Date.now();
  print(f.name + ' отработал за ' +
        (end - start) + ' ms.');
}
```

```
measure(doManyLookups);
measure(lookupAndCache);
```

```
$ node prototype.js
Всего: 10000000000
doManyLookups отработал за 8243 ms.
Всего: 10000000000
lookupAndCache отработал за 1058 ms.
```

```
$ node prototype.js
Всего: 1000000000
doManyLookups отработал за 8243 ms.
Всего: 1000000000
doLookupAndCache отработал за 1058 ms.
```

ПОДТВЕРЖДЕНО

усложним задачу

```
-    Object.create({ prop: 10 }))))));
+    Object.create({ get prop () { return 10 } }))))));
```

```
$ node prototype.js
Всего: 10000000000
doManyLookups отработал за 1082 ms.
Всего: 10000000000
lookupAndCache отработал за 1061 ms.
```

«что за уличная магия?»

```
 B13
 v96 BlockEntry
 v97 Simulate id=90
 v98 StackCheck changes[NewSpacePromotion]
v103 Simulate id=113 push d81, push t99
v104 EnterInlined prop, id=4
v106 LeaveInlined
v107 Simulate id=111 push t105
v108 Goto B14

 B14
v109 BlockEntry
d110 Add d81 d147 !
i                                    483647,1000,m0=0]
v113 Simulate id=86 pop 2 / var[2] = d110, var
v114 Goto B11
```

inlined

Constant 10 range[10,10,m0=0]

```
B13
v96  BlockEntry
v97  Simulate id=90
v98  StackCheck changes[NewSpacePromotion]
t99  LoadContextSlot t30[4]
t100 LoadNamedGeneric t99.prop changes[*]
v101 Simulate id=111 push d81, push t100
d139 Change t100 t to d
d102 Add d81 d139 !
i104 Add i84 i103 range[-2147483647,1000,m0=0
v105 Simulate id=86 pop 2 / var[2] = d102, va
v106 Goto B11
```

очень старая V8 не умела инлайнить обращение к data свойствам объявленным на прототипе

попробуем новую V8

```
$ d8 prototype.js
Всего: 10000000000
doManyLookups отработал за 1294 ms.
Всего: 10000000000
lookupAndCache отработал за 1189 ms.
```

```
$ d8 prototype.js
Всего: 10000000000
doManyLookups отработал за 1294 ms.
Всего: 10000000000
lookupAndCache отработал за 1189 ms.
```

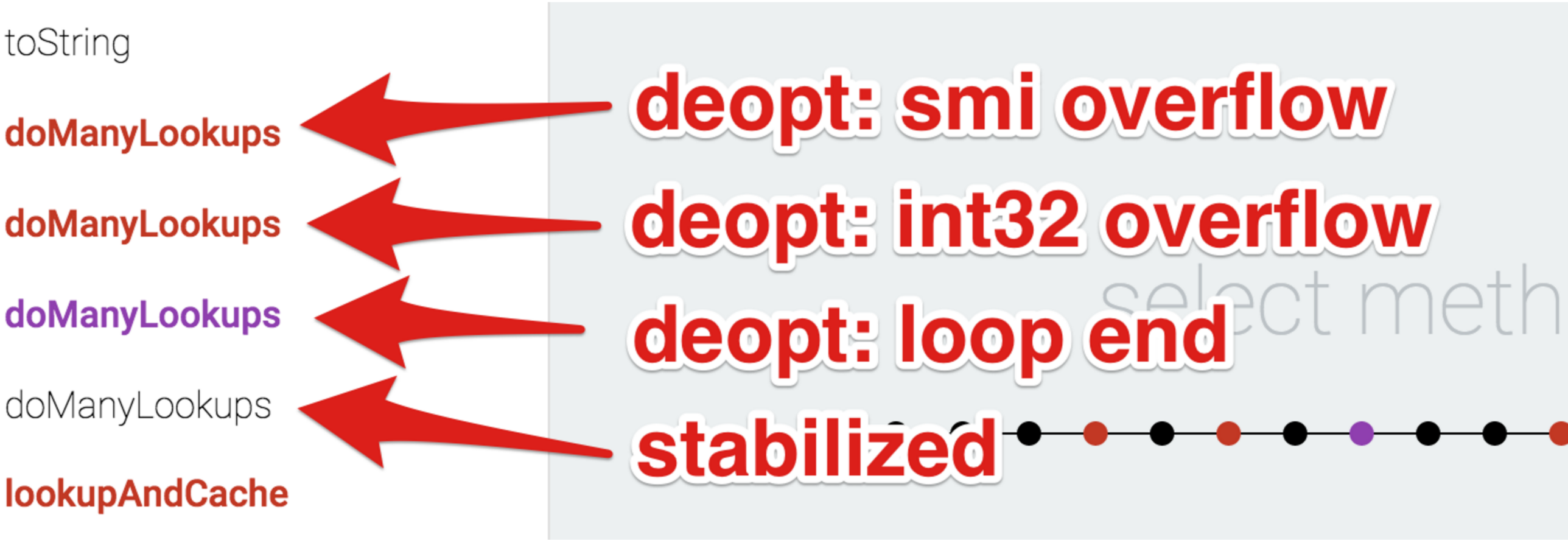
проход по цепочке прототипов — инвариант цикла

а что если
запустить
дважды?

```
measure(doManyLookups);
measure(doManyLookups);
measure(lookupAndCache);
measure(lookupAndCache);
```

```
$ d8 prototype.js
doManyLookups отработал за 1301 ms.
doManyLookups отработал за 3408 ms.
lookupAndCache отработал за 1204 ms.
lookupAndCache отработал за 3406 ms.
```

```
$ d8 prototype.js
doManyLookups отработал за 1301 ms.
doManyLookups отработал за 3408 ms.
lookupAndCache отработал за 1204 ms.
lookupAndCache отработал за 3406 ms.
```


toString
doManyLookups
doManyLookups
doManyLookups
doManyLookups
lookupAndCache
deopt: smi overflow
deopt: int32 overflow
deopt: loop end
stabilized

```
B13
            counter += obj.prop;
d221 Change t87 t to d allow-undefined-as-nan
            counter += obj.prop;
d120 Add d221 d222 !
          for(var k = 0; k < 1000; k++) {
s125 Add s90 s124
          for(var k = 0; k < 1000; k++) {
t223 Change d120 d to t changes[NewSpacePromotNumber
v128 Goto B11
```

unbox

add 10

rebox

Выражение `'Всего: ' + counter` влияло на представление выбираемое для переменной `counter`

workaround:
спрятать + от
выбора представлений

```
-    print('Всего: ' + counter);
+    print('Всего: ' + counter.toString());
```

```
$ d8 prototype.js
doManyLookups отработал за 1298 ms.
doManyLookups отработал за 1119 ms.
lookupAndCache отработал за 1188 ms.
lookupAndCache отработал за 982 ms.
```

```
Benchmark.prototype.setup = function () {
  var obj = { }

  var _a = 0;
  Object.defineProperty(obj, 'a', {
    get: function () { return _a; },
    set: function (val) { _a = val; }
  });
};
```

```
suite.add('Accessor', function () {
  obj.a = 2;
  obj.a;
});
```

```
$ v5.3/d8 concat.js
Accessor  x   3,840,922 ops/sec ±1.54%
```

```
$ v5.3/d8 concat.js
Accessor  x   3,840,922 ops/sec ±1.54%

$ v3.31/d8 concat.js
Accessor  x 874,731,387 ops/sec ±1.09%
```

```
$ v5.3/d8 concat.js
Accessor  x   3,840,922 ops/sec ±1.54%

$ v3.31/d8 concat.js
Accessor  x 874,731,387 ops/sec ±1.09%
```

```
function f() {
  var obj = { }
  var _a = 0;
  Object.defineProperty(obj, 'a', {
    get: function () { return _a; },
    set: function(val) { _a = val; }
  });
  for (var i = 0; i < 1e6; i++) {
    obj.a = 2;
    obj.a;
  }
}
```

```
f();  // =>   2ms
```

```
f();  // =>   2ms
f();  // => 287ms
```

```
f();   // =>    2ms
f();   // => 287ms
```

```
Object.defineProperty(obj, 'a', {
  /* ... */
});
print('obj имеет быстрые свойства: ' +
      %HasFastProperties(obj));
```

```
$ d8 --allow-natives-syntax
obj имеет быстрые свойства: true
f() took 2ms
obj имеет быстрые свойства: false
f() took 287ms
```

hidden class transition clash

```
var obj1 = { }
Object.defineProperty(obj1, 'a', {
  get: function () { /* ... */ },
  set: function(val) { /* ... */ }
});
var obj2 = { }
Object.defineProperty(obj2, 'a', {
  get: function () { /* ... */ },
  set: function(val) { /* ... */ }
});
// Начальный скрытый класс для obj1/obj2 одинаков
// а вот после defineProperty они пришли к разным
```

```
Object.defineProperty(obj, 'a', {
  /* ... */
});
Object.create(obj);
```

```
f();  // =>  4ms
f();  // =>  3ms
```

```
f();   // =>    4ms
f();   // =>    3ms
f();   // =>    3ms
f();   // =>    9ms
f();   // =>   28ms
```

```
function g() {
  var obj = { }
  var _a = 0;
  obj.a = {
    get: function () { return _a; },
    set: function(val) { _a = val; }
  };

  for (var i = 0; i < 1e6; i++) {
    obj.a.set(2);
    obj.a.get();
  }
}
```

```
g();   // =>  23ms
g();   // =>  15ms
g();   // =>  16ms
g();   // =>  22ms
g();   // =>  16ms
```

```
function Box(_a) {
  this._a = _a;
}

Box.prototype.get = function () {
  return this._a;
};

Box.prototype.set = function (val) {
  this._a = val;
};
```

```
function h() {
  var obj = { }
  obj.a = new Box(0);
  for (var i = 0; i < 1e6; i++) {
    obj.a.set(2);
    obj.a.get();
  }
}
```

```
h();  // =>   2ms
h();  // =>   1ms
h();  // =>   0ms
h();  // =>   1ms
h();  // =>   1ms
```

а сейчас десерт!

вызов метода против вызова функции

```
function mk(word) {
  var len = word.length;
  if (len > 255) return undefined;
  var i = len >> 2;
  return String.fromCharCode(
    (word.charCodeAt(    0) & 0x03) << 14 |
    (word.charCodeAt(    i) & 0x03) << 12 |
    (word.charCodeAt(  i+i) & 0x03) << 10 |
    (word.charCodeAt(i+i+i) & 0x03) <<  8 |
    len
  );
}
```

```
Benchmark.prototype.setup = function() {
  function mk(word) {
    /* ... */
  }

  var MK = function() { };
  MK.prototype.mk = mk;
  var mker = new MK;
};
```

```
suite
  .add('Function', function() {
    var key = mk('www.wired.com');
    key = mk('www.youtube.com');
    key = mk('scorecardresearch.com');
    key = mk('www.google-analytics.com');
  })
  .add('Method', function() {
    var key = mker.mk('www.wired.com');
    key = mker.mk('www.youtube.com');
    key = mker.mk('scorecardresearch.com');
    key = mker.mk('www.google-analytics.com');
  })
```

```
$ d8 method-vs-function.js
Function x   4,149,776 ops/sec ±0.62%
Method   x 682,273,122 ops/sec ±0.72%
Fastest is Method
```

вызов метода

быстрее?

СОХРАНИТЕ В СЕКРЕТЕ ТО, ЧТО ВЫ СЕЙЧАС УВИДИТЕ

```
--- a/method-function.js
+++ b/method-function.js
@@ -2,6 +2,9 @@
 load("../benchmark.js");

 Benchmark.prototype.setup = function() {
+    "Разгони" + "свой" + "Жава" + "Скрипт" +
+    "этим" + "супер" + "секретным" + "трюком";
+
     function mk(word) {
         var len = word.length;
         if (len > 255) return undefined;
```

```
$ d8 method-vs-function.js
Function x 695,708,197 ops/sec ±0.38%
Method   x 692,496,013 ops/sec ±0.29%
Fastest is Function,Method
```

```
$ d8 method-vs-function.js
Function x 695,708,197 ops/sec ±0.38%
Method   x 692,496,013 ops/sec ±0.29%
Fastest is Function,Method
```

isFunction

no optimized instances!

select

filter by name

Function

Function не была оптимизирована

— эвристики, принимающие решение об оптимизации, основаны в том числе и на количестве инициализированных встроенных кэшей (inline cache);

— вызов функции не делался через IC, в отличие от вызова метода;

— `Method` содержал достаточно инициализированных IC для срабатывания оптимизаций, а вот `Function` — нет!

— Добавленные операторы + увеличили количество инициализированных IC.

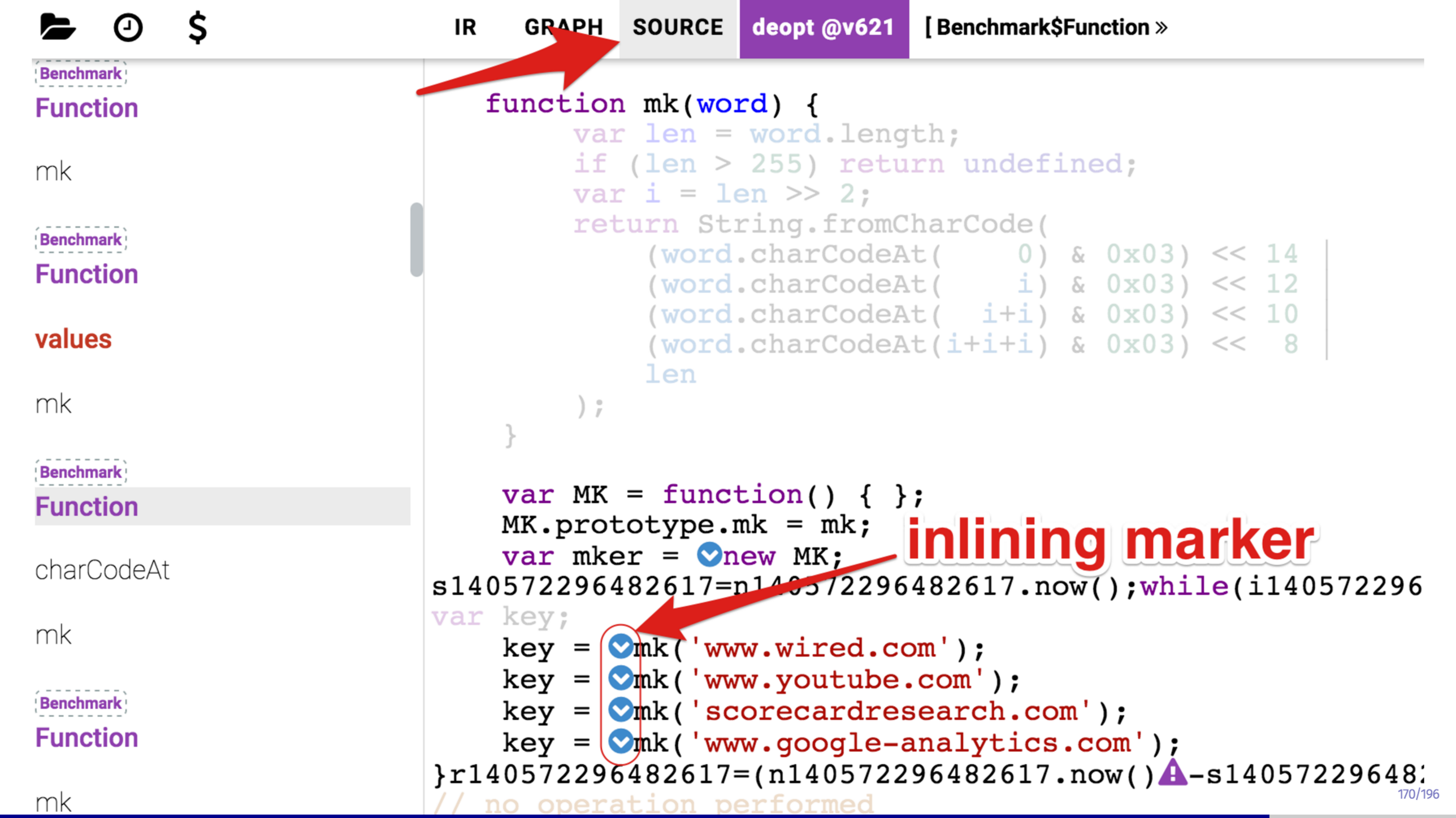
— эвристики, принимающие решение об оптимизации, основанн в том числе и на количестве инициализированных встроенных кэшей (inline cache);

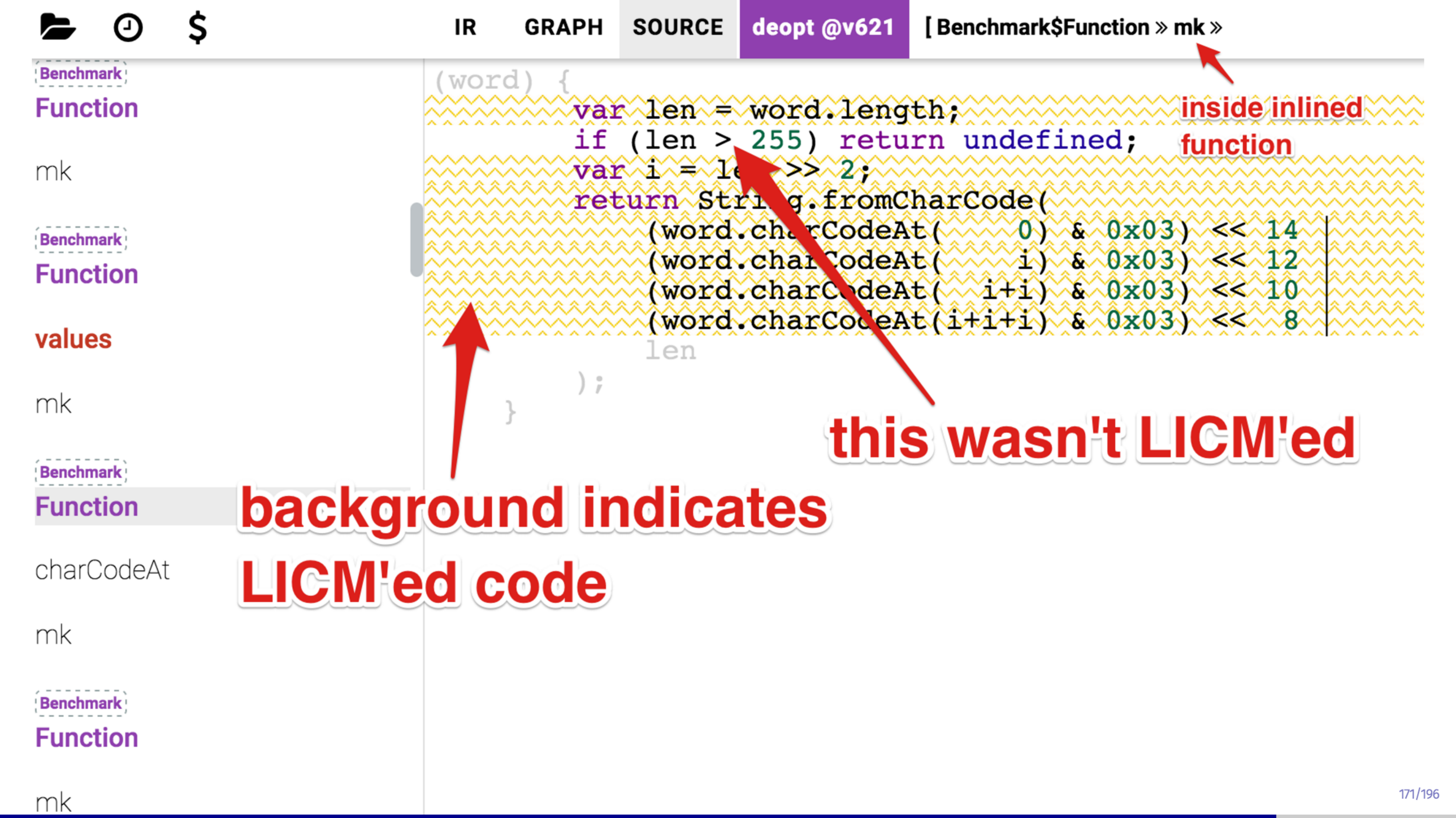
— Вызов функции не делался через IC, в отличие от вызова метода;

— `Method` содержал достаточно инициализированных IC для срабатывания оптимизаций, а вот `Function` — нет!

— Добавленные операторы + увеличили количество инициализированных IC.

БЛА БЛА БЛА


IR
GRAPH
SOURCE
deopt @v621
[Benchmark$Function »
Benchmark
Function
mk
Benchmark
Function
values
mk
Benchmark
Function
charCodeAt
mk
Benchmark
Function
mk
function mk(word) {
var len = word.length;
if (len > 255) return undefined;
var i = len >> 2;
return String.fromCharCode(
(word.charCodeAt(0) & 0x03) << 14
(word.charCodeAt(i) & 0x03) << 12
(word.charCodeAt(i+i) & 0x03) << 10
(word.charCodeAt(i+i+i) & 0x03) << 8
len
);
}
var MK = function() { };
MK.prototype.mk = mk;
var mker = new MK;
s140572296482617=n140572296482617.now();while(i140572296
var key;
key = mk('www.wired.com');
key = mk('www.youtube.com');
key = mk('scorecardresearch.com');
key = mk('www.google-analytics.com');
}r140572296482617=(n140572296482617.now()-s140572296482
// no operation performed
inlining marker
170/196


IR
GRAPH
SOURCE
deopt @v621
[Benchmark$Function » mk »
Benchmark
Function
mk
Benchmark
Function
values
mk
Benchmark
Function
charCodeAt
mk
Benchmark
Function
mk
(word) {
var len = word.length;
if (len > 255) return undefined;
var i = le >> 2;
return Stri g.fromCharCode(
(word.charCodeAt(0) & 0x03) << 14
(word.charCodeAt(i) & 0x03) << 12
(word.charCodeAt(i+i) & 0x03) << 10
(word.charCodeAt(i+i+i) & 0x03) << 8
len
);
}
inside inlined function
this wasn't LICM'ed
background indicates LICM'ed code

опять замеряем

практически

пустой цикл!

а теперь
наоборот

```
// for (var item in list) { ... }
var sum = 0;
for (var i = 0, L = arr.length;
     i < arr.length;
     ++i) {
  if (arr.length !== L)
    H.throwConcurrentModificationError(arr);
  var item = list[i];
  sum += item;
}
```

```
// for (var item in list) { ... }
var sum = 0;
for (var i = 0, L = arr.length;
     i < arr.length;
     ++i) {
  // if (arr.length !== L)
  //   H.throwConcurrentModificationError(arr);
  var item = list[i];
  sum += item;
}
```

```
$ d8 iteration.js
WithCheck    x 396,792 ops/sec ±0.37%
WithoutCheck x 456,653 ops/sec ±0.82%
Fastest is WithoutCheck (by 18%)
```

```
// for (var item in list) { ... }
var sum = 0;
for (var i = 0, L = arr.length;
     i < arr.length;
     ++i) {
  if (arr.length !== L)
    H.throwConcurrentModificationError(arr);
  var item = list[i];
  sum += item;
}
```

```
// for (var item in list) { ... }
var sum = 0;
for (var i = 0, L = arr.length;
     i < arr.length;
     ++i) {
  if (arr.length !== L)
    (0,H.throwConcurrentModificationError)(arr);
  var item = list[i];
  sum += item;
}
```

```
$ d8 iteration.js
WithCheck    x 396,792 ops/sec ±0.37%
WithoutCheck x 456,653 ops/sec ±0.82%
WithHack     x 462,698 ops/sec ±0.48%
Fastest is WithoutCheck,WithHack
```

```
$ d8 iteration.js
WithCheck     x 396,792 ops/sec ±0.37%
WithoutCheck  x 456,653 ops/sec ±0.82%
WithHack      x 462,698 ops/sec ±0.48%
Fastest is WithoutCheck,WithHack
```

разогнали на **18%** заменив
`o.f()` на `(0,o.f)()`
в коде, который
никогда на исполняется

```
(window,t1431360483467) { var global = window, clearTimeout = global.cl
var r1431360483467,s1431360483467,m1431360483467=this,f1431360483467
  for (var i = 0; i < 1000; i++) arr.push(i);
s1431360483467=n1431360483467.now();while(i1431360483467--){
var sum = 0;
    for (var i = 0, l = arr.length; i < arr.length; ++i) {
      if (arr.length !== l)
        H.throwConcurrentModificationError(arr);
      sum += arr[i];
    }
}r1431360483467=(n14[illegible]34667)/1e3;
// no operation perfor[illegible]
return{elapsed:r143136[illegible]
```

never executed and

V8 doesn't know where it goes

```
(window,t14313604834667) { var global = window, clearTimeout = global.cl
var r14313604834667,s1431360483[illegible] m14313604834667=this.f1431360483466
  for (var i = 0; i < 1000; i++[illegible]
s14313604834667=n14313604834667.now();while(i14313604834667--){
var sum = 0;
    for (var i [illegible], l = arr.length; i < arr.length; ++i) {
      if (arr.length !== l)
        H.throwConcurrentModificationError(arr);
      sum += arr[i];
    }
}r14313604834667=(n14313604834667.now()-s14313604834667)/1e3;
// no operation performed
return{elapsed:r14313604834667,uid:"uid14313604834667"}}
```

has to assume length

can change

```
(window,t143136048346657) { var global = window, clearTimeout = global.c
var r143136048346657,s143136048346657,m143136048346657=this,f14313604834
  for (var i = 0; i < 1000; i++) arr.push(i);
s143136048346657=n143136048346657.now();while(i143136048346657--){
var sum = 0;
    for (var i = 0, l = arr.length; i < arr.length; ++i) {
      if (arr.length !== l)
        (0, H.throwConcurrentModificationError)(arr);
      sum += ar[i];
    }
}r143136048346657=(n143136048346657.now()-s143136048346657)/1e3;
// no operation perform
return{elapsed:r143136
```

also never executed

but property loads are special

```
(window,t143136048346657) { var global = window, clearTimeout = global.c
var r143136048346657,s143136048346657=n143136048346657.now();while(i<334
  for (var i = 0; i < 1000; i++) arr.push(i);
s143136048346657=n143136048346657.now();while(i<334
var sum = 0;
    for (var i = 0, l = arr.length; i < arr.length; ++i) {
      if (arr.length !== l)
        (0, H.throwConcurrentModificationError)(arr);
      sum += arr[i];
    }
}r143136048346657=(n143136048346657.now()-s143136048346657)/1e3;
// no operation performed
return{elapsed:r143136048346657,uid:"uid143136048346657"}}
```

assume that paths leading to never executed loads are never taken

```
(window,t143136048346657) { var global = window, clearTimeout = global.c
var r143136048346657,s143136048346657,m143136048346657=this,f14313604834
  for (var i = 0; i < 1000; i++) arr.push(i);
s143136048346657=n143136048346657.now();while(i143136048346657--){
var sum = 0;
    for (var i = 0, l = arr.length; i < arr.length; ++i) {
      if (arr.length !== l)
        (0, H.throwConcurrentModificationError)(arr);
      sum += arr[i];
    }
}r143136048346657=(n143136048346657.now()-s143136048346657)/1e3;
// no operation performed
return{elapsed:r143136048346657,uid:"uid143136048346657"}}
```

hoisted

folded away

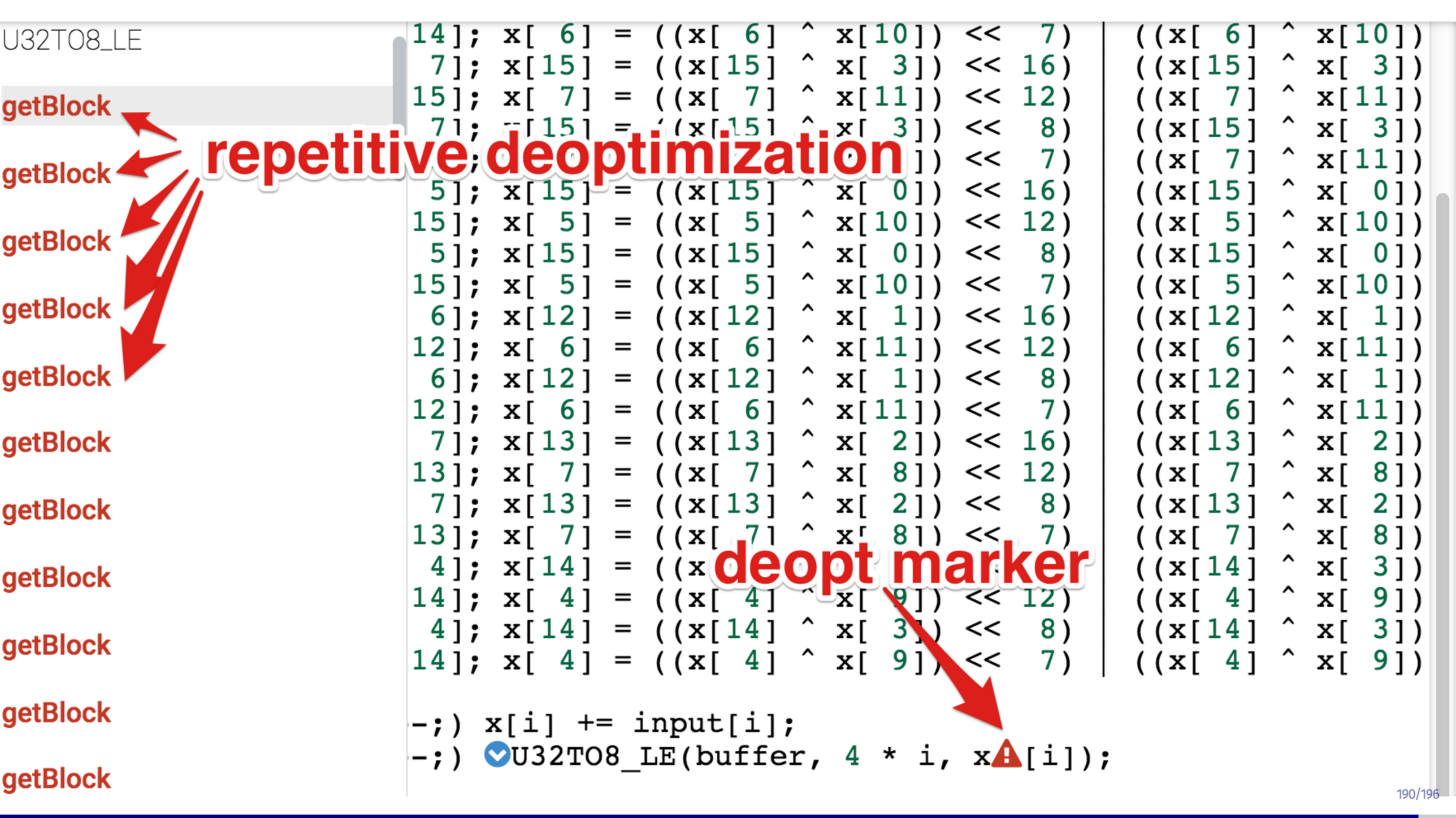
СПАСИБО

ПОСТОЙТЕ

а как же тот пример с ChaCha20?

```
function getBlock(buffer) {
  var x = new Uint32Array(16);

  for (var i = 16; i--;) x[i] = input[i];
  for (var i = 20; i > 0; i -= 2) {
    // quarterRound(x, 0, 4, 8,12);
    x[ 0] += x[ 4]; x[12] = ((x[12] ^ x[ 0]) << 16) | ((x[12] ^ x[ 0])
    x[ 8] += x[12]; x[ 4] = ((x[ 4] ^ x[ 8]) << 12) | ((x[ 4] ^ x[ 8])
    x[ 0] += x[ 4]; x[12] = ((x[12] ^ x[ 0]) <<  8) | ((x[12] ^ x[ 0])
    x[ 8] += x[12]; x[ 4] = ((x[ 4] ^ x[ 8]) <<  7) | ((x[ 4] ^ x[ 8])
    // ... so on ...
  }
  for (i = 16; i--;) x[i] += input[i];
  for (i = 16; i--;) U32TO8_LE(buffer, 4 * i, x[i]);
  input[12]++;
  return buffer;
}
```


U32TO8_LE
getBlock
getBlock
getBlock
getBlock
getBlock
getBlock
getBlock
getBlock
getBlock
getBlock
getBlock
getBlock
repetitive deoptimization
deopt marker
14]; x[6] = ((x[6] ^ x[10]) << 7) | ((x[6] ^ x[10])
7]; x[15] = ((x[15] ^ x[3]) << 16) | ((x[15] ^ x[3])
15]; x[7] = ((x[7] ^ x[11]) << 12) | ((x[7] ^ x[11])
7]; x[15] = ((x[15] ^ x[3]) << 8) | ((x[15] ^ x[3])
0]) << 7) | ((x[7] ^ x[11])
5]; x[15] = ((x[15] ^ x[0]) << 16) | ((x[15] ^ x[0])
15]; x[5] = ((x[5] ^ x[10]) << 12) | ((x[5] ^ x[10])
5]; x[15] = ((x[15] ^ x[0]) << 8) | ((x[15] ^ x[0])
15]; x[5] = ((x[5] ^ x[10]) << 7) | ((x[5] ^ x[10])
6]; x[12] = ((x[12] ^ x[1]) << 16) | ((x[12] ^ x[1])
12]; x[6] = ((x[6] ^ x[11]) << 12) | ((x[6] ^ x[11])
6]; x[12] = ((x[12] ^ x[1]) << 8) | ((x[12] ^ x[1])
12]; x[6] = ((x[6] ^ x[11]) << 7) | ((x[6] ^ x[11])
7]; x[13] = ((x[13] ^ x[2]) << 16) | ((x[13] ^ x[2])
13]; x[7] = ((x[7] ^ x[8]) << 12) | ((x[7] ^ x[8])
7]; x[13] = ((x[13] ^ x[2]) << 8) | ((x[13] ^ x[2])
13]; x[7] = ((x[7] ^ x[8]) << 7) | ((x[7] ^ x[8])
4]; x[14] = ((x | ((x[14] ^ x[3])
14]; x[4] = ((x[4] ^ x[9]) << 12) | ((x[4] ^ x[9])
4]; x[14] = ((x[14] ^ x[3]) << 8) | ((x[14] ^ x[3])
14]; x[4] = ((x[4] ^ x[9]) << 7) | ((x[4] ^ x[9])
-;) x[i] += input[i];
-;) U32TO8_LE(buffer, 4 * i, x[i]);
190/196

повторяющаяся деоптимизация всегда баг V8

— инлайн `U32T08_LE` ломал *safe-uint32* анализ

— длинный комментарий в `U32T08_LE` отключал инлайн

— есть и вменяемые способы обойти проблему

```
function getBlock(buffer) {
  var x = new Uint32Array(16);

  for (var i = 16; i--;) x[i] = input[i];
  for (var i = 20; i > 0; i -= 2) {
    // ...
  }
  for (i = 16; i--;) x[i] += input[i];
  for (i = 16; i--;) U32TO8_LE(buffer, 4 * i, x[i]|0);
  //                                  ОБРЕЗАТЬ ДО INT32 ^^
  input[12]++;
  return buffer;
}
```

разработчики VM
ваши друзья!
(шлите баги)

СПАСИБО

Q&A

[me@mrale.ph | @mraleph]