

Copy Summary

View

Closed

Bug 1271473

Opened 6 years ago

Closed 6 years ago

Constructors are slower when there's an if (false) { /* access arguments */ } statement in them than if actually accessing arguments

Categories

Product: Core

Component: JavaScript Engine

Version: 46 Branch

Platform: x86_64 Windows 10

Type: defect

Priority: Not set

Severity: normal

Tracking

Status: RESOLVED FIXED

People

Reporter: alexgorisse, Unassigned

References

Depends on 1 open bug


Details

Keywords: testcase, Whiteboard: javascript engine

Bottom

Tags

Timeline

 alex gorisse Reporter
Description • 6 years ago

—

User Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:46.0) Gecko/20100101 Firefox/46.0
Build ID: 20160502172042

Steps to reproduce:

```
var time = null,
    msg = "";

function Foo(toto) {
    this.args = toto;
}

function Foo0() {
    var args = arguments;
}

function Foo1() {
    this.args = arguments;
}

var dumb = false;
function Foo2() {
    if (dumb) this.args = arguments;
}

var isFirefox = typeof InstallTrigger !== 'undefined';
function Foo3() {
    if (!isFirefox) this.args = arguments;
}

function Foo4() {
    this.args = [];
    for (var i = 0; i < arguments.length; i++) {
        this.args[i] = arguments[i];
    }
}

var before = Date.now();
for (var i = 0; i < 1000000 ; i++) { var toto = new Foo() }
console.log("Time : "+ (Date.now() - before));

var before = Date.now();
for (var i = 0; i < 1000000 ; i++) { var toto = new Foo0() }
console.log("Time : "+ (Date.now() - before));

var before = Date.now();
for (var i = 0; i < 1000000 ; i++) { var toto = new Foo1() }
console.log("Time : "+ (Date.now() - before));

var before = Date.now();
for (var i = 0; i < 1000000 ; i++) { var toto = new Foo2() }
console.log("Time : "+ (Date.now() - before));

var before = Date.now();
for (var i = 0; i < 1000000 ; i++) { var toto = new Foo3() }
console.log("Time : "+ (Date.now() - before));


var before = Date.now();
for (var i = 0; i < 1000000 ; i++) { var toto = new Foo4() }
console.log("Time : "+ (Date.now() - before));
```

Actual results:

Slow whenever you don't want to set arguments on firefox...

Expected results:

Should be fast, on direct set, or when you forbid the set

 alex gorisse Reporter
Updated • 6 years ago

—

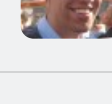
Severity: normal → critical

Component: Untriaged → General

OS: Unspecified → Windows 10

Hardware: Unspecified → x86_64

Whiteboard: javascript engine

 :Gijs (he/him)
Updated • 6 years ago

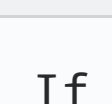
—

Component: General → JavaScript Engine

Keywords: testcase

Product: Firefox → Core

Summary: arguments are always called → Constructors are slower when there's an if (false) { /* access arguments */ } statement in them than if actually accessing arguments


 Jan de Mooij [jandem]
Comment 1 • 6 years ago

—

If a function needs an arguments object, we create it when we *enter* the function, not when we first use |arguments|. So even if |arguments| is only used inside an |if (alwaysFalse) {}| block, we will create an arguments object.

Arguments objects are pretty slow in all engines. The Foo4 case is optimized, we don't create an actual arguments object in that case.

Is this actually hurting performance somewhere?

 alex gorisse Reporter
Comment 2 • 6 years ago

—

Its hurts when you try to chain functions or manipulate instances.

I really don't know how to bind a prototype of an other function, when you don't know its arguments... I could use Object.keys but performances are worst.

Here's a quick example :

```
function Test() {

    this.A = function(a1) {
        this.args = arguments;
        this.a1 = a1;
    }


    this.A.prototype.foo = function() {
        console.log("I am " + this.a1);
    }

    this.B = function(a) {
        if (a.args) {
            a.constructor.apply(this, a.args)
        }
    }

    this.B.prototype = this.A.prototype;
}

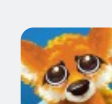
var test = new Test();

var a = new test.A("mother A")
var b = new test.B(a);
b.foo();
```

 alex gorisse Reporter
Comment 3 • 6 years ago

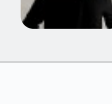
—

In the end, the biggest issue is that I have to use foo4() in every browser, while foo1() is faster in Chrome, unless you can solve the problem above...

 Loïc
Updated • 6 years ago

—

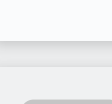
Severity: critical → normal

 Till Schneiderreit [till]
Comment 4 • 6 years ago

—

In my testing, foo4 is fastest in Chrome (current Chrome Canary, specifically):
Time : 20
Time : 13
Time : 27
Time : 25
Time : 30
Time : 14

This makes sense, because as jandem explained creating an arguments object is expensive, and there's just no way not to do it if 'arguments' escapes from the function.

 alex gorisse Reporter
Comment 5 • 6 years ago

—

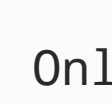
try with args :

```
new Foo1("a", "b", "c", "d", "e");

new Foo4("a", "b", "c", "d", "e");
```

Chrome is faster with foo1() ;), and so is safari and almost IE

Only FF gives the worst results of all ! almost 5 time slower, with no way to use foo3() to avoid that...

 Jan de Mooij [jandem]
Comment 6 • 6 years ago

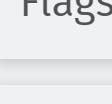
—

Tomorrow I'll look at arguments object allocation performance.

I've optimized this before ([bug-1175466](#), [bug-1175511](#)) but |arguments| is still fairly common in some frameworks, so it's worth spending some more time on it I think.

Depends on: [1132004](#)

Flags: needinfo?(jdemooij)

 alex gorisse Reporter
Comment 7 • 6 years ago

—

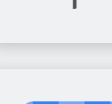
Thanks a lot !

Don't hesitate to put me in touch, in order to improve my framework ^^

 Jan de Mooij [jandem]
Updated • 6 years ago

—


Depends on: [1271929](#)

 Jan de Mooij [jandem]
Comment 8 • 6 years ago

—


I've a patch stack that improves Foo1 (and the other tests that require an arguments object) from 152 ms to 44 ms or so. Foo4 is about 2x faster at this point.

There's another optimization I want to try tomorrow, but I think these numbers are pretty nice (arguments objects used to be horribly slow).

 Jan de Mooij [jandem]
Updated • 6 years ago

—

Depends on: [1272596](#)

 alex gorisse Reporter
Comment 9 • 6 years ago

—

Thanks again, hope to see this great patch include in FF one day !

By the way, I found an other strange bug :

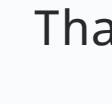
can you try this :

```
var before = Date.now();
for (var i = 0; i < 1000000 ; i++) { var toto = new Foo4("a1", "a2", "a3", "a4", "a5", "a6") }
console.log("Time 41 : "+ (Date.now() - before));

var before = Date.now();
for (var i = 0; i < 1000000 ; i++) { var toto = new Foo4("a1", "a2", "a3", "a4", "a5", "a6", "a7") }
console.log("Time 41 : "+ (Date.now() - before));
```

In my browser, with 6 arguments (or less), I've got a result of 150ms, and just when I put 7 arguments, it goes to 450 ms. At 14 arguments, I still have 450ms, and when I put 15 arguments, It goes to 1000 ms !

Hope the patch solve that too, and that is not an Array issue...

 Jan de Mooij [jandem]
Comment 10 • 6 years ago

—

(In reply to alex gorisse from [comment-#9](#))

> Thanks again, hope to see this great patch include in FF one day !

[Bug-1272596](#) will be in Firefox 50. You can download a Nightly build if you want to test it.

> In my browser, with 6 arguments (or less), I've got a result of 150ms, and
> just when I put 7 arguments, it goes to 450 ms. At 14 arguments, I still
> have 450ms, and when I put 15 arguments, It goes to 1000 ms !

That's likely unrelated: I'd guess 6 elements fit inline in the array and when you add more, we need to allocate external elements and take a slower path. We can make that faster with some work but it's an unrelated issue.

I'll close this bug because the main arguments perf issue is fixed. Thanks for the bug report and feel free to file more bugs!

Status: UNCONFIRMED → RESOLVED
Closed: 6 years ago
Flags: needinfo?(jdemooij)
Resolution: --- → FIXED