Ruby trunk - Bug #1720

[NaN] == [NaN] □ true □□□

07/03/2009 09:43 PM - tadf (tadayoshi funaba)

Status:	Closed		
Priority:	Normal		
Assignee:	mrkn (Kenta Murata)		
Target version:	2.0.0		
ruby -v:	ruby 1.9.2dev (2009-07-03 trunk 23945) [i686-linux]	**Backport:**	2.3: UNKNOWN, 2.4: UNKNOWN

Description

=begin
NaN = 0.0/0
[NaN] == [NaN] □ true □□□□□□□

NaN == NaN #=> false
[1] == [1.0] #=> true

□□□□□□□□□□□□□□□□□□□□□□□□□
=end

Related issues:

Has duplicate Ruby trunk - Bug #7676: Comparison of Float::NAN in array behav...	**Open**

Associated revisions

Revision 6ea34efa - 11/08/2012 12:34 AM - mrkn (Kenta Murata)

- numeric.c: Add description of that the results of the comparing operations of two NaNs are undefined. [#1720] [ruby-dev:38725] [ruby-core:36966]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@37546 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 37546 - 11/08/2012 12:34 AM - mrkn (Kenta Murata)

- numeric.c: Add description of that the results of the comparing operations of two NaNs are undefined. [#1720] [ruby-dev:38725] [ruby-core:36966]

Revision 37546 - 11/08/2012 12:34 AM - mrkn (Kenta Murata)

- numeric.c: Add description of that the results of the comparing operations of two NaNs are undefined. [#1720] [ruby-dev:38725] [ruby-core:36966]

Revision 37546 - 11/08/2012 12:34 AM - mrkn (Kenta Murata)

- numeric.c: Add description of that the results of the comparing operations of two NaNs are undefined. [#1720] [ruby-dev:38725] [ruby-core:36966]

Revision 37546 - 11/08/2012 12:34 AM - mrkn (Kenta Murata)

- numeric.c: Add description of that the results of the comparing operations of two NaNs are undefined. [#1720] [ruby-dev:38725] [ruby-core:36966]

Revision 37546 - 11/08/2012 12:34 AM - mrkn (Kenta Murata)

- numeric.c: Add description of that the results of the comparing operations of two NaNs are undefined. [#1720] [ruby-dev:38725] [ruby-core:36966]

Revision 37546 - 11/08/2012 12:34 AM - mrkn (Kenta Murata)

- numeric.c: Add description of that the results of the comparing operations of two NaNs are undefined. [#1720] [ruby-dev:38725]

> [ruby-core:36966]

History

#1 - 07/05/2009 01:14 AM - matz (Yukihiro Matsumoto)

まつもと ゆきひろです

In message "Re: [ruby-dev:38725] [Bug #1720] [NaN] == [NaN] が true になる"
on Fri, 3 Jul 2009 21:43:24 +0900, tadayoshi funaba redmine@ruby-lang.org writes:

> NaN = 0.0/0
> [NaN] == [NaN] が true になりますが、
>
> ```
> NaN == NaN #=> false
> [1] == [1.0] #=> true
> ```
>
> という結果からするとおかしいように思います。

確かに。rb_equal()が両辺が同じオブジェクトのときにメソッド呼び出しを行わずに真を返しているせいですね。NaNというのは、equal?が成立しても==が成立しないという特異なオブジェクトであるためにこの問題が発生しています。

とはいえ、こんな特殊な例のために同値性チェックを遅くしたくないし、困ったものです。

#2 - 07/05/2009 01:31 AM - matz (Yukihiro Matsumoto)

まつもと ゆきひろです

In message "Re: [ruby-dev:38734] Re: [Bug #1720] [NaN] == [NaN] が true になる"
on Sun, 5 Jul 2009 01:14:16 +0900, Yukihiro Matsumoto matz@ruby-lang.org writes:

> > NaN = 0.0/0
> > [NaN] == [NaN] が true になりますが、
> >
> > ```
> > NaN == NaN #=> false
> > [1] == [1.0] #=> true
> > ```
> >
> > という結果からするとおかしいように思います。
>
> 確かに。rb_equal()が両辺が同じオブジェクトのときにメソッド呼び出しを行わずに真を返しているせいですね。NaNというのは、equal?が成立しても==が成立しないという特異なオブジェクトであるためにこの問題が発生しています。

考えられる対処は

(1) NaN == NaN も true にする
一貫性はあるが NaN の本来の挙動ではない
(2) rb_equal()でまずequal?でのチェックをやめる
性能が劣化するので避けたい
(3) rb_equal()でT_FLOATを特別扱い
2ほどではないにしても性能劣化が気になる
特別扱いは後悔することが多い
(4) このまま。これは例外的なケースとする

くらいでしょうか。

私自身は、どれが良いという意見を現時点では持たないのですが、どれが好きかと言われれば、(1)が好きです。

#3 - 07/05/2009 06:35 PM - tadf (tadayoshi funaba)

そんなに速度的にきついんですかね。
であれば、あまり妙なことはしないほうがいいと思うので、
既知の問題として当面は(4)とするとか。

#4 - 07/13/2009 08:53 PM - yugui (Yuki Sonoda)

- *Status changed from Open to Assigned*

- Assignee set to matz (Yukihiro Matsumoto)

- Priority changed from Normal to 3

私も(4)を推します。

NaNが生じるケースって、要は例外を発生すべきところ伝統的なモデルを尊重してこうなっている、と理解していますから。無理に整合性を持たせようとしても難しいんじゃないかなーと。だとすると一番被害が少ないのが(4)だと思います。

#5 - 07/13/2009 09:14 PM - keiju (Keiju Ishitsuka)

けいじゅ@いしつかです.

In [ruby-dev:38735] the message: "[ruby-dev:38735] Re: [Bug #1720]
[NaN] == [NaN] が true になる", on Jul/05 01:31(JST) Yukihiro
Matsumoto writes:

> まつもと ゆきひろです
>
> (1) NaN == NaN も true にする
> 一貫性はあるが NaN の本来の挙動ではない
> (2) rb_equal()でまずequal?でのチェックをやめる
> 性能が劣化するので避けたい
> (3) rb_equal()でT_FLOATを特別扱い
> 2ほどではないにしても性能劣化が気になる
> 特別扱いは後悔することが多い
> (4) このまま。これは例外的なケースとする
>
> 私自身は、どれが良いという意見を現時点では持たないのですが、
> どれが好きかと言われれば、(1)が好きです。

わたしも, (1)のような気がします.

というか, (1')ですか:

(1')

```
nan1 = 0.0/0
nan2 = 0.0/0
```

として,

```
nan1 == nan1 => true
nan1 == nan2 => false
```

現行では,

```
nan1.equal?(nan1)
```

なのに,

```
nan1 == nan1 => false
```

となるのは,オブジェクト指向的にかなり気分の悪い仕様だと思います. nan1
とnan1の値はやはり同じだとしてよいと思います.

> 一貫性はあるが NaN の本来の挙動ではない

とありますが, それに関しては nan1 == nan2 => false になれば問題ないき
がします.

___
-->> 石塚 圭樹 <<---
---------------------------------->> e-mail: keiju@ishitsuka.com <<---

#6 - 06/11/2011 02:30 PM - ko1 (Koichi Sasada)

誰に聞けばいいのかわかりませんが, これはどうなりますでしょうか.

#7 - 07/14/2012 02:13 PM - matz (Yukihiro Matsumoto)

- Status changed from Assigned to Closed

Rubyの言語仕様的にはNaNとNaNの比較は未定義と言うことにします。

処理系によっては等しいかもしれないし、そうでないかもしれない。

#8 - 07/14/2012 02:15 PM - ko1 (Koichi Sasada)

- Category set to doc

- Status changed from Closed to Assigned

- Assignee changed from matz (Yukihiro Matsumoto) to mrkn (Kenta Murata)

NaN と NaN の挙動は未定義ということで，実装は変えませんが，
ドキュメントの改訂が必要です．mrkn が引き受けてくれました．

#9 - 11/08/2012 10:18 AM - mrkn (Kenta Murata)

- Status changed from Assigned to Closed

#10 - 10/30/2017 08:58 PM - Eregon (Benoit Daloze)

Could someone summarize in English the rationale?

```
Float::NAN == Float::NAN # => false
```

The documentation says:

```
The result of NaN == NaN is undefined, so the implementation-dependent
value is returned.
```

But the result is false no matter the environment, isn't it? (NaN is the only numeric value never equal to itself)

And then we have:

```
[Float::NAN] == [Float::NAN] # => true
[0.0/0] == [Float::NAN] # => false
```

Which sounds to me like a bad side effect of short-circuiting in rb_equal on

```
if (a == b)
    return Qtrue;
else
    return rb_funcall(a, "==", 1, b);
```

#11 - 10/30/2017 10:59 PM - Eregon (Benoit Daloze)

At least IEEE 754 has NaN == NaN # => false.

#12 - 10/31/2017 03:07 AM - mame (Yusuke Endoh)

Summary:

ko1: [NaN] == [NaN] evaluates to true. This looks awkward since NaN == NaN is false and [1] == [1.0] is true.

matz: rb_equal first checks if the two sides are the same, which causes this behavior. NaN is a special object since equal? returns true but == returns false. However, I don't want to make equivalence check slow for such a special case. There are some approaches to fix this issue: (1) make NaN == NaN, which is consistent but unnatural, (2) change rb_equal() not to check equal?, which will cause performance degradation, (3) change rb_equal() to handle T_FLOAT specially, which will also cause performance degradation, and (4) do nothing. ... I decide that the comparison of NaN and NaN is undefined in Ruby.

#13 - 10/31/2017 12:23 PM - Eregon (Benoit Daloze)

mame (Yusuke Endoh): Thank you for the summary, that's very helpful!

#14 - 11/01/2017 08:56 AM - Hanmac (Hans Mackowiak)

Eregon (Benoit Daloze)

checkout the object id Float::NAN.object_id != (0.0/0).object_id
while NAN is a constant, (0.0/0) returns a new object each time

thats why your Array compare shows a difference