Ruby master - Feature #11286

[PATCH] Add case equality arity to Enumerable's sequence predicates.

06/19/2015 07:11 AM - 0x0dea (D.E. Akers)

Status:	Closed
Priority:	Normal
Assignee:	matz (Yukihiro Matsumoto)
Target version:	2.5

Description

Proposal

It is proposed that Enumerable's sequence predicates (#all?, #any?, #none?, and #one?) be augmented to return, in the case of a single argument, whether their query holds when each element is supplied to the argument's #=== method.

Rationale

Enumerable#grep filters by case equality, allowing us to write very natural and expressive code:

```
strs.select { |str| /foo/ === str }
strs.grep(/foo/)

nums.select { |num| (5..10) === num }
nums.grep(5..10)
```

In addition to taking advantage of the versatility of case equality, it lets us do away with the syntactic noise incurred by opening a block. #grep is a very nice method! Let's make #all? and friends more like #grep.

Related issues:

Related to Ruby master - Feature #13067: TrueClass,FalseClass to provide `===...	**Closed**
Related to Ruby master - Feature #14197: `Enumerable#{select,reject}` accept ...	**Open**

Associated revisions

Revision 61098 - 12/10/2017 10:36 PM - marcandre (Marc-Andre Lafortune)

Add case equality arity to Enumerable#all?, any?, none? and one?,
and specialized Array#any? and Hash#any?
Based on patch by D.E. Akers [#11286]

Revision 61098 - 12/10/2017 10:36 PM - marcandre (Marc-Andre Lafortune)

Add case equality arity to Enumerable#all?, any?, none? and one?,
and specialized Array#any? and Hash#any?
Based on patch by D.E. Akers [#11286]

Revision 61098 - 12/10/2017 10:36 PM - marcandre (Marc-Andre Lafortune)

Add case equality arity to Enumerable#all?, any?, none? and one?,
and specialized Array#any? and Hash#any?
Based on patch by D.E. Akers [#11286]

History

#1 - 06/19/2015 08:48 AM - zenspider (Ryan Davis)

I rather like this proposal. I hope it gets accepted.

#2 - 06/19/2015 02:22 PM - marcandre (Marc-Andre Lafortune)

- Assignee set to matz (Yukihiro Matsumoto)

I agree. Assigning to Matz

#3 - 06/19/2015 09:54 PM - nobu (Nobuyoshi Nakada)

```
+     struct MEMO *memo = MEMO_NEW(Qtrue, *argv, 0);
+     rb_check_arity(argc, 0, 1);
```

Why dereference argv before checking argc.

#4 - 06/19/2015 10:14 PM - 0x0dea (D.E. Akers)

- File case_equality_sequence_predicates-check_argc_before_deref.patch added

Nobuyoshi Nakada wrote:

```
+     struct MEMO *memo = MEMO_NEW(Qtrue, *argv, 0);
+     rb_check_arity(argc, 0, 1);
```

> Why dereference argv before checking argc.

I assumed *argv would resolve to Qnil in the case of no arguments, but this is indeed not the case. I have attached a modified patch which takes this into account.

#5 - 06/20/2015 12:12 AM - nobu (Nobuyoshi Nakada)

D.E. Akers wrote:

> I assumed *argv would resolve to Qnil in the case of no arguments, but this is indeed not the case.

Nobody guarantees it.

And argument check should be before making a memo object, I think.

```
diff --git a/enum.c b/enum.c
index c5b9d77..d6f11e5 100644
--- a/enum.c
+++ b/enum.c
@@ -1045,6 +1045,8 @@ enum_sort_by(VALUE obj)

 #define ENUMFUNC(name, argc) argc ? name##_eqq : rb_block_given_p() ? name##_iter_i : name##_i

+#define MEMO_ENUM_NEW(v1) (rb_check_arity(argc, 0, 1), MEMO_NEW((v1), (argc ? *argv : 0), 0))
+
 #define DEFINE_ENUMFUNCS(name) \
 static VALUE enum_##name##_func(VALUE result, struct MEMO *memo); \
 \
@@ -1104,8 +1106,7 @@ DEFINE_ENUMFUNCS(all)
 static VALUE
 enum_all(int argc, VALUE *argv, VALUE obj)
 {
-    struct MEMO *memo = MEMO_NEW(Qtrue, argc ? *argv : 0, 0);
-    rb_check_arity(argc, 0, 1);
+    struct MEMO *memo = MEMO_ENUM_NEW(Qtrue);
     rb_block_call(obj, id_each, 0, 0, ENUMFUNC(all, argc), (VALUE)memo);
     return memo->v1;
 }
@@ -1145,8 +1146,7 @@ DEFINE_ENUMFUNCS(any)
 static VALUE
 enum_any(int argc, VALUE *argv, VALUE obj)
 {
-    struct MEMO *memo = MEMO_NEW(Qfalse, argc ? *argv : 0, 0);
-    rb_check_arity(argc, 0, 1);
+    struct MEMO *memo = MEMO_ENUM_NEW(Qfalse);
     rb_block_call(obj, id_each, 0, 0, ENUMFUNC(any, argc), (VALUE)memo);
     return memo->v1;
 }
@@ -1438,8 +1438,7 @@ DEFINE_ENUMFUNCS(none)
 static VALUE
 enum_none(int argc, VALUE *argv, VALUE obj)
 {
-    struct MEMO *memo = MEMO_NEW(Qtrue, argc ? *argv : 0, 0);
-    rb_check_arity(argc, 0, 1);
+    struct MEMO *memo = MEMO_ENUM_NEW(Qtrue);
     rb_block_call(obj, id_each, 0, 0, ENUMFUNC(none, argc), (VALUE)memo);
     return memo->v1;
```

```
}
```

#6 - 06/20/2015 01:13 AM - 0x0dea (D.E. Akers)

- File case_equality_sequence_predicates-all_updates.patch added

I knew the call to rb_check_arity() should come first, but your very nice solution to doing so without mixing declarations and code did not occur to me. The attached contains all of the improvements discussed so far, but maybe there are more to be found? Is there something cleaner than passing argc to ENUMFUNC()?

#7 - 07/16/2015 05:30 PM - 0x0dea (D.E. Akers)

matz, would you mind voicing your opinion of this proposal?

#8 - 11/05/2015 09:53 PM - shan (Shannon Skipper)

This seems quite nice. Any downsides?

#9 - 11/05/2015 10:09 PM - 0x0dea (D.E. Akers)

I've come to realize that it might not be entirely clear what is being proposed. In essence, all of the following examples feel very "Ruby" and should, in my opinion, Just Work.

```
[1, 3.14, 2ri].all?(Numeric) # => true

if should_be_all_symbols.any?(String)
  ...
end

some_strings.none?(/aeiou/i)

lotto.all?(1..49) && lotto.one?(43..49)
```

#10 - 10/06/2016 12:22 AM - zenspider (Ryan Davis)

Why is this still open a year later?

Matz? A ruling?

#11 - 10/06/2016 09:27 PM - shevegen (Robert A. Heiler)

I like this one in particular:

```
some_strings.none?(/aeiou/i)
```

I am not so sure about:

```
nums.grep(5..10)
```

But I have no strong feelings either way. I do disagree with the statement "syntactic noise incurred by opening a block" though - blocks feel very natural in ruby to me. I guess the main part I would agree is that the .grep(range) variant is shorter to write than the block variant. This also reminds me of other proposals about allowing arguments to e. g. .map(&:chomp) invocations - I use this quite a lot myself but the & there still "feels" sort of strange.

#12 - 12/21/2016 02:56 PM - shyouhei (Shyouhei Urabe)

We looked at this issue in today's developer meeting.

Attendees were positive about the proposed functionality. But Matz wanted separate method(s) than to extend all? etc, like we have separate select and grep methods.

#13 - 12/21/2016 04:03 PM - matz (Yukihiro Matsumoto)

I am positive about the idea too. But as a duty of the final decision maker, I have to consider every option before the judgment. I see a few additional options.

- takes optional argument that takes === (proposed)
- takes optional keyword argument, match: for example
- introduces another set of methods with different names

Right now, I am wandering between option 1 and 2.

Matz.

#14 - 12/24/2016 05:08 PM - matz (Yukihiro Matsumoto)

- Related to Feature #13067: TrueClass,FalseClass to provide `===` to match truthy/falsy values. added

#15 - 11/28/2017 04:26 AM - marcandre (Marc-Andre Lafortune)

matz (Yukihiro Matsumoto) wrote:

> I am positive about the idea too. But as a duty of the final decision maker, I have to consider every option before the judgment. I see a few additional options.
>
> - takes optional argument that takes === (proposed)
> - takes optional keyword argument, match: for example
> - introduces another set of methods with different names
>
> Right now, I am wandering between option 1 and 2.

I hope a decision is made on this.

While I love keyword arguments in general, I am not convinced they are needed here and I would favor solution #1.

#16 - 11/29/2017 09:37 AM - matz (Yukihiro Matsumoto)

OK, I made a decision. I took option 1 above as proposed.
We can do as following:

```
[1, 3.14, 2ri].all?(Numeric) # => true

if should_be_all_symbols.any?(String)
  ...
end

some_strings.none?(/aeiou/i)
```

Matz.

#17 - 12/10/2017 10:41 PM - marcandre (Marc-Andre Lafortune)

- Target version set to 2.5

- Status changed from Open to Closed

Thank you Matz

D.E. Akers: thanks for the patch! I adapted it for better handling of enumerables yielding multiple arguments, and also changing Hash#any? / Array#any? which have their own versions.

Merged.

#18 - 12/18/2017 02:12 PM - znz (Kazuhiro NISHIYAMA)

- Related to Feature #14197: `Enumerable#{select,reject}` accept a pattern argument added

Files

0001-enum.c-add-case-equality-arity-to-sequence-predicates.patch	10 KB	06/19/2015	0x0dea (D.E. Akers)
case_equality_sequence_predicates-check_argc_before_deref.patch	10 KB	06/19/2015	0x0dea (D.E. Akers)
case_equality_sequence_predicates-all_updates.patch	9.94 KB	06/20/2015	0x0dea (D.E. Akers)