Ruby master - Feature #13067

TrueClass,FalseClass to provide `===` to match truthy/falsy values.

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

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

Description

I propose to make TrueClass, FalseClass to provide === method to match truthy values (TrueClass), and falsy values (FalseClass), so that we can use true and false for case pattern matching. And we can pick truthy values using grep e.g. ary.grep(true).

Matz.

Related issues:

Related to Ruby master - Feature #11286: [PATCH] Add case equality arity to E...	**Closed**

History

#1 - 12/24/2016 05:08 PM - matz (Yukihiro Matsumoto)

- Related to Feature #11286: [PATCH] Add case equality arity to Enumerable's sequence predicates. added

#2 - 12/25/2016 01:22 AM - nobu (Nobuyoshi Nakada)

It affects case/when clause too, and breaks optparse and rubygems at least.

```
diff --git i/lib/optparse.rb w/lib/optparse.rb
index afeff80740..216dd38732 100644
--- i/lib/optparse.rb
+++ w/lib/optparse.rb
@@ -394,9 +394,9 @@
 #
 class OptionParser
   # :stopdoc:
-  NoArgument = [NO_ARGUMENT = :NONE, nil].freeze
-  RequiredArgument = [REQUIRED_ARGUMENT = :REQUIRED, true].freeze
-  OptionalArgument = [OPTIONAL_ARGUMENT = :OPTIONAL, false].freeze
+  NoArgument = [NO_ARGUMENT = :NONE, NilClass].freeze
+  RequiredArgument = [REQUIRED_ARGUMENT = :REQUIRED, TrueClass].freeze
+  OptionalArgument = [OPTIONAL_ARGUMENT = :OPTIONAL, FalseClass].freeze
   # :startdoc:

   #
diff --git i/object.c w/object.c
index 5f0055fb5a..c566bc57c5 100644
--- i/object.c
+++ w/object.c
@@ -1260,6 +1260,7 @@ true_to_s(VALUE obj)
 /*
  *  call-seq:
  *     true & obj    -> true or false
+ *     true === obj  -> true or false
  *
  *  And---Returns <code>false</code> if <i>obj</i> is
  *  <code>nil</code> or <code>false</code>, <code>true</code> otherwise.
@@ -1297,6 +1298,7 @@ true_or(VALUE obj, VALUE obj2)
 /*
  *  call-seq:
  *     true ^ obj   -> !obj
+ *     false === obj -> !obj
  *
  *  Exclusive Or---Returns <code>true</code> if <i>obj</i> is
  *  <code>nil</code> or <code>false</code>, <code>false</code>
@@ -3603,7 +3605,7 @@ InitVM_Object(void)
     rb_define_method(rb_cTrueClass, "&", true_and, 1);
     rb_define_method(rb_cTrueClass, "|", true_or, 1);
     rb_define_method(rb_cTrueClass, "^", true_xor, 1);
-    rb_define_method(rb_cTrueClass, "===", rb_equal, 1);
```

```
+    rb_define_method(rb_cTrueClass, "===", true_and, 1);
     rb_undef_alloc_func(rb_cTrueClass);
     rb_undef_method(CLASS_OF(rb_cTrueClass), "new");
     /*
@@ -3618,7 +3620,7 @@ InitVM_Object(void)
     rb_define_method(rb_cFalseClass, "&", false_and, 1);
     rb_define_method(rb_cFalseClass, "|", false_or, 1);
     rb_define_method(rb_cFalseClass, "^", false_xor, 1);
-    rb_define_method(rb_cFalseClass, "===", rb_equal, 1);
+    rb_define_method(rb_cFalseClass, "===", true_xor, 1);
     rb_undef_alloc_func(rb_cFalseClass);
     rb_undef_method(CLASS_OF(rb_cFalseClass), "new");
     /*
```

#3 - 12/27/2016 03:55 PM - shevegen (Robert A. Heiler)

\o/

#4 - 12/28/2016 04:05 AM - matz (Yukihiro Matsumoto)

Nobu, I think the benefit outpass the drawbacks.

Regarding your patch, I think === should be its own RDoc entries.

Matz.

#5 - 12/28/2016 07:54 PM - marcandre (Marc-Andre Lafortune)

- Assignee set to matz (Yukihiro Matsumoto)

1. This would break a lot of code. As an example, I found 13 instances of when true/false in Rails' code that would break.
2. There is no good replacement for the current uses of when true and when false. In particular, when true would basically require separate if.

```
# From rails/actionpack/lib/action_dispatch/middleware/ssl.rb
# Current:
def normalize_hsts_options(options)
  case options
  when false
    # ...
  when nil, true
    # ...
  else
    # ...
  end
end

# Ugly rewrite necessary:
def normalize_hsts_options(options)
  if options == true
    # ...
  else
    case options
    when nil
      # repeated code from `options == true` above...
    when false  # `nil` case must be already taken care of
      # ...
    else
      # ...
    end
end
```

3. This makes when true almost meaningless and very misleading. when false is not useful as it can already be written easily with when false, nil
4. Increases the confusion of true vs "truthy" and false vs "falsey"
5. More importantly, I can not think of a single valid usecase.

ary.grep(true) was mentioned. I very much doubt there's much need to do that. We often want to exclude nil from a list (that's why we have compact), but excluding both nil and false seem odd, like the values were not computed properly. If there was such a need: ary.select(&:itself) works fine and is clear and concise enough.

ary.grep(false) seems completely meaningless to me (what can be the use of an array of nil and false values?) but if it isn't, ary.reject(&:itself) works too.

In short, I find the proposal both completely useless and confusing. In case it is somehow deemed necessary, then please define TRUTHY and FALSEY instead of changing the definitions that have been valid for 20 years.

#6 - 12/31/2016 04:37 PM - shevegen (Robert A. Heiler)

> 4) Increases the confusion of true vs "truthy" and false vs "falsey"

I think that the words "true" and "false" are a lot easier to understand than "truthy" and "falsy".

There should be some document that states the valid values for these in ruby.

Off the top of my head, true being everything except for:

- nil
- false

Perhaps I missed something, but that seems to be like a very small list to remember for what is false? I am confused about the confusion comment. I guess the part about code that breaks is a valid concern though. There is a long way to go for ruby 3.x anyway. ;)

#7 - 01/02/2017 12:16 AM - snood1205 (Eli Sadoff)

I think that a possible middle ground replacement would be to introduce truthy and falsy constants into TrueClass and FalseClass respectively. You could then do, for example,

```
case 1
when true then puts 'This will not match'
when TrueClass::TRUTHY puts 'But this will'
end
```

#8 - 01/03/2017 03:10 PM - sawa (Tsuyoshi Sawada)

Marc-Andre Lafortune wrote:

> 2) There is no good replacement for the current uses of when true and when false. In particular, when true would basically require separate if.

```
# From rails/actionpack/lib/action_dispatch/middleware/ssl.rb
# Current:
def normalize_hsts_options(options)
  case options
  when false
    # ...
  when nil, true
    # ...
  else
    # ...
  end
end
```

```
# Ugly rewrite necessary:
def normalize_hsts_options(options)
  if options == true
    # ...
  else
    case options
    when nil
      # repeated code from `options == true` above...
    when false  # `nil` case must be already taken care of
      # ...
    else
      # ...
    end
end
```

I think there is an alternative.

```
case options
when FalseClass
```

```
  ...
when NilClass, TrueClass
  ...
...
end
```

which is actually my preferred way of writing.

In fact, I had often felt there is redundancy, or unnecessary freedom, between writing when true etc. and when TrueClass etc., and had to wonder which one to use (before I came to the conclusion to use the latter). If they would mean different things, as would be the case if this proposal is implemented, then that would be a desired move, I think.

#9 - 01/03/2017 06:24 PM - tenderlovemaking (Aaron Patterson)

IMO the backwards incompatibility risks outweigh the rewards.

As Marc-Andre says, array.grep(true) and array.grep(false) could be replaced with array.select(&:itself) and array.reject(&:itself). Since case / when just sends ===, we can use a proc like this:

```
def foo val
  case val
  when true
    'true value'
  when 1
    'one'
  when :itself.to_proc
    'truthy'
  else
    'falsy'
  end
end

p foo(nil)          # falsy
p foo(false)        # falsy
p foo(true)         # true value
p foo(Object.new)   # truthy
p foo(1)            # one
```

The :itself.to_proc doesn't look so great as an alternative, but maybe we could make case / when support &:itself syntax.

#10 - 01/03/2017 08:44 PM - marcandre (Marc-Andre Lafortune)

While :itself.to_proc works, I still can't see when one would need it, and it's much easier and clearer to check for nil and false

```
def foo val
  case val
  when true
    'true value'
  when 1
    'one'
  when nil, false
    'falsy'
  else
    'truthy'
  end
end
```

#11 - 01/04/2017 07:23 AM - nobu (Nobuyoshi Nakada)

Aaron Patterson wrote:

> The :itself.to_proc doesn't look so great as an alternative, but maybe we could make case / when support &:itself syntax.

case / when isn't the original concern, but just an unexpected side effect.
Everybody can write that foo method, and select(&method(:foo)) and so on, but it won't look nice.

#12 - 01/19/2017 09:22 AM - matz (Yukihiro Matsumoto)

- Status changed from Open to Closed

Compatibility issue was bigger than I expected. I withdraw this proposal.

Matz.