**Ruby master - Feature #6023**
Add "a ?= 2" support for meaning "a = a.nil? ? 2 : a"

02/15/2012 12:31 AM - rosenfeld (Rodrigo Rosenfeld Rosas)

<table>
<thead>
<tr>
<th>Status:</th>
<th>Rejected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td>matz (Yukihiro Matsumoto)</td>
</tr>
<tr>
<td>Target version:</td>
<td>2.6</td>
</tr>
</tbody>
</table>

**Description**

I've just proposed this idea to Groovy and I thought the same semantics would be interesting to have in Ruby too:

[http://jira.codehaus.org/browse/GROOVY-5306](http://jira.codehaus.org/browse/GROOVY-5306)

This is a minor, but important, difference to the "a ||= 2" syntax.

This would be a caching/memoization operator, and it would allow code like this:

```ruby
a = nil
a ?= false # a is false now
a ?= true  # a is still false
```

This contrasts with

```ruby
a = nil
a ||= false # a is false now
a ||= true  # a is true now
```

**Related issues:**

Has duplicate Ruby master - Feature #6561: ?= operator

<table>
<thead>
<tr>
<th>Related issue</th>
<th>Status</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rejected</td>
<td>06/09/2012</td>
</tr>
</tbody>
</table>

**History**

#1 - 02/20/2012 05:53 AM - judofyr (Magnus Holm)

I've just proposed this idea to Groovy and I thought the same semantics would be interesting to have in Ruby too:

[http://jira.codehaus.org/browse/GROOVY-5306](http://jira.codehaus.org/browse/GROOVY-5306)

This is a minor, but important, difference to the "a ||= 2" syntax.

This would be a caching/memoization operator, and it would allow code like this:

```ruby
a = nil
a ?= false # a is false now
a ?= true  # a is still false
```

This contrasts with

```ruby
a = nil
a ||= false # a is false now
a ||= true  # a is true now
```

If we want something like this, we should provide a non-assignment version too. Perl uses // for the same purpose:

```perl
sub foo {
  my ($foo, %options) = @_;  
  $foo //= 1;  
  my $bar = $options{bar} //= 2;  
  return ($foo, $bar)
}
```

foo(undef, bar => undef)  # => (1, 2)
foo(0, bar => 0)  # => (0, 0)  (0 is false in Perl)
Although I suspect we rather want to use // for float/exact-division in the future.

#2 - 02/20/2012 06:53 AM - Anonymous
Magnus Holm judolyr@gmail.com writes:

If we want something like this, we should provide a non-assignment version too. Perl uses // for the same purpose:

```perl
sub foo {
    my ($foo, %options) = @_;  
    $foo //= 1;  
    my $bar = $options{bar} // 2;  
    return ($foo, $bar)
}
foo(undef, bar => undef)  # => (1, 2)  
foo(0, bar => 0)          # => (0, 0)  (0 is false in Perl)  

Although I suspect we rather want to use // for float/exact-division in the future.

How about ‘??’?
```

#3 - 03/18/2012 06:46 PM - shyouhei (Shyouhei Urabe)
- Status changed from Open to Assigned

#4 - 06/12/2012 10:22 PM - shevegen (Robert A. Heiler)
The perl example is not very elegant.
When I see code like this:

```perl
$foo // 1;  
```

I first think that someone wants to divide via / somehow.

The:

```perl
a ?? true  # a is still false  
```

looks a bit weird. Was ? not used to get the ASCII value of characters before?
I also rarely see x = x ?? y and it reminds me of ternary operator.

#5 - 06/13/2012 09:31 PM - slayer (Vlad Moskovets)
I think #6561 more convenient to store nil'able and false'able items

#6 - 10/25/2012 07:38 PM - yhara (Yutaka HARA)
- Target version changed from 2.0.0 to 2.6

#7 - 10/26/2012 11:59 AM - nobu (Nobuyoshi Nakada)
- Status changed from Assigned to Rejected