**Ruby master - Bug #16343**

**Inconsistent behavior of 'circular argument reference' error**

11/11/2019 06:28 PM - ibylich (Ilya Bylich)

<table>
<thead>
<tr>
<th>Status:</th>
<th>Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td></td>
</tr>
<tr>
<td>Target version:</td>
<td></td>
</tr>
<tr>
<td>ruby -v:</td>
<td>ruby 2.7.0dev (2019-11-04T16:32:26Z trunk a087e027bf) [x86_64-darwin19]</td>
</tr>
<tr>
<td>Backport:</td>
<td>2.5: UNKNOWN, 2.6: UNKNOWN</td>
</tr>
</tbody>
</table>

**Description**

The following code gives a syntax error:

```ruby
m { |foo = proc { || foo }|  }  
# => SyntaxError (-e:14: circular argument reference - foo)
```

But this code is valid:

```ruby
m { |foo = proc { |bar| foo }|  }  
```

Is it intentional? foo becomes a proc that returns itself, so there must be a circular dependency.

**Associated revisions**

Revision ed90ec3e - 11/12/2019 12:40 AM - nobu (Nobuyoshi Nakada)

Clear current argument name at empty block argument [Bug #16343]

**History**

**#1 - 11/11/2019 06:53 PM - jeremyevans0 (Jeremy Evans)**

On Ruby 2.2-2.6, there is a warning instead of an error in the first case, but I think that is invalid. The reason for the warning/error is to complain about this code:

```ruby
def a(b=b)
  b
end
```

Here, a would return nil, because of how the default argument values are handled. The reason the warning was added is because in Ruby 2.1 and below, calling a with no arguments would call the b method to get the value for the b argument.

```ruby
m { |foo = proc { || foo }|  }  is not a case I think we should warn about, since it doesn't have the same issue.  m { |foo = proc { || foo }.call|  } could have the same issue, but it seems unlikely to occur in real world code.
```

**#2 - 11/12/2019 12:43 AM - nobu (Nobuyoshi Nakada)**

- Status changed from Open to Closed

Applied in changeset gitied90e3e0d8f77b956b40ba36d3bce0e706e6.

Clear current argument name at empty block argument [Bug #16343]