Ruby master - Bug #16908
Strange behaviour of Hash#shift when used with `default_proc`.
05/23/2020 02:15 AM - ioquatix (Samuel Williams)

<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>jeremyevans0 (Jeremy Evans)</td>
</tr>
<tr>
<td>Target version:</td>
<td>ruby -v: 2.7.0</td>
</tr>
<tr>
<td>Backport:</td>
<td>2.5: UNKNOWN, 2.6: UNKNOWN, 2.7: UNKNOWN</td>
</tr>
</tbody>
</table>

**Description**

I don't have any strong opinion about this, but I observed the following behaviour which I thought was confusing. Maybe it's okay, or maybe we should change it to be more consistent.

```ruby
hash = Hash.new{|k,v| k[v] = 0}
hash.shift # => 0
hash.shift # => [nil, 0]
```

My feeling was, both cases should return [nil, 0].

**Associated revisions**

Revision a93cc3e2 - 01/14/2022 08:17 PM - jeremyevans (Jeremy Evans)
Make Hash#shift return nil for empty hash
Fixes [Bug #16908]

**History**

#1 - 05/23/2020 02:26 AM - jeremyevans0 (Jeremy Evans)

While your particular example is non-intuitive, there is a simple explanation for it. The first time hash.shift is called, hash is empty, so it returns the default value (0). It gets the default value by calling the default_proc for hash with a nil key. There is no better option since hash.shift isn't provided a key. The second time hash.shift is called, the hash is not empty, so it returns the first entry as a key value pair. I agree Hash#shift semantics with a default_proc are questionable, but I'm not sure if it could be improved.

I don't think we should change this behavior. It is expected that Hash.new.shift should return nil, as should Hash.new(nil).shift and Hash.new{}.shift.

hash.shift is used in conditionals:

```ruby
hash = {a: 1, b: 2}
while (k,v = hash.shift)
  p [k, v]
end
```

If you change Hash#shift to return an array when the hash is empty, you've turned this into an infinite loop.

#2 - 05/23/2020 02:38 AM - ioquatix (Samuel Williams)

jeremyevans0 (Jeremy Evans) I agree with your assessment, however that hash does not have default_proc so I assume that it would return nil after all key-value pairs are shifted out. That's different from invoking default_proc and returning [nil, 0].

#3 - 05/23/2020 02:53 AM - jeremyevans0 (Jeremy Evans)

ioquatix (Samuel Williams) wrote in #note-2:

jeremyevans0 (Jeremy Evans) I agree with your assessment, however that hash does not have default_proc so I assume that it would return nil after all key-value pairs are shifted out. That's different from invoking default_proc and returning [nil, 0].

Same principle applies when using a default_proc:

```ruby
# simple hash with indifferent access
hash = Hash.new{|h,k| h[k.to_s] unless k.is_a? String}
hash['a'] = 1
hash['b'] = 2
```
while (k, v = hash.shift)
  p [k, v]
end

#4 - 05/23/2020 12:37 PM - Eregon (Benoit Daloze)
Maybe Hash#shift should not call the default_proc or use Hash#default?
I.e., it would always return nil if Hash#empty?.
I think that would be more intuitive and probably compatible enough.

#5 - 05/23/2020 12:39 PM - Eregon (Benoit Daloze)
In other words, the current semantics of return hash.default(nil) if hash.empty? feel hacky and actually harmful to me.
The user probably never expects to have the default_proc called with a nil key in many cases.

#6 - 05/24/2020 12:53 AM - Dan0042 (Daniel DeLorme)
I would expect Hash#shift to return either a key-value tuple or nil. Returning the default value is, honestly, incomprehensible.

#7 - 03/17/2021 05:31 AM - matz (Yukihiro Matsumoto)
I don't remember why I made this behavior. Now I think #shift should return nil for an empty hash, without calling its default value, in the long run. naruse (Yui NARUSE) claims the change should be postponed to 3.2 or later if we make the change.

Matz.

#8 - 09/27/2021 05:37 AM - ioquatix (Samuel Williams)
Should we introduce some kind of deprecation or warning in 3.1?

#9 - 12/03/2021 02:23 AM - naruse (Yui NARUSE)
ioquatix (Samuel Williams) wrote in #note-8:

Should we introduce some kind of deprecation or warning in 3.1?

Not allowed. Ruby 3.1 shouldn't introduce anything which requests application developers to change something in their application code. Deprecation warning is just a way of communications to request application developers to fix their application code.

#10 - 12/27/2021 10:46 PM - jeremyevans0 (Jeremy Evans)
I've submitted a pull request to make Hash#shift return nil if the hash is empty: https://github.com/ruby/ruby/pull/5360

Not sure if we want that behavior in 3.2, or if we want to issue a deprecation warning in 3.2 and change in 3.3. Considering there would be no way to avoid the deprecation warning if the hash has a default value, I think it's best to just change the behavior without deprecation.

#11 - 01/01/2022 07:12 AM - ioquatix (Samuel Williams)
Pretty much anything will be better than the current behaviour. I think your proposal makes sense.

#12 - 01/14/2022 03:01 AM - mame (Yusuke Endoh)
  - Assignee set to jeremyevans0 (Jeremy Evans)
  - Status changed from Open to Assigned

Jeremy's approach (make Hash#shift return nil if the hash is empty) was approved by matz (Yukihiro Matsumoto).

#13 - 01/14/2022 08:18 PM - jeremyevans (Jeremy Evans)
  - Status changed from Assigned to Closed

Applied in changeset git@93cc3e23b4044762e80820fc7a45606587e11db.

Make Hash#shift return nil for empty hash

Fixes [Bug #16908]

#14 - 01/15/2022 01:39 AM - ioquatix (Samuel Williams)
Thank you jeremyevans0 (Jeremy Evans)