Ruby - Feature #6869

Do not treat `_` parameter exceptionally

08/15/2012 06:50 AM - alexeymuranov (Alexey Muranov)

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

Description

I started by commenting on #6693, but i have realized that this is a slightly different request.

I propose to not treat the variable name "_" exceptionally. Current behavior:

```
{0=>1}.each_with_index { |_,_| p _ } # [0, 1]
```

prints "[0, 1]", but

```
{1=>2}.each_with_index { |x,x| p x } # SyntaxError: (eval):2: duplicated argument name
```

raises "SyntaxError: (eval):2: duplicated argument name".

Similarly for methods:

```
def f(_, _)
  _
end
f(0, 1) # => 0

def f(x, x)
  x
end # => SyntaxError: (eval):2: duplicated argument name
```

Observe also that the use of repeated _ parameter is not consistent between methods and blocks: for methods the value is the first assigned value, and for blocks it is the array of all the assigned values.

1. I propose to use the same rule for all variables, without distinguishing _ specially.

In particular i propose to allow to repeat any variable, not only _, in block or method arguments without raising an error.

There may be several solutions what the repeated argument will hold: it may hold the array of all assigned values, the first assigned value, the last assigned value, the first non-nil assigned value, or the last non-nil assigned value.

1. I propose to treat repeated arguments in methods and in blocks the same way (do not know which one).

2. For unused variables i propose to introduce a special placeholder, for example "-" not followed by anything other than a delimiter (comma or bracket):

```
each_with_index { |-, value| puts value }

-, -, suffix = parse(name)
```

History

#1 - 08/15/2012 10:59 AM - drbrain (Eric Hodel)

- Category set to core

- Assignee set to matz (Yukihiro Matsumoto)

Seems to be part of variable shadowing checks. The check was added before r8857 (which was a refactor of the feature) and checking for '_' was removed in r14186.

Since it was committed by matz I think your chances at acceptance are low.

#2 - 08/15/2012 01:03 PM - marcandre (Marc-Andre Lafortune)

Hi,

alexeymuranov (Alexey Muranov) wrote:

> I propose to not treat the variable name "_" exceptionally.

Sorry for the naive question, but why? What are you trying to achieve? What real world problem do you want to fix?

> 1. For unused variables i propose to introduce a special placeholder

I feel that unused variables do not warrant a change to the already complex Ruby syntax.

#3 - 08/15/2012 04:23 PM - alexeymuranov (Alexey Muranov)

marcandre (Marc-Andre Lafortune) wrote:

> Hi,
>
> alexeymuranov (Alexey Muranov) wrote:
>
> > I propose to not treat the variable name "_" exceptionally.
>
> Sorry for the naive question, but why? What are you trying to achieve? What real world problem do you want to fix?

I do not like exceptions. When i was first learning Ruby, i thought that the underscore is a letter like any other, but sometimes it behaves like any other, and sometimes not.

It also seems to me more natural to use a placeholder for a discarded value than to assign it to a variable first and then discard.

> > 1. For unused variables i propose to introduce a special placeholder
>
> I feel that unused variables do not warrant a change to the already complex Ruby syntax.

In my opinion, treating variables differently based on their names is also a part of syntax, and in my opinion such rules are harder to follow than a rule for a single placeholder. As there is no dedicated placeholder in Ruby now, this one may be adapted later to other situations as well, i think.

Update: The most important real world problem this addresses is reading the code! With a placeholder, it is immediately clear that the value is discarded, but with a special variable one needs to look through the code to be sure it is not used somewhere.

Plus one needs to remember currently what a repeated variable is holding in different situations.

#4 - 04/12/2014 01:05 PM - alexeymuranov (Alexey Muranov)

It looks like the use of the underscore _ as a "placeholder" is quite common in other languages ("black hole" register in Vim, "whatever" pattern that matches everything in Haskell), but there it is really a placeholder and not a variable: values "assigned" to _ cannot be retrieved.

With this in view, maybe, instead of this my proposal, the underscore can be "downgraded" to a "placeholder" (or "black hole" peudo-variable)?

#5 - 04/13/2014 05:11 AM - nobu (Nobuyoshi Nakada)

- Description updated

Alexey Muranov wrote:

> Observe also that the use of repeated _ parameter is not consistent between methods and blocks: for methods the value is the first assigned value, and for blocks it is the array of all the assigned values.

It is unrelated to _, but because of Enumerable#each_with_index.
Try:

```
{0=>1}.each_with_index {|x,y| p x} # [0, 1]
```

Alexey Muranov wrote:

> It looks like the use of the underscore _ as a "placeholder" is quite common in other languages ("black hole" register in Vim, "whatever" pattern that matches everything in Haskell), but there it is really a placeholder and not a variable: values "assigned" to _ cannot be retrieved.

Isn't it more exceptional?

#6 - 04/13/2014 08:50 AM - alexeymuranov (Alexey Muranov)

Nobuyoshi Nakada wrote:

> Alexey Muranov wrote:
>
> > Observe also that the use of repeated _ parameter is not consistent between methods and blocks: for methods the value is the first assigned value, and for blocks it is the array of all the assigned values.
>
> It is unrelated to _, but because of Enumerable#each_with_index.
> Try:
>
> ```
> {0=>1}.each_with_index {|x,y| p x} # [0, 1]
> ```

Thanks, i do not know what i was thinking.

> Alexey Muranov wrote:
>
> > It looks like the use of the underscore _ as a "placeholder" is quite common in other languages ("black hole" register in Vim, "whatever" pattern that matches everything in Haskell), but there it is really a placeholder and not a variable: values "assigned" to _ cannot be retrieved.
>
> Isn't it more exceptional?

Yes, so this proposal would need to be closed, and i would need to open a new one. When i opened this one, i did not know that the underscore was a common "placeholder" in other languages and i thought that Ruby documentation presents the underscore in identifiers roughly as equivalent to a lowercase letter (doesn't it?).

Here is a sentence from the online version of *Programming Ruby*:

> In these descriptions, lowercase letter means the characters "a" though "z", as well as "_", the underscore.

In any case, in Ruby the following works perfectly, and in my opinion this all is confusing:

```
_ = 1
p _
```

So, yes, my new proposal would be to downgrade the underscore to a placeholder, so that in something like this

```
foo do |_,x|
  # 10 lines of code
end
```

or

```
_, _, suffix = parse something
```

it would be immediately clear the values "assigned" to _ are discarded.

#7 - 06/11/2020 08:25 AM - docx (Lukas Dolezal)

Hi. This is interesting and I can see that the inconsistent treatment of _ can be confusing (I never noticed tho because I never tried to access _).

I wonder however, I always thought that _ is exactly explicitly part of syntax as "unused parameter". Am I wrong?

So if that is the case, what about going the other direction to remove the inconsistency of what value it takes, and just make it explicitly "unusable" - can we raise SyntaxError when any _ is being accessed inside of method/block?

#8 - 06/11/2020 08:28 AM - docx (Lukas Dolezal)

Re

> In these descriptions, lowercase letter means the characters "a" though "z", as well as "_", the underscore.

I think what they mean here is that you can use it inside of a variable. Probably just did not realized the single underscore case :) But that is my interpretation.

#9 - 12/10/2020 08:58 AM - naruse (Yui NARUSE)

- Target version deleted (3.0)

#10 - 04/03/2024 03:50 AM - hsbt (Hiroshi SHIBATA)

- Status changed from Open to Assigned