Ruby master - Feature #13763

Trigger "unused variable warning" for unused variables in parameter lists

07/24/2017 12:16 PM - rovf (Ronald Fischer)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

Consider the following program nowa.rb:

```
def foo(a)
end
%w(x).each {|y|}
foo(1)
z=5
```

If I syntax-check it with *ruby -cw nowa.rb* I get the following warning:

```
nowa.rb:5: warning: assigned but unused variable - z
```

Ruby complains about z, but does not complain about a and y, even though these are also variables which receive a value which never is used. I suggest to issue a warning in these cases too.

Tested with: ruby 2.3.3p222 (2016-11-21 revision 56859) [x86_64-cygwin]

History

#1 - 07/24/2017 04:25 PM - Hanmac (Hans Mackowiak)

i am against this, becauese such functions could be used as hookups too for other functions to overwrite them.

like:

```
def xyz
  do_something(temp)
end

def do_something(x)
end
```

then something else can overwrite do_something with something else and hook to the parameters

ruby does something with functions like method_defined

#2 - 07/24/2017 06:09 PM - shevegen (Robert A. Heiler)

I am indifferent, so neither pro or con. I can see both points, more warnings or "hints"
and less warnings. There may be a practical reason to not change towards this as it may
lead to many more warnings all of a sudden? I don't know, just mentioning it - I can be
wrong here of course.

I think we can all agree that most "forgotten" local variables will not have a big negative
impact - like in the above example, the world will not end if variable "z" is not used.

I think there was some other suggestion, by Jeremy Evans (or someone else), who pointed
out something in regards to ruby issuing warnings. Since I do not remember it, I can
not quote it but I think that one suggestion was to control or "fine tune" warnings,
which I think should be possible. Perhaps ruby 3.x will have a warning-system that allows
people more to customize it. A bit like rubocop, no? The default style guide of rubocop
is not very useful to me, but you can customize the styles, and you can even let rubocop
autocorrect code, which I think as an idea, is awesome.

What I think should be possible, though, is for ruby hackers to have some more control
over warnings issued in general, ideally even in a custom manner - like the above could
perhaps be specified by Ronald Fischer and stored in some file, like irbrc, or some
config parameter for your local ruby variant. And/or also as command-line switch like ...

I don't know ...

--use-excessive-warnings

or

--use-lots-of-nagging-warning-messages

or something like that. :D

Thing is that while hanmac's example is perfectly fine, we can also give counter-examples of code or accidental code or forgotten/partially refactored code where the situation by Ronald Fischer would be valid.

Last but not least, and I shall finish this - while I think in general that it would be nice for ruby to have a better, more flexible and sophisticated warning system, I am not sure if there is a huge use case or need for the above scenario detailed by Ronald Fischer.

Then again, with flexibility, it would not matter since you could easily create your own warning-triggering codes (one could even say this for errors, and then allow people to have ruby continue to work just fine, even when errors happen ... but this is perhaps too much flexibility even for ruby. I just envision some crazy schemes, all without having to use "rescue" clauses ... like embedding lisp code right into ruby and crazy stuff like this).

#3 - 11/28/2017 04:49 AM - marcandre (Marc-Andre Lafortune)

Hanmac (Hans Mackowiak) wrote:

> i am against this, becauese such functions could be used as hookups too for other functions to overwrite them.

This is particularly true for named parameters, which can not be renamed with a leading underscore.

FWIW, rubocop can detect those already, for any version of Ruby, and is customizable by the user.

I feel that it's probably best to handle such cases at a higher level than MRI itself.