Ruby master - Feature #13207

Allow keyword local variable names like `class` or `for`

02/11/2017 03:47 PM - kaspth (Kasper Timm Hansen)

Status:	Feedback	
Priority:	Normal	
Assignee:		
Target version:		

Description

Sometimes when trying to write expressive Ruby you enevitably hit a case
that would sound just right if the variable name matches a Ruby keyword.

E.g. writing a method to output HTML tags:

```
def label_tag(text, class:)
  %(<label class=""#{class.camelize}>#{text}</label>")
end
```

Or a method to generate a representation for a specific purpose:

```
def to_gid(for:)
  for ||= :universal
  GlobalID.generate(self.class.name, id, for)
end
```

Currently Ruby's keywords get in the way of the type of code we'd like to write.
Instead we have to use variable names like klass or modjule:

```
[ A::B, C::D ].each { |klass| puts klass.name }
```

It would make me a happier programmer if I could write more naturally instead of
worrying about keywords clashing with local variable names.

It also stands to me that far more often when there is a potential clash I want
the variable name to win out. It's unlikely I will be defining classes or
modules within a method say. In those rare cases where I do,
we could expose keywords like this: Keyword.class, Keyword.for etc.

I propose that renaming a keyword is fair game anywhere except the root scope and
that a rename follows local variable scoping.

NOTE: I think this could also remove the self currently needing to be
prepended to self.class.

I hope this can be yet another case in Ruby's quest to go a bit out of its way
to make programmers lives happier. Thanks!

History

#1 - 02/12/2017 12:53 PM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Feedback

Binding#local_variable_get is for that purpose.

Or do you have any concrete proposal?

#2 - 02/13/2017 06:22 AM - shevegen (Robert A. Heiler)

I don't have any pro or con on the proposal itself. I also have no idea if
this is a difficult change or whether matz likes it or not, but I wanted
to comment on just one other part.

Kasper Timm Hansen gave this example with keyword arguments:

```
def label_tag(text, class:)
  %(<label class=""#{class.camelize}>#{text}</label>")
end
```

I remember that some years ago, for my pseudo-webframework, I needed to
autogenerate HTML tags and also some CSS optionally for these tags.

I was using something like the following:

```
def h2(content = '', class = 'border_black_1px')
  # code here to generate a h2 tag with a css class called border_black_1px
end
```

To explain the above, I actually meant class here in the context of CSS,
so more a CSS class and not a ruby class.

Back then to my surprise ruby did not like it, and of course it was easy
to understand why not - ruby expects something such as "class Foo". Fair
enough.

It was not a big deal for me to actually change all the arguments there
from class to css_class instead. No big deal. So it would look like
this:

```
def h2(content = '', css_class = 'mars1em')
```

May be easier to read for others, too.

But what I am actually trying to say here is, and this is not a pro or
con opinion on the proposal itself - I actually UNDERSTAND what he is
saying, since I myself encountered a somewhat similar situation. Whether
ruby should allow for it or not is another matter, as said I have no particular
pro or con opinion here. I just wanted to mention that I think that
the example given, even if it was different from my example (since it
used keywords), appears to be a valid example in my opinion.

I have not had the same for "for" though, but for "class" I have. (I
also think that these two are slightly different from the english
language point of view itself, but I digress here.)

Thanks.