Ruby master - Feature #6739

One-line rescue statement should support specifying an exception class

07/16/2012 06:49 AM - Quintus (Marvin Gülker)

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

Description

Hi there,

When wrapping up a single line inside a begin/rescue block I feel constantly annoyed that I have to create a whole lot of bloated code just to rescue from a specific exception. For example:

```
begin
  File.read("myfile.txt")
rescue Errno::ENOENT
  puts "No file there"
end
```

Now it's possible to do this:

```
File.read("myfile.txt") rescue puts "No file there"
```

But this forces me to rescue from StandardError which is not really what I want, because it swallows exceptions I'd rather have wanted to see, e.g. if I mistyped `File.read' as `File.raed' this would be swallowed as well. I know it is possible to compress the multiline statements above into a single line by using semicolons, but it's better to avoid them as they decrease readability.

So my suggestion is to add something like the following syntax to Ruby:

```
File.read("myfile.txt") rescue Errno::ENOENT, puts "No file there"
```

This way it is more concise than having to write five lines (instead of just one) and still reads good (as opposed to the semicolon trick). Maybe the syntax isn't ideal as the comma operator is already used elsewhere, but the general idea should be clear though.

Valete,
Marvin

Related issues:

Related to Ruby master - Feature #10042: Deprecate postfix rescue syntax for ...	**Open**

History

#1 - 07/16/2012 11:45 PM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Feedback

A rescue clause (non-modifier) can be followed by arbitrary expressions which would return a class or a module,
so the syntax in your proposal is unclear.

We had discussed about arguments to rescue-modifier when adding it, but no suitable syntax was found.

#2 - 07/16/2012 11:50 PM - nobu (Nobuyoshi Nakada)

- Description updated

#3 - 07/19/2012 05:45 PM - Quintus (Marvin Gülker)

> A rescue clause (non-modifier) can be followed by arbitrary expressions which would return a class or a module, so the syntax in your proposal is unclear.

As said, I wasn't happy with the syntax either, it was more about the actual *possibility* of only rescueing specific exceptions.

> We had discussed about arguments to rescue-modifier when adding it, but no suitable syntax was found.

Maybe it could be similar to how the case statement can be used. There it is possible to do:

```
when x then y
```

So what about a syntax like this:

```
File.read("myfile.txt") rescue Errno::ENOENT then puts "No file there"
```

I.e. using the then keyword to distinguish the expression for the exception class from the actual rescue code.

Valete,
Marvin

#4 - 07/20/2012 05:23 AM - Anonymous

On Jul 19, 2012, at 3:45 AM, Quintus (Marvin Gülker) wrote:

> ```
> File.read("myfile.txt") rescue Errno::ENOENT then puts "No file there"
> ```

I like "when" and "then"

```
File.read("myfile.txt") rescue when Errno::ENOENT then puts "No file there"
```

#5 - 07/24/2012 07:41 AM - Quintus (Marvin Gülker)

> ```
> File.read("myfile.txt") rescue when Errno::ENOENT then puts "No file there"
> ```

Putting two keywords right after one another doesn’t seem like a good idea to me. If you really like this when there, what about this one:

```
File.read("myfile.txt") when Errno::ENOENT rescue puts "No file there"
```

However, this may conflict with a surrounding case statement. I’d prefer the rescue/then to this as it seems more clear to me.

#6 - 07/24/2012 07:43 AM - Quintus (Marvin Gülker)

> I’d prefer the rescue/then to this as it seems more clear to me.

Sorry, not a native speaker here... Should be "I’d prefer the rescue/then combination over this as it seems more clear to me.".

#7 - 07/24/2012 09:25 AM - nobu (Nobuyoshi Nakada)

I don't think it would be possible to tell "rescue exceptions then" from mere "rescue", unfortunately.

#8 - 07/24/2012 09:35 AM - trans (Thomas Sawyer)

What about an add-"on"?

```
File.read("myfile.txt") rescue on Errno::ENOENT puts "No file there"
```

Although better to read, maybe?

```
File.read("myfile.txt") rescue puts "No file there" on Errno::ENOENT
```

#9 - 07/25/2012 01:41 PM - nobu (Nobuyoshi Nakada)

New keyword is too big deal for this trivial syntax extension, I think.

#10 - 10/27/2012 07:22 AM - ko1 (Koichi Sasada)

- Assignee set to mame (Yusuke Endoh)

#11 - 10/27/2012 12:02 PM - mame (Yusuke Endoh)

- Target version changed from 2.0.0 to Next Major

#12 - 10/27/2012 08:09 PM - mame (Yusuke Endoh)

- Assignee changed from mame (Yusuke Endoh) to matz (Yukihiro Matsumoto)

#13 - 07/26/2014 04:22 PM - Anonymous

How about introducing shorter "resc" keyword for error-specific inline rescue?

```
do_messy_job resc TypeError: 42, NameError: 43
```

Tbh, inline rescue always leaves me with a feeling that I'm doing something underhanded. Keyword "resc" would save precious 2 characters, while still being highly unique, so it could be argued that its introduction would not be too big deal. This omnivorous inline rescue has been really pissing me off for a long time. Together with my other proposal to officially rename ArgumentError to ArgError, "resc" keyword for specific inline rescue would be a godsend.

#14 - 07/27/2014 04:12 PM - funny_falcon (Yura Sokolov)

```
do_something rescue SomeError with puts "SomeError occured"
```

#15 - 07/27/2014 07:28 PM - nobu (Nobuyoshi Nakada)

- Description updated

As far as I tried, using when can't parse well.

#16 - 10/22/2014 11:10 PM - Anonymous

@NobuyoshiNakada, how then about my resc suggestion?

#17 - 10/23/2014 05:59 AM - nobu (Nobuyoshi Nakada)

No new reserved word, as possible.
I don't think it is acceptable.

#18 - 04/28/2016 06:32 PM - Quintus (Marvin Gülker)

I noticed we have a keyword already in Ruby that's used nearly nowhere and could be a nice fit here. Might not be linguistically optimal, but is not too bad: in. That keyword is currently only used for the for loop and should thus be fairly clear to tell if not used in a for loop.

Suggestion 1:

```
File.read("myfile.txt") rescue Errno::ENOENT in puts "No file there"
```

Suggestion 2, building on top the "on" example given earlier:

```
File.read("myfile.txt") rescue puts "No file there" in Errno::ENOENT
```

A third suggestion, diggin up the when suggestion from earlier in an adapted way (like suggestion 2 above):

```
File.read("myfile.txt") rescue puts "No file there" when Errno::ENOENT
```

Using if here is probably not possible because that would clash with the postcondition modifier.

Greetings
Marvin

#19 - 05/18/2016 12:34 AM - javawizard (Alex Boyd)

What's the problem with then? I'm getting a syntax error when I try it:

```
irb(main):002:0> foo rescue Bar then baz
SyntaxError: (irb):2: syntax error, unexpected keyword_then, expecting end-of-input
foo rescue Bar then baz
                   ^
    from /Users/aboyd/.rbenv/versions/2.1.2/bin/irb:11:in `<main>'
```

I don't believe there's anywhere then can appear without something like if leading into it, in which case the then would obviously belong to the if, like:

```
foo rescue if bar then Baz else Qux end then blah
```

(which probably isn't actually advisable, but would be perfectly syntactically unambiguous.)

#20 - 05/18/2016 12:42 AM - shyouhei (Shyouhei Urabe)

We looked at it in yesterday's developer meeting. Nobu told us that all the proposed syntax so far renders conflicts to existing syntax.

My feeling is that rescue modifier is "too easy to use" compared to ordinary begin-rescue-end. People tend to inappropriately use modifier style because it's much shorter. I personally suggest a coding style to restrict its usage to contain no side effects; like expr rescue nil or expr rescue false.

#21 - 02/07/2018 01:14 AM - duerst (Martin Dürst)

- Related to Feature #10042: Deprecate postfix rescue syntax for removal in 3.0 added