Ruby master - Bug #11143

it should always be possible to return from an if

05/12/2015 05:42 PM - bughit (bug hit)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:		**Backport:**	2.5: REQUIRED, 2.6: REQUIRED

Description

```
irb(main):001:0> def foo; a = if true then return end end
SyntaxError: (irb):1: void value expression
```

it should not matter that you are not producing a value for the if expression, since you are leaving the method immediately. This should also apply to other jumps.

Related issues:

Related to Ruby master - Bug #15932: wrong "void value expression" error for ...	**Closed**

History

#1 - 05/13/2015 03:21 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Rejected

Why do you need such assignment?

#2 - 05/18/2015 05:09 PM - bughit (bug hit)

Nobuyoshi Nakada wrote:

> Why do you need such assignment?

What does my "need" have to do with whether something should be a syntax error or not?

Just about everything in ruby is an expression and all expressions can potentially fail to produce a value (due to some jump: raise, throw, return, etc). That does not make such expressions syntactically invalid (e.g. a = (1; return; 2)), except for the if? Why?

#3 - 05/18/2015 09:08 PM - bughit (bug hit)

What is the objection to providing some sort explanation when rejecting a bug report?

#4 - 05/19/2015 09:35 AM - nobu (Nobuyoshi Nakada)

bug hit wrote:

> Nobuyoshi Nakada wrote:
>
> > Why do you need such assignment?
>
> What does my "need" have to do with whether something should be a syntax error or not?

I wanted to know the use-case.

> Just about everything in ruby is an expression and all expressions can potentially fail to produce a value (due to some jump: raise, throw, return, etc). That does not make such expressions syntactically invalid (e.g. a = (1; return; 2)), except for the if? Why?

Thank you, it's a bug.
That return should be invalid too.

#5 - 05/19/2015 09:38 AM - nobu (Nobuyoshi Nakada)

- Status changed from Rejected to Open

#6 - 05/19/2015 02:15 PM - bughit (bug hit)

Nobuyoshi Nakada wrote:

> Just about everything in ruby is an expression and all expressions can potentially fail to produce a value (due to some jump: raise, throw, return, etc). That does not make such expressions syntactically invalid (e.g. a = (1; return; 2)), except for the if? Why?

Thank you, it's a bug.
That return should be invalid too.

There is nothing special about return, there are other jumps (return, break, next, redo, retry, throw, raise)

For your proposal to make sense, you would have to detect every possible jump in every possible expression whose value is not discarded, and produce a syntax error. This is crazy. You still have not stated why that's a good idea. AFAICT there is nothing wrong and in many cases useful to be able to jump out of an expression when conditions warrant, especially because in ruby just about everything is an expression.

#7 - 05/19/2015 03:34 PM - bughit (bug hit)

bug hit wrote:

> Nobuyoshi Nakada wrote:
>
> > Just about everything in ruby is an expression and all expressions can potentially fail to produce a value (due to some jump: raise, throw, return, etc). That does not make such expressions syntactically invalid (e.g. a = (1; return; 2)), except for the if? Why?
>
> Thank you, it's a bug.
> That return should be invalid too.
>
> There is nothing special about return, there are other jumps (return, break, next, redo, retry, throw, raise)
>
> For your proposal to make sense, you would have to detect every possible jump in every possible expression whose value is not discarded, and produce a syntax error. This is crazy. You still have not stated why that's a good idea. AFAICT there is nothing wrong and in many cases useful to be able to jump out of an expression when conditions warrant, especially because in ruby just about everything is an expression.

To expand on why this is crazy.

1. All the jumps (return, break, next, redo, retry, throw, raise) should be treated the same, because they all have the same effect, break out of the expression, preventing it from producing a value
2. If you forbid jumping out of expressions, you will undoubtedly break a lot of code, and for no good reason.
3. because if you are in an expression, (e.g. begin/end, if) and certain conditions arise such that it does not make sense to continue the normal flow, jumping out of it by raising or returning (or whatever jump is appropriate) is perfectly reasonable, and it makes zero sense to forbid it, and you have not even tried to justify it.

So the reasonable thing to do is not to expand this restriction, but to lift it entirely.

#8 - 07/05/2019 11:53 PM - jeremyevans0 (Jeremy Evans)

- Related to Bug #15932: wrong "void value expression" error for 'next' or 'break' statements inside an 'if' assignment added

#9 - 07/05/2019 11:54 PM - jeremyevans0 (Jeremy Evans)

- Status changed from Open to Closed

Fixed by 01b3a3804334be19d013526d3edde2b84399ae43.

#10 - 04/16/2020 09:30 PM - marcandre (Marc-Andre Lafortune)

Just got hit by this. Is this difficult to backport to supported versions?

#11 - 04/16/2020 09:34 PM - marcandre (Marc-Andre Lafortune)

- Backport changed from 2.0.0: UNKNOWN, 2.1: UNKNOWN, 2.2: UNKNOWN to 2.5: REQUIRED, 2.6: REQUIRED