Ruby trunk - Feature #12079

Loosening the condition for refinement

02/16/2016 01:29 PM - sawa (Tsuyoshi Sawada)

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

Description

There are a few non-standard ways of calling a method that cannot be used when the relevant method is a refined method:

- a symbol used with & as a block via symbol to proc
- a symbol used with send or __send__

For example, the following will fail:

```
    module Foo
      refine String
        def baz; end
      end
    end

    using Foo
    ["a", "b", "c"].map(&:baz) # => undefined method error
    "a".send(:baz) # => undefined method error
```

I would like to propose to loosen the condition for refinement so that as long as the relevant construction (such as the use of & to provoke Symbol#to_proc or calling of send or __send__) is within the valid scope of refinement, allow the call to the relevant methods.

Related issues:

Has duplicate Ruby trunk - Bug #12530: Module Refinements	**Rejected**
Is duplicate of Ruby trunk - Feature #9451: Refinements and unary & (to_proc)	**Closed**
Is duplicate of Ruby trunk - Feature #11476: Methods defined in Refinements c...	**Closed**

History

#1 - 02/20/2016 03:34 PM - sawa (Tsuyoshi Sawada)

To the list of relevant constructions, I would also like to add inject when it takes a symbol argument.

```
module Foo
  refine String do
    def baz a, b; a + b * 2 end
  end
end

using Foo
["x", "y", "z"].inject(:baz) # => undefined method error
["x", "y", "z"].inject("", :baz) # => undefined method error
```

So my generalization for the target of my proposal is: Ruby core methods/constructions in which a(nother) method is called in the form of a symbol or a string. There may be a few more of them that I have missed.

#2 - 02/21/2016 02:14 AM - sawa (Tsuyoshi Sawada)

There is a point that needs to be made clear regarding this proposal: whether the symbol or string used in the construction has to be a literal. I think there would be use cases where the symbol/string is expressed as a more complex expression:

```
module Foo
  refine String do
    def baz; end
  end
end

def a
```

```
  case some_expression
  when "x" then :baz
  when "y" then :bar
  end
end

using Foo
["a", "b", "c"].map(&(some_condition ? :baz : :bar))
"a".__send__("BAZ".downcase)
"a".send(a)
```

In order for the proposal to be useful, I think the relevant symbol/string should not be restricted to literals. Furthermore, the location where the expression is expanded to a symbol/string should not matter; solely the location of &, __send__, or send, etc. should matter. In above, while send is within the scope of refinement for String#baz, a, which can be possibly expanded to :baz, is expanded outside of the scope of refinement. In such cases too, I think the refinement should be effective.

#3 - 05/28/2016 09:55 AM - hsbt (Hiroshi SHIBATA)

- Assignee set to shugo (Shugo Maeda)

- Status changed from Open to Assigned

#4 - 06/24/2016 06:22 AM - shugo (Shugo Maeda)

- Assignee changed from shugo (Shugo Maeda) to matz (Yukihiro Matsumoto)

> I would like to propose to loosen the condition for refinement so that as long as the relevant construction (such as the use of & to provoke Symbol#to_proc or calling of send or **send**) is within the valid scope of refinement, allow the call to the relevant methods.

What do you think, Matz?

#5 - 06/29/2016 03:25 AM - shugo (Shugo Maeda)

- Has duplicate Bug #12530: Module Refinements added

#6 - 06/29/2016 03:27 AM - matz (Yukihiro Matsumoto)

I agree that would be nicer to users. My concern is performance penalty.

Matz.

#7 - 09/08/2016 06:49 AM - shugo (Shugo Maeda)

- Is duplicate of Feature #9451: Refinements and unary & (to_proc) added

#8 - 09/08/2016 06:49 AM - shugo (Shugo Maeda)

- Is duplicate of Feature #11476: Methods defined in Refinements cannot be called via send added

#9 - 09/08/2016 06:51 AM - shugo (Shugo Maeda)

- Status changed from Assigned to Rejected

This issue will be addressed by #9451 and #11476.