Ruby - Feature #10064

&:symbol in when clause

07/18/2014 05:42 PM - mrkn (Kenta Murata)

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

Description

Now we can put Proc objects in a when clause as this code.

```
require 'prime'

def test(n)
  print "#{n} is "
  case n
  when :zero?.to_proc
    puts 'zero'
  when :even?.to_proc
    puts 'an even number'
  when :prime?.to_proc
    puts 'a prime number'
  else
    puts 'a non-prime odd number'
  end
end

(1..10).each &method(:test)
```

I would like to write it as below:

```
require 'prime'

def test(n)
  print "#{n} is "
  case n
  when &:zero?
    puts 'zero'
  when &:even?
    puts 'an even number'
  when &:prime?
    puts 'a prime number'
  else
    puts 'a non-prime odd number'
  end
end

(1..10).each &method(:test)
```

How do you think about this new syntax, Matz?

History

#1 - 07/19/2014 02:41 AM - ko1 (Koichi Sasada)

just idea.

```
class Symbol
  def ~@
    self.to_proc
  end
end

case 0
```

```
when ~:zero?
  p :zero
end
```

#2 - 07/19/2014 11:11 AM - matz (Yukihiro Matsumoto)

- Status changed from Open to Feedback

The original idea includes dispatch based on type (Symbol), which is not a good idea in general.
The one proposed by ko1 is slightly better, yet I still don't love it.

Matz.

#3 - 10/01/2014 09:47 AM - sawa (Tsuyoshi Sawada)

What about allowing syntax like this:

```
->&Proc.new{|y| y.zero?} #=> Should be equivalent to ->{|x| x.zero?}
->&:zero?                #=> Should be equivalent to ->{|x| x.zero?} (Symbol#to_proc applies implicitly)
```

This would be a parallel to

```
[*["foo"]] #=> ["foo"]
[*{foo: "bar"}] #=> [[:foo, "bar"]] (Symbol#to_a applies implicitly)
```

under the understanding that what [] is to * (converse operation) is what -> is to &.

Then, we would be able to do:

```
case 0
when ->&:zero?
  p :zero
end
```

#4 - 10/01/2014 05:48 PM - marcandre (Marc-Andre Lafortune)

Yukihiro Matsumoto wrote:

> The original idea includes dispatch based on type (Symbol), which is not a good idea in general.

I'm not sure I understand. I think the original idea would work for other types too; the syntax would just be a shortcut for a call to to_proc.

For example, I'd expect the following to work too:

```
x = Object.new
def x.to_proc
  -> { true }
end

case 42
  when &x
    puts "Always true"
end
```

#5 - 01/05/2018 09:01 PM - naruse (Yui NARUSE)

- Target version deleted (2.2.0)