Ruby master - Bug #15880

Wrong precedence of the if modifier in pattern matching

05/27/2019 01:51 PM - ibylich (Ilya Bylich)

Status:	Rejected		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.7.0dev (2019-05-20 trunk ab0f2deab1) [x86_64-darwin18]	**Backport:**	2.4: UNKNOWN, 2.5: UNKNOWN, 2.6: UNKNOWN

Description

When "If" is used as an "If modifier" it runs before the expression that it wraps:

```
=> puts 1 if (puts 2; true)
2
1
```

However, when it's used in the pattern matching destructuring runs first:

```
class A
  def deconstruct
    puts 'deconstruct called'
    [1]
  end
end

p case A.new
in A[1] if (puts 'if check'; true)
  'yes'
else
  'no'
end

# prints
# deconstruct called
# if check
# "yes"
```

I personally think that it's very confusing as it doesn't reflect the code. I assumed it to not run destructuring if the check returns false (especially because destructuring is allowed to have side-effects)

History

#1 - 05/27/2019 04:37 PM - marcandre (Marc-Andre Lafortune)

- Status changed from Open to Rejected

This is necessary so the if can depend on the variables of the matching, e.g. in [Integer => x] if x.odd?

#2 - 05/28/2019 06:59 AM - duerst (Martin Dürst)

It's clear that the order of evaluation has to be the way it is currently. But in this case, if just sounds wrong, not only because of examples such as puts 1 if condition, but also because of general English.

So I think changing if to something else is highly desirable. But I unfortunately don't have any good proposal.