Ruby master - Feature #12957

A more OO way to create lambda Procs

11/18/2016 07:09 PM - justcolin (Colin Fulton)

Status:	Feedback	
Priority:	Normal	
Assignee:		
Target version:		

Description

Currently to create a lambda Proc one has to use lambda { } or -> { }. For doing metaprogramming it would be nice to have a more OO way to generate them. Something like LambdaProc.new. That way one could write:

```
class MetaThingy
  def initialize proc_class
    @anonymous_function = proc_class.new do
      # Some Code
    end
  end
end
```

and pass in either Proc or LambdaProc depending on their needs, instead of:

```
class MetaThingy
  def initialize proc_type
    @anonymous_function = case proc_type
                          when :proc
                            proc do
                              # Some Code
                            end
                          when :lambda
                            lambda do
                              # Some Code
                            end
                          end
    end
  end
end
```

This is not a common use case, but would help make the language more orthogonal.

Related issues:

Related to Ruby master - Feature #7314: Convert Proc to Lambda doesn't work i...	**Assigned**
Related to Ruby master - Feature #15973: Let Kernel#lambda always return a la...	**Closed**

History

#1 - 11/20/2016 01:38 PM - shyouhei (Shyouhei Urabe)

Problem is, when you allow LambdaProc.new, that have to accept non-lambda procs, like LambdaProc.new(&nonlambda). This way, a proc would be converted from/between lambda and non-lambda.

This is not a good idea. Right now a lambda is born to be a lambda and there is no way to turn it into a proc. If such property breaks, a programmer cannot guarantee what to expect to a block they wrote's parameter, because that proc can later be changed into a lambda by someone else. And there is no way to detect such change from that block itself before it is called.

#2 - 11/20/2016 01:41 PM - shyouhei (Shyouhei Urabe)

- Related to Feature #7314: Convert Proc to Lambda doesn't work in MRI added

#3 - 01/12/2017 05:07 PM - dsisnero (Dominic Sisneros)

I want this too

```
MyLambda = Class.new Proc
```

I want MyLambda{|x| x + 1}.lambda? to == true

I only want it when initializing the new lambdas. I want easy initialization with a block

Maybe allow Proc.new to accept true or false

```
class Proc

  def initialize(lmbda = false, &block)
       if lmbda
         __lambda__ = true
       end
       block = block
  end

end
```

then

```
class MyLambda
    def initialize(&block)
       super(true,&block)
    end
end

MyLambda.new{|x| x + 1}.lambda? == true
```

#4 - 01/19/2017 08:40 AM - akr (Akira Tanaka)

- Status changed from Open to Feedback

It is not impossible to generate MyLambda.new {} as lambda.

```
% ruby -e '
class MyLambda
end
class << MyLambda
  alias new lambda
  public :new
end
x = MyLambda.new { p :foo }
p x.lambda?
x.call
'
true
:foo
```

I'm not sure this is enough for actual usages, though.

#5 - 01/19/2017 09:00 AM - matz (Yukihiro Matsumoto)

The code above does not return a MyLamnda instance.

Matz.

#6 - 01/24/2017 08:43 PM - Eregon (Benoit Daloze)

Actually, it is possible to create a single block of code that can be proc or lambda with #send:

```
> Kernel.send(:lambda) {}.lambda?
=> true
> Kernel.send(:proc) {}.lambda?
=> false
```

I was surprised as well this worked, the Kernel#lambda method is quite magic:
https://github.com/graalvm/truffleruby/pull/12#discussion_r96889356

#7 - 07/03/2019 01:59 AM - shyouhei (Shyouhei Urabe)

- Related to Feature #15973: Let Kernel#lambda always return a lambda added