Backport191 - Backport #5807

"define_method" not consistent with "def"

12/26/2011 03:28 AM - silvioricardoc (Silvio Cordeiro)

Status:	Rejected	
Priority:	Normal	
Assignee:		

Description

=begin
Under at least Ruby 1.8.7 and 1.9.1, the following definitions are not equivalent:

def foo1
puts "foo1 called"
end

class << self
define_method(:foo2) do
puts "foo2 called"
end
end

While ((|foo1|)) always works, ((|foo2|)) is considered *undefined* in some contexts. For example, this method works:

(({
def call_foos
foo1
foo2
end
}))

but ((|Bad.new|)) below would not work, failing because ((|foo2|)) is undefined:

(({
class Bad
def initialize
foo1
foo2
end
end
}))

See attached file with an example code that is even stranger than this.
=end

History

#1 - 12/26/2011 06:47 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Rejected

=begin
These two are different things.

def foo1
puts "foo1 called"
end

class << self
define_method(:foo2) do
puts "foo2 called"
end
end

While (({foo1})) is a private instance method of (({Object})), (({foo2})) is a singleton method of (({main})).
=end

Files

bad.rb	277 Bytes	12/26/2011	silvioricardoc (Silvio Cordeiro)