Ruby master - Feature #13083

{String|Symbol}#match{?} with nil returns falsy as Regexp#match{?}

12/28/2016 06:34 PM - kachick (Kenichi Kamiya)

Status:	Open	
Priority:	Normal	
Assignee:		
Target version:		

Description

Just for consistency

- patch: https://github.com/ruby/ruby/pull/1506
- spec: https://github.com/ruby/spec/pull/380

Currently behaves as (ruby --version: ruby 2.5.0dev (2016-12-28 trunk 57228) [x86_64-darwin16])

```
'string'.__send__(:=~, nil) #=> nil
'string'.match(nil)         #=> TypeError: wrong argument type nil (expected Regexp)
'string'.match?(nil)        #=> TypeError: wrong argument type nil (expected Regexp)
:symbol.__send__(:=~, nil)  #=> nil
:symbol.match(nil)          #=> TypeError: wrong argument type nil (expected Regexp)
:symbol.match?(nil)         #=> TypeError: wrong argument type nil (expected Regexp)
/regex/.__send__(:=~, nil)  #=> nil
/regex/.match(nil)          #=> nil
/regex/.match?(nil)         #=> false
```

Expected to

```
'string'.__send__(:=~, nil) #=> nil
'string'.match(nil)         #=> nil
'string'.match?(nil)        #=> false
:symbol.__send__(:=~, nil)  #=> nil
:symbol.match(nil)          #=> nil
:symbol.match?(nil)         #=> false
/regex/.__send__(:=~, nil)  #=> nil
/regex/.match(nil)          #=> nil
/regex/.match?(nil)         #=> false
```

History

#1 - 12/29/2016 12:15 AM - nobu (Nobuyoshi Nakada)

- Description updated

#2 - 01/02/2017 12:22 AM - snood1205 (Eli Sadoff)

I think that the way it currently exists is logical. As =~ is an Object method, it is worth while keeping the more permissible behavior whereas errors should be thrown for match and match? if nil is passed. The change that I would propose would actually to have Regexp#match and Regexp#match? both raise a type error if nil is passed as an argument. The inconsistency seems to be stronger there.

#3 - 02/22/2017 07:39 AM - matz (Yukihiro Matsumoto)

Those methods (but =~) should consistently raise exceptions.

Matz.