Ruby - Feature #18788

Support passing Regexp options as String to Regexp.new

05/17/2022 12:11 PM - janosch-x (Janosch Müller)

Status:	Closed	
Priority:	Normal	
Assignee:		
Target version:		

Description

Current situation

Regexp.new takes an integer as second argument which needs to be ORed together from multiple constants:

```
Regexp.new('foo', Regexp::IGNORECASE | Regexp::MULTILINE | Regexp::EXTENDED) # => /foo/imx
```

Any other non-nil value is treated as i flag:

```
Regexp.new('foo', Object.new) # => /foo/i
```

Suggestion

Regexp.new should support passing the regexp flags not only as an Integer, but also as a String, like so:

```
Regexp.new('foo', 'i')   # => /foo/i
Regexp.new('foo', 'imx') # => /foo/imx

# edge cases
Regexp.new('foo', 'iii') # => /foo/i
Regexp.new('foo', '')    # => /foo/

# unsupported flags should probably emit a warning
Regexp.new('foo', 'jmq') # => /foo/m
Regexp.new('foo', '-m')  # => /foo/m
```

Reasons

1. The constants are a bit cumbersome to use, particularly when building the regexp from variable data:

```
def make_regexp(regexp_body, opt_string)
  opt_int = 0
  opt_int |= Regexp::IGNORECASE if opt_string.include?('i')
  opt_int |= Regexp::MULTILINE  if opt_string.include?('m')
  opt_int |= Regexp::EXTENDED   if opt_string.include?('x')

  Regexp.new(regexp_body, opt_int)
end
```

1. Passing a String is already silently accepted, and people might get the wrong impression that it works:

```
Regexp.new('foo', 'i') # => /foo/i
```

... but it doesn't really work:

```
Regexp.new('foo', 'x') # => /foo/i
```

Backwards compatibility

This change would not be fully backwards compatible.

Code that relies on the second argument being a String which does not contain "i" in order to make the Regexp case insensitive would break.

Note: originally I suggested supporting Symbols in the same way as Strings, but removed that in light of the discussion.

Associated revisions

Revision 1e9939dae24db232d6f3693630fa37a382e1a6d7 - 06/20/2022 10:35 AM - nobu (Nobuyoshi Nakada)

[Feature #18788] Support options as String to Regexp.new

Regexp.new now supports passing the regexp flags not only as an
Integer, but also as a `String. Unknown flags raise errors.

Revision 883d13dc4127b5fde617b584ebb89714eac19965 - 06/20/2022 10:35 AM - nobu (Nobuyoshi Nakada)

[Feature #18788] Spec for options as String to Regexp.new

Co-Authored-By: Janosch Müller janosch.mueller@betterplace.org

Revision 4a6facc2d683d1dbb67ded8a9f4d7cd10a9fd8ad - 06/20/2022 10:35 AM - nobu (Nobuyoshi Nakada)

[Feature #18788] [DOC] String options to Regexp.new

Co-Authored-By: Janosch Müller janosch.mueller@betterplace.org

Revision 1e9939dae24db232d6f3693630fa37a382e1a6d7 - 06/20/2022 10:35 AM - nobu (Nobuyoshi Nakada)

[Feature #18788] Support options as String to Regexp.new

Regexp.new now supports passing the regexp flags not only as an
Integer, but also as a `String. Unknown flags raise errors.

Revision 883d13dc4127b5fde617b584ebb89714eac19965 - 06/20/2022 10:35 AM - nobu (Nobuyoshi Nakada)

[Feature #18788] Spec for options as String to Regexp.new

Co-Authored-By: Janosch Müller janosch.mueller@betterplace.org

Revision 4a6facc2d683d1dbb67ded8a9f4d7cd10a9fd8ad - 06/20/2022 10:35 AM - nobu (Nobuyoshi Nakada)

[Feature #18788] [DOC] String options to Regexp.new

Co-Authored-By: Janosch Müller janosch.mueller@betterplace.org

Revision 1e9939da - 06/20/2022 10:35 AM - nobu (Nobuyoshi Nakada)

[Feature #18788] Support options as String to Regexp.new

Regexp.new now supports passing the regexp flags not only as an
Integer, but also as a `String. Unknown flags raise errors.

Revision 883d13dc - 06/20/2022 10:35 AM - nobu (Nobuyoshi Nakada)

[Feature #18788] Spec for options as String to Regexp.new

Co-Authored-By: Janosch Müller janosch.mueller@betterplace.org

Revision 4a6facc2 - 06/20/2022 10:35 AM - nobu (Nobuyoshi Nakada)

[Feature #18788] [DOC] String options to Regexp.new

Co-Authored-By: Janosch Müller janosch.mueller@betterplace.org

History

#1 - 05/17/2022 01:15 PM - Dan0042 (Daniel DeLorme)

+1, I've often wanted that.

#2 - 05/17/2022 02:10 PM - sawa (Tsuyoshi Sawada)

I think this is a good idea, but I also think Regexp.new is not used so often so it does not matter if we have such feature or not. Most of the time, a regex is created using a regex literal. What is the use case of Regexp.new?

#3 - 05/17/2022 08:26 PM - janosch-x (Janosch Müller)

I agree that the use cases are fairly limited. They are not totally uncommon, though.

Regexp.new is mostly needed for recursion on Ruby and metaprogramming, and is used this way e.g. in rubocop, ruby_parser, and parser.

Sometimes it is used to build Regexps based on input, e.g. in capybara, prawn, and psych. There might also be a few CMSes that allow admins to type in validation patterns.

Many people are also seemingly unaware that Regexp literals support interpolation, or maybe they just find this interpolation hard to read. Either way, they often use Regexp.new instead, passing it an interpolated or concatenated String or Regexp.union output, as can be seen e.g. in css_parser, haml, net-ssh, or uri. (css_parser actually uses the only coincidentally working "i" as a second argument.)

In some other cases, Regexp.new is used to avoid a SyntaxError or a warning on older Rubies, e.g. sinatra does this.

#4 - 05/18/2022 04:03 AM - duerst (Martin Dürst)

Please don't allow symbols. It may look cute (in some cases), but the options are essentially a set of single letters, and that's not what Symbols are about.

If you really want symbols, please make it something like

```
Regexp.new('foo', :ignorecase, :multiline, :extend) # => /foo/imx
```

Even something a bit shorter, such as the following, would be okay with me:

```
Regexp.new('foo', :ignore, :multi, :ext) # => /foo/imx
```

#5 - 05/18/2022 08:21 AM - janosch-x (Janosch Müller)

> Please don't allow symbols. It may look cute (in some cases), but the options are essentially a set of single letters, and that's not what Symbols are about.

My reasoning for supporting them was not cuteness but avoiding confusion.

If we change only the processing of Strings, we will have this behavior:

```
Regexp.new('foo', 'i') # => /foo/i
Regexp.new('foo', 'm') # => /foo/m
Regexp.new('foo', :i)  # => /foo/i # looks like it also works
Regexp.new('foo', :m)  # => /foo/i # slightly surprising
```

I'm happy to support only Strings, though. In this case we might want to consider raising an ArgumentError or emitting a deprecation warning when a Symbol is passed, or even for anything that is not nil/true/false/Int/String?

> please make it something like Regexp.new('foo', :ignorecase, :multiline, :extend) # => /foo/imx

I don't think this is a viable option. Regexp.new already accepts up to 3 arguments. The third one is undocumented as far as I can tell, but it is used in the wild. If a String starting with "n" or "N" is passed as third argument, the Regexp encoding is set to ASCII. Arguably that makes consistency another reason for my proposal.

#6 - 05/22/2022 07:45 AM - nobu (Nobuyoshi Nakada)

I agree that symbols should be disallowed.
The examples uses arbitrary combinations of flags and the order of the flags doesn't have meanings.
In other words, it is a set of chars and OK for a string.
But a symbol is not such thing.

For the compatibility sake, we'll need migration period to warn anything other than integer and valid string, and then error.

#7 - 05/22/2022 01:55 PM - nobu (Nobuyoshi Nakada)

```
Regexp.new("(?#{options}:#{code})")

Regexp.new(code, eval("//#{options}").options)
```

#8 - 05/23/2022 10:17 AM - janosch-x (Janosch Müller)

@nobu (Nobuyoshi Nakada)

Thank you for the explanation regarding symbols!

> we'll need migration period to warn anything other than inter and valid string

I think the optional third argument should be also deprecated as described in #18797. Otherwise we might still want to allow nil as "stopgap" value for the second argument.

```
Regexp.new(code, eval("//#{options}").options)
```

This feels too unsafe for some of the cases mentioned above.

```
Regexp.new("(?#{options}:#{code})")
```

This is clever and will work for most of the use cases mentioned above. Just a few minor downsides:

1. The group options syntax isn't so well-known and universally understood.
2. A few use cases might need to preserve notation (e.g. codemod).
3. The result may become verbose in case of recursion or nesting of Regexps.

#9 - 05/23/2022 11:48 AM - janosch-x (Janosch Müller)

- Description updated

#10 - 06/16/2022 06:26 AM - matz (Yukihiro Matsumoto)

Accepted. Unknown flags should raise errors.

Matz.

#11 - 06/20/2022 10:35 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Closed

Applied in changeset git|1e9939dae24db232d6f3693630fa37a382e1a6d7.

[Feature #18788] Support options as String to Regexp.new

Regexp.new now supports passing the regexp flags not only as an
Integer, but also as a `String. Unknown flags raise errors.