Refinements are currently lexically scoped, which makes their use burdensome when one wants to apply a refinement to an entire project, requiring boiler plate at the top of every project file. I propose that there ought to be a method of applying refinements to an entire namespace (module), similar to how constants are scoped.

For example, here's a trivial example of how one is able to scope a new RuntimeError class inside a module.

```ruby
# a.rb
require 'b'
module MyProject
  class RuntimeError < ::RuntimeError; end
end

# b.rb
module MyProject
  class SomeClass
    def initialize
      raise RuntimeError, "Application error occurred" # Raises MyProject::RuntimeError exception
    end
  end
end
```

Could refinements be scoped in a similar way, without the requirement for the using clause in every file. For example:

```ruby
# a.rb
require 'b'
module MyProject
  refine! String do
    def exclaim
    self + "!!!"
    end
  end
end

# b.rb
module MyProject
  class SomeClass
    def initialize
      puts "Hello".exclaim # Outputs "Hello!!!"
    end
  end
end
```

I believe this would be an intuitive means of making refinements global to a project, which seems to be a highly desired feature, and something that needs to happen to mostly eliminate monkey patching. Of course, using the refine name would clash with the current behaviour of refine and break backwards compatibility, so I'd propose introducing refine! as an alternative to invoke this proposed new behaviour.

Apologies if this has already been proposed/discussed. I did search for similar proposals, but couldn't find anything. I'm interested to hear what some of the potential pitfalls of this would be.
First message:

#1 - 08/11/2017 05:00 AM - wardrop (Tom Wardrop)
- Subject changed from Make refinement scope like that of constants to Make refinement scoping to be like that of constants

Second message:

#2 - 08/11/2017 05:42 AM - jeremyevans0 (Jeremy Evans)
You probably want to read the very long issue that introduced refinements (#4085), which contains the reasoning.

Third message:

#3 - 08/12/2017 03:53 AM - shevegen (Robert A. Heiler)
I do not think that issue #4085 necessarily has to be the "one and only one true refinement", in particular if we keep in mind that towards ruby 3.x, even syntax changes could happen if they may make sense (and matz would approve).

On a general side note, I wonder whether I am the only one who dislikes the syntax in regards to refinements; that includes both the current form but also the "refine!" variant. For some reason, the syntax does not seem to fit into ruby code I write but perhaps that is just me. It's weird because I am in full agreement that refinements are good to be had, but the syntax is ... strange.

Especially the "using" clause, so in one way or another, although I dislike the "refine!" variant used by Tom above, I actually understand his syntax proposal a bit. Then again perhaps I misunderstood it.

Of course, using the refine name would clash with the current behaviour of refine and break backwards compatibility, so I'd propose introducing refine! as an alternative to invoke this proposed new behaviour.

Fourth message:

Well, in the worst case, you could target ruby 3.x - matz said at a conference that the major reason why he will not eliminate (too much?) existing functionality is to stay backwards compatible in the 2.x branch, whereas the 3.x could possibly include such changes. So I think you could actually propose any alternative syntax if it were to target 3.x (which is destined loosely towards the year ~2020, give or take).

Fifth message:

#4 - 08/13/2017 10:52 PM - wardrop (Tom Wardrop)
jeremyevans0 (Jeremy Evans) wrote:

You probably want to read the very long issue that introduced refinements (#4085), which contains the reasoning.

I thought there’d be one of these long discussions floating around, thanks for the links. Didn't find a lot of discussion on why refinements weren't implemented as module-scoped. The discussion pretty much started off as being lexically/file scoped. Either way, ~4 years on and with refinements now in the wild for some time, it's definitely something worth raising again in my opinion.

Sixth message:

#5 - 09/09/2017 02:44 AM - duerst (Martin Dürst)
- Related to Feature #13109: 'using' in refinements is required to be physically placed before the refined method call added