Description
Push a namespace into another namespace

Hello.

Consider this code here:

```ruby
class Konsole; def initialize; puts 'hello world'; end; end
```

This may reside in a file called konsole.rb

So far so fine. Now, as more and more other projects are required, the ruby developer may not want to have a toplevel class called Konsole. There are other examples, such as "module Config" or "module Configuration"; also see the change towards RbConfig, which I assume happened to not too easily conflict with a Config namespace.

Anyway, the ruby developer would like to shuffle this namespace into another one so that the toplevel "Konsole" goes away.

This is already possible in ruby code, I think.

We can add:

```ruby
module Foobar; end
```

And then put Konsole into the module Foobar "namespace", possibly so, like via this line here:

```ruby
module Foobar; Konsole = ::Konsole; end
```

And then delete the old namespace via:

```ruby
Object.send :remove_const, :Konsole
```

Then we can instantiate it still and I verified that this works:

```ruby
konsole = Foobar::Konsole.new
```

The following code demonstrates this; the last line is the one that will fail, which was what we wanted to achieve (to get rid of toplevel class Konsole):

```ruby
class Konsole; def initialize; puts 'hello world'; end; end
module Foobar; end
module Foobar; Konsole = ::Konsole; end
Object.send :remove_const, :Konsole
konsole = Foobar::Konsole.new
Konsole.new
```

So if you look at the above code, all I am doing here is basically to define a class and a module, and then "putting" the toplevel class Konsole into that module "namespace"; and then using :remove_const to get rid of the toplevel Konsole.

07/29/2016 11:55 AM - shevegen (Robert A. Heiler)
So far so fine, we can already do so in ruby.

But! I was thinking that this is a bit cumbersome. I am not sure if anyone ever wants to have this, but just in the event that others may wish to reshuffle namespaces more easily, perhaps there could be some API to support this.

I can not think of a good name - giving things a proper name is one of the hardest task in programming. :)

Perhaps we could add a new module called Namespace or something for ruby 3.x or some other name. Or it could be added to Kernel or Object, but I am not sure - it probably would not belong to either that. Or we could have a new module where we can add lots of fancy tricks, a bit like the old evil.rb and so forth. (Or like the did-you-mean gem showed, with extra requires such as the require 'did_you_mean/experimental' or require 'evil' haha, sorry, I just like the name evil)

A few examples could be:

```ruby
module Foobar; end # First we must create the new namespace.

Konsole.relocate to: Foobar

Might be a good name perhaps? Not sure.

With Namespace, it could be:

Namespace.assign Konsole, to: Foobar
Namespace.push Konsole, to: Foobar

Might be better, not sure.

Note that the above line, would also perform the above actions:

- Push the "namespace" Konsole into Foobar
- Delete the toplevel Konsole (or, if we want to make this more general, to get rid of wherever it is defined)

Anyway! I think it may perhaps be not worth to implement this, but I still thought that I could make the suggestion at the least.

Perhaps it also helps the generation of new, other ideas.

I assume that in the long run, with other ideas such as "isolated changes" being possible to "namespaces" (constants), like via refinements, and matz saying that the path to ruby 3.x is still open (aka no feature freezes and no "idea freezes"), I thought it is ok to suggest it even if it can not be implemented. :)

Thanks for reading!

May ruby make people happier.

PS:

I also was thinking of making this here:

```ruby
Object.send :remove_const, :Foobar
```n

Perhaps somewhat easier to read.

```ruby
Object.remove_const :Foobar
```n

The last line there does not work because it is a private method. Unfortunately, I again have the problem that I can not think of a better name either.
Perhaps:

Object.delete_namespace
Object.delete_constant

Or some other name.

I simply found the .send() variant a bit verbose. Anyway, I digress, I am using it here and there. :)

**History**

#1 - 08/01/2016 12:22 AM - shyouhei (Shyouhei Urabe)

I have wanted this kind of feature for a long time. Not sure if this proposed API is the answer though.

#2 - 08/03/2016 01:28 AM - kernigh (George Koehler)

We can call the private method with class Object; remove_const :Konsole; end

Your technique for moving Konsole into the Foobar namespace doesn't work if Konsole refers to itself. For example, I add code to Konsole that calls Konsole.new:

```ruby
class Konsole
  def initialize
    puts 'hello world'
  end

  def another
    Konsole.new
  end
end

module Foobar
  Konsole = ::Konsole
end

class Object
  remove_const :Konsole
end

konsole = Foobar::Konsole.new
konsole2 = konsole.another
```

It doesn't work...

```
$ ruby scratch.rb
hello world
scratch.rb:7:in `another': uninitialized constant Konsole::Konsole (NameError)
  from scratch.rb:19:in `<main>'
```

...because Konsole#another can't find the constant that we moved to Foobar::Konsole. This is because the definition of Konsole#another isn't inside module Foobar; ...; end, so it doesn't search Foobar for the constant.

The same error happens with classes in the standard library, like OpenStruct:

```ruby
require 'ostruct'

module Foobar
  OpenStruct = ::OpenStruct
end

class Object
  remove_const :OpenStruct
end

thing = Foobar::OpenStruct.new
p thing == thing
```

```
$ ruby scratch.rb
/home/kernigh/prefix/lib/ruby/2.4.0/ostruct.rb:312:in `==': uninitialized constant OpenStruct::OpenStruct (NameError)
  from scratch.rb:11:in `<main>'
```

To make your feature, it isn't enough to set Foobar::OpenStruct and remove Object::OpenStruct.
Probably you want an isolated loading system like Java's or Python's. That's fine. It must be a good idea. But I don't think it can be accomplished by modifying existing require or adding anything to it. You have to design the new one from scratch.

If you come up with any idea, please submit the proposal. That would be a nice addition to the future Ruby.

Matz.