Ruby trunk - Feature #6817

Partial application

07/31/2012 03:38 PM - citizen428 (Michael Kohl)

Status:	Open
Priority:	Normal
Assignee:	matz (Yukihiro Matsumoto)
Target version:	

Description

I know that what I propose here is a significant change to Ruby, but it's part of my ongoing quest to get some more functional programming features into Ruby (see also #4539 and #6373).

I was wondering if it would make sense to maybe introduce partial application to Ruby? So that instead of

```
(1..3).map { |i| i + 2 }
```

or the somewhat unwieldy

```
(1..3).map(&2.method(:+))
```

one could just write

```
(1..3).map(&2.+)
```

which I think has a quite rubyish feel to it. I have a POC implementation in Ruby (I tried it with various Fixnum methods) over at my blog, but that was just a quick hack and obviously I don't want to monkey-patch every method with arity 1, but it was a nice way of exploring possible syntax.

Related issues:

Related to Ruby trunk - Feature #4539: Array#zip_with	**Assigned**
Related to Ruby trunk - Feature #6373: public #self	**Closed**
Related to Ruby trunk - Feature #7939: Alternative curry function creation	**Feedback**
Related to Ruby trunk - Feature #13765: Add Proc#bind	**Open**

History

#1 - 07/31/2012 03:52 PM - shyouhei (Shyouhei Urabe)

- Description updated

#2 - 10/25/2012 07:36 PM - yhara (Yutaka HARA)

- Target version changed from 2.0.0 to 2.6

#3 - 02/26/2013 11:30 AM - ko1 (Koichi Sasada)

- Assignee set to matz (Yukihiro Matsumoto)

This ticket is related to [ruby-core:52797] [ruby-trunk - Feature #7939]?
(definitely no?)

Basically, I like this proposal.
But I'm not sure this notation can be acceptable.

FYI: Scheme has similar, but more flexible proposal:
http://srfi.schemers.org/srfi-26/srfi-26.html

```
(cut cons (+ a 1) <>)   is the same as  (lambda (x2) (cons (+ a 1) x2))
(cut list 1 <> 3 <> 5)   is the same as  (lambda (x2 x4) (list 1 x2 3 x4 5))
(cut list)   is the same as  (lambda () (list))
(cut list 1 <> 3 <...>)   is the same as  (lambda (x2 . xs) (apply list 1 x2 3 xs))
(cut <> a b)  is the same as  (lambda (f) (f a b))
```

Of course, it is not ruby's way. This is only sample of the other language.

#4 - 08/02/2014 08:59 AM - citizen428 (Michael Kohl)

Koichi Sasada wrote:

> Basically, I like this proposal.
> But I'm not sure this notation can be acceptable.

In that case, how about making Symbol#to_proc accept additional arguments?

```
(1..3).map(:+, 2)
```

The syntax would be very straightforward, but it doesn't go well with the current implementation of Symbol#to_proc's proc cache. Also this does go away a bit from the original point of partial application, though tbh this sort of scenario is what I mostly had in mind anyway.

#5 - 08/02/2014 02:18 PM - nobu (Nobuyoshi Nakada)

- Description updated

Michael Kohl wrote:

> In that case, how about making Symbol#to_proc accept additional arguments?
>
> ```
> (1..3).map(:+, 2)
> ```

It doesn't relate to Symbol#to_proc.

#6 - 07/26/2017 03:35 PM - k0kubun (Takashi Kokubun)

- Related to Feature #7939: Alternative curry function creation added

#7 - 07/26/2017 03:36 PM - k0kubun (Takashi Kokubun)

- Related to Feature #13765: Add Proc#bind added

#8 - 12/25/2017 06:15 PM - naruse (Yui NARUSE)

- Target version deleted (2.6)