# Ruby master - Feature #13765

## Add Proc#bind

07/24/2017 09:04 PM - davidcornu (David Cornu)

| | |
|---|---|
| **Status:** | Open |
| **Priority:** | Normal |
| **Assignee:** | |
| **Target version:** | |

**Description**

Proc has curry but no method to do partial application. Something like Proc#bind might be handy.

A naive implementation might look something like

```
class Proc
  def bind(*bound_args)
    -> (*args) { self.call(*bound_args, *args) }
  end
end

irb(main):001:0> foo = -> (first, second) { puts first, second }
=> #<Proc:0x007fc93a091f90@(irb):6 (lambda)>
irb(main):002:0> foo.bind(1).call(2)
1
2
=> nil
irb(main):003:0> foo.bind(1).bind(2).call
1
2
```

which does the job with the downside of only reporting argument mismatches when the returned Proc is called.

```
irb(main):004:0> foo3 = foo.bind(1).bind(2).bind(3)
=> #<Proc:0x007fc9378bcb00@(irb):3 (lambda)>
irb(main):005:0> foo.call
ArgumentError: wrong number of arguments (given 0, expected 2)
    from (irb):6:in `block in irb_binding'
    from (irb):35
    from /usr/local/bin/irb:11:in `<main>'
```

**Related issues:**

| | |
|---|---|
| Related to Ruby master - Feature #6817: Partial application | **Open** |
| Related to Ruby master - Feature #7939: Alternative curry function creation | **Feedback** |

### History

#### #1 - 07/25/2017 10:50 AM - k0kubun (Takashi Kokubun)

Could you show a real Ruby application or code which you can write more effectively if we have partial application?

#### #2 - 07/25/2017 02:51 PM - shevegen (Robert A. Heiler)

I do not have any pro or con opinion per se; my slight worry is about the name "bind".

When I read .bind, I wonder what is actually bound, and to what it is bound.

#### #3 - 07/25/2017 07:36 PM - davidcornu (David Cornu)

> I do not have any pro or con opinion per se; my slight worry is about the name "bind".

Yeah I share that concern. Ruby has a concept of bound methods which might get confused with this.

Lodash/Underscore refer to this as partial (https://lodash.com/docs/#partial) which could be a better name.

#### #4 - 07/26/2017 01:31 PM - davidcornu (David Cornu)

> Could you show a real Ruby application or code which you can write more effectively if we have partial application?

The use case is similar to that of Proc#curry, but I'd agree that typical Ruby code doesn't rely on Procs much. The lack of partial application on Proc just seemed like an odd omission.

The particular code I was writing that led to this implemented pagination by returning the current page of results and a proc to fetch the next page.

Example:

```
bind = -> (fn, *bound_args) {
  -> (*args) { fn.(*bound_args, *args) }
}

fetch_page = -> (page = 1) {
  # Perform request
  [results, bind.(fetch_page, page + 1)]
}
```

which lets you use it as follows

```
results, next_page = fetch_page.()

until results.empty?
  # Process results
  results, next_page = next_page.()
end
```

#### #5 - 07/26/2017 03:36 PM - k0kubun (Takashi Kokubun)

*- Related to Feature #6817: Partial application added*

#### #6 - 07/26/2017 03:36 PM - k0kubun (Takashi Kokubun)

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