After introduction of safe navigation operator and other latest features, it could be pretty idiomatic to do this:

```ruby
next_page = response.dig('meta', 'pagination', 'next')&.match(/&page=(\d+)/)&.[](1)
```

The ugly thing here is, obviously, &.[](1). When dealing with other classes we can use dig(one_key) or at(index) for arrays, but for MatchData there are no good synonym/alias.

Two options come to mind:

- &.capture(1) — singular from "captures"
- &.at(1) — like it is in Array, from which MatchData already borrows values_at

Yes, several &. in a row could be said a code smell and example may feel a bit extreme.

But for simple matches, the principle stays the same: "match something and extract if matched" otherwise should be written in much less readable code like this:

```ruby
if url =~ /&page=(\d+)/
  next_page = Regexp.last_match[1]
end
```

# or
```
if (m = url.match(/&page=(\d+)/))
  next_page = m[1]
end
```

# this seems the most reasonable syntax anyways:
```
next_page = url.match(/&page=(\d+)/)&.at(1)
```

This of course assumes that the safe navigation operator is "idiomatic". ;-)

I guess one can say that every possible feature of ruby is "idiomatic", but I feel that certain ruby style fits less well together than other particular styles in ruby. Anyway, I am just nitpicking, I understand what you want to convey (I think).

Yes, several &. in a row could be said a code smell and example may feel a bit extreme.

I don't know if we can state this because one could also say that "&." may be supoptimal, such as in any lines that already contain other use of ".", such as .map(&:strip). I am fine with the latter, but the lonely person & preceding and staring at a . is a bit ... weird to my eyes. But it depends a lot on how something is used. I still prefer the { 'key' => 'notation' } for Hashes, but I also make heavy use of the { key: 'notation' }, even though the latter is just "symbols" - it simply is a lot shorter for long hashes, and I like being able to be
succinct. (I also have no problem with Symbols at all, by the way.)

Code like this:

```ruby
next_page = url.match(/&page=(\d+)/)&.at(1)
```

also seems strange to my eyes.

I don't mind your feature request at all, mind you. If I understood it correctly then you wish to be able to replace e. g.

```ruby
$.[](l)
```

with a named method such as:

```ruby
$.capture(l)
```

So if this is the suggestion then I am in no way against it, independent of the "&" there. My reason is primarily because while I myself love method calls such as [] and variable "names" such as _, and while I use [] a LOT in my own code as an alternative to "Foo.new" aka "Foo[]", having a named method to invoke is in my opinion good and makes sense.

We'd get the best of two worlds here - a general, specific name and a non-"name" such as[]. So we can choose what we prefer, which is fine.

I don't have any particular opinion on the name itself, be it .capture() or .at() or any other name. I think good names are very important but the even more important thing is whether the general suggestion is ok or not; and if it is ok, then I guess we can easily find a name.

So in short, I think the alias suggestion in general for[] is ok.

I think that the & is not the prettiest character though.

---

**#2 - 02/05/2018 04:09 PM - zverok (Victor Shepelev)**

Idiomaticity of & is from the same point of view as yield_self's (here is my small blog post about the latter and why it is idiomatic): It allows to express data processing flow in a single chain of methods, instead of local variables, if's and so on.

I tend to use it only for the last step of some chain, which could eventually drop nil. For me subjectively, this:

```ruby
foo.bar.baz.qwak&.length
```

... is way more "Ruby idiomatic" than this:

```ruby
qwak = foo.bar.baz.qwak
qwak.length if qwak
# or
qwak & qwak.length
# or
if (qwak = foo.bar.baz.qwak)
  qwak.length
end
```

---

**#3 - 01/10/2019 04:29 AM - naruse (Yui NARUSE)**

- Status changed from Open to Feedback

Following is better

```ruby
next_page = response.dig('meta', 'pagination', 'next').slice(/&page=(\d+)/, l)
```

---

**#4 - 01/10/2019 10:14 AM - zverok (Victor Shepelev)**

Ugh. I feel ashamed, honestly :)

Never thought about using .slice this way, somehow! It is definitely cleaner.

Can be closed.

---

**#5 - 10/15/2021 06:49 PM - jeremyevans0 (Jeremy Evans)**

- Status changed from Feedback to Closed