Ruby master - Feature #16374

Object#nullify to provide scalars with the effect similar to Enumerable#reject

11/26/2019 04:12 PM - cvss (Kirill Vechera)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

How about adding a new method to Object class?

```
class Object
    def nullify &block
        block.call(self) ? nil : self
    end
end

'asdf'.nullify(&:empty?) #=> "asdf"
''.nullify(&:empty?) #=> nil
```

It can be used together for chaining several methods with conditions. E.g. with &. operator and #then:

```
[1, 2].nullify(&:empty?)&.then(&:join) #=> "12"
[].nullify(&:empty?)&.then(&:join) #=> nil
[].join #=> ""
'a b'.nullify(&:empty?)&.then(&:split) #=> ["a", "b"]
''.nullify(&:empty?)&.then(&:split) #=> nil
''.split #=> []
```

P.S. A similar opposite operation is available as a chain of then.detect

History

#1 - 11/26/2019 05:24 PM - cvss (Kirill Vechera)

- Subject changed from Object#nullify to make for scalars the similar to effect of #reject for Enumerable to Object#nullify to provide scalars with the effect similar to Enumerable#reject

#2 - 11/26/2019 08:50 PM - shevegen (Robert A. Heiler)

Personally I don't quite like method names that end in "fy alone. Crystal has .stringify
such as in **&.name.stringify** , which I always felt was a weird name - does not tell me much at
all. I have no particular opinion on the feature suggested itself, only on the name - although
the syntax example looks quite strange to me. Is it easy for others to decode on
[].nullify(&:empty?)&.then(&:join)? There seems to be so much information density packed
in and it just doesn't "feel" like oldschool ruby. Ruby can be very simple. Somehow from the
functional side of things, either it is all quite complex ... or just so different that it
does not feel simple. Or just different - but to me it "feels" as if this all starts from a
higher complexity base. Perhaps a separate ruby language could be added, just to appease
more functional-in-style ruby users. ;)

May be interesting to hear what zverok thinks about the idea either way since he put
forward ideas that are somewhat related to the suggestion here. :)

#3 - 12/03/2019 05:16 AM - mame (Yusuke Endoh)

IMO, [1, 2].nullify(&:empty?)&.then(&:join) is just cryptic.

BTW, why do you write &.then(&:join)? I'm afraid you have been corrupted by &:sym.

#4 - 12/03/2019 05:40 AM - nobu (Nobuyoshi Nakada)

I think I've proposed Object#not (and Object#!) for this purpose.

#5 - 12/07/2019 07:38 PM - jonathanhefner (Jonathan Hefner)

This is an interesting idea. It is like a generalized version of Active Support's Object#presence.

However, I agree that the name reads awkwardly. What about Object#unless?

```
[1, 2].unless(&:empty?)&.join  # == "12"
[].unless(&:empty?)&.join      # == nil
```

And then we could have Object#if to match:

```
[2].if(&:one?)&.first     # == 2
[1, 2].if(&:one?)&.first  # == nil
```

#6 - 12/08/2019 02:24 AM - sawa (Tsuyoshi Sawada)

This is a duplicate of #13807, which I have proposed and withdrawn. I have proposed an alternative #15557.

#7 - 12/08/2019 02:30 AM - sawa (Tsuyoshi Sawada)

- Description updated