Ruby master - Feature #14680
Adding +@ and -@ to hash and array
04/11/2018 02:49 PM - kddeisz (Kevin Newton)

Status: Closed
Priority: Normal
Assignee:
Target version:

Description
Since we have -@ and +@ for strings and it's very useful (and reads better than .freeze) I'd like to propose adding the same API to
hash and array. Especially with constants, linters are always telling me to freeze them, and I'd like to be able to accomplish that with
-@. I've attached the necessary code to make that happen.

History
#1 - 04/11/2018 07:48 PM - shevegen (Robert A. Heiler)
I like .dup and .freeze, more than + and - on class String.

https://ruby-doc.org/core-2.5.0/String.html#method-i-2B-40

The only good thing is that I don't have to use any of these
methods so ultimately I don't have to care either way. :)

As for linters, I assume you refer to rubocop since that is
by far the most common use case I could see, with people
putting things such as:

FOO = 'foo'.freeze
BAR = 'bar'.freeze

into files. I wonder if matz thinks that this is a good way;
to me, I always wondered when people do so. Probably
lint-based development ... should that not be superfluous
with frozen strings now? One comment on top of the file
and avoid all the explicit .freeze calls?

#2 - 04/11/2018 08:02 PM - kddeisz (Kevin Newton)
I don't think it's as common these days to see .freeze all over the place, since the linters explicitly tell you to use the frozen string literal comment. But
that's explicitly for literals. There are plenty of cases where you'd want to freeze strings without them being a literal in the code. In this case there's
also plenty of occurrences where you'd want to freeze hashes and arrays without them being literals. Maybe I should have rephrased the ticket, it's
definitely not just for the benefit of lint-based development.

#3 - 04/12/2018 02:24 AM - nobu (Nobuyoshi Nakada)
- Subject changed from Adding @+ and @- to hash and array to Adding +@ and -@ to hash and array
- Description updated

#4 - 05/17/2018 11:26 PM - rab (Rob Biedenharn)
If -@ is supposed to return a frozen copy, is that patch really doing that? The tests don't seem to ensure that. Do rb_hash_freeze and rb_ary_freeze
make copies first or .freeze the given object?

#5 - 06/10/2018 06:27 PM - kddeisz (Kevin Newton)
This patch is adding -@ and +@ to array and hash, mirroring the logic of string's version of those methods. So for -@ if the array or hash is already
frozen, it just returns the object, otherwise it returns a frozen copy of it. The opposite for +@.

#6 - 06/11/2018 02:38 PM - rab (Rob Biedenharn)
My question is whether this would pass:

def test_uminus
  a = { a: 1, b: 2, c: 3 }
  refute a.frozen?
  assert (-a).frozen?

08/29/2022
```ruby
a = { a: 1, b: 2, c: 3 }.freeze
assert_equal a, -a

# * If the hash is frozen, then return the hash itself.
fa = { a: 1, b: 2, c: 3 }.freeze
assert_equal fa, -fa
assert_equal fa.object_id, (-fa).object_id

# * If the hash is not frozen, return a frozen copy of it.
ua = { a: 1, b: 2, c: 3 }
assert_equal ua, -ua
refute_equal ua.object_id, (-ua).object_id
end
```
as the code simply return `rb_hash_freeze(hsh)` when `not OBJ_FROZEN(hsh)`.

#7 - 06/11/2018 03:26 PM - Hanmac (Hans Mackowiak)
we need to specify it, that `-@ and +@ are not doing deep freeze (they don't freeze the values of hash or elements of array)
people might do this:

```ruby
VAL = -['abc']
```

and suspect the string inside the array to be frozen too

or are we assume now that frozen-string-literal is true for this?

#8 - 06/11/2018 03:31 PM - kddeisz (Kevin Newton)
Happy to specify it in the docs, but it would be odd to assume that it did since `.freeze` itself doesn't.

#9 - 02/07/2019 06:33 AM - matz (Yukihiro Matsumoto)
- Status changed from Open to Closed

The `+@ and -@ for strings are strongly coupled with frozen-string-literals magic comment. Since we don't have frozen-array-expression nor frozen-hash-expression magic comments, I don't think we have a strong reason for `+@ and `-@ methods for arrays and hashes.

Matz.

Files

| freeze.patch | 3.86 KB | 04/11/2018 | kddeisz (Kevin Newton) |