Ruby master - Feature #14546

Hash#delete!

02/23/2018 06:52 PM - rringler (Ryan Ringler)

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

Description

Hash#delete currently returns nil if a given key is not found in the hash. It would be nice to have a way to check that the key was present in the hash. This can be accomplished with with a block, but it would be nice to have some sugar for this.

```
{ a: 'a' }.delete(:b)                                                    # => nil
{ a: 'a' }.delete(:b) { |key| raise KeyError, "key not found #{key}" }
# => KeyError (key not found: b)
```

I'd like to propose a Hash#delete! method:

```
{ a: 'a' }.delete!(:a) # => 'a'
{ a: 'a' }.delete!(:b) # => KeyError (key not found: :b)
```

History

#1 - 02/24/2018 02:20 AM - nobu (Nobuyoshi Nakada)

Hash#delete is destructive, so appending ! doesn't seem making sense.
Rather, a destructive version of Hash#fetch feels better.

#2 - 02/24/2018 04:42 AM - rringler (Ryan Ringler)

nobu (Nobuyoshi Nakada) wrote:

> Hash#delete is destructive, so appending ! doesn't seem making sense.

I'm as guilty of it as anyone, but as I understand ! does not mean destructive, but rather 'more dangerous', which I believe a possible exception qualifies for.

#3 - 02/24/2018 12:07 PM - shevegen (Robert A. Heiler)

> as I understand ! does not mean destructive, but rather
> 'more dangerous', which I believe a possible exception
> qualifies for.

You may still have the issue of semantics.

For example, .fetch() semantic is different than .delete().

The way how I remember methods with a "!", even if this is
not the official one, is to think of the "!" as a "modify
in place" operation. If one thinks about it in this way
then .delete() already modifies in place so .delete!()
is not making a lot of sense. However had, please do not
think that I really mind either way - I am neutral on the
suggestion. Perhaps someone could mention it to matz
briefly in the next developer meeting since Ryan referred
to a comment from matz almost 10 years ago. :-)

#4 - 02/26/2018 04:57 PM - marcandre (Marc-Andre Lafortune)

- Assignee set to matz (Yukihiro Matsumoto)

As Matz clearly stated, even if a long time ago, bang is not strictly for mutating versions of methods. Note that there is Process#exit! (which is not a mutating version of Process#exit). Rails also has many bang methods which aren't mutating, in particular ActiveRecord#save!.

I'm in favor of delete!.

#5 - 02/27/2018 01:36 AM - duerst (Martin Dürst)

marcandre (Marc-Andre Lafortune) wrote:

> As Matz clearly stated, even if a long time ago, bang is not strictly for mutating versions of methods. Note that there is Process#exit! (which is not a mutating version of Process#exit). Rails also has many bang methods which aren't mutating, in particular ActiveRecord#save!.

It's clear that bang methods are not only for mutating versions of methods. However, it would be a bad idea to use a bang method in a context (such as delete) where it can be very easily mistaken as a mutating version (and by opposition, the non-bang method would be misunderstood as a non-mutating version). That's why I think it's a bad idea to use delete! in the sense proposed above.

#6 - 02/28/2018 06:30 AM - sawa (Tsuyoshi Sawada)

What is wrong with using fetch?

```
{a: "a"}.fetch(:b) # => KeyError: key not found: :b
```

#7 - 03/08/2018 04:42 AM - rringler (Ryan Ringler)

sawa (Tsuyoshi Sawada) wrote:

> What is wrong with using fetch?
>
> ```
> {a: "a"}.fetch(:b) # => KeyError: key not found: :b
> ```

Hash#fetch is not mutative. This proposal is for a whiny version of #delete.

```
hsh = { a: 'a' }; hsh.delete!(:a); hsh # => {}
hsh = { a: 'a' }; hsh.delete!(:b); hsh # => KeyError (key not found: :b)
```

#8 - 05/30/2018 08:24 PM - janosch-x (Janosch Müller)

duerst (Martin Dürst) wrote:

> it would be a bad idea to use a bang method in a context (such as delete) where it can be very easily mistaken as a mutating version (and by opposition, the non-bang method would be misunderstood as a non-mutating version).

I'm not sure that point holds in such a general way. E.g. I've never seen ActiveRecord::Base#destroy being mistaken for a non-destructive version of #destroy!.

In my opinion, the main reason why #delete! feels confusing is that there are already #delete and #delete! implementations in Ruby that differ by destructiveness, e.g. on String. That is an unfortunate inconsistency when compared to Hash#delete.

Maybe something like Hash#yank could be introduced as an alias for delete and #yank! as a whiny version. That could both serve the need stated by the OP and also undercut this existing inconsistency.

Files

hash_delete_bang.patch	2.37 KB	02/23/2018	rringler (Ryan Ringler)