Ruby master - Feature #6219

Return value of Hash#store

03/29/2012 06:11 AM - MartinBosslet (Martin Bosslet)

Status:	Feedback	
Priority:	Normal	
Assignee:	matz (Yukihiro Matsumoto)	
Target version:	Next Major	

Description

Hash#store returns the value that was just assigned, for example:

h[:a] = b # => b

Does anyone rely on this behavior, are there cases when this becomes handy?

If however the return value is discarded most of the time, I was thinking it
might be beneficial if we would return the previous value of a given key (nil if
none was assigned yet) instead. That way we could assign and check for a collision
in one pass, something that right now can only be done in two separate steps.

History

#1 - 03/29/2012 06:29 AM - aprescott (Adam Prescott)

On Wed, Mar 28, 2012 at 22:18, Adam Prescott adam@aprescott.com wrote:

> Assignment always returns the right hand side, so this isn't quite an
> accurate demonstration of #store.

Although, I should also mention, this is due to the syntax of assignment
here being also a method call, and not simply that #foo= will always return
the argument it's given.

Or, since it's easier shown than described, what I mean is:

X.new.send(:foo=, 10) #=> :nope

#2 - 03/29/2012 06:40 AM - MartinBosslet (Martin Bosslet)

Right, good point. OK, then let me rephrase it for explicitly calling store. What I would prefer is:

h = { a: 1 }
h.store(:a, 2) # => 1
h.store(:b, 3) # => nil

That way I can check for collisions in one pass without having to call has_key? first. Otherwise
the result is evaluating the internal hash function twice, once for has_key? and once for the
following call to store.

#3 - 03/29/2012 07:24 AM - cjheath (Clifford Heath)

On 29/03/2012, at 8:40 AM, MartinBosslet (Martin Bosslet) wrote:

> Right, good point. OK, then let me rephrase it for explicitly calling store.
> ...
> That way I can check for collisions in one pass without having to call has_key? first.

It's a nice thought, but about fifteen years too late - you can't just change
fundamental things like this without introducing all sorts of subtle bugs into
people's programs. Write a new method (e.g. put(x)) that has the behaviour
you want.

Clifford Heath.

#4 - 03/29/2012 08:09 AM - MartinBosslet (Martin Bosslet)

cjheath (Clifford Heath) wrote:

> On 29/03/2012, at 8:40 AM, MartinBosslet (Martin Bosslet) wrote:
>
> > Right, good point. OK, then let me rephrase it for explicitly calling store.
> > ...
> > That way I can check for collisions in one pass without having to call has_key? first.
>
> It's a nice thought, but about fifteen years too late - you can't just change
> fundamental things like this without introducing all sorts of subtle bugs into
> people's programs. Write a new method (e.g. put(x)) that has the behaviour
> you want.

Doing so in one pass is only possible on the C level. No use in composing it in
Ruby, this still means that the hash function is evaluated twice.

Hash is probably way too popular for no one to rely on the current
return value, I agree. But put isn't taken yet - how about Hash#put with the described
behavior?

#5 - 03/29/2012 11:44 AM - shyouhei (Shyouhei Urabe)

=begin

Hmm, here you are a patch (not tested though).

From c55a9c9fab30d51be77821bce36054fe365b49af Mon Sep 17 00:00:00 2001
Message-Id: c55a9c9fab30d51be77821bce36054fe365b49af.1332988729.git.shyouhei@ruby-lang.org
From: URABE, Shyouhei shyouhei@ruby-lang.org
Date: Thu, 29 Mar 2012 11:38:17 +0900
Subject: [PATCH 1/1] Hash#store to return former content [feature #6219]

Signed-off-by: URABE, Shyouhei shyouhei@ruby-lang.org

diff --git a/hash.c b/hash.c
index e9937ff..0cd4e43 100644
--- a/hash.c
+++ b/hash.c
@@ -1099,6 +1099,20 @@ copy_str_key(st_data_t str)
return (st_data_t)rb_str_new4((VALUE)str);
}

+struct hash_aset_tuple {

- st_data_t old;
- st_data_t new; +}; + +static int +hash_aset_i(st_data_t key, st_data_t *value, st_data_t arg) +{
- struct hash_aset_tuple *ptr = (void *)arg;
- ptr->old = *value;
- *value = ptr->new;
- return ST_CONTINUE; +} + /*
  - call-seq:
  - hsh[key] = value   -> value @@ -1120,15 +1134,19 @@ copy_str_key(st_data_t str) VALUE rb_hash_aset(VALUE hash, VALUE key, VALUE val) {
- struct hash_aset_tuple t;   rb_hash_modify(hash);   hash_update(hash, key);
- t.new = val;
- t.old = Qundef;   if (RHASH(hash)->ntbl->type == &identhash || rb_obj_class(key) != rb_cString) {
- st_insert(RHASH(hash)->ntbl, key, val);
- st_update(RHASH(hash)->ntbl, key, hash_aset_i, (st_data_t)&t);   }   else {
- st_insert2(RHASH(hash)->ntbl, key, val, copy_str_key);
- st_data_t k = copy_str_key(key);
- st_update(RHASH(hash)->ntbl, k, hash_aset_i, (st_data_t)&t);   }
- return val;
- return t.old; }

static int
--
1.7.0.4

=end

#6 - 03/29/2012 08:38 PM - MartinBosslet (Martin Bosslet)

shyouhei (Shyouhei Urabe) wrote:

> =begin
>
> Hmm, here you are a patch (not tested though).

Awesome, thank you Shyouhei. I'll be happy to test this tonight!

#7 - 03/29/2012 09:33 PM - shyouhei (Shyouhei Urabe)

Note however, that nobu changed st_update API1 after I posted that. So it might not apply cleanly to the latest ruby trunk...

#8 - 03/29/2012 10:56 PM - nobu (Nobuyoshi Nakada)

And, what should be returned if the key wasn't set?

#9 - 03/29/2012 11:32 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

nobu (Nobuyoshi Nakada) wrote:

> And, what should be returned if the key wasn't set?

Its default value?

h = Hash.new {[]}
h.store('key', [1, 2]) # => []

#10 - 03/29/2012 11:42 PM - MartinBosslet (Martin Bosslet)

rosenfeld (Rodrigo Rosenfeld Rosas) wrote:

> nobu (Nobuyoshi Nakada) wrote:
>
>> And, what should be returned if the key wasn't set?
>
> Its default value?

Seems reasonable to me. To detect whether a key hasn't been set yet,
one would check for the presence of the default value. That
doesn't catch cases where a key with a default value is actually
set, but that seems fine to me. Voices against this?

#11 - 03/30/2012 12:25 AM - mame (Yusuke Endoh)

- Status changed from Open to Assigned

- Assignee set to matz (Yukihiro Matsumoto)

- Target version set to Next Major

Hello,

I tentatively mark this as 3.0 issue because it changes the behavior.
But if matz accepts the change, 2.0 may be able to include it.

Personally, I'm neutral to the proposal itself.

--
Yusuke Endoh mame@tsg.ne.jp

#12 - 03/30/2012 01:09 AM - nobu (Nobuyoshi Nakada)

- File 0001-Hash-store-return-old-value.patch added

rosenfeld (Rodrigo Rosenfeld Rosas) wrote:

>> And, what should be returned if the key wasn't set?
>
> Its default value?

Also calling its default proc?

MartinBosslet (Martin Bosslet) wrote:

> Seems reasonable to me. To detect whether a key hasn't been set yet,
> one would check for the presence of the default value. That
> doesn't catch cases where a key with a default value is actually
> set, but that seems fine to me. Voices against this?

Calling default proc can be expensive. So I don't think it's good idea to call it always.
A patch to separate Hash#store from Hash#[]=, and let it return the old value.

#13 - 03/30/2012 01:27 AM - rosenfeld (Rodrigo Rosenfeld Rosas)

How about hash.store(new_value, nil_value: :default)

This way, one could write "a = hash.store(1, nil_value: nil)", while the default :default would call the default proc if it exists.

#14 - 03/30/2012 01:29 AM - rosenfeld (Rodrigo Rosenfeld Rosas)

Or Hash#store(new_value, return_nil_if_not_set: false)

#15 - 03/30/2012 07:48 AM - MartinBosslet (Martin Bosslet)

nobu (Nobuyoshi Nakada) wrote:

> Calling default proc can be expensive. So I don't think it's good idea to call it always.
> A patch to separate Hash#store from Hash#[]=, and let it return the old value.

You are right, the default proc is a real "party pooper" here... Could something like Rodrigo
proposed work for the default proc case? Did you mean something like it when you proposed
separating Hash#store from #[]=, or did you think about something different?

#16 - 03/31/2012 10:07 AM - matz (Yukihiro Matsumoto)

by spec, all assignment should return the assigning value (to enable assignment chain), so that I have to reject the original proposal to change the value from "a[k] = v".
but there's still room to change the return value of Hash#store.

Matz.

#17 - 03/31/2012 11:13 AM - matz (Yukihiro Matsumoto)

- Status changed from Assigned to Feedback

#18 - 04/26/2012 09:20 AM - MartinBosslet (Martin Bosslet)

Yes, I neglected assignment in my initial proposal, silly idea :) Hash#store would be great, though! Apart from the problem of how to handle the default proc, are there still other reasons against this?

Files

0001-Hash-store-return-old-value.patch	6.8 KB	03/30/2012	nobu (Nobuyoshi Nakada)