Ruby - Feature #11813

Extend safe navigation operator for [] and []= with syntax sugar

12/13/2015 08:09 AM - sawa (Tsuyoshi Sawada)

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

Description

Now we have the safe navigation operator &.. But this cannot be used with syntax sugar form of the methods [] and []=, which are more frequent than their ordinary forms of method call. For example, when a can be either an array or nil, we can do:

```
a &.[](3)
a &.[]= 2, :foo
```

but we cannot do:

```
a &.[3]
a &.[2] = :foo
```

It would be nice if we can extend the use of &. to cover syntactic sugar as above.

Related issues:

Related to Ruby - Bug #11618: Safe call syntax with aref or aset is	**Rejected**
Has duplicate Ruby - Feature #13645: Syntactic sugar for indexing when using ...	**Open**

History

#1 - 12/13/2015 11:42 AM - yugui (Yuki Sonoda)

- Assignee set to matz (Yukihiro Matsumoto)

#2 - 12/13/2015 06:17 PM - usa (Usaku NAKAMURA)

IMO, we can write &. only for replacement of ..
As you know, ary.[idx] is not valid, then ary&.[idx] should not be valid, too.

#3 - 12/14/2015 05:04 AM - nobu (Nobuyoshi Nakada)

Usaku NAKAMURA wrote:

> IMO, we can write &. only for replacement of ..
> As you know, ary.[idx] is not valid, then ary&.[idx] should not be valid, too.

That is same as matz's opinion and the reason it was removed at r52430.

```
parse.y: revert lbracket

* parse.y (lbracket): remove .? before aref.  [Feature #11537]
  revert r52422 and r52424
```

I don't think this proposal will be accepted.
We'll need a better notation.

#4 - 12/18/2015 02:03 AM - yui-knk (Kaneko Yuichiro)

- Related to Bug #11618: Safe call syntax with aref or aset is added

#5 - 07/15/2016 06:49 PM - Anonymous

It seems to me that a "safe subscript operator" should simply add a & between the receiver and the subscript operator (making a[3] safe would mean changing it to a&[3]), just like safe navigation adds a & between the receiver and the method invocation operator (a.foo => a&.foo).

Unfortunately, & is also a method name and is defined for several corelib classes (bitwise AND for Fixnum, set intersection for Array, boolean AND for FalseClass/NilClass/TrueClass). So if variable a above were an array, a&[3] would return the set intersection of a and [3]. It is true that a&.[](3) accomplishes the desired outcome, but this involves using the subscript operator as a method name -- which obscures semantic intent.

Is it possible to define a "safe subscript operator" with simple and unique syntax?

#6 - 07/19/2016 06:32 AM - matz (Yukihiro Matsumoto)

- Status changed from Open to Rejected

Use #dig for referencing the value.
For updating, show us use cases.

Matz.

#7 - 06/16/2017 01:27 PM - znz (Kazuhiro NISHIYAMA)

- Has duplicate Feature #13645: Syntactic sugar for indexing when using the safe navigation operator added