Ruby trunk - Feature #8811

Counterpart to `Hash#key?` for `Array`

08/23/2013 04:04 AM - sawa (Tsuyoshi Sawada)

Status:	Feedback	
Priority:	Normal	
Assignee:		
Target version:		

Description

=begin
Hash hash key? to tell if a key exists without checking the value. It would be convenient if there were a counterpart in Array. Suppose it is called Array#index?. Then it should behave as follows:

```
[1, 2, 3].index?(2) # => true
[1, 2, 3].index?(3) # => false
[1, 2, 3].index?(-3) # => true
[1, 2, 3].index?(-4) # => false
```

This is useful when we want to insert/move/delete elements to/from a certain position of an array. Without checking if a value exists, it can be messed up. Implementing a check is cumbersome now. With the proposed method, it would become easy.

=end

History

#1 - 08/23/2013 01:20 PM - matz (Yukihiro Matsumoto)

- Status changed from Open to Feedback

Concrete use-case please? Besides that, I don't think #index? is a good name for the function.

Matz.

#2 - 08/27/2013 09:04 AM - phluid61 (Matthew Kerwin)

- File rb_ary_has_index.patch added

matz (Yukihiro Matsumoto) wrote:

> Concrete use-case please? Besides that, I don't think #index? is a good name for the function.
>
> Matz.

I think the name is reasonable. Hash defines:

- #key(value) => key of value
- #key?(key) => true if key present
- #has_key?(key) => alias of #key?

Array defines:

- #index(value) => index of value It makes sense that this method would be called #index? and/or #has_index?

#3 - 08/29/2013 06:14 AM - alexeymuranov (Alexey Muranov)

phluid61 (Matthew Kerwin) wrote:

> matz (Yukihiro Matsumoto) wrote:
>
> > Concrete use-case please? Besides that, I don't think #index? is a good name for the function.
> >
> > Matz.
>
> I think the name is reasonable. Hash defines:

- #key(value) => key of value
- #key?(key) => true if key present
- #has_key?(key) => alias of #key?

Array defines:

- #index(value) => index of value It makes sense that this method would be called #index? and/or #has_index?

There are two ways of using #index with an array: with non-negaitve integers, and with negaitve integers. It does not seem to make much sense to have an #index? method that returns true whenever an index fits one or the other convention. It seems to me indeed hard to come up with a use-case where the user wants to know if a number can be used as an index, but does not care if it is negative or positive. Two separate methods like #non_negative_index? and #negative_index? would look more natural but cumbersome to me.

#4 - 09/03/2013 01:05 PM - zzak (Zachary Scott)

Shouldn't it be called Hash#value then?

#5 - 09/03/2013 01:06 PM - zzak (Zachary Scott)

Ahh, sorry I misunderstood the feature request.

Request is for Array#index?

#6 - 09/03/2013 09:27 PM - sawa (Tsuyoshi Sawada)

=begin

- Matz: As for the method name, phluid61's explanation is exactly what I had in mind. And, some use cases are as shown below.
- phluid61: Your implementation looks perfect to me.

==== Use case 1
I want to delete and get from an array a an element at position i if there is such element. That element can be possibly nil.

With index?, it works like this:

```
a = ["a", nil, "b"]
deleted = :none
deleted = a.delete_at(3) if a.index?(3) # => Deletion did not happen.
deleted # => :none # I can be sure that deletion did not happen.

a = ["a", nil, "b"]
deleted = :none
deleted = a.delete_at(1) if a.index?(1) # => Deleted `nil`.
deleted # => nil # I can be sure that an element `nil` was deleted.
```

Without checking the index, I cannot tell whether deletion happened:

```
a = ["a", nil, "b"]
deleted = :none
deleted = a.delete_at(3) # => Deletion did not happen.
deleted # => nil # I cannot tell whether an element `nil` was deleted or deletion did not happen.

a = ["a", nil, "b"]
deleted = :none
deleted = a.delete_at(1) if a.index?(1) # => Deleted `nil`.
deleted # => nil # I cannot tell whether an element `nil` was deleted or deletion did not happen.
```

=== Use case 2
I want to prepend with :foo an element at position i if there is such element.

With index?, it works like this:

```
a = ["a", "b"]
a.insert(1, :foo) if a.index?(1)
# => ["a", :foo, "b"]

a = ["a", "b"]
a.insert(3, :foo) if a.index?(3)
# => ["a", "b"]
```

Without index?, unwanted nil will be inserted

```
a = ["a", "b"]
a.insert(1, :foo)
```

```
# => ["a", :foo, "b"] # This is okay.

a = ["a", "b"]
a.insert(3, :foo)
# => ["a", "b", nil, :foo] # Unwanted result
```

=end

#7 - 09/04/2013 02:12 AM - alexeymuranov (Alexey Muranov)

Why not to use

3 < a.size

instead of

a.index?(3)

etc.?

Files

rb_ary_has_index.patch	2.08 KB	08/27/2013	phluid61 (Matthew Kerwin)