Ruby master - Feature #18617
Allow multiples keys in Hash[] acting like Hash#dig
03/08/2022 09:52 PM - ddfznt (Ederson Fuzinato)

<table>
<thead>
<tr>
<th>Status:</th>
<th>Open</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td></td>
</tr>
<tr>
<td>Target version:</td>
<td></td>
</tr>
</tbody>
</table>

Description

Abstract

Since is very common the hash nested hashes, especially in the API world, I would love to use Hash[] syntax as alias for Hash#dig.

Background

Since a start to learn ruby, Hashes are the most powerful structure I can use to build API. It's intuitive and concise way to represent data.
But something always make me uncomfortable, the excess of square brackets([]) to access nested data.
Everytime is a "nasty", to access things like

```ruby
purchase[:customer][:addresses][:delivery][:street]
```

even worse when data is missing anywhere.
So, I meet the Hash#dig. Wonderful, easy, and powerful as well.

But .dig is not so intuitive, and I think, why not use the most common way to access data with multiple keys.

Why not use the most powerful method, with all powerfulness.
Why limitate Hash[] to one single param. :(

Proposal

So, my proposal is to allow Hash[] to take undefined params, and act like .dig, more concise to access nested data, more powerful, more happy :D.

Stop:

```ruby
hash[:a][:b][:c][:d][:e][:f][:u]
```

Just:

```ruby
hash[:a, :b, :c, :d, :e, :lov, :u]
```

Implementation

Since Hash[] and Hash#dig, both calling Hash::new when key is not found, just check the arity for performance.
Currently, I use something like:

```ruby
module AwesomeAccess
  def [] *keys
    if keys.many?
      dig *keys
    else
      super
    end
  end
end

class Hash
  prepend AwesomeAccess
end
```
a = {foo: {bar: :baz}} # => { foo: { bar: :baz } }

a[:foo][:bar] == a[:foo, :bar] # => true

Evaluation

It'll be awesome. ( Roses )•.೧۝ ৾ ༼ つ・*:・゚✧ ✧゚・: *ﾉ(◕ヮ◕ﾉ)

Discussion

I would love listen you guys.
Sincerely...ಥ_ಥ

Summary

Faces by textfac.es ?!

History

#1 - 03/09/2022 12:17 AM - hsbt (Hiroshi SHIBATA)
- Description updated

#2 - 03/10/2022 08:31 AM - janosch-x (Janosch Müller)
i think this would be confusing.

- it's not obvious what hash[a, b] does, it could also mean hash.values_at(a, b)
- no other bracket method on data structures behaves this way, e.g.
  - Array[] (and this can't be changed without breaking current 2-argument usage)
  - Hash[]
  - Set[]
  - Struct[]

#3 - 03/10/2022 01:36 PM - Eregon (Benoit Daloze)
janosch-x (Janosch Müller) wrote in #note-2:

  i think this would be confusing.

Agreed. Also IMHO accessing deep chains with many potential nils should be explicit (i.e, using dig).