Ruby - Feature #19634

Pattern matching dynamic key

05/09/2023 07:05 PM - baweaver (Brandon Weaver)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

I found myself in a situation (stable marriage problem) where I would like to match against a dynamic key, like so:

```
mentor_proposals = { mentor_name: ['mentee_1', 'mentee_2'] }
mentor_name = :mentor_name
mentee_name = 'mentee_1'

mentor_proposals in ^key: [*, ^mentee_name, *] # SyntaxError
```

Currently this is not supported syntax, but there are some use cases in which I might see myself wanting to use it including this one. As deconstruct_keys currently accepts an Array of keys this would not break compatibility but would introduce syntactic complexity in capturing keys on hash-like matches.

I believe the tradeoff is worthwhile, but would like to hear others opinions on the matter.

Granted this case has some oddities of Symbol and String interchangeability as an implementation detail, and I will not be arguing for key irreverence in this issue as that's a much more involved topic.

History

#1 - 05/18/2023 03:41 AM - matz (Yukihiro Matsumoto)

Probably, you are proposing mentor_proposals in ^mentor_name => [*, ^mentee_name, *]. I still cannot imagine the case where dynamic key is useful.
As far as I know, no other language support dynamic key matching. Please be more concrete to persuade us (preferably, with pseudo code).

Matz.

#2 - 05/18/2023 04:28 AM - marcandre (Marc-Andre Lafortune)

FWIW, Elixir actually supports it, but I don't recall seeing it used in the wild

```
map = %{mentor_name: "Joe"}
value = "Joe"
key = :mentor_name
match?(%{^key => ^value}, map) # => true
```

OTOH, Elixir does not support [*, ^mentee_name, *]...

#3 - 05/18/2023 04:39 AM - marcandre (Marc-Andre Lafortune)

Some actual examples of dynamic key matching in Elixir: https://github.com/search?q=%2F%25%5C%7B%5C%5E%2F+&type=code

#4 - 05/19/2023 04:02 PM - Anonymous

#5 - 05/19/2023 04:23 PM - austin (Austin Ziegler)

I count 44 instances of this in our production code (~100k lines of Elixir), but I don't think I've ever used key and value pinning as shown in the examples above.

But dynamic key matching is precisely what is required when it is required, although it's rare enough.

Here's a sample GraphQL response parser or the Shopify Customers API:

```
  def parse_customer_response(%{} = body, resource, action) do
    gql_action = resource <> String.capitalize(action)
```

```
    case body do
      %{status: 200, body: %{^gql_action => %{^resource => %{} = result, "userErrors" => []}}} ->
        {:ok, result}

      %{status: 200, body: %{^gql_action => %{"userErrors" => []} = result}} ->
        {:ok, result}

      %{errors: errors} ->
        {:error, Enum.map(errors, &Map.get(&1, "message"))}

      %{body: %{^gql_action => %{"userErrors" => errors}}} ->
        {:error, Enum.map(errors, &Map.get(&1, "message"))}

      %{body: %{^gql_action => nil}} ->
        {:error, [%{message: "#{gql_action} access denied"}]}

      %{body: nil} ->
        {:error, [%{message: "#{gql_action} access denied"}]}

      _ ->
        {:error, [%{message: "Unknown error"}]}
    end
  end
```

I haven't started using pattern matching in Ruby, but I could see some simplified code with dynamic key matching.

(As a separate note, it appears that email replies to redmine aren't necessarily working, as this should have come in by email.)