# Ruby - Feature #21358

## Advanced filtering support for #dig

05/21/2025 11:01 PM - lovro-bikic (Lovro Bikić)

| | |
|---|---|
| **Status:** | Feedback |
| **Priority:** | Normal |
| **Assignee:** | |
| **Target version:** | |

**Description**

Currently, #dig can be used to access nested data structures using "simple" keys, such as array indices or hash keys.

Real-world applications sometimes require non-trivial data access, for example, finding an item in an array based on some criteria, then returning some property of that item.

This feature request is to add support for such non-trivial access to #dig, concretely by allowing Procs as dig keys.

## Introductory example

Given the following data structure (similar to one from dig docs):

```
item = {
  id: '0001',
  batters: {
    batter: [
      { id: '1001', type: 'Regular' },
      { id: '1002', type: 'Chocolate' },
      { id: '1003', type: 'Blueberry' },
      { id: '1004', type: 'Devils Food' }
    ]
  }
}
```

and the requirement "find ID of batter with type 'Chocolate'" (assuming we don't know its position in the array), currently the datum can be retrieved like so:

```
item.dig(:batters, :batter)&.find { it[:type] == 'Chocolate' }&.[](:id)
# => "1002"
```

If #dig supported Proc as keys, the solution could be:

```
item.dig(:batters, :batter, -> { it[:type] == 'Chocolate' }, :id)
# => "1002"
```

## Implementation

Here's a monkey patch which adds Proc support to Array#dig, to play around with:

```
class Array
  alias_method :original_dig, :dig

  def dig(key, *identifiers)
    case key
    when Proc
      val = find(&key)

      identifiers.any? ? val&.dig(*identifiers) : val
    else
      original_dig(key, *identifiers)
    end
  end
end
```

This code also shows how I would define Proc argument behavior for arrays. The proc is called with each array item, and the *first* one for which a truthy value is evaluated is returned.

## Precedence

I see this feature similar to JSONPath's filter selector, where the equivalent path for the introductory example would be:

```
$.batters.batter[?(@.type == 'Chocolate')].id
```

I am not familiar with similar implementations in other programming languages, so I cannot draw parallels there.

## Real-world examples

Here's a GitHub code search with potential candidates that could be refactored with this new feature. There are some false positives on this link, sorry about that. The regex is also probably incomplete, so there could be more results.

I will highlight two examples from the results and how they could be refactored:

```
# https://github.com/moraki-finance/ruby-experian/blob/84f7def9987b6377f4718a0730fdb564d6e9a0fb/li
b/experian/trade_report.rb#L89
value_section&.find { |d| d["Tipo"] == value_name }&.dig("ListaValores", "Valor")&.find { |v| v[
"Periodo"] == period.to_s }&.dig("Individual")&.to_i
value_section&.dig(-> { it["Tipo"] == value_name }, "ListaValores", "Valor", -> { it["Periodo"] ==
 period.to_s }, "Individual")&.to_i

# https://github.com/cyfronet-fid/marketplace/blob/f84947777aa79d02fb987092416a3cb143db3d01/lib/im
port/datasources.rb#L59
datasource_data&.dig("identifiers", "alternativeIdentifiers")&.find { |id| id["type"] == "PID" }&.
[]("value")
datasource_data&.dig("identifiers", "alternativeIdentifiers", -> { it["type"] == "PID" }, "value")
```

## Final thoughts

The main benefit of this feature would be to not have to insert other methods in-between #dig, for example #find in the above examples. This could lead to cleaner code when handling complex data structures (such as returned from JSON APIs), with fewer usages of safe navigation.

On the other hand, Proc argument could be misinterpreted by developers to mean something else (e.g. return *all* items for which the proc returns a truthy value). Proc arguments could also be seen as non-idiomatic Ruby.

It should also be clearly defined how many results such digs return, just the first one for which the Proc is truthy (#find behavior), or all for which it's truthy (#select behavior). I admit at this point I'm not entirely sure, though I'm leaning towards the former (#find).

Final note: in this feature request I considered only Array#dig, but Hash#dig could be considered as well (the proc receives each key/value pair).

### History

#### #1 - 05/22/2025 10:00 AM - nobu (Nobuyoshi Nakada)

You might be able to achieve similar behavior using pattern matching.

```
item => batters: {batter: [*, {type: "Chocolate", id:}, *]}
id #=> "1002"
```

#### #2 - 06/05/2025 07:13 AM - mame (Yusuke Endoh)

*- Status changed from Open to Feedback*

#### #3 - 06/05/2025 09:08 AM - matz (Yukihiro Matsumoto)

I prefer pattern matching.

Matz.