Ruby - Feature #17165

Add `filter` and `flatten` keywords to `Enumerable#map`

09/12/2020 02:46 PM - sawa (Tsuyoshi Sawada)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

I had a use case to do map on an enumerable, with 1-level flattening, while skipping nil values.

There are convenient Enumerable#flat_map and Enumerable#filter_map methods, but the problem is that they cannot be used at the same time. I had to chose to do either of the following:

```
array
.filter_map do |foo|
  bar = baz(foo)
  next unless bar
  bar.map{...}
end
.flatten(1)

array
.flat_map do |foo|
  bar = baz(foo)
  next unless bar
  bar.map{...}
end
.compact

array
.flat_map do |foo|
  bar = baz(foo)
  next [] unless bar
  bar.map{...}
end
```

The last one of the above may not look so bad, but it requires an extra consideration, and is a bit hacky. When you are in a hurry, it just might not come to your mind.

This led me to realize that flat_map and filter_map should not be independent operations, but are rather some different modes of the operation map. There is no reason for the modes to be mutually exclusive of one another, and a use case that I mentioned above may arise.

I propose to add filter and flatten as optional keyword arguments to Enumerable#map.

```
array
.map(filter: true, flatten: 1) do |foo|
  bar = baz(foo)
  next unless bar
  bar.map{...}
end
```

In fact, even when the two parameters are not used together, I believe it would be easier to the brain and I would feel much more comfortable to pass filter: true or flatten: 1 to map when necessary rather than having to deicide whether to use map or flat_map or use map or filter_map.

Furthermore, this would make it possible to do flattening of an arbitrary depth (as specified by the parameter) during map.

History

#1 - 09/13/2020 09:38 AM - Eregon (Benoit Daloze)

What's the problem with the obvious:

```
array.map { |foo|
  baz(foo)
}.select { |bar|
  condition(bar)
}.flat_map { |bar|
  bar.map{...}
}
```

I think it's so much more readable.
And I don't think the extra allocations matter much.

IMHO Enumerable#map should map elements, and nothing else.
That's also seem to have been the opinion of many others.

#2 - 09/13/2020 09:41 AM - Eregon (Benoit Daloze)

And I'd argue if one wants to do everything in one block, just enjoy the freedom of imperative programming:

```
result = []
array.each do |foo|
  if bar = baz(foo)
    result.concat bar.map{...}
  end
end
```