Ruby - Feature #16296

Alternative behavior for `...` in method body if `...` is not in method definition

11/05/2019 04:53 PM - Dan0042 (Daniel DeLorme)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

In #16253 we settled on a syntax where the remainder arguments captured via ... in the method definition can be forwarded via ... in the method body. I think that was the correct decision.

But I can't forget about the use case presented by zverok (and in #15049) where the method definition is used to specify mandatory and default parameters and then forward all of them to another method. I've also experienced that same use case in my code. Using the current syntax we would need to do this:

```
def get(path:, accept: :json, headers: {}, ...)
  _request(method: :get, path: path, accept: accept, headers: headers, ...)
end

def post(path:, body:, accept: :json, headers: {}, ...)
  _request(method: :post, path: path, body: body, accept: accept, headers: headers, ...)
end
```

Which feels pointlessly repetitive to me. So I was thinking that maybe if ... is not present in the method definition, then in the method body ... could take on the meaning of "all arguments of the method". Then the code would look like this:

```
def get(path:, accept: :json, headers: {}, **opts)
  _request(method: :get, ...)
end

def post(path:, body:, accept: :json, headers: {}, **opts)
  _request(method: :post, ...)
end
```

In those examples (no positional parameters) it would also allow Hash[...] or {}.replace(...) to get the hash of all keyword arguments.

Pro: it allows a new useful and powerful behavior
Con: some may consider it 'unclean' to change the behavior of ... based on the method definition

Related issues:

Related to Ruby - Feature #16253: Shorthand "forward everything" syntax	**Closed**

History

#1 - 11/06/2019 12:56 AM - shevegen (Robert A. Heiler)

This proposal would change ... though and add a new meaning.
People then have to remember that ... can be omitted in
definitions too.

I am not sure that this is the same use case as the prior one
where ... had to be used in both the method definition and
the method body.

Maybe I am too used to oldschool ruby, but I think that we
are beginning to have too much special syntax in general;
and people often like to build up on prior ideas, e. g. the
.: addition.

To me just sticking to an options hash seems so much better
compared to these alternatives ideas ;) - I understand it
is not the same as keyword arguments but it is so much
simpler too.

#2 - 11/10/2019 10:15 AM - Eregon (Benoit Daloze)

- Related to Feature #16253: Shorthand "forward everything" syntax added

#3 - 11/10/2019 10:20 AM - Eregon (Benoit Daloze)

On current master def m(a, ...); end is a syntax error.
So I think we need to confirm first whether we will allow any other parameter alongside ....