rescue *[] should be equivalent to `rescue` as `method_call(*[])` is equivalent to `method_call`
This is the expected behavior. `rescue *array` should mean rescue only exception classes in the array. It should not mean rescue only exception classes in the array, unless the array is empty, in which case rescue `StandardError`. Otherwise you end up changing the meaning of things like:

```ruby
exceptions= []
exceptions << ArgumentError if ENV["ArgumentError"]
begin
  raise ArgumentError, "x"
rescue *exceptions
  puts "caught"
end
```

That's not consistent with the meaning of splatting an empty array, whereas the opposite is.

In a construct that takes a coma separated list, splatting an empty array produces a void list (no values)

so rescue `*[Class1, Class2]` translates to rescue `Class1, Class2`
rescue `*[Class1]` translates to rescue `Class1`
and rescue `[*]` to a plain rescue which does not mean rescue nothing

That would be logical and consistent.

There is no explicit syntax for rescue nothing which would be something like rescue(), so rescue `[*]` has to mean rescue and not the non-existent rescue()

Your argument is basically that rescue `[*]` should mean rescue. In reality, rescue is a shortcut for rescue `*[StandardError]`. If you look at it from that perspective, it is obvious that rescue `[*]` and rescue `*[StandardError]` should not be the same thing.

It should not mean rescue only exception classes in the array, unless the array is empty

That's not consistent with the meaning of splatting an empty array, whereas the opposite is.

In a construct that takes a coma separated list, splatting an empty array produces a void list (no values)

so rescue `*[Class1, Class2]` translates to rescue `Class1, Class2`
rescue `*[Class1]` translates to rescue `Class1`
and rescue `[*]` to a plain rescue which does not mean rescue nothing

That would be logical and consistent.

There is no explicit syntax for rescue nothing which would be something like rescue(), so rescue `[*]` has to mean rescue and not the non-existent rescue()
The explicit syntax for rescue nothing is rescue "[] :) . As I showed in my earlier example, changing rescue *array to mean rescue StandardError if the array is empty will break backwards compatibility.

Assigning to matz to make a decision on this.

#7 - 09/06/2019 03:59 AM - sawa (Tsuyoshi Sawada)
- Description updated

#8 - 09/06/2019 05:37 PM - bughit (bug hit)

The explicit syntax for rescue nothing is rescue "[] :) 

Splat is not part of the rescue syntax, it composes with it, the same way it composes with other constructs that take a comma separated list (invocations, not sure if there are others).

Here's an excerpt from "The ruby programming language"

Here's how we would write a rescue clause to handle exceptions of either of these types and assign the exception object to the variable error:

rescue ArgumentError, TypeError => error

Here, finally, we see the syntax of the rescue clause at its most general. The rescue keyword is followed by zero or more comma-separated expressions, each of which must evaluate to a class object that represents the Expression class or a subclass. These expressions are optionally followed by => and a variable name.

It documents the specific syntax of rescue but does not even mention the splat, which does not have any special meaning in this context and its general meaning is "[] == a void list, so rescue "[] == rescue

#9 - 09/07/2019 10:06 AM - Eregon (Benoit Daloze)

The core of this is that rescue (which means rescue StandardError) vs rescue *classes (which means rescue any of classes) is detected at parse time, not at runtime.

I think the current logic makes sense in that regard, and I think it's is less surprising than rescue "no_classes to "magically" rescue StandardError.

#10 - 09/09/2019 05:16 PM - bughit (bug hit)

I think it's is less surprising than rescue "no_classes to "magically" rescue StandardError

It is the current behavior that's magical. If you try to deduce what rescue "[] means from the primitives, it goes like this:

- "[] means a void (non-existent) list
- therefore rescue "[] means rescue. It can't mean rescue() (like super()) because rescue() does not exist

Anything but the above is special-casing, i.e. magic

#11 - 09/19/2019 05:26 AM - matz (Yukihiro Matsumoto)
- Status changed from Assigned to Closed

This "[] is not just exception list omitted, but explicitly specifies zero exceptions to catch. Thus the current behavior is intended.

Matz.