Ruby - Bug #8841

Module#included_modules and prepended modules

08/31/2013 08:17 AM - marcandre (Marc-Andre Lafortune)

Status:	Closed	
Priority:	Normal	
Assignee:	matz (Yukihiro Matsumoto)	
Target version:		
ruby -v:	r42735	**Backport:**

Description

The documentation for Module#included_modules currently states "Returns the list of modules included in +mod+."

This was never perfectly accurate, as the list also contains modules included in +mod+'s ancestors.

It now also includes prepended modules.

This is consistent with include? that returns true for prepended modules, but not quite consistent with included that does not get called for prepended modules.

Matz, could you confirm that current behavior is what you want?

If so, we should fix the documentation of include? and included_modules.

Related issues:

Related to Ruby - Feature #8026: Need Module#prepended_modules **Feedback**

Associated revisions

Revision 8ab11096ef3e0cf594308da285af2257cb4f5291 - 01/16/2020 07:31 PM - Marc-Andre Lafortune

Clarify documentation for Module#included_modules and Module#included?

[DOC] [ci skip] [Bug #8841]

Revision 8ab11096ef3e0cf594308da285af2257cb4f5291 - 01/16/2020 07:31 PM - Marc-Andre Lafortune

Clarify documentation for Module#included_modules and Module#included?

[DOC] [ci skip] [Bug #8841]

Revision 8ab11096 - 01/16/2020 07:31 PM - Marc-Andre Lafortune

Clarify documentation for Module#included_modules and Module#included?

[DOC] [ci skip] [Bug #8841]

History

#1 - 12/25/2017 06:15 PM - naruse (Yui NARUSE)

- Target version deleted (2.6)

#2 - 12/20/2019 12:56 AM - mame (Yusuke Endoh)

Summary:

- Module#include? and Module#included_modules regard prepended modules as included (not well documented)
- Module#included is not called when the module is prepended

Is this right?

#3 - 12/20/2019 02:33 PM - byroot (Jean Boussier)

> Module#included is not called when the module is prepended

I think that's fine, because it calls Module#prepended.

> Module#include? and Module#included_modules regard prepended modules as included (not well documented)

That yes:

```
class Base
end

module A
end

module B
end

Base.prepend(A)
Base.include(B)

p Base.included_modules
```

outputs:

```
[A, B, Kernel]
```

#4 - 01/16/2020 06:19 AM - matz (Yukihiro Matsumoto)

- Status changed from Open to Closed

This intentional.

Matz.

#5 - 01/16/2020 07:32 PM - marcandre (Marc-Andre Lafortune)

Thank you.

I clarified the documentation to reflect this.