Ruby master - Feature #11927

Return value for `Module#include` and `Module#prepend`

12/30/2015 11:40 AM - sawa (Tsuyoshi Sawada)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

Currently, Module#include and Module#prepend return the receiver, regardless of whether the ancestor chain has been modified. It is not straightforward to know whether it actually had effect.

```
module A; end
module B; end
A.include B # => A
A.ancestors # => [A, B]
A.prepend B # => A
A.ancestors # => [A, B]
```

I propose that, when Module#include and Module#prepend have no effect, they should either:

(1) return nil
(2) return false, or
(3) raise an exception

This is similar to Kernel#require, which returns false when it has no effect. To make it parallel with Kernel#require, it might be even better to return true when Module#include and Module#prepend have effect, and false otherwise. It makes no sense to return the receiver because that is known.

Some relevant cases with expectations are:

- prepend after include

```
module A; end
module B; end
A.include B # => A/true
A.prepend B # => nil/false/exception
```

- include after prepend

```
module A; end
module B; end
A.prepend B # => A/true
A.include B # => nil/false/exception
```

- include/prepend after include/include at superclass

```
class A; end
module B; end
A.include M # => A/true
class B < A; end
B.include M # => nil/false/exception
```

History

#1 - 08/29/2020 05:03 PM - fatkodima (Dima Fatko)

I would like to have this. Please, reconsider this feature.

#2 - 08/29/2020 08:21 PM - sawa (Tsuyoshi Sawada)

- Description updated

#3 - 08/30/2020 02:18 AM - sawa (Tsuyoshi Sawada)

- Description updated

#4 - 09/02/2020 12:40 AM - marcandre (Marc-Andre Lafortune)

It would help to:

1) have an example of use case

2) discuss why B.include M unless B < M is not equivalent / sufficient