

RUBY

bits

MODULES

PRESS START

NAMESPACE

IMAGE_UTILS.REB

```
def preview(image)  
end
```

```
def transfer(image, destination)  
end
```



pollutes global namespace

potential conflicts with
methods with same name

RUN.REB

```
require 'image_utils'
```

```
image = user.image  
preview(image)
```

RUBY
BITS

NAMESPACE

IMAGE_UTILS.RB

```
module ImageUtils
  def self.preview(image)
  end

  def self.transfer(image, destination)
  end
end
```



RUN.RB

```
require 'image_utils'

image = user.image
ImageUtils.preview(image)
```

RUBY
BITS

MIXIN

IMAGE_UTILS.REB

```
module ImageUtils
  def preview
  end

  def transfer(destination)
  end
end
```



... can access properties
on objects from host class

AVATAR.REB

```
require 'image_utils'
class Image
  include ImageUtils
end
```

Included as instance methods

RUN.REB

```
image = user.image
image.preview
```

RUBY
BITS

ANCESTORS

```
class Image
  include ImageUtils
end
```

Adding module to Image's
ancestors chain

Image.ancestors
[Image, ImageUtils, Object, Kernel, BasicObject]

Image.included_modules
[ImageUtils, Kernel]

modules only

MIXINS VS CLASS INHERITANCE

```
class Post
  def share_on_facebook
  end
end
```

```
class Image
  def share_on_facebook
  end
end
```

```
class Tweet
  def share_on_facebook
  end
end
```

```
class Shareable
end
```

MIXINS VS CLASS INHERITANCE

```
class Post < Shareable  
end
```

```
class Image < Shareable  
end
```

```
class Tweet < Shareable  
end
```



```
class Shareable  
  def share_on_facebook  
  end  
end
```

**A class can only have one superclass.
Inheritance suggests specialization.
Some behaviors are not fit for classes.**

MIXINS VS CLASS INHERITANCE

```
class Post  
  include Shareable  
end
```

```
class Image  
  include Shareable  
end
```

```
class Tweet  
  include Shareable  
end
```



```
module Shareable  
  def share_on_facebook  
  end  
end
```


MIXINS VS CLASS INHERITANCE

```
class Post
  include Shareable
  include Favoritable
end
```

```
class Image
  include Shareable
  include Favoritable
end
```

```
class Tweet
  include Shareable
  include Favoritable
end
```



```
module Shareable
  def share_on_facebook
  end
end
```

```
module Favoritable
  def add_to_delicious
  end
end
```

MIXIN

```
class Tweet  
  extend Searchable  
end
```

```
module Searchable  
  def find_all_from(user)  
  end  
end
```



•
•
•
•
•
•
•



Included as class methods

```
Tweet.find_all_from('@GreggPollack')
```

MIXIN

```
class Tweet  
  extend Searchable  
end
```

- use extend to expose
- methods as class methods

▼

```
Tweet.find_all_from('@GreggPollack')
```

```
class Image  
  include ImageUtils  
end
```

- use include to expose
- methods as instance methods

▶

```
image = user.image  
image.preview
```

MIXIN

```
class Image  
end
```

```
image = Image.new  
image.extend(ImageUtils)  
image.preview
```

```
module ImageUtils  
  def preview  
  end  
end
```

.....▶
an object is extending the module

the module will not be available in other objects

```
image = Image.new  
image.preview
```

```
NoMethodError: undefined method `preview' for #<Image:0x10b448a98>
```

HOOKS - SELF.INCLUDED

```
module ImageUtils
```

```
  def preview  
  end
```

```
  def transfer(destination)  
  end
```

```
  module ClassMethods
```

```
    def fetch_from_twitter(user)  
    end  
  end
```

```
end
```

```
class Image
```

```
  include ImageUtils
```

```
  extend ImageUtils::ClassMethods
```

```
end
```

```
image = user.image  
image.preview
```

```
Image.fetch_from_twitter('gregg')
```


HOOKS - SELF.INCLUDED

```
module ImageUtils
  def self.included(base)
    base.extend(ClassMethods)
  end

  def preview
  end

  def transfer(destination)
  end

  module ClassMethods
    def fetch_from_twitter(user)
    end
  end
end
```

```
class Image
  include ImageUtils
  extend ImageUtils::ClassMethods
end
```

self.included is called by Ruby
when a module is included in a class

```
image = user.image
image.preview

Image.fetch_from_twitter('gregg')
```

in this case, the module
name can be anything

NEW
BITS

ACTIVESUPPORT::CONCERN

```
module ImageUtils
```

```
  def self.included(base)
    base.extend(ClassMethods)
    base.cleanup
  end
```

```
module ClassMethods
```

```
  def fetch_from_twitter(user)
  end
```

```
  def cleanup
  end
```

```
end
end
```

```
class Image
  include ImageUtils
end
```

RUBY
BITS

ACTIVESUPPORT::CONCERN

gem install activesupport

```
require 'active_support/concern'

module ImageUtils

  extend ActiveSupport::Concern

  included do
    clean_up
  end

  module ClassMethods
    def fetch_from_twitter(user)
    end

    def clean_up
    end
  end
end
```

```
class Image
  include ImageUtils
end
```

included block is executed in
the context of the Image class

ActiveSupport::Concern looks
for a module named ClassMethods

RUBY
BITS

ACTIVESUPPORT::CONCERN

```
module ImageUtils
```

```
  def self.included(base) ◀ .....  
    base.extend(ClassMethods)  
  end
```

```
  module ClassMethods
```

```
    def clean_up; end  
  end
```

```
end
```

```
module ImageProcessing
```

```
  def self.included(base)
```

```
    base.clean_up ◀ .....
```

```
  end
```

```
end
```

```
class Image
```

```
  include ImageUtils
```

```
  include ImageProcessing
```

```
end
```



ImageProcessing depends
on ImageUtils

base is Image class

calls method on
Image class

RUBY
BITS

ACTIVESUPPORT::CONCERN

```
module ImageUtils
```

```
  def self.included(base) ◀ .....  
    base.extend(ClassMethods)  
  end
```

```
  module ClassMethods  
    def clean_up; end  
  end
```

```
end
```

```
module ImageProcessing
```

```
  include ImageUtils
```

```
  def self.included(base)  
    base.clean_up ◀ .....  
  end
```

```
end
```



```
class Image  
  include ImageProcessing  
end
```



base is ImageProcessing module

undefined method error

RUBY
BITS

ACTIVESUPPORT::CONCERN

```
module ImageUtils
  def self.included(base) ◀ .....
    base.extend(ClassMethods)
  end

  module ClassMethods
    def clean_up; end
  end
end

module ImageProcessing
  extend ActiveSupport::Concern
  include ImageUtils
  def self.included(base)
    base.clean_up
  end
end
```



```
class Image
  include ImageProcessing
end
```

Dependencies are properly resolved

base is Image class

RUBY
BITS

ACTIVESUPPORT::CONCERN

```
module ImageUtils
  extend ActiveSupport::Concern
  module ClassMethods
    def clean_up; end
  end
end
```

```
module ImageProcessing
```



```
  extend ActiveSupport::Concern
  include ImageUtils
  included do
    clean_up
  end
end
```

```
class Image
  include ImageProcessing
end
```

Dependencies are
properly resolved

RUBY
BITS

RUBY

bits

MODULES

PRESS START