```ruby
# rack p(params) messes up p
# so I changed it to ph (params hash)
# and puts loses array structure.
# so a little display patch to get new lines
def d(data)
  data.send :display
  "\n".send :display
end

# This might be useful some time
require "rack"

def ph(params)
  Rack::Utils.parse_nested_query(params)
end

d ph("id1[]=a&id2[]=b&id3[]=c")
d ph("id[][1]=a&id[][2]=b&id[][3]=c")

mg = [] # Module groups
tmp = [] # temporary array

modules = Enumerator.new do |yielder|
  Module.constants.sort.each do |m|
    yielder.yield m
  end
end

# The methods to call
def method_strings
  [
    ".methods",
    ".methods(false)",
    ".instance_methods",
    ".instance_methods(false)",
    ".public_methods",
    ".public_methods(false)",
    ".public_instance_methods",
    ".public_instance_methods(false)",
    ".protected_methods",
    ".protected_methods(false)",
    ".private_methods",
    ".private_methods(false)"
  ]
```

```
end

methods = Enumerator.new do |yielder|
  method_strings.each do |ms|
    yielder.yield ms
  end
end

modules.each do |m|
  methods.each do |s|
    begin
      ea = eval(m.to_s + s.to_s + ".sort")
      tmp << (m.to_s + s.to_s + ".sort") << ea
    rescue
      # make empty array if no method error for module.
      tmp << (m.to_s + s.to_s + ".sort") << []
    end
    # add either result to module group array.
    mg << tmp
    # clear the temp array
    tmp  = []
  end
end

# convert array to a hash
mg = mg.to_h

# Inspect some values of the given key strings(12 for each module)
# mg.keys.each {|key| p key}
d "Object.private_methods(false).sort"
d mg.fetch("Object.private_methods(false).sort")
d "Class.private_methods(false).sort"
d mg.fetch("Class.private_methods(false).sort")
d "Module.private_methods(false).sort"
d mg.fetch("Module.private_methods(false).sort")
d "Object.public_methods(false).sort"
d mg.fetch("Object.public_methods(false).sort")
d "Class.instance_methods(false).sort"
d mg.fetch("Class.instance_methods(false).sort")
d "Class.public_methods(false).sort"
mg.fetch("Class.public_methods(false).sort")
d "Module.public_methods(false).sort"
d mg.fetch("Module.public_methods(false).sort")
d "Module.methods(false).sort"
d mg.fetch("Module.methods(false).sort")
```

```ruby
d "Module.instance_methods(false).sort"
mg.fetch("Module.instance_methods(false).sort").each do |im|
  d im
end
puts "Object.methods.sort"
mg.fetch("Object.methods.sort").each do |om|
  d om
end

mg.fetch("Class.methods.sort").each do |cm|
  d cm
end
d "Object.instance_methods.sort"
mg.fetch("Object.instance_methods.sort").each do |im|
  d im
end
d "Object.protected_methods.sort"
d mg.fetch("Object.protected_methods.sort")
d "Class.protected_methods.sort"
d mg.fetch("Class.protected_methods.sort")
d "Module.protected_methods.sort"
d mg.fetch("Module.protected_methods.sort")
```