I'd like to know whether my struct was initialized with `keyword_init: true` or not. This information is useful when writing a deserializer (attached an example below).

```ruby
S1 = Struct.new(:a, :b, keyword_init: true)
S2 = Struct.new(:a, :b)

# Specialized for Struct
def serialize(d)
  d.to_h.merge(__class_name: d.class.name)
end

def deserialize(h)
  klass = Object.const_get(h.delete(__class_name))
  if keyword_init?(klass)
    # If the class is created with `keyword_init: true`, the parameter should be passed as keywords
    klass.new(**h)
  else
    # Otherwise, each values are passed in the order of members.
    klass.new(*klass.members.map { |sym| h[sym] })
  end
end

def keyword_init?(klass)
  # I don't want to do this...
  # `klass.keyword_init?` looks cool.
  klass.inspect.end_with?('({keyword_init: true})')
end

s1 = S1.new(a: 1, b: 2)
p s1
p s1_ = deserialize(serialize(s1))
p s1 == s1_

s2 = S2.new(1, 2)
p s2
p s2_ = deserialize(serialize(s2))
p s2 == s2_
```

Associated revisions

Revision 1a637544 - 07/15/2021 09:14 AM - hkdnet (Ko Sato)
struct.c: Add `keyword_init?` singleton method for StructClass (#4609)

Fixes [Feature #18008]

Revision 835c63cd - 07/15/2021 09:21 AM - naruse (Yui NARUSE)
Add tests and NEWS [Feature #18008]

Revision 3e7a7fb2 - 07/15/2021 01:24 PM - nobu (Nobuyoshi Nakada)
Make Struct#`keyword_init?` return nil by default [Feature #18008]
+1. While we aim to obviate `keyword_init` in [Feature #16806], unless we also intend to deprecate the option (for now we don't), we'd need to have the check to correctly write a deserializer of Structs.

#2 - 07/15/2021 09:11 AM - matz (Yukihiro Matsumoto)
Accepted.

Matz.

#3 - 07/15/2021 09:11 AM - naruse (Yui NARUSE)
hkdnet's patch is https://github.com/ruby/ruby/pull/4609

#4 - 07/15/2021 09:15 AM - hkdnet (Ko Sato)
- Status changed from Open to Closed

Applied in changeset gil1a637544166eca6b917fb6f32baeb771f4914b7a.

struct.c: Add `keyword_init?` singleton method for StructClass (#4609)

Fixes [Feature #18008]