# Best scenario

# Best case scenario

# Best case scenario

when life.gives_you lemon

Sandjiv @Yaquelqun

# case presentation

# case presentation

1. Just in case

# case presentation

1. Just in case
2. case implementation

# case presentation

1. Just in case
2. case implementation
3. use cases

# case presentation

1. Just in case
2. case implementation
3. use cases
4. Pattern matching

# Just in case

# Just in case

```
case status
```

# Just in case

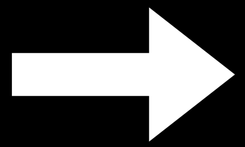
```
case status
when :success

when :error

else

end
```

# Just in case

```
case status
when :success
  process
when :error
  fail
else
  fail_harder
end
```

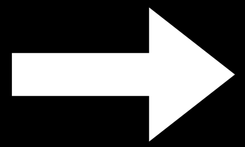
# Just in case

```
case status
when :success
  process
when :error
  fail
else
  fail_harder
end
```



# Just in case

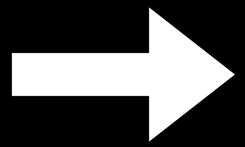
```
case status
when :success
  process
when :error
  fail
else
  fail_harder
end
```



```
case status
when :success then process
when :error then fail
else
  fail_harder
end
```

# Just in case

```
case status
when :success
  process
when :error
  fail
else
  fail_harder
end
```



```
case status
when :success then process
when :error, :timeout then fail
else
  fail_harder
end
```

# Just in case

```
case status
when :success
  process
when :error
  fail
else
  fail_harder
end
```

# Just in case

```
case
when :success
  process
when :error
  fail
else
  fail_harder
end
```

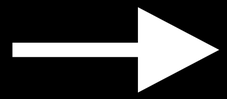
# Just in case

```
case
when status < 400
  process
when status ⩾ 400
  fail
else
  fail_harder
end
```

# case implementation

# case implementation

- Custom method



```
Task.new.method(:assigned?).source_location
# ["/Users/sandjiv/Workspace/ringtwice/app/models/tasks/has_state.rb", 148]
```

# case implementation

- Custom method → `Task.new.method(:assigned?).source_location`

```
# [“/Users/sandjiv/Workspace/ringtwice/app/models/tasks/has_state.rb", 148]
```

- Ruby method → RTFM 🤓

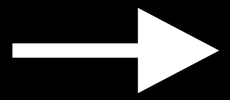
# case implementation

- Custom method → `Task.new.method(:assigned?).source_location`

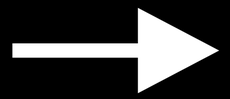
```
# ["/Users/sandjiv/Workspace/ringtwice/app/models/tasks/has_state.rb", 148]
```

- Ruby method → RTFM 🤓
- Ruby keyword → VM Instructions

# case implementation

- Ruby keyword  VM Instructions

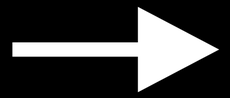
# case implementation

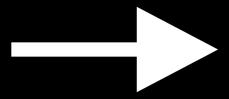
- Ruby keyword  VM Instructions

Ruby code

The one you write

# case implementation

- Ruby keyword  VM Instructions

Ruby code  Tokens

The one you write

Your code
but exploded

# `case` implementation

- Ruby keyword → VM Instructions

Ruby code → Tokens → AST

The one you write

Your code
but exploded

The exploded code
but formatted

# `case` implementation

- Ruby keyword → VM Instructions

Ruby code → Tokens → AST → VM Instructions

The one you write

Your code
but exploded

The exploded code
but formatted

The formatted steps
but translated

# case implementation

- Ruby keyword → VM Instructions

```
case status
when :success
  process
when :error
  fail
else
  fail_harder
end
```

```
== disasm: #<ISeq:<main>@./case.rb:1 (1,0)-(8,3)> (catch: FALSE)
0000 putself                                                  (   1)[Li]
0001 opt_send_without_block               <calldata!mid:status, argc:0, FCALL|VCALL|ARGS_SIMPLE>
0003 dup
0004 opt_case_dispatch                    <cdhash>, 23
0007 putobject                            :success            (   2)
0009 topn                                 1
0011 opt_send_without_block               <calldata!mid:===, argc:1, FCALL|ARGS_SIMPLE>
0013 branchif                             28
0015 putobject                            :error              (   4)
0017 topn                                 1
0019 opt_send_without_block               <calldata!mid:===, argc:1, FCALL|ARGS_SIMPLE>
0021 branchif                             33
0023 pop                                                      (   7)
0024 putself                              [Li]
0025 opt_send_without_block               <calldata!mid:fail_harder, argc:0, FCALL|VCALL|ARGS_SIMPLE>
0027 leave
0028 pop                                                      (   2)
0029 putself                                                  (   3)[Li]
0030 opt_send_without_block               <calldata!mid:proceed, argc:0, FCALL|VCALL|ARGS_SIMPLE>
0032 leave                                                    (   7)
0033 pop                                                      (   4)
0034 putself                                                  (   5)[Li]
0035 opt_send_without_block               <calldata!mid:fail, argc:0, FCALL|VCALL|ARGS_SIMPLE>
0037 leave                                                    (   7)
```

# case implementation

- Ruby keyword → VM Instructions

```
case status
when :success
  process
when :error
  fail
else
  fail_harder
```

```
== disasm: #<ISeq:<main>@./case.rb:1 (1,0)-(8,3)> (catch: FALSE)
0000 putself                                                  (   1)[Li]
0001 opt_send_without_block               <calldata!mid:status, argc:0, FCALL|VCALL|ARGS_SIMPLE>
0003 dup
0004 opt_case_dispatch                    <cdhash>, 23
```

# case implementation

- Ruby keyword → VM Instructions

```
case status
```

```
== disasm: #<ISeq:<main>@./case.rb:1 (1,0)-(8,3)> (catch: FALSE)
0000 putself                                                  (   1)[Li]
0001 opt_send_without_block                <calldata!mid:status, argc:0, FCALL|VCALL|ARGS_SIMPLE>
0003 dup
0004 opt_case_dispatch                     <cdhash>, 23
0007 putobject                             :success           (   2)
0009 topn                                  1
0011 opt_send_without_block                <calldata!mid:===, argc:1, FCALL|ARGS_SIMPLE>
0013 branchif                              28
0015 putobject                             :error             (   4)
0017 topn                                  1
0019 opt_send_without_block                <calldata!mid:===, argc:1, FCALL|ARGS_SIMPLE>
0021 branchif                              33
0023 pop                                                      (   7)
0024 putself                               [Li]
```

```
0009 topn                          1
0011 opt_send_without_block        <calldata!mid:===, argc:1, FCALL|ARGS_SIMPLE>
0013 branchif                      28
0015 putobject                     :error                      (   4)
0017 topn                          1
0019 opt_send_without_block        <calldata!mid:===, argc:1, FCALL|ARGS_SIMPLE>
0021 branchif                      33
0023 pop                                                       (   7)
0024 putself                       [Li]
0025 opt_send_without_block        <calldata!mid:fail_harder, argc:0, FCALL|VCALL|ARGS_SIM
0027 leave
0028 pop                                                       (   2)
0029 putself                                                   (   3)[Li]
0030 opt_send_without_block        <calldata!mid:proceed, argc:0, FCALL|VCALL|ARGS_SIMPLE>
0032 leave                                                     (   7)
0033 pop                                                       (   4)
0034 putself                                                   (   5)[Li]
0035 opt_send_without_block        <calldata!mid:fail, argc:0, FCALL|VCALL|ARGS_SIMPLE>
0037 leave                                                     (   7)
```

```
end
```

# Use case

What implements “===“ ?

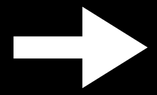
# Use case

What implements “===“ ?

- **Strings, Integers,  Float, Array…**

# Use case

What implements “===“ ?

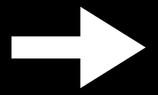
- **Strings, Integers,  Float, Array…** 

# Use case

What implements “===“ ?

**Checks equality**

- **Strings, Integers,  Float, Array…**



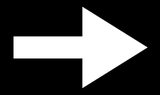
```
case params[:response]
```

# Use case

What implements “===“ ?

**Checks equality**

- **Strings, Integers,  Float, Array…**



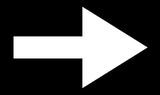
```
case params[:response]
when 200, 'success', :success
  'success'
```

# Use case

What implements “===“ ?

**Checks equality**

- **Strings, Integers,  Float, Array…**

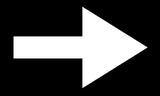


```
case params[:response]
when 200, 'success', :success
  'success'
when ['error1', 'error2']
  ‘errors array, really ?'
```

# Use case

What implements “===“ ?

- **Strings, Integers,  Float, Array…**

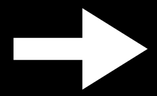


**Checks equality**

```ruby
case params[:response]
when 200, 'success', :success
  'success'
when ['error1', 'error2']
  ‘errors array, really ?'
when { status: 'error' }
  'whyy ??'
else
  'i give up'
end
```

# Use case

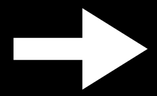
What implements “===“ ?

- Strings, Integers,  Float, Array…
- **Classes, Modules …** 

# Use case

What implements “===“ ?

- Strings, Integers,  Float, Array…
- **Classes, Modules …**



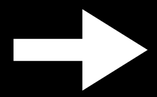
**Checks type/ancestry**

```
case error
```

# Use case

What implements “===“ ?

- Strings, Integers,  Float, Array…
- **Classes, Modules …**



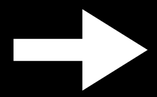
**Checks type/ancestry**

```
case error
when Ignorable
  head :ok
```

# Use case

What implements “===“ ?

- Strings, Integers,  Float, Array…
- **Classes, Modules …**



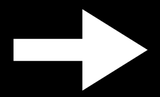
**Checks type/ancestry**

```
case error
when Ignorable
  head :ok
when RecordNotFound, Unauthorized
  render :not_found
```

# Use case

What implements “===“ ?

- Strings, Integers,  Float, Array…
- **Classes, Modules …**



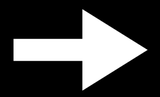
**Checks type/ancestry**

```
case error
when Ignorable
  head :ok
when RecordNotFound, Unauthorized
  render :not_found
when NoMethodError
  'who forgot about safe navigation ?’
```

# Use case

What implements “===“ ?

- Strings, Integers,  Float, Array…
- **Classes, Modules …**



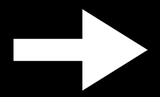
**Checks type/ancestry**

```
case error
when Ignorable
  head :ok
when RecordNotFound, Unauthorized
  render :not_found
when NoMethodError
  'who forgot about safe navigation ?’
when StandardError
  raise error
```

# Use case

What implements “===“ ?

- Strings, Integers,  Float, Array…
- **Classes, Modules …**



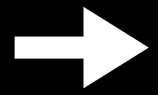
**Checks type/ancestry**

```
case error
when Ignorable
  head :ok
when RecordNotFound, Unauthorized
  render :not_found
when NoMethodError
  'who forgot about safe navigation ?’
when StandardError
  raise error
else
  'you have been cursed by an unknown
  error. Forward this message to 20
  admins before midnight
  or face untold consequences'
end
```

# Use case

What implements “===“ ?

- Strings, Integers,  Float, Array…
- Classes, Modules …
- **Ranges, IPAddr**

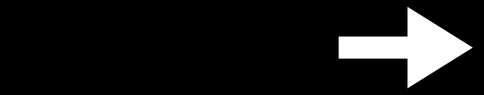


# Use case

What implements “===“ ?

**Checks inclusion**

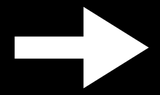
- Strings, Integers,  Float, Array…
- Classes, Modules …
- **Ranges, IPAddr**



# Use case

What implements “===“ ?

- Strings, Integers,  Float, Array…
- Classes, Modules …
- **Ranges, IPAddr**



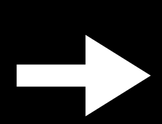
**Checks inclusion**

```
case feature_flag.user_count
when (..30_000)
  puts 'flop'
when (100_000..)
  puts 'complete_success'
else
  puts 'standard engagement'
end
```

# Use case

What implements “===“ ?

- Strings, Integers,  Float, Array…
- Classes, Modules …
- **Ranges, IPAddr**



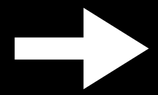
**Checks inclusion**

```
case request.ip
when IPAddr.new("192.168.1.1/25")
  request.tag(vlan: :guests)
when IPAddr.new("192.168.1.128/25")
  request.tag(vlan: :pro)
else
  raise 'innapropriate ip'
end
```

# Use case

What implements “===“ ?

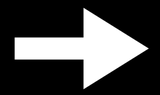
- Strings, Integers,  Float, Array…
- Classes, Modules …
- Ranges, IPAddr
- **Regex**



# Use case

What implements “===“ ?

- Strings, Integers,  Float, Array…
- Classes, Modules …
- Ranges, IPAddr
- **Regex**



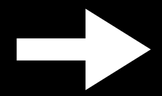
**Checks match**

```
case message
when EMAIL_REGEX
  'email_detected'
when ADDRESS_REGEX
  check_country(message)
when INNAPROPRIATE_REGEX
  message.user.ban_account
end
```

# Use case

What implements “===“ ?

- Strings, Integers,  Float, Array…
- Classes, Modules …
- Ranges, IPAddr
- Regex
- **Procs, lambdas**

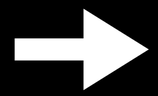


# Use case

What implements “===“ ?

**Checks execution**

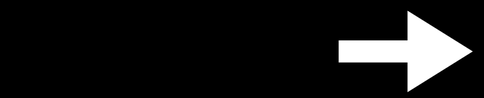
- Strings, Integers,  Float, Array…
- Classes, Modules …
- Ranges, IPAddr
- Regex
- **Procs, lambdas**



# Use case

What implements “===“ ?

- Strings, Integers,  Float, Array…
- Classes, Modules …
- Ranges, IPAddr
- Regex
- **Procs, lambdas**



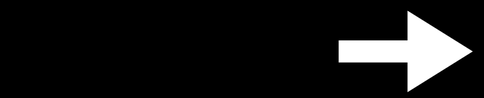
**Checks execution**

```
unknown_host = → { !check_host(_1) }
```

# Use case

What implements “===“ ?

- Strings, Integers,  Float, Array…
- Classes, Modules …
- Ranges, IPAddr
- Regex
- **Procs, lambdas**



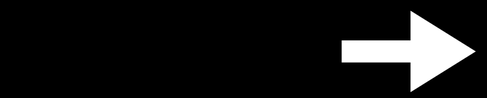
**Checks execution**

```
unknown_host = → { !check_host(_1) }
unknown_action → { !check_action(_1) }
```

# Use case

What implements “===“ ?

- Strings, Integers,  Float, Array…
- Classes, Modules …
- Ranges, IPAddr
- Regex
- **Procs, lambdas**



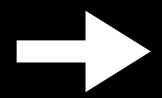
**Checks execution**

```
unknown_host = → { !check_host(_1) }
unknown_action → { !check_action(_1) }
whitelisted = lambda do
```

# Use case

What implements “===“ ?

- Strings, Integers,  Float, Array…
- Classes, Modules …
- Ranges, IPAddr
- Regex
- **Procs, lambdas**



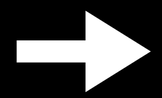
**Checks execution**

```
unknown_host = → { !check_host(_1) }
unknown_action → { !check_action(_1) }
whitelisted = lambda do
  !(unknown_host(_1) || unknown_action(_1))
```

# Use case

What implements “===“ ?

- Strings, Integers,  Float, Array…
- Classes, Modules …
- Ranges, IPAddr
- Regex
- **Procs, lambdas**



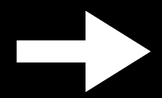
**Checks execution**

```
unknown_host = → { !check_host(_1) }
unknown_action → { !check_action(_1) }
whitelisted = lambda do
  !(unknown_host(_1) || unknown_action(_1))
end
```

# Use case

What implements “===“ ?

- Strings, Integers,  Float, Array…
- Classes, Modules …
- Ranges, IPAddr
- Regex
- **Procs, lambdas**



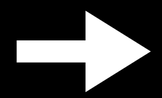
**Checks execution**

```
unknown_host = → { !check_host(_1) }
unknown_action → { !check_action(_1) }
whitelisted = lambda do
  !(unknown_host(_1) || unknown_action(_1))
end
```

# Use case

What implements “===“ ?

- Strings, Integers,  Float, Array…
- Classes, Modules …
- Ranges, IPAddr
- Regex
- **Procs, lambdas**



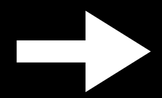
**Checks execution**

```
unknown_host = → { !check_host(_1) }
unknown_action → { !check_action(_1) }
whitelisted = lambda do
  !(unknown_host(_1) || unknown_action(_1))
end

case webhook
```

# Use case

What implements “===“ ?

- Strings, Integers,  Float, Array…
- Classes, Modules …
- Ranges, IPAddr
- Regex
- **Procs, lambdas**



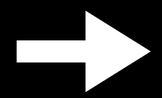
**Checks execution**

```
unknown_host = → { !check_host(_1) }
unknown_action → { !check_action(_1) }
whitelisted = lambda do
  !(unknown_host(_1) || unknown_action(_1))
end

case webhook
when whitelisted then 'ok'
```

# Use case

What implements “===“ ?

- Strings, Integers,  Float, Array…
- Classes, Modules …
- Ranges, IPAddr
- Regex
- **Procs, lambdas**



**Checks execution**

```
unknown_host = → { !check_host(_1) }
unknown_action → { !check_action(_1) }
whitelisted = lambda do
  !(unknown_host(_1) || unknown_action(_1))
end

case webhook
when whitelisted then 'ok'
when unknown_host then 'who?'
```

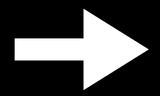
# Use case

What implements “===“ ?

- Strings, Integers,  Float, Array…
- Classes, Modules …
- Ranges, IPAddr
- Regex
- **Procs, lambdas**

**Checks execution**

```
unknown_host = → { !check_host(_1) }
unknown_action → { !check_action(_1) }
whitelisted = lambda do
  !(unknown_host(_1) || unknown_action(_1))
end
```



```
case webhook
when whitelisted then 'ok'
when unknown_host then 'who?'
when unknown_action then 'what?'
```

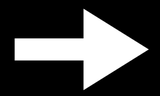
# Use case

What implements “===“ ?

- Strings, Integers,  Float, Array…
- Classes, Modules …
- Ranges, IPAddr
- Regex
- **Procs, lambdas**

**Checks execution**

```
unknown_host = → { !check_host(_1) }
unknown_action → { !check_action(_1) }
whitelisted = lambda do
  !(unknown_host(_1) || unknown_action(_1))
end
```

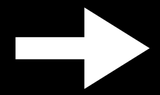


```
case webhook
when whitelisted then 'ok'
when unknown_host then 'who?'
when unknown_action then 'what?'
end
```

# Use case

What implements “===“ ?

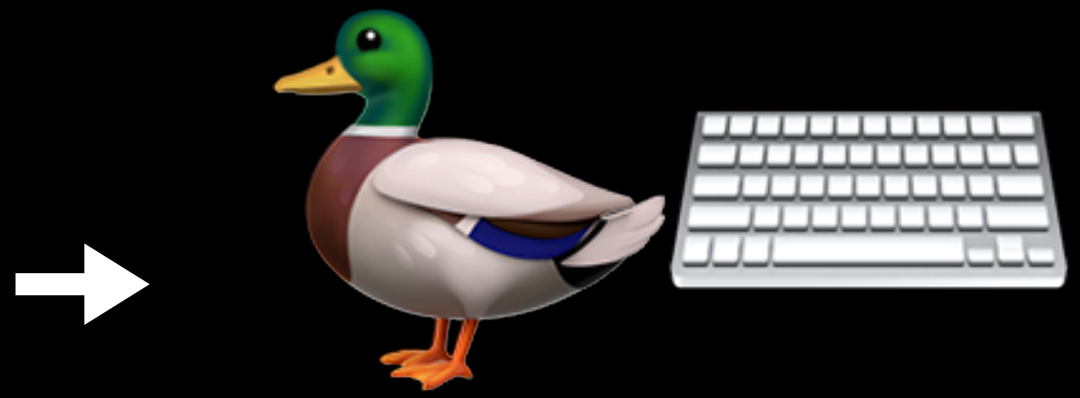
- Strings, Integers,  Float, Array…
- Classes, Modules …
- Ranges, IPAddr
- Regex
- Procs, lambdas
- **Everything else**



# Use case

What implements “===“ ?

- Strings, Integers,  Float, Array…
- Classes, Modules …
- Ranges, IPAddr
- Regex
- Procs, lambdas
- **Everything else**



# Use case

# Use case

```ruby
module Responses
  class Success
    def ===(obj)
      obj.status == :success
    end
  end
end
```

# Use case

```ruby
module Responses
  class Success
    def ===(obj)
      obj.status == :success
    end
  end
end

module Responses
  class Error
    def ===(obj)
      obj.status == :error && obj.errors.any?
    end
  end
end
```

# Use case

```ruby
module Responses
  class Success
    def ===(obj)
      obj.status == :success
    end
  end
end

module Responses
  class Error
    def ===(obj)
      obj.status == :error && obj.errors.any?
    end
  end
end
```

```ruby
case response
when Responses::Success.new
  proceed(response.body)
when Responses::Error.new
  render_errors(response.errors)
end
```

# Pattern Matching

# Pattern Matching

```
{
  status: 200,
  body: {
    user: {
      name: name,
      age: age,
    }
  }
}
```

# Pattern Matching

```
{
  status: 200,
  body: {
    user: {
      name: name,
      age: age,
    }
  }
}
```

```
{
  status: 200,
  body: {
    user: {
      name: 'Roger',
      age: 48,
    }
  }
}
```

# Pattern Matching

```
{
  status: 200,
  body: {
    user: {
      name: name,
      age: age,
    }
  }
}
```

```
{
  status: 200,
  body: {
    user: {
      name: name,
      age: age,
    }
  }
}
```

```
name = 'Roger'
age = 48
```

# Pattern Matching

```
{
  status: 200,
  body: {
    user: {
      name: name,
      age: age,
    }
  }
}
```

# Pattern Matching

```
{
  status: 200,
  body: {
    user: {
      name: name,
      age: age,
    }
  }
}
```

```
{
  status: 401
  body: {
    errors: ['User must be logged in'
  }
}
```

# Pattern Matching

```
{
  status: 200,
  body: {
    user: { name: name, age: age }
  }
}
```

```
{
  status: 422,
  body: { errors: [ 'User must be logged in' ] }
}
```

# Pattern Matching: simple use case

# Pattern Matching: simple use case

```
case response
```

# Pattern Matching: simple use case

```
case response
in { body: success_body, status: :success }

in { message: error_message, status: :error }

else

end
```

# Pattern Matching: simple use case

```
case response
in { body: success_body, status: :success }
  process(success_body)
in { message: error_message, status: :error }
  fail(error_message)
else
  fail_harder
end
```

# Pattern Matching: simple use case

```
case response
in { body: success_body, status: :success } unless ongoing_maintenance?
  process(success_body)
in { message: error_message, status: :error }
  fail(error_message)
else
  fail_harder
end
```

# Pattern Matching: finding and pinning values

# Pattern Matching: finding and pinning values

```
id = 69
```

# Pattern Matching: finding and pinning values

```
id = 69
users = [{id: 1, admin: true, name: 'joe'}, {id: 69, admin: true, name: 'roger'}, ...]
```

# Pattern Matching: finding and pinning values

```
id = 69
users = [{id: 1, admin: true, name: 'joe'}, {id: 69, admin: true, name: 'roger'}, ...]
case users
```

# Pattern Matching: finding and pinning values

```
id = 69
users = [{id: 1, admin: true, name: 'joe'}, {id: 69, admin: true, name: 'roger'}, ...]
case users
in [*somewhere, {id: ^id, admin: admin, **keys}, *in_there]
```

# Pattern Matching: finding and pinning values

```
id = 69
users = [{id: 1, admin: true, name: 'joe'}, {id: 69, admin: true, name: 'roger'}, ...]
case users
in [*somewhere, {id: ^id, admin: admin, **keys}, *in_there]
  puts admin
```

# Pattern Matching: finding and pinning values

```
id = 69
users = [{id: 1, admin: true, name: 'joe'}, {id: 69, admin: true, name: 'roger'}, ...]
case users
in [*somewhere, {id: ^id, admin: admin, **keys}, *in_there]
  puts admin
else
```

# Pattern Matching: finding and pinning values

```
id = 69
users = [{id: 1, admin: true, name: 'joe'}, {id: 69, admin: true, name: 'roger'}, ...]
case users
in [*somewhere, {id: ^id, admin: admin, **keys}, *in_there]
  puts admin
else
  puts 'not found'
```

# Pattern Matching: finding and pinning values

```
id = 69
users = [{id: 1, admin: true, name: 'joe'}, {id: 69, admin: true, name: 'roger'}, ...]
case users
in [*somewhere, {id: ^id, admin: admin, **keys}, *in_there]
  puts admin
else
  puts 'not found'
end
```

# Pattern Matching: How does it work ?

# Pattern Matching: How does it work ?

# Pattern Matching: How does it work ?

```
== disasm: #<ISeq:<main>@case.rb:1 (1,0)-(118,3)> (catch: FALSE)
0000 putnil                                                           ( 114)[Li]
0001 putself                                                          ( 113)
0002 opt_send_without_block                <calldata!mid:response, argc:0, FCALL|VCALL|ARGS_SIMPLE>
0004 dup                                                              ( 114)
0005 dup
0006 putobject                             :deconstruct_keys
0008 opt_send_without_block                <calldata!mid:respond_to?, argc:1, ARGS_SIMPLE>
0010 branchunless                          65
0012 duparray                              [:status, :body]
0014 opt_send_without_block                <calldata!mid:deconstruct_keys, argc:1, ARGS_SIMPLE>
0016 dup
0017 checktype                             T_HASH
0019 branchunless                          56
0021 dup
0022 putobject                             :status
0024 opt_send_without_block                <calldata!mid:key?, argc:1, ARGS_SIMPLE>
0026 branchunless                          65
0028 dup
0029 putobject                             :body
0031 opt_send_without_block                <calldata!mid:key?, argc:1, ARGS_SIMPLE>
0033 branchunless                          65
0035 dup
0036 putobject                             :status
0038 opt_aref                              <calldata!mid:[], argc:1, ARGS_SIMPLE>[CcCr]
0040 putobject                             :success
0042 checkmatch                            2
0044 branchunless                          65
0046 dup
0047 putobject                             :body
0049 opt_aref                              <calldata!mid:[], argc:1, ARGS_SIMPLE>[CcCr]
0051 setlocal_WC_0                         response_body@0
0053 pop
0054 jump                                  140
0056 putspecialobject                      1
0058 putobject                             TypeError
0060 putobject                             "deconstruct_keys must return Hash"
0062 opt_send_without_block                <calldata!mid:core#raise, argc:2, ARGS_SIMPLE>
0064 pop
0065 pop
0066 dup                                                              ( 116)
0067 dup
```

# Pattern Matching: How does it work ?

```
0012 duparray                   [:status, :body]
0014 opt_send_without_block     <calldata!mid:deconstruct_keys, argc:1, ARGS_SIMPLE>
0016 dup
0017 checktype                  T_HASH
0019 branchunless               56
0021 dup
0022 putobject                  :status
0024 opt_send_without_block     <calldata!mid:key?, argc:1, ARGS_SIMPLE>
0026 branchunless               65
0028 dup
0029 putobject                  :body
0031 opt_send_without_block     <calldata!mid:key?, argc:1, ARGS_SIMPLE>
0033 branchunless               65
0035 dup
0036 putobject                  :status
0038 opt_aref                   <calldata!mid:[], argc:1, ARGS_SIMPLE>[CcCr]
0040 putobject                  :success
0042 checkmatch                 2
0044 branchunless               65
0046 dup
0047 putobject                  :body
0049 opt_aref                   <calldata!mid:[], argc:1, ARGS_SIMPLE>[CcCr]
0051 setlocal_WC_0              response_body@0
0053 pop
0054 jump                       140
0056 putspecialobject           1
0058 putobject                  TypeError
0060 putobject                  "deconstruct_keys must return Hash"
0062 opt_send_without_block     <calldata!mid:core#raise, argc:2, ARGS_SIMPLE>
0064 pop
0065 pop
0066 dup                                    ( 116)
0067 dup
0068 putobject                  :deconstruct_keys
0070 opt_send_without_block     <calldata!mid:respond_to?, argc:1, ARGS_SIMPLE>
0072 branchunless               127
0074 duparray                   [:status, :errors]
0076 opt_send_without_block     <calldata!mid:deconstruct_keys, argc:1, ARGS_SIMPLE>
0078 dup
0079 checktype                  T_HASH
0081 branchunless               118
0083 dup
0084 putobject                  :status
0086 opt_send_without_block     <calldata!mid:key?, argc:1, ARGS_SIMPLE>
0088 branchunless               127
0090 dup
0091 putobject                  :errors
0093 opt_send_without_block     <calldata!mid:key?, argc:1, ARGS_SIMPLE>
0095 branchunless               127
```

# Pattern Matching: Custom matching

# Pattern Matching: Custom matching

```
class Location
  def deconstruct_keys(_)
    {
      latitude:self.latitude,
      longitude: self.longitude
    }
  end
end
```

# Pattern Matching: Custom matching

```ruby
class Location
  def deconstruct_keys(_)
    {
      latitude:self.latitude,
      longitude: self.longitude
    }
  end
end
```

```ruby
case location
in { latitude: (0..90) ⇒ latitude }
  puts "latitude: #{latitude}"
  northern_hemisphere(location)
else
  southern_hemisphere(location)
end
```

# That’s it

Thank you very much for listening 🙏