DRY RUBY

Who am I?

Ruby developer @ironin

Functional Miners host

@mentero

artur@codebakery.eu

Who am I?

Ruby developer @ironin

Functional Miners host

@mentero

artur@codebakery.eu

Who am I?

Ruby developer @ironin

Functional Miners host

@mentero

artur@codebakery.eu

Who am I?

Ruby developer @ironin

Functional Miners host

@mentero

artur@codebakery.eu

Let’s start!

Won’t talk about

Don’t repeat yourself

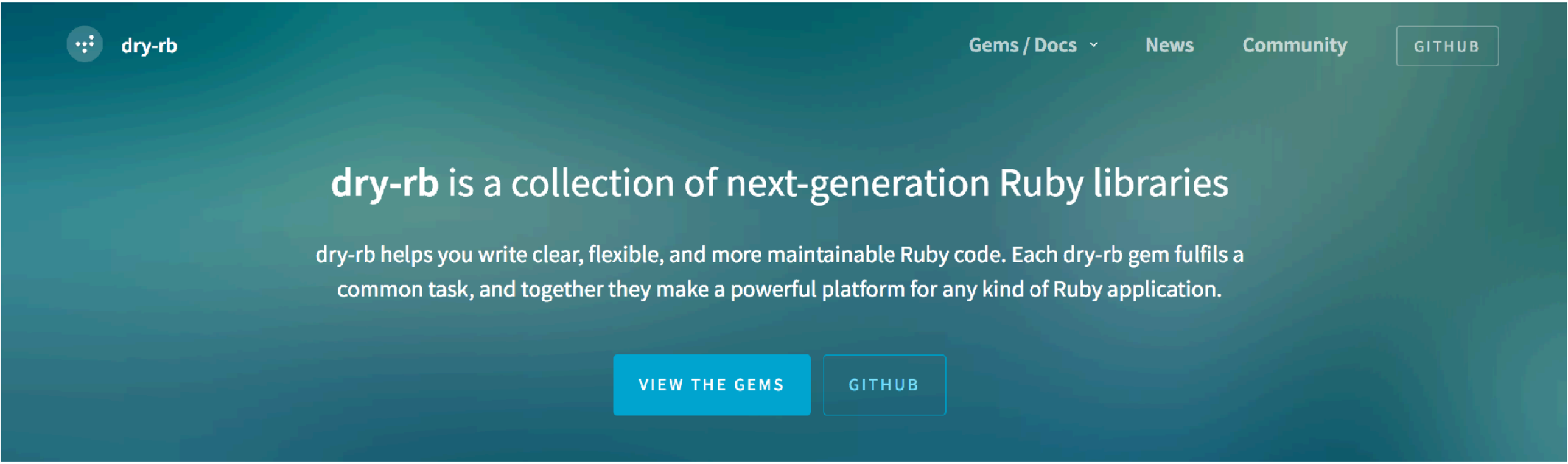
Or rather

Won’t talk about how to have less code

We will talk about

Dry-rb

Dry-rb?


dry-rb
Gems / Docs
News
Community
GITHUB
dry-rb is a collection of next-generation Ruby libraries
dry-rb helps you write clear, flexible, and more maintainable Ruby code. Each dry-rb gem fulfils a common task, and together they make a powerful platform for any kind of Ruby application.
VIEW THE GEMS
GITHUB

dry-rb

dry-rb is a collection of next-generation Ruby libraries, each intended to encapsulate a common task

http://dry-rb.org

Repositories 28 | People 9

Search repositories... | Type: All | Language: All

dry-inflector

Inflector for Ruby

ruby | dry-rb | inflection

Ruby ★ 23 ⑂ 3 MIT Updated 17 hours ago

dry-struct

Typed struct and value objects

ruby | data | types | constraints | dry-rb | type-safety

Ruby ★ 89 ⑂ 17 MIT 2 issues need help Updated 2 days ago

dry-schema

Schema validation DSL on top of dry-logic

dry-rb

Top languages

Ruby CSS

Most used topics

dry-rb | ruby | ioc | constraints | dependency-injection

People 9 >

Gems GitHub Gitter

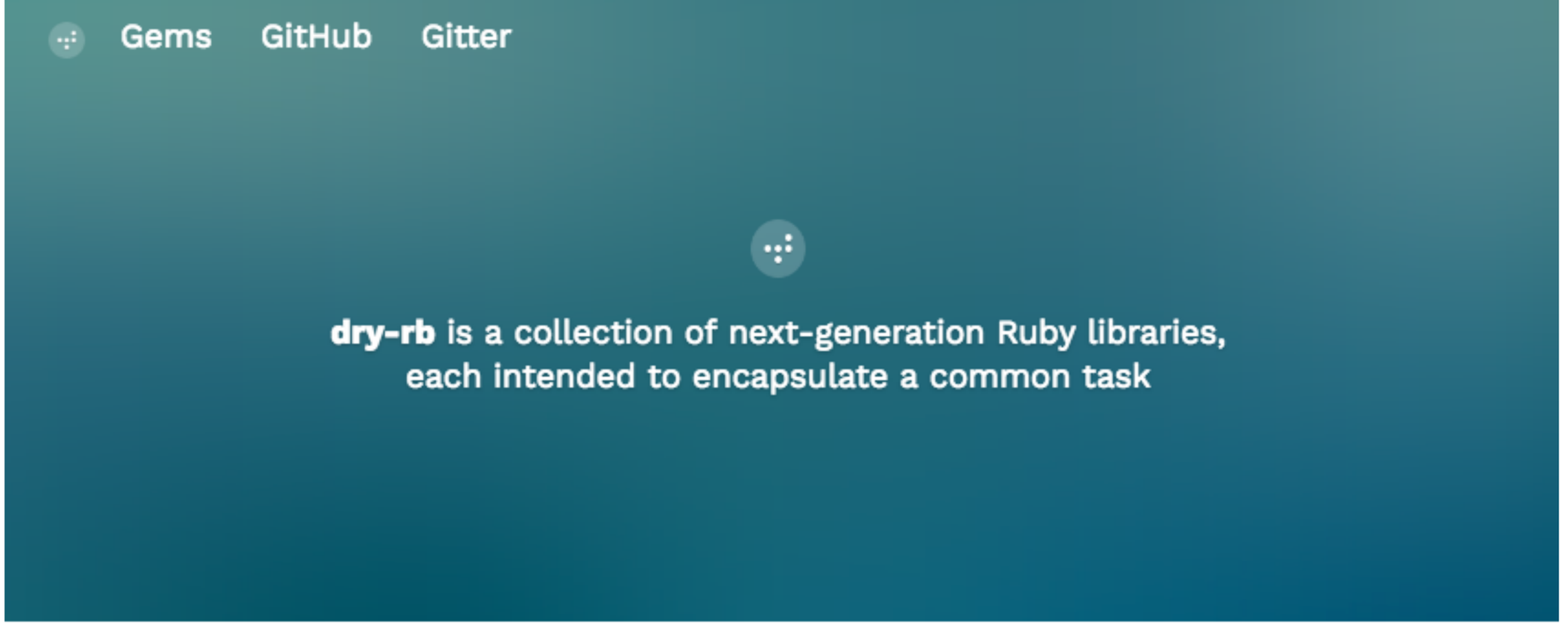
dry-rb is a collection of next-generation Ruby libraries,
each intended to encapsulate a common task

Libraries belonging to the dry-rb organisation try to follow a few basic principles

Non-intrusive

Flexible

Fast!



Libraries belonging to the dry-rb organisation try to follow a few basic principles

Non-intrusive

Flexible

Fast!

WHY ARE YOU NOT USING THIS AWESOME MONKEY-PATCHED FEATURE I JUST PREPARED FOR YOU!

Typical Rails Guy

NON-INTRUSIVE

SORRY, BUT YOU CANNOT CHANGE THAT.

Highly coupled code

FLEXIBILITY

CAN I GO GRAB A COFFEE BEFORE PRINTING RESULTS BACK TO YOU?

Slow and shameless code

FAST

FAST

Do you want numbers?

```
~/C/P/R/dry-validation (master|✔) $ be ruby benchmarks/benchmark_form_invalid.rb
#<Dry::Validation::Result output={:email=>nil, :age=>18} errors={:email=>["must be filled"], :age=>["must be greater than 18"]}>
#<Dry::Validation::Result output={:email=>"", :age=>18} errors={:email=>["must be filled"], :age=>["must be greater than 18"]}>
false
Warming up --------------------------------------
ActiveModel::Validations
                        113.000  i/100ms
dry-validation / schema
                        634.000  i/100ms
dry-validation / form
                        568.000  i/100ms
Calculating -------------------------------------
ActiveModel::Validations
                          1.193k (±12.7%) i/s -      5.989k in   5.124050s
dry-validation / schema
                          6.340k (±12.9%) i/s -     31.066k in   5.010648s
dry-validation / form
                          5.759k (±14.0%) i/s -     28.400k in   5.097339s

Comparison:
dry-validation / schema:     6339.9 i/s
dry-validation / form:     5759.0 i/s - same-ish: difference falls within error
ActiveModel::Validations:     1193.3 i/s - 5.31x  slower
```

Let’s take a look at few gems

dry-types
dry-struct

```ruby
module Types
  include Dry::Types.module
end
```

```ruby
module Types
  include Dry::Types.module
end
```

```
~/C/P/R/dryrb-test (master|✚2…) $ bundle console
[1] pry(main)> Types::Coercible::String["foo"]
=> "foo"
[2] pry(main)> Types::Coercible::String[1]
=> "1"
[3] pry(main)> Types::Coercible::String[:foo]
=> "foo"
```

```ruby
module Types
  include Dry::Types.module
end
```

```
~/C/P/R/dryrb-test (master|✚2…) $ bundle console
[1] pry(main)> Types::Coercible::String["foo"]
=> "foo"
[2] pry(main)> Types::Coercible::String[1]
=> "1"
[3] pry(main)> Types::Coercible::String[:foo]
=> "foo"
[4] pry(main)> Types::Strict::String["foo"]
=> "foo"
[5] pry(main)> Types::Strict::String[1]
Dry::Types::ConstraintError: 1 violates constraints (type?(String, 1) failed)
from /Users/mentero/Code/Playground/RRUG/dryrb-test/.bundle/gems/gems/dry-types-0.12
ed.rb:28:in `block in call'
[6] pry(main)> Types::Strict::String[:foo]
Dry::Types::ConstraintError: :foo violates constraints (type?(String, :foo) failed)
from /Users/mentero/Code/Playground/RRUG/dryrb-test/.bundle/gems/gems/dry-types-0.12
ed.rb:28:in `block in call'
```

```ruby
module Types
  include Dry::Types.module

  Email = String.constrained(format: /\A[\w+\-.]+@[a-z\d\-]+(\.[a-z]+)*\.[a-z]+\z/i)
  Age   = Int.constrained(gt: 18)
end
```

```ruby
module Types
  include Dry::Types.module

  Email = String.constrained(format: /\A[\w+\-.]+@[a-z\d\-]+(\.[a-z]+)*\.[a-z]+\z/i)
  Age   = Int.constrained(gt: 18)
end
```

```
~/C/P/R/dryrb-test (master|✚2…) $ bundle console
[1] pry(main)> Types::Age.call(27)
=> 27
[2] pry(main)> Types::Age.call(10)
Dry::Types::ConstraintError: 10 violates constraints (gt?(18, 10) failed)
from /Users/mentero/Code/Playground/RRUG/dryrb-test/.bundle/gems/gems/dry-
ed.rb:28:in `block in call'
```

```ruby
module Types
  include Dry::Types.module

  Email = String.constrained(format: /\A[\w+\-.]+@[a-z\d\-]+(\.[a-z]+)*\.[a-z]+\z/i)
  Age   = Int.constrained(gt: 18)
end
```

```
[3] pry(main)> Types::Email.call("artur@codebakery.eu")
=> "artur@codebakery.eu"
[4] pry(main)> Types::Email.call("artur<>codebakery.eu")
Dry::Types::ConstraintError: "artur<>codebakery.eu" violates constraints (
-z]+)*\.[a-z]+\z/i, "artur<>codebakery.eu") failed)
from /Users/mentero/Code/Playground/RRUG/dryrb-test/.bundle/gems/gems/dry-
ed.rb:28:in `block in call'
[5] pry(main)> Types::Email.call(3)
TypeError: no implicit conversion of Fixnum into String
from /Users/mentero/Code/Playground/RRUG/dryrb-test/.bundle/gems/gems/dry-
.rb:187:in `match'
```

Look ma, types everywhere

Built-in Types

Built-in types are grouped under 5 categories:

- `definition` - base type definitions with primitive class and options
- `strict` - constrained types with a primitive type check applied to input
- `coercible` - types with constructors using kernel coercions
- `form` - types with constructors performing non-strict coercions specific to HTTP params
- `json` - types with constructors performing non-strict coercions specific to JSON
- `maybe` - types accepting either nil or a specific primitive type

Structs on steroids

Structs on steroids

Virtus alternative

Structs on steroids

Virtus alternative

```ruby
class User < Dry::Struct
  attribute :name, Types::String
  attribute :email, Types::Email
  attribute :age, Types::Age
end
```

Structs on steroids

Virtus alternative

```ruby
class User < Dry::Struct
  attribute :name, Types::String
  attribute :email, Types::Email
  attribute :age, Types::Age
end
```

```ruby
class Message < Dry::Struct
  attribute :body, Types::String
  attribute :to, User
end
```

Structs on steroids

Virtus alternative

```ruby
class User < Dry::Struct
  attribute :name, Types::String
  attribute :email, Types::Email
  attribute :age, Types::Age
end
```

```ruby
class Message < Dry::Struct
  attribute :body, Types::String
  attribute :to, User
end
```

```ruby
class User < Dry::Struct
  attribute :name, Types::String
  attribute :email, Types::Email
  attribute :age, Types::Age
end
```

```ruby
class Message < Dry::Struct
  attribute :body, Types::String
  attribute :to, User
end
```

```ruby
class User < Dry::Struct
  attribute :name, Types::String
  attribute :email, Types::Email
  attribute :age, Types::Age
end
```

```ruby
class Message < Dry::Struct
  attribute :body, Types::String
  attribute :to, User
end
```

```
~/C/P/R/dryrb-test (master|✚2…) $ bundle console
[1] pry(main)> user = User.call(name: 'Artur', email: 'artur@codebakery.eu', age: 27)
=> #<User name="Artur" email="artur@codebakery.eu" age=27>
[2] pry(main)> message = Message.call(body: 'Lorem Ipsum', to: user)
=> #<Message body="Lorem Ipsum" to=#<User name="Artur" email="artur@codebakery.eu" age=27>>
```

```ruby
class User < Dry::Struct
  attribute :name, Types::String
  attribute :email, Types::Email
  attribute :age, Types::Age
end
```

```ruby
class Message < Dry::Struct
  attribute :body, Types::String
  attribute :to, User
end
```

```
~/C/P/R/dryrb-test (master|✚2…) $ bundle console
[1] pry(main)> user = User.call(name: 'Artur', email: 'artur@codebakery.eu', age: 27)
=> #<User name="Artur" email="artur@codebakery.eu" age=27>
[2] pry(main)> message = Message.call(body: 'Lorem Ipsum', to: user)
=> #<Message body="Lorem Ipsum" to=#<User name="Artur" email="artur@codebakery.eu" age=27>>
[3] pry(main)> message.to.email
=> "artur@codebakery.eu"
[4] pry(main)> user.name
=> "Artur"
[5] pry(main)> message.to_h
=> {:body=>"Lorem Ipsum", :to=>{:name=>"Artur", :email=>"artur@codebakery.eu", :age=>27}}
[6] pry(main)> message[:body]
=> "Lorem Ipsum"
```

```ruby
class User < Dry::Struct
  attribute :name, Types::String
  attribute :email, Types::Email
  attribute :age, Types::Age
end
```

```ruby
class Message < Dry::Struct
  attribute :body, Types::String
  attribute :to, User
end
```

```
[8] pry(main)> User.call(name: 42)
Dry::Struct::Error: [User.new] :email is missing in Hash input
from /Users/mentero/Code/Playground/RRUG/dryrb-test/.bundle/gems
terface.rb:98:in `rescue in new'
```

```ruby
class User < Dry::Struct
  attribute :name, Types::String
  attribute :email, Types::Email
  attribute :age, Types::Age
end
```

```ruby
class Message < Dry::Struct
  attribute :body, Types::String
  attribute :to, User
end
```

```
[9] pry(main)> User.call(name: 42, email: nil, age: nil)
Dry::Struct::Error: [User.new] nil (NilClass) has invalid type for :email violates constraints (format?(/\A[\w
+\-.]+@[a-z\d\-]+(\.[a-z]+)*\.[a-z]+\z/i, nil) failed)
from /Users/mentero/Code/Playground/RRUG/dryrb-test/.bundle/gems/gems/dry-struct-0.4.0/lib/dry/struct/class_in
terface.rb:98:in `rescue in new'
```

Other features

Optional Values

Other features

Optional values

```
[1] pry(main)> optional_string = Types::Strict::String.optional;
```

Other features

Optional values

```
[1] pry(main)> optional_string = Types::Strict::String.optional;
[2] pry(main)> optional_string["foo"]
=> "foo"
```

Other features

Optional values

```
[1] pry(main)> optional_string = Types::Strict::String.optional;
[2] pry(main)> optional_string["foo"]
=> "foo"
[3] pry(main)> optional_string[nil]
=> nil
```

Other features

Optional values

```
[1] pry(main)> optional_string = Types::Strict::String.optional;
[2] pry(main)> optional_string["foo"]
=> "foo"
[3] pry(main)> optional_string[nil]
=> nil
[4] pry(main)> optional_string[:foo]
Dry::Types::ConstraintError: :foo violates constraints (type?(String, :foo) failed)
from /Users/mentero/Code/Playground/RRUG/dryrb-test/.bundle/gems/gems/dry-types-0.1
:in `block in call'
```

Other features

Default Values

Other features

Default values

```
[1] pry(main)> PostStatus = Types::Strict::String.default('draft');
```

Other features

Default values

```
[1] pry(main)> PostStatus = Types::Strict::String.default('draft');
[2] pry(main)> PostStatus[nil]
=> "draft"
```

Other features

Default values

```
[1] pry(main)> PostStatus = Types::Strict::String.default('draft');
[2] pry(main)> PostStatus[nil]
=> "draft"
[3] pry(main)> PostStatus['published']
=> "published"
```

Other features

Default values

```
[1] pry(main)> PostStatus = Types::Strict::String.default('draft');
[2] pry(main)> PostStatus[nil]
=> "draft"
[3] pry(main)> PostStatus['published']
=> "published"
[4] pry(main)> PostStatus[2]
Dry::Types::ConstraintError: 2 violates constraints (type?(String, 2) failed)
from /Users/mentero/Code/Playground/RRUG/dryrb-test/.bundle/gems/gems/dry-type
ed.rb:28:in `block in call'
```

Other features

Arrays

Other features

Array

```
[1] pry(main)> Users = Types::Strict::Array.of(User);
```

Other features

Array

```
[1] pry(main)> Users = Types::Strict::Array.of(User);
[2] pry(main)> Users.call([1,2])
TypeError: can't convert Fixnum into Hash
from /Users/mentero/Code/Playground/RRUG/dryrb-test/.bundle/gems/gems/dry-types-0.12.2/lib/dry/types/construct
or.rb:47:in `Hash'
```

Other features

Array

```
[1] pry(main)> Users = Types::Strict::Array.of(User);
[2] pry(main)> Users.call([1,2])
TypeError: can't convert Fixnum into Hash
from /Users/mentero/Code/Playground/RRUG/dryrb-test/.bundle/gems/gems/dry-types-0.12.2/lib/dry/types/construct
or.rb:47:in `Hash'
[3] pry(main)> Users.call([{}, {}])
Dry::Types::ConstraintError: [{}, {}] violates constraints ([#<Dry::Types::Result::Failure input={} error="[Us
er.new] :name is missing in Hash input">, #<Dry::Types::Result::Failure input={} error="[User.new] :name is mi
ssing in Hash input">] failed)
from /Users/mentero/Code/Playground/RRUG/dryrb-test/.bundle/gems/gems/dry-types-0.12.2/lib/dry/types/constrain
ed.rb:28:in `block in call'
```

Other features

Array

```
[1] pry(main)> Users = Types::Strict::Array.of(User);
[2] pry(main)> Users.call([1,2])
TypeError: can't convert Fixnum into Hash
from /Users/mentero/Code/Playground/RRUG/dryrb-test/.bundle/gems/gems/dry-types-0.12.2/lib/dry/types/construct
or.rb:47:in `Hash'
[3] pry(main)> Users.call([{}, {}])
Dry::Types::ConstraintError: [{}, {}] violates constraints ([#<Dry::Types::Result::Failure input={} error="[Us
er.new] :name is missing in Hash input">, #<Dry::Types::Result::Failure input={} error="[User.new] :name is mi
ssing in Hash input">] failed)
from /Users/mentero/Code/Playground/RRUG/dryrb-test/.bundle/gems/gems/dry-types-0.12.2/lib/dry/types/constrain
ed.rb:28:in `block in call'
[4] pry(main)> Users.call([{name: 'Artur', email: 'artur@codebakery.eu', age: 27}, {name: 'Foo', email: 'foo@b
ar.com', age: 30}])

=> [#<User name="Artur" email="artur@codebakery.eu" age=27>, #<User name="Foo" email="foo@bar.com" age=30>]
```

Other features

Enums

Other features

Enum

```
[1] pry(main)> Statuses = Types::Strict::String.enum('draft', 'published', 'archived');
```

Other features

Enum

```
[1] pry(main)> Statuses = Types::Strict::String.enum('draft', 'published', 'archived');
[2] pry(main)> Statuses.values
=> ["draft", "published", "archived"]
```

Other features

Enum

```
[1] pry(main)> Statuses = Types::Strict::String.enum('draft', 'published', 'archived');
[2] pry(main)> Statuses.values
=> ["draft", "published", "archived"]
[3] pry(main)> Statuses[0]
=> "draft"
```

Other features

Enum

```
[1] pry(main)> Statuses = Types::Strict::String.enum('draft', 'published', 'archived');
[2] pry(main)> Statuses.values
=> ["draft", "published", "archived"]
[3] pry(main)> Statuses[0]
=> "draft"
[4] pry(main)> Statuses['draft']
=> "draft"
```

Other features

Enum

```
[1] pry(main)> Statuses = Types::Strict::String.enum('draft', 'published', 'archived');
[2] pry(main)> Statuses.values
=> ["draft", "published", "archived"]
[3] pry(main)> Statuses[0]
=> "draft"
[4] pry(main)> Statuses['draft']
=> "draft"
[5] pry(main)> Statuses[nil]
Dry::Types::ConstraintError: nil violates constraints (type?(String, nil) AND included_in?(["draft", "publishe
d", "archived"], nil) failed)
from /Users/mentero/Code/Playground/RRUG/dryrb-test/.bundle/gems/gems/dry-types-0.12.2/lib/dry/types/constrain
ed.rb:28:in `block in call'
```

Other features

Custom types?

Other features

Custom types

```ruby
module Types
  include Dry::Types.module

  Email = String.constrained(format: /\A[\w+\-.]+@[a-z\d\-]+(\.[a-z]+)*\.[a-z]+\z/i)
  Age = Int.constrained(gt: 18)

  # Money = Dry::Types::Definition.new(Money)
  #                               .constructor { |input| ::Money.new(input) }

  Money = Constructor(Money)
  GUID  = String
end
```

Other features

Custom types

```
[1] pry(main)> Types::Money.call(1000)
=> #<Money fractional:1000 currency:USD>
```

Other features

Custom types

```
[1] pry(main)> Types::Money.call(1000)
=> #<Money fractional:1000 currency:USD>
[2] pry(main)> Types::Money.call(Money.new(1000))
=> #<Money fractional:1000 currency:USD>
```

Other features

Custom types

```
[1] pry(main)> Types::Money.call(1000)
=> #<Money fractional:1000 currency:USD>
[2] pry(main)> Types::Money.call(Money.new(1000))
=> #<Money fractional:1000 currency:USD>
[3] pry(main)> Types::Money.call(true)
=> #<Money fractional:0 currency:USD>
```

Other features

Custom types

```
[1] pry(main)> Types::Money.call(1000)
=> #<Money fractional:1000 currency:USD>
[2] pry(main)> Types::Money.call(Money.new(1000))
=> #<Money fractional:1000 currency:USD>
[3] pry(main)> Types::Money.call(true)
=> #<Money fractional:0 currency:USD>
[4] pry(main)> class Transaction < Dry::Struct
[4] pry(main)*   attribute :amount, Types::Money
[4] pry(main)* end
=> Transaction
```

Other features

Custom types

```
[1] pry(main)> Types::Money.call(1000)
=> #<Money fractional:1000 currency:USD>
[2] pry(main)> Types::Money.call(Money.new(1000))
=> #<Money fractional:1000 currency:USD>
[3] pry(main)> Types::Money.call(true)
=> #<Money fractional:0 currency:USD>
[4] pry(main)> class Transaction < Dry::Struct
[4] pry(main)*   attribute :amount, Types::Money
[4] pry(main)* end
=> Transaction
[5] pry(main)> tx = Transaction.call(amount: 1000)
=> #<Transaction amount=#<Money fractional:1000 currency:USD>>
```

Other features

Custom types

```
[1] pry(main)> Types::Money.call(1000)
=> #<Money fractional:1000 currency:USD>
[2] pry(main)> Types::Money.call(Money.new(1000))
=> #<Money fractional:1000 currency:USD>
[3] pry(main)> Types::Money.call(true)
=> #<Money fractional:0 currency:USD>
[4] pry(main)> class Transaction < Dry::Struct
[4] pry(main)*   attribute :amount, Types::Money
[4] pry(main)* end
=> Transaction
[5] pry(main)> tx = Transaction.call(amount: 1000)
=> #<Transaction amount=#<Money fractional:1000 currency:USD>>
[6] pry(main)> tx2 = Transaction.call(amount: true)
=> #<Transaction amount=#<Money fractional:0 currency:USD>>
```

Other features

Constructor Types

Other features

Constructor Types

```ruby
class PermissiveTest < Dry::Struct
  constructor_type(:permissive) # default

  attribute :number, Types::Strict::Int.default(2)
  attribute :text,   Types::Strict::String.optional
end
```

Other features

Constructor Types

```
[1] pry(main)> PermissiveTest.call(number: 1, text: "foo")
=> #<PermissiveTest number=1 text="foo">
```

Other features

Constructor Types

```
[1] pry(main)> PermissiveTest.call(number: 1, text: "foo")
=> #<PermissiveTest number=1 text="foo">
[2] pry(main)> PermissiveTest.call(number: 1, text: nil)
=> #<PermissiveTest number=1 text=nil>
```

Other features

Constructor Types

```
[1] pry(main)> PermissiveTest.call(number: 1, text: "foo")
=> #<PermissiveTest number=1 text="foo">
[2] pry(main)> PermissiveTest.call(number: 1, text: nil)
=> #<PermissiveTest number=1 text=nil>
[3] pry(main)> PermissiveTest.call(number: nil, text: nil)
=> #<PermissiveTest number=2 text=nil>
```

Other features

Constructor Types

```
[1] pry(main)> PermissiveTest.call(number: 1, text: "foo")
=> #<PermissiveTest number=1 text="foo">
[2] pry(main)> PermissiveTest.call(number: 1, text: nil)
=> #<PermissiveTest number=1 text=nil>
[3] pry(main)> PermissiveTest.call(number: nil, text: nil)
=> #<PermissiveTest number=2 text=nil>
[4] pry(main)> PermissiveTest.call(number: nil, text: nil, foo: :bar)
=> #<PermissiveTest number=2 text=nil>
```

Other features

Constructor Types

```
[1] pry(main)> PermissiveTest.call(number: 1, text: "foo")
=> #<PermissiveTest number=1 text="foo">
[2] pry(main)> PermissiveTest.call(number: 1, text: nil)
=> #<PermissiveTest number=1 text=nil>
[3] pry(main)> PermissiveTest.call(number: nil, text: nil)
=> #<PermissiveTest number=2 text=nil>
[4] pry(main)> PermissiveTest.call(number: nil, text: nil, foo: :bar)
=> #<PermissiveTest number=2 text=nil>
[5] pry(main)> PermissiveTest.call(text: nil)
Dry::Struct::Error: [PermissiveTest.new] :number is missing in Hash input
from /Users/mentero/Code/Playground/RRUG/dryrb-test/.bundle/gems/gems/dry-st
b:98:in `rescue in new'
```

Other features

Constructor Types

```
[1] pry(main)> PermissiveTest.call(number: 1, text: "foo")
=> #<PermissiveTest number=1 text="foo">
[2] pry(main)> PermissiveTest.call(number: 1, text: nil)
=> #<PermissiveTest number=1 text=nil>
[3] pry(main)> PermissiveTest.call(number: nil, text: nil)
=> #<PermissiveTest number=2 text=nil>
[4] pry(main)> PermissiveTest.call(number: nil, text: nil, foo: :bar)
=> #<PermissiveTest number=2 text=nil>
[5] pry(main)> PermissiveTest.call(text: nil)
Dry::Struct::Error: [PermissiveTest.new] :number is missing in Hash input
from /Users/mentero/Code/Playground/RRUG/dryrb-test/.bundle/gems/gems/dry-st
b:98:in `rescue in new'
[6] pry(main)> PermissiveTest.call(number: :foo, text: "foo")
Dry::Struct::Error: [PermissiveTest.new] :foo (Symbol) has invalid type for
, :foo) failed)
from /Users/mentero/Code/Playground/RRUG/dryrb-test/.bundle/gems/gems/dry-st
b:98:in `rescue in new'
```

Other features

Constructor Types

```
class SchemaTest < Dry::Struct
  constructor_type(:schema)

  attribute :number, Types::Strict::Int.default(2)
  attribute :text,   Types::Strict::String.optional
end
```

Other features

Constructor Types

```
[1] pry(main)> SchemaTest.call(number: 1, text: "foo")
=> #<SchemaTest number=1 text="foo">
```

Other features

Constructor Types

```
[1] pry(main)> SchemaTest.call(number: 1, text: "foo")
=> #<SchemaTest number=1 text="foo">
[2] pry(main)> SchemaTest.call(number: 1, text: nil)
=> #<SchemaTest number=1 text=nil>
```

Other features

Constructor Types

```
[1] pry(main)> SchemaTest.call(number: 1, text: "foo")
=> #<SchemaTest number=1 text="foo">
[2] pry(main)> SchemaTest.call(number: 1, text: nil)
=> #<SchemaTest number=1 text=nil>
[3] pry(main)> SchemaTest.call(number: nil, text: nil)
=> #<SchemaTest number=2 text=nil>
```

Other features

Constructor Types

```
[1] pry(main)> SchemaTest.call(number: 1, text: "foo")
=> #<SchemaTest number=1 text="foo">
[2] pry(main)> SchemaTest.call(number: 1, text: nil)
=> #<SchemaTest number=1 text=nil>
[3] pry(main)> SchemaTest.call(number: nil, text: nil)
=> #<SchemaTest number=2 text=nil>
[4] pry(main)> SchemaTest.call(number: nil, text: nil, foo: :bar)
=> #<SchemaTest number=2 text=nil>
```

Other features

Constructor Types

```
[1] pry(main)> SchemaTest.call(number: 1, text: "foo")
=> #<SchemaTest number=1 text="foo">
[2] pry(main)> SchemaTest.call(number: 1, text: nil)
=> #<SchemaTest number=1 text=nil>
[3] pry(main)> SchemaTest.call(number: nil, text: nil)
=> #<SchemaTest number=2 text=nil>
[4] pry(main)> SchemaTest.call(number: nil, text: nil, foo: :bar)
=> #<SchemaTest number=2 text=nil>
[5] pry(main)> SchemaTest.call(text: nil)
=> #<SchemaTest number=2 text=nil>
```

Other features

Constructor Types

```
[1] pry(main)> SchemaTest.call(number: 1, text: "foo")
=> #<SchemaTest number=1 text="foo">
[2] pry(main)> SchemaTest.call(number: 1, text: nil)
=> #<SchemaTest number=1 text=nil>
[3] pry(main)> SchemaTest.call(number: nil, text: nil)
=> #<SchemaTest number=2 text=nil>
[4] pry(main)> SchemaTest.call(number: nil, text: nil, foo: :bar)
=> #<SchemaTest number=2 text=nil>
[5] pry(main)> SchemaTest.call(text: nil)
=> #<SchemaTest number=2 text=nil>
[6] pry(main)> SchemaTest.call(number: :foo, text: nil)
Dry::Struct::Error: [SchemaTest.new] :foo (Symbol) has invalid type for
oo) failed)
from /Users/mentero/Code/Playground/RRUG/dryrb-test/.bundle/gems/gems/dr
b:98:in `rescue_in new'
```


Validate meow!

dry-validation

```ruby
EMAIL_REGEX = /\A[\w+\-.]+@[a-z\d\-]+(\.[a-z]+)*\.[a-z]+\z/i

UserSchema = Dry::Validation.Schema do
  required(:name).filled

  required(:email).filled(format?: EMAIL_REGEX)

  required(:age).maybe(:int?)

  required(:address).schema do
    required(:street).filled
    required(:city).filled
    required(:zipcode).filled
  end
end
```

```ruby
EMAIL_REGEX = /\A[\w+\-.]+@[a-z\d\-]+(\.[a-z]+)*\.[a-z]+\z/i

UserSchema = Dry::Validation.Schema do
  required(:name).filled

  required(:email).filled(format?: EMAIL_REGEX)

  required(:age).maybe(:int?)

  required(:address).schema do
    required(:street).filled
    required(:city).filled
    required(:zipcode).filled
  end
end
```

```
[1] pry(main)> UserSchema.call(name: 'Jane', email: 'jane@doe.org', address: {
[1] pry(main)*     street: 'Street 1', city: 'NYC', zipcode: '1234'
[1] pry(main)* })
=> #<Dry::Validation::Result output={:name=>"Jane", :email=>"jane@doe.org", :address=>{:street=>"Street 1", :city=>"NYC
", :zipcode=>"1234"}} errors={:age=>["is missing"]}>
```

```
EMAIL_REGEX = /\A[\w+\-.]+@[a-z\d\-]+(\.[a-z]+)*\.[a-z]+\z/i

UserSchema = Dry::Validation.Schema do
  required(:name).filled

  required(:email).filled(format?: EMAIL_REGEX)

  required(:age).maybe(:int?)

  required(:address).schema do
    required(:street).filled
    required(:city).filled
    required(:zipcode).filled
  end
end
```

```
EMAIL_REGEX = /\A[\w+\-.]+@[a-z\d\-]+(\.[a-z]+)*\.[a-z]+\z/i

UserSchema = Dry::Validation.Schema do
  required( Attribute illed

  required(:email).filled(format?: EMAIL_REGEX)

  required(:age).maybe(:int?)

  required(:address).schema do
    required(:street).filled
    required(:city).filled
    required(:zipcode).filled
  end
end
```

```
EMAIL_REGEX = /\A[\w+\-.]+@[a-z\d\-]+(\.[a-z]+)*\.[a-z]+\z/i

UserSchema = Dry::Validation.Schema do
  required(:name).filled

  required(:email).filled(format?: EMAIL_REGEX)

  required(:age).maybe(:int?)

  required(:address).schema do
    required(:street).filled
    required(:city).filled
    required(:zipcode).filled
  end
end
```

Value

Attributes

```
optional(:key)
required(:key)
```

Attributes

```
optional(:key)
required(:key)
```

Values

```
optional(:key).maybe(:int?, gt?: 18)
required(:key).filled(:int?, gt?: 18)
```

```ruby
TestSchema = Dry::Validation.Schema do
  optional(:optional).maybe(:int?, gt?: 18)
  required(:required).filled(:int?, gt?: 18)
end
```

```ruby
TestSchema = Dry::Validation.Schema do
  optional(:optional).maybe(:int?, gt?: 18)
  required(:required).filled(:int?, gt?: 18)
end
```

```
[1] pry(main)> TestSchema.call({})
=> #<Dry::Validation::Result output={} errors={:required=>["is missing"]}>
```

```ruby
TestSchema = Dry::Validation.Schema do
  optional(:optional).maybe(:int?, gt?: 18)
  required(:required).filled(:int?, gt?: 18)
end
```

```
[1] pry(main)> TestSchema.call({})
=> #<Dry::Validation::Result output={} errors={:required=>["is missing"]}>
[2] pry(main)> TestSchema.call(required: 20)
=> #<Dry::Validation::Result output={:required=>20} errors={}>
```

```ruby
TestSchema = Dry::Validation.Schema do
  optional(:optional).maybe(:int?, gt?: 18)
  required(:required).filled(:int?, gt?: 18)
end
```

```
[1] pry(main)> TestSchema.call({})
=> #<Dry::Validation::Result output={} errors={:required=>["is missing"]}>
[2] pry(main)> TestSchema.call(required: 20)
=> #<Dry::Validation::Result output={:required=>20} errors={}>
[3] pry(main)> TestSchema.call(required: 10)
=> #<Dry::Validation::Result output={:required=>10} errors={:required=>["must be greater than 18"]}>
```

```ruby
TestSchema = Dry::Validation.Schema do
  optional(:optional).maybe(:int?, gt?: 18)
  required(:required).filled(:int?, gt?: 18)
end
```

```
[1] pry(main)> TestSchema.call({})
=> #<Dry::Validation::Result output={} errors={:required=>["is missing"]}>
[2] pry(main)> TestSchema.call(required: 20)
=> #<Dry::Validation::Result output={:required=>20} errors={}>
[3] pry(main)> TestSchema.call(required: 10)
=> #<Dry::Validation::Result output={:required=>10} errors={:required=>["must be greater than 18"]}>
[4] pry(main)> TestSchema.call(required: 20, optional: 20)
=> #<Dry::Validation::Result output={:required=>20, :optional=>20} errors={}>
```

```ruby
TestSchema = Dry::Validation.Schema do
  optional(:optional).maybe(:int?, gt?: 18)
  required(:required).filled(:int?, gt?: 18)
end
```

```
[1] pry(main)> TestSchema.call({})
=> #<Dry::Validation::Result output={} errors={:required=>["is missing"]}>
[2] pry(main)> TestSchema.call(required: 20)
=> #<Dry::Validation::Result output={:required=>20} errors={}>
[3] pry(main)> TestSchema.call(required: 10)
=> #<Dry::Validation::Result output={:required=>10} errors={:required=>["must be greater than 18"]}>
[4] pry(main)> TestSchema.call(required: 20, optional: 20)
=> #<Dry::Validation::Result output={:required=>20, :optional=>20} errors={}>
[5] pry(main)> TestSchema.call(required: 20, optional: 10)
=> #<Dry::Validation::Result output={:required=>20, :optional=>10} errors={:optional=>["must be greater than 18"]}>
```

```ruby
TestSchema = Dry::Validation.Schema do
  optional(:optional).maybe(:int?, gt?: 18)
  required(:required).filled(:int?, gt?: 18)
end
```

```
[1] pry(main)> TestSchema.call({})
=> #<Dry::Validation::Result output={} errors={:required=>["is missing"]}>
[2] pry(main)> TestSchema.call(required: 20)
=> #<Dry::Validation::Result output={:required=>20} errors={}>
[3] pry(main)> TestSchema.call(required: 10)
=> #<Dry::Validation::Result output={:required=>10} errors={:required=>["must be greater than 18"]}>
[4] pry(main)> TestSchema.call(required: 20, optional: 20)
=> #<Dry::Validation::Result output={:required=>20, :optional=>20} errors={}>
[5] pry(main)> TestSchema.call(required: 20, optional: 10)
=> #<Dry::Validation::Result output={:required=>20, :optional=>10} errors={:optional=>["must be greater than 18"]}>
[6] pry(main)> TestSchema.call(required: nil, optional: nil)
=> #<Dry::Validation::Result output={:required=>nil, :optional=>nil} errors={:required=>["must be filled"]}>
```

```ruby
TestSchema = Dry::Validation.Schema do
  optional(:optional).maybe(:int?, gt?: 18)
  required(:required).filled(:int?, gt?: 18)
end
```

```
[1] pry(main)> TestSchema.call({})
=> #<Dry::Validation::Result output={} errors={:required=>["is missing"]}>
[2] pry(main)> TestSchema.call(required: 20)
=> #<Dry::Validation::Result output={:required=>20} errors={}>
[3] pry(main)> TestSchema.call(required: 10)
=> #<Dry::Validation::Result output={:required=>10} errors={:required=>["must be greater than 18"]}>
[4] pry(main)> TestSchema.call(required: 20, optional: 20)
=> #<Dry::Validation::Result output={:required=>20, :optional=>20} errors={}>
[5] pry(main)> TestSchema.call(required: 20, optional: 10)
=> #<Dry::Validation::Result output={:required=>20, :optional=>10} errors={:optional=>["must be greater than 18"]}>
[6] pry(main)> TestSchema.call(required: nil, optional: nil)
=> #<Dry::Validation::Result output={:required=>nil, :optional=>nil} errors={:required=>["must be filled"]}>
[7] pry(main)> TestSchema.call(required: :foo, optional: :foo)
=> #<Dry::Validation::Result output={:required=>:foo, :optional=>:foo} errors={:optional=>["must be an integer"], :requ
ired=>["must be an integer"]}>
```

```ruby
TestSchema = Dry::Validation.Schema do
  optional(:optional).maybe(:int?, gt?: 18)
  required(:required).filled(:int?, gt?: 18)
end
```

```
[1] pry(main)> result = TestSchema.call(required: 20, optional: 10)
=> #<Dry::Validation::Result output={:required=>20, :optional=>10} errors={:optional=>["must be greater than 18"]}>
```

```ruby
TestSchema = Dry::Validation.Schema do
  optional(:optional).maybe(:int?, gt?: 18)
  required(:required).filled(:int?, gt?: 18)
end
```

```
[1] pry(main)> result = TestSchema.call(required: 20, optional: 10)
=> #<Dry::Validation::Result output={:required=>20, :optional=>10} errors={:optional=>["must be greater than 18"]}>
[2] pry(main)> result.success?
=> false
[3] pry(main)> result.failure?
=> true
```

```ruby
TestSchema = Dry::Validation.Schema do
  optional(:optional).maybe(:int?, gt?: 18)
  required(:required).filled(:int?, gt?: 18)
end
```

```
[1] pry(main)> result = TestSchema.call(required: 20, optional: 10)
=> #<Dry::Validation::Result output={:required=>20, :optional=>10} errors={:optional=>["must be greater than 18"]}>
[2] pry(main)> result.success?
=> false
[3] pry(main)> result.failure?
=> true
[4] pry(main)> result.to_h
=> {:required=>20, :optional=>10}
```

```ruby
TestSchema = Dry::Validation.Schema do
  optional(:optional).maybe(:int?, gt?: 18)
  required(:required).filled(:int?, gt?: 18)
end
```

```
[1] pry(main)> result = TestSchema.call(required: 20, optional: 10)
=> #<Dry::Validation::Result output={:required=>20, :optional=>10} errors={:optional=>["must be greater than 18"]}>
[2] pry(main)> result.success?
=> false
[3] pry(main)> result.failure?
=> true
[4] pry(main)> result.to_h
=> {:required=>20, :optional=>10}
[5] pry(main)> result.errors
=> {:optional=>["must be greater than 18"]}
[6] pry(main)> result.errors(full: true)
=> {:optional=>["optional must be greater than 18"]}
```

Uniqueness validation

Uniqueness validation

```ruby
module EmailUniqueness
  include Dry::Logic::Predicates

  predicate(:unique_email?) { |email| Repo::User.new.email_unique?(email) }
end
```

Uniqueness validation

```ruby
module EmailUniqueness
  include Dry::Logic::Predicates

  predicate(:unique_email?) { |email| Repo::User.new.email_unique?(email) }
end
```

```yaml
en:
  errors:
    unique_email?: 'already in use'
    password_confirmation: 'does not match'
```

Uniqueness validation

```
Validation = Dry::Validation.Schema do
  configure do
    config.messages_file = 'config/locales/validations/errors.yml'
    predicates(EmailUniqueness)
  end

  required(:ssn).filled
  required(:email).filled(size?: 12..64)
  required(:password).filled.confirmation
end
```

Intentionally left blank

Intentionally left blank

In order to build some tension

Intentionally left blank

In order to build some tension

...

THE END

@mentero
artur@codebakery.eu