



**Full credit is given to the above companies including the Operating System (OS) that this PDF file was generated!**

***Rocky Enterprise Linux 9.2 Manual Pages on command 'Type::Tiny::Enum.3pm'***

***\$ man Type::Tiny::Enum.3pm***

Type::Tiny::Enum(3pm)      User Contributed Perl Documentation      Type::Tiny::Enum(3pm)

**NAME**

Type::Tiny::Enum - string enum type constraints

**STATUS**

This module is covered by the Type-Tiny stability policy.

**DESCRIPTION**

Enum type constraints.

This package inherits from Type::Tiny; see that for most documentation. Major differences are listed below:

**Attributes**

"values"

Arrayref of allowable value strings. Non-string values (e.g. objects with overloading) will be stringified in the constructor.

"constraint"

Unlike Type::Tiny, you cannot pass a constraint coderef to the constructor. Instead rely on the default.

"inlined"

Unlike `Type::Tiny`, you cannot pass an inlining coderef to the constructor. Instead rely on the default.

"parent"

Parent is always `Types::Standard::Str`, and cannot be passed to the constructor.

"unique\_values"

The list of "values" but sorted and with duplicates removed. This cannot be passed to the constructor.

"coercion"

If "coercion => 1" is passed to the constructor, the type will have a coercion using the "closest\_match" method.

## Methods

"as\_regexp"

Returns the enum as a regexp which strings can be checked against. If you're checking a lot of strings, then using this regexp might be faster than checking each string against

```
my $enum = Type::Tiny::Enum->new(...);
my $check = $enum->compiled_check;
my $re = $enum->as_regexp;

# fast
my @valid_tokens = grep $enum->check($_), @all_tokens;

# faster
my @valid_tokens = grep $check->($_), @all_tokens;

# fastest
my @valid_tokens = grep /$re/, @all_tokens;
```

You can get a case-insensitive regexp using "\$enum->as\_regexp('i')".

## "closest\_match"

Returns the closest match in the enum for a string.

```
my $enum = Type::Tiny::Enum->new(
    values => [ qw( foo bar baz quux ) ],
);

say $enum->closest_match("FO"); # ==> foo
```

It will try to find an exact match first, fall back to a case-insensitive match, if it still can't find one, will try to find a head substring match, and finally, if given an integer, will use that as an index.

```
my $enum = Type::Tiny::Enum->new(
    values => [ qw( foo bar baz quux ) ],
);

say $enum->closest_match( 0 ); # ==> foo
say $enum->closest_match( 1 ); # ==> bar
say $enum->closest_match( 2 ); # ==> baz
say $enum->closest_match( -1 ); # ==> quux
```

## Overloading

? Arrayrefification calls "values".

## BUGS

Please report any bugs to <<https://github.com/tobyink/p5-type-tiny/issues>>.

## SEE ALSO

Type::Tiny::Manual.

Type::Tiny.

Moose::Meta::TypeConstraint::Enum.

## AUTHOR

Toby Inkster <tobyink@cpan.org>.

## COPYRIGHT AND LICENCE

This software is copyright (c) 2013-2014, 2017-2021 by Toby Inkster.

This is free software; you can redistribute it and/or modify it under the same terms as the Perl 5 programming language system itself.

## DISCLAIMER OF WARRANTIES

THIS PACKAGE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

perl v5.32.1

2021-08-31

Type::Tiny::Enum(3pm)