



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

C:\>man Type::Tiny::Manual::AllTypes.3pm

Type::Tiny::Manual::AllTypesUser)Contributed Perl DocumentType::Tiny::Manual::AllTypes(3pm)

NAME

Type::Tiny::Manual::AllTypes - alphabetical list of all type constraints bundled with Type::Tiny

MANUAL

The following is a list of type constraints bundled with Type::Tiny, with very brief descriptions. For more information, see the type library's documentation, and the test cases in "t/21-types/".

GitHub link: <<https://github.com/tobyink/p5-type-tiny/tree/master/t/21-types>>.

? Any in Types::Standard

Anything. Absolutely anything.

? ArrayLike in Types::TypeTiny

Arrayrefs and objects overloading arrayfication.

? ArrayRef [parameterizable] in Types::Standard

Arrayrefs.

? Bool [has coercion] in Types::Standard

Booleans; the numbers or strings "0" and "1", the empty string, or undef.

? ClassName in Types::Standard

Any loaded package name.

? CodeLike in Types::TypeTiny

Coderefs and objects overloading codereification.

? CodeRef in Types::Standard

Coderefs.

? ConsumerOf [parameterizable] in Types::Standard

An object that DOES a particular role.

? CycleTuple [parameterizable] in Types::Standard

An arrayref with a repeating pattern of constraints on its values.

? Defined in Types::Standard

Any value other than undef.

? Dict [parameterizable] in Types::Standard

A hashref with constraints on each of its values.

? Enum [parameterizable] in Types::Standard

A string from an allowed set of strings.

? FileHandle in Types::Standard

A reference where Scalar::Util::openhandle returns true, or a blessed object in the IO::Handle class.

? GlobRef in Types::Standard

Globrefs

? HashLike in Types::TypeTiny

Hashrefs and objects overloading hashrefification.

? HashRef [parameterizable] in Types::Standard

Hashrefs.

? HasMethods [parameterizable] in Types::Standard

An object that can do particular methods.

? InstanceOf [parameterizable] in Types::Standard

An object that isa particular class.

? Int in Types::Standard

A whole number, either positive, negative, or zero.

? `IntRange` [parameterizable] in `Types::Common::Numeric`

An integer within a particular numeric range.

? `Item` in `Types::Standard`

Any single item; effectively the same as `Any`.

? `LaxNum` in `Types::Standard`

A number; relaxed constraint that allows "inf".

? `LowerCaseSimpleStr` [has coercion] in `Types::Common::String`

A string less than 256 characters long with no line breaks or uppercase letters.

? `LowerCaseStr` [has coercion] in `Types::Common::String`

A string with no uppercase letters.

? `Map` [parameterizable] in `Types::Standard`

A hashref with a constraint for the values and keys.

? `Maybe` [parameterizable] in `Types::Standard`

When parameterized, the same as its parameter, but also allows `undef`.

? `NegativeInt` in `Types::Common::Numeric`

An integer below 0.

? NegativeNum in Types::Common::Numeric

A number below 0.

? NegativeOrZeroInt in Types::Common::Numeric

An integer below 0, or 0.

? NegativeOrZeroNum in Types::Common::Numeric

A number below 0, or 0.

? NonEmptySimpleStr in Types::Common::String

A string with more than 0 but less than 256 characters with no line breaks.

? NonEmptyStr in Types::Common::String

A string with more than 0 characters.

? Num in Types::Standard

The same as LaxNum or StrictNum depending on environment.

? NumericCode [has coercion] in Types::Common::String

A string containing only digits.

? NumRange [parameterizable] in Types::Common::Numeric

A number within a particular numeric range.

? Object in Types::Standard

A blessed object.

? Optional [parameterizable] in Types::Standard

Used in conjunction with Dict, Tuple, or CycleTuple.

? OptList in Types::Standard

An arrayref of arrayrefs, where each of the inner arrayrefs are two values, the first value being a string.

? Overload [parameterizable] in Types::Standard

An overloaded object.

? Password in Types::Common::String

A string at least 4 characters long and less than 256 characters long with no line breaks.

? PositiveInt in Types::Common::Numeric

An integer above 0.

? PositiveNum in Types::Common::Numeric

A number above 0.

? PositiveOrZeroInt in Types::Common::Numeric

An integer above 0, or 0.

? PositiveOrZeroNum in Types::Common::Numeric

An number above 0, or 0.

? Ref [parameterizable] in Types::Standard

Any reference.

? RegexpRef in Types::Standard

A regular expression.

? RoleName in Types::Standard

Any loaded package name where there is no `new` method.

? ScalarRef [parameterizable] in Types::Standard

Scalarrefs.

? SimpleStr in Types::Common::String

A string with less than 256 characters with no line breaks.

? SingleDigit in Types::Common::Numeric

A single digit number. This includes single digit negative numbers!

? Str in Types::Standard

A string.

? StrictNum in Types::Standard

A number; strict constant.

? StringLike in Types::TypeTiny

Strings and objects overloading stringification.

? StrLength [parameterizable] in Types::Common::String

A string with length in a particular range.

? StrMatch [parameterizable] in Types::Standard

A string matching a particular regular expression.

? StrongPassword in Types::Common::String

A string at least 4 characters long and less than 256 characters long with no line breaks and at least one non-alphabetic character.

? Tied [parameterizable] in Types::Standard

A reference to a tied variable.

? Tuple [parameterizable] in Types::Standard

An arrayref with constraints on its values.

? TypeTiny [has coercion] in Types::TypeTiny

Blessed objects in the Type::Tiny class.

? Undef in Types::Standard

undef.

? UpperCaseSimpleStr [has coercion] in Types::Common::String

A string less than 256 characters long with no line breaks or lowercase letters.

? UpperCaseStr [has coercion] in Types::Common::String

A string with no lowercase letters.

? Value in Types::Standard

Any non-reference value, including undef.

NEXT STEPS

Here's your next step:

? Type::Tiny::Manual::Policies

Policies related to Type::Tiny development.

AUTHOR

Toby Inkster <tobyink@cpan.org>.

COPYRIGHT AND LICENCE

This software is copyright (c) 2013-2014, 2017-2019 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 v5.30.0

2019-12-28

Type::Tiny::Manual::AllTypes(3pm)