

## NAME

Type::Tiny::Manual::UsingWithTestMore – Type::Tiny for test suites

## MANUAL

### Test::TypeTiny

This is a module for testing that types you've defined accept and reject the values you think they should.

```
should_pass($value, $type);
should_fail($othervalue, $type);
```

Easy. (But yeah, I always forget whether the type goes first or second!)

There's also a function to test that subtype/supertype relationships are working okay.

```
ok_subtype($type, @subtypes);
```

Of course you can just check a type like this:

```
ok($type->check($value) );
```

But the advantage of `should_pass` is that if the `EXTENDED_TESTING` environment variable is set to true, `should_pass` will also perform a strict check on the value, which involves climbing up the type's inheritance tree (its parent, its parent's parent, etc) to make sure the value passes all their constraints.

If a normal check and strict check differ, this is usually a problem in the inlining code somewhere.

See `Test::TypeTiny` for more information.

### Type::Tiny as a Replacement for Test::Deep

Here's one of the examples from the `Test::Deep` documentation:

```
my $name_re = re('^(Mr|Mrs|Miss) \w+ \w+$');
cmp_deeply(
  $person,
  {
    Name      => $name_re,
    Phone     => re('^0d{6}$'),
    ChildNames => array_each($name_re)
  },
  "person ok"
);
```

It's pretty easy to rewrite this to use `Types::Standard`:

```
my $name = StrMatch[ qr/^(Mr|Mrs|Miss) \w+ \w+$/ ];
should_pass(
  $person,
  Dict[
    Name      => $name,
    Phone     => StrMatch[ qr/^0d{6}$/ ],
    ChildNames => ArrayRef[$name]
  ]
);
```

There's nothing especially wrong with `Test::Deep`, but if you're already familiar with `Type::Tiny`'s built-in types and you've maybe written your own type libraries too, it will save you having to switch between using two separate systems of checks.

## NEXT STEPS

Here's your next step:

- [Type::Tiny::Manual::Params](#)

Advanced information on `Type::Params`, and using `Type::Tiny` with other signature modules like `Function::Parameters` and `Kavorka`.

## **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.