NAME

Moo::Role – Minimal Object Orientation support for Roles

SYNOPSIS

```
package My::Role;

use Moo::Role;
use strictures 2;

sub foo { ... }

sub bar { ... }

has baz => (
  is => 'ro',
);

1;
```

And elsewhere:

```
package Some::Class;

use Moo;
use strictures 2;

# bar gets imported, but not foo
with('My::Role');

sub foo { ... }

1;
```

DESCRIPTION

`Moo::Role` builds upon Role::Tiny, so look there for most of the documentation on how this works (in particular, using `Moo::Role` also enables strict and warnings). The main addition here is extra bits to make the roles more "Moosey;" which is to say, it adds "has".

IMPORTED SUBROUTINES

See "IMPORTED SUBROUTINES" in Role::Tiny for all the other subroutines that are imported by this module.

has

```
has attr => (
  is => 'ro',
);
```

Declares an attribute for the class to be composed into. See "has" in Moo for all options.

CLEANING UP IMPORTS

Moo::Role cleans up its own imported methods and any imports declared before the `use Moo::Role` statement automatically. Anything imported after `use Moo::Role` will be composed into consuming packages. A package that consumes this role:

```
package My::Role::ID;

use Digest::MD5 qw(md5_hex);
use Moo::Role;
use Digest::SHA qw(sha1_hex);
```

```
  requires 'name';

  sub as_md5  { my ($self) = @_; return md5_hex($self->name);  }
  sub as_sha1 { my ($self) = @_; return sha1_hex($self->name); }

  1;
```

..will now have a `$self->sha1_hex()` method available to it that probably does not do what you expect. On the other hand, a call to `$self->md5_hex()` will die with the helpful error message: `Can't locate object method "md5_hex"`.

See "CLEANING UP IMPORTS" in Moo for more details.

SUPPORT

See Moo for support and contact information.

AUTHORS

See Moo for authors.

COPYRIGHT AND LICENSE

See Moo for the copyright and license.