



## ***Rocky Enterprise Linux 9.2 Manual Pages on command 'Class::XSAccessor.3pm'***

**C:\>man Class::XSAccessor.3pm**

Class::XSAccessor(3pm) User Contributed Perl Documentation Class::XSAccessor(3pm)

### NAME

Class::XSAccessor - Generate fast XS accessors without runtime compilation

### SYNOPSIS

```
package MyClass;

use Class::XSAccessor

replace    => 1, # Replace existing methods (if any)

constructor => 'new',

getters    => {

    get_foo => 'foo', # 'foo' is the hash key to access

    get_bar => 'bar',

},

setters    => {

    set_foo => 'foo',

    set_bar => 'bar',

},

accessors  => {

    foo => 'foo',

    bar => 'bar',

},

# "predicates" is an alias for "defined_predicates"

defined_predicates => {
```

```

    defined_foo => 'foo',
    defined_bar => 'bar',
},
exists_predicates => {
    has_foo => 'foo',
    has_bar => 'bar',
},
lvalue_accessors => { # see below
    baz => 'baz', # ...
},
true => [ 'is_token', 'is_whitespace' ],
false => [ 'significant' ];
# The imported methods are implemented in fast XS.
# normal class code here.

```

As of version 1.05, some alternative syntax forms are available:

```

package MyClass;

# Options can be passed as a HASH reference, if preferred,
# which can also help Perl::Tidy to format the statement correctly.
use Class::XSAccessor {
    # If the name => key values are always identical,
    # the following shorthand can be used.
    accessors => [ 'foo', 'bar' ],
};

```

## DESCRIPTION

Class::XSAccessor implements fast read, write and read/write accessors in XS.

Additionally, it can provide predicates such as "has\_foo()" for testing whether the attribute "foo" exists in the object (which is different from "is defined within the object"). It only works with objects that are implemented as ordinary hashes.

Class::XSAccessor::Array implements the same interface for objects that use arrays for their internal representation.

Since version 0.10, the module can also generate simple constructors (implemented in XS). Simply supply the "constructor => 'constructor\_name'" option or the "constructors => ['new', 'create', 'spawn']" option. These constructors do the

equivalent of the following Perl code:

```
sub new {  
    my $class = shift;  
    return bless { @_ }, ref($class)||$class;  
}
```

That means they can be called on objects and classes but will not clone objects entirely. Parameters to "new()" are added to the object.

The XS accessor methods are between 3 and 4 times faster than typical pure-Perl accessors in some simple benchmarking. The lower factor applies to the potentially slightly obscure "sub set\_foo\_pp {\$\_[0]->{foo} = \$\_[1]}", so if you usually write clear code, a factor of 3.5 speed-up is a good estimate. If in doubt, do your own benchmarking!

The method names may be fully qualified. The example in the synopsis could have been written as "MyClass::get\_foo" instead of "get\_foo". This way, methods can be installed in classes other than the current class. See also: the "class" option below.

By default, the setters return the new value that was set, and the accessors (mutators) do the same. This behaviour can be changed with the "chained" option - see below. The predicates return a boolean.

Since version 1.01, "Class::XSAccessor" can generate extremely simple methods which just return true or false (and always do so). If that seems like a really superfluous thing to you, then consider a large class hierarchy with interfaces such as PPI. These methods are provided by the "true" and "false" options - see the synopsis.

"defined\_predicates" check whether a given object attribute is defined.

"predicates" is an alias for "defined\_predicates" for compatibility with older versions of "Class::XSAccessor". "exists\_predicates" checks whether the given attribute exists in the object using "exists".

## OPTIONS

In addition to specifying the types and names of accessors, additional options can be supplied which modify behaviour. The options are specified as key/value pairs in the same manner as the accessor declaration. For example:

```
use Class::XSAccessor
```

```
getters => {  
  get_foo => 'foo',  
},  
replace => 1;
```

The list of available options is:

#### replace

Set this to a true value to prevent "Class::XSAccessor" from complaining about replacing existing subroutines.

#### chained

Set this to a true value to change the return value of setters and mutators (when called with an argument). If "chained" is enabled, the setters and accessors/mutators will return the object. Mutators called without an argument still return the value of the associated attribute.

As with the other options, "chained" affects all methods generated in the same "use Class::XSAccessor ..." statement.

#### class

By default, the accessors are generated in the calling class. The the "class" option allows the target class to be specified.

## LVALUES

Support for lvalue accessors via the keyword "lvalue\_accessors" was added in version 1.08. At this point, THEY ARE CONSIDERED HIGHLY EXPERIMENTAL. Furthermore, their performance hasn't been benchmarked yet.

The following example demonstrates an lvalue accessor:

```
package Address;  
use Class::XSAccessor  
  constructor => 'new',  
  lvalue_accessors => { zip_code => 'zip' };  
package main;  
my $address = Address->new(zip => 2);  
print $address->zip_code, "\n"; # prints 2  
$address->zip_code = 76135; # <--- This is it!  
print $address->zip_code, "\n"; # prints 76135
```

## CAVEATS

Probably won't work for objects based on tied hashes. But that's a strange thing to do anyway.

Scary code exploiting strange XS features.

If you think writing an accessor in XS should be a laughably simple exercise, then please contemplate how you could instantiate a new XS accessor for a new hash key that's only known at run-time. Note that compiling C code at run-time a la Inline::C is a no go.

Threading. With version 1.00, a memory leak has been fixed. Previously, a small amount of memory would leak if "Class::XSAccessor"-based classes were loaded in a subthread without having been loaded in the "main" thread. If the subthread then terminated, a hash key and an int per associated method used to be lost. Note that this mattered only if classes were only loaded in a sort of throw-away thread. In the new implementation, as of 1.00, the memory will still not be released, in the same situation, but it will be recycled when the same class, or a similar class, is loaded again in any thread.

#### SEE ALSO

? Class::XSAccessor::Array

? AutoXS

#### AUTHOR

Steffen Mueller <smueller@cpan.org>

chocolateboy <chocolate@cpan.org>

#### COPYRIGHT AND LICENSE

Copyright (C) 2008, 2009, 2010, 2011, 2012, 2013 by Steffen Mueller

This library is free software; you can redistribute it and/or modify it under the same terms as Perl itself, either Perl version 5.8 or, at your option, any later version of Perl 5 you may have available.

perl v5.30.0

2019-10-18

Class::XSAccessor(3pm)