```
        my $whatnot =  sub { $some_obj->obfuscate(@args) };
        func($whatnot);
        sub func {
            my $code = shift;
            &$code();
        }
```

You could also investigate the can() method in the UNIVERSAL class (part of the standard perl distribution).

23.1.15 How do I create a static variable?

As with most things in Perl, TMTOWTDI. What is a "static variable" in other languages could be either a function-private variable (visible only within a single function, retaining its value between calls to that function), or a file-private variable (visible only to functions within the file it was declared in) in Perl.

Here's code to implement a function-private variable:

```
    BEGIN {
        my $counter = 42;
        sub prev_counter { return --$counter }
        sub next_counter { return $counter++ }
    }
```

Now prev_counter() and next_counter() share a private variable $counter that was initialized at compile time.

To declare a file-private variable, you'll still use a my(), putting the declaration at the outer scope level at the top of the file. Assume this is in file Pax.pm:

```
    package Pax;
    my $started = scalar(localtime(time()));

    sub begun { return $started }
```

When `use Pax` or `require Pax` loads this module, the variable will be initialized. It won't get garbage-collected the way most variables going out of scope do, because the begun() function cares about it, but no one else can get it. It is not called $Pax::started because its scope is unrelated to the package. It's scoped to the file. You could conceivably have several packages in that same file all accessing the same private variable, but another file with the same package couldn't get to it.

See Persistent Private Variables in *perlsub* for details.

23.1.16 What's the difference between dynamic and lexical (static) scoping? Between local() and my()?

`local($x)` saves away the old value of the global variable `$x` and assigns a new value for the duration of the subroutine *which is visible in other functions called from that subroutine*. This is done at run-time, so is called dynamic scoping. local() always affects global variables, also called package variables or dynamic variables.

`my($x)` creates a new variable that is only visible in the current subroutine. This is done at compile-time, so it is called lexical or static scoping. my() always affects private variables, also called lexical variables or (improperly) static(ly scoped) variables.

For instance:

```
    sub visible {
        print "var has value $var\n";
    }
```