```
sub logger_maker {
    my $level = shift;
    my $cb    = shift // sub { say STDERR "[" . shift . "] ", shift };

    return sub {
        $cb->($level, $_) for @_;
    };
}

my $debug = logger_maker('debug');
my $info  = logger_maker('info');
$debug->("Value of X is 3", "Value of Y is 9");
[debug] Value of X is 3
[debug] Value of Y is 9

$info->("You are here");
[info] You are here
```

```
my $debug = logger_maker('debug');
```

```
my $info  = logger_maker('info');
```

```
sub { say STDERR "[" . shift . "] ", shift }
```

```
        sub {
    $cb->($level, $_) for @_;
};
```


my $cb

```
$info->("You are here");
[info] You are here
```

```
$debug->("Value of X is 3", "Value of Y is 9");
[debug] Value of X is 3
[debug] Value of Y is 9
```

Currying

```
sub logger_maker {
    my $level = shift;
    my $cb    = shift // sub { say STDERR "[" . shift . "] ", shift };

    return sub {
        $cb->($level, $_) for @_;
    };
}

my $debug = logger_maker('debug');
my $info  = logger_maker('info');
$debug->("Value of X is 3", "Value of Y is 9");
[debug] Value of X is 3
[debug] Value of Y is 9

$info->("You are here");
[info] You are here
```

Function Composition

```
sub logger_maker {
    my $level = shift;
    my $cb    = shift // sub { say STDERR "[" . shift . "] ", shift };

    return sub {
        $cb->($level, $_) for @_;
    };
}

my $fatal = logger_maker('fatal', sub { die "[" . shift . "] ", shift . "\n" });
$fatal->("Watch this!");
[fatal] Watch this!

say "Not reached.";
```