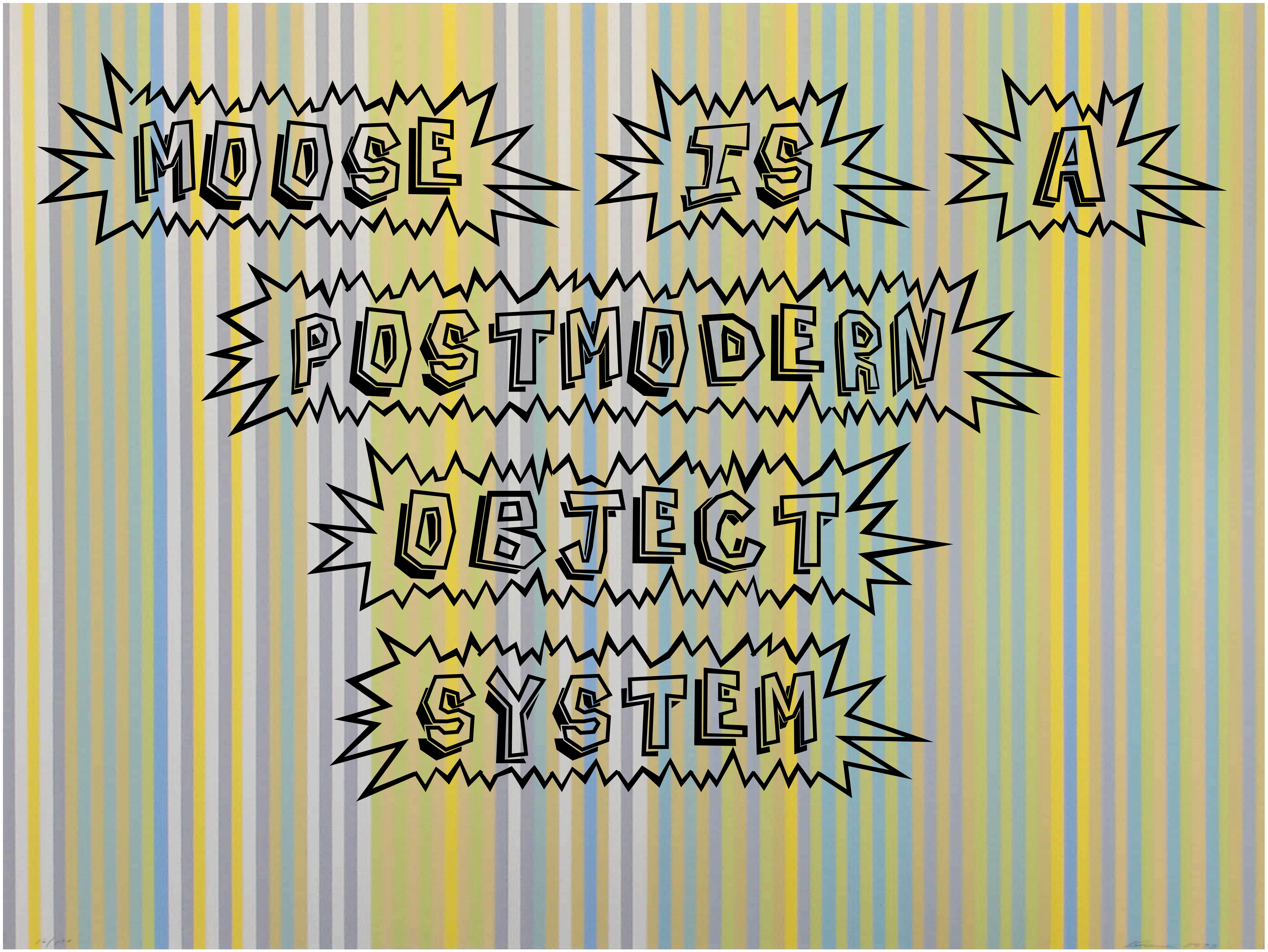
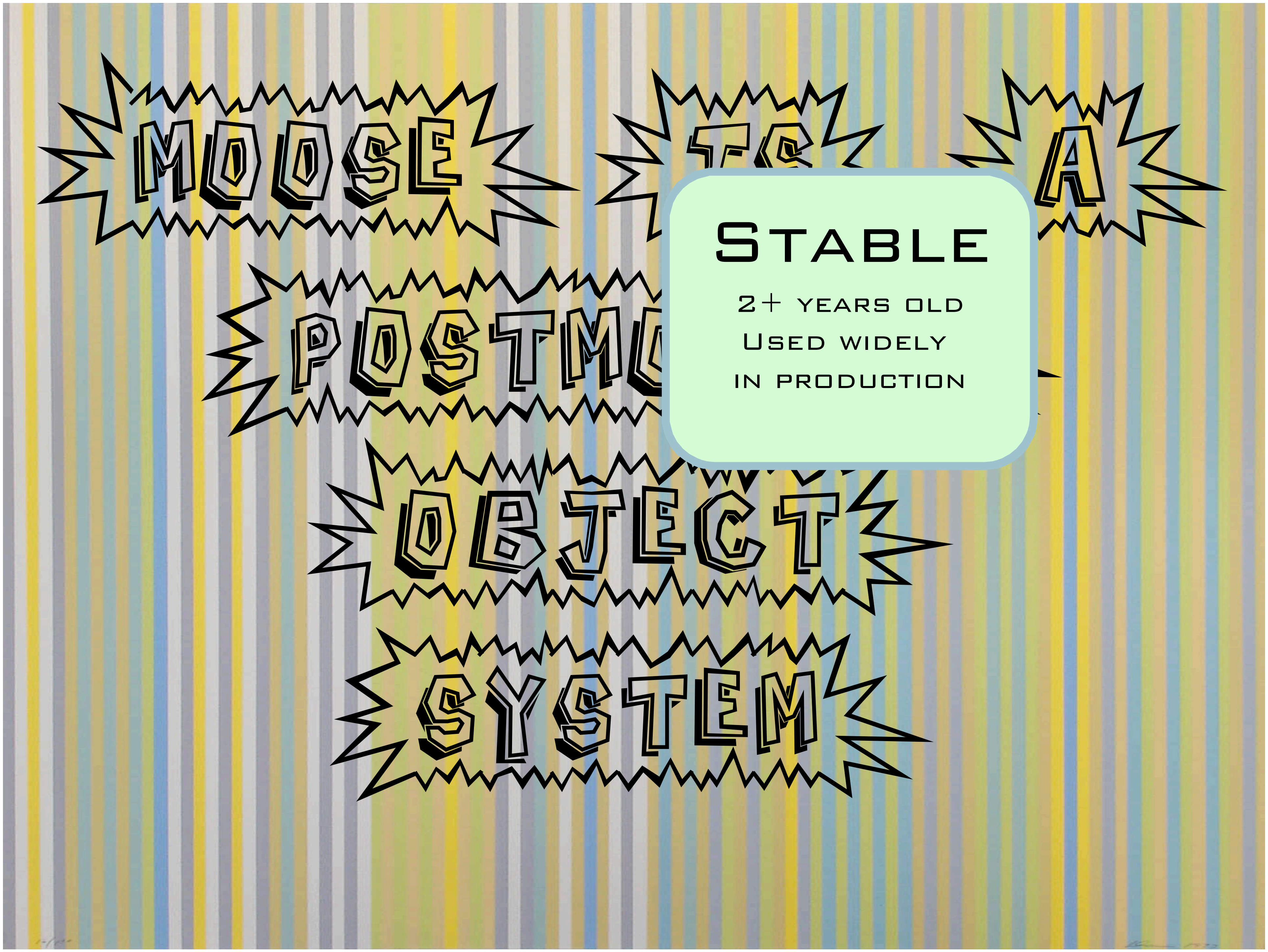
moose


MOOSE
IS
A
POSTMODERN
OBJECT
SYSTEM

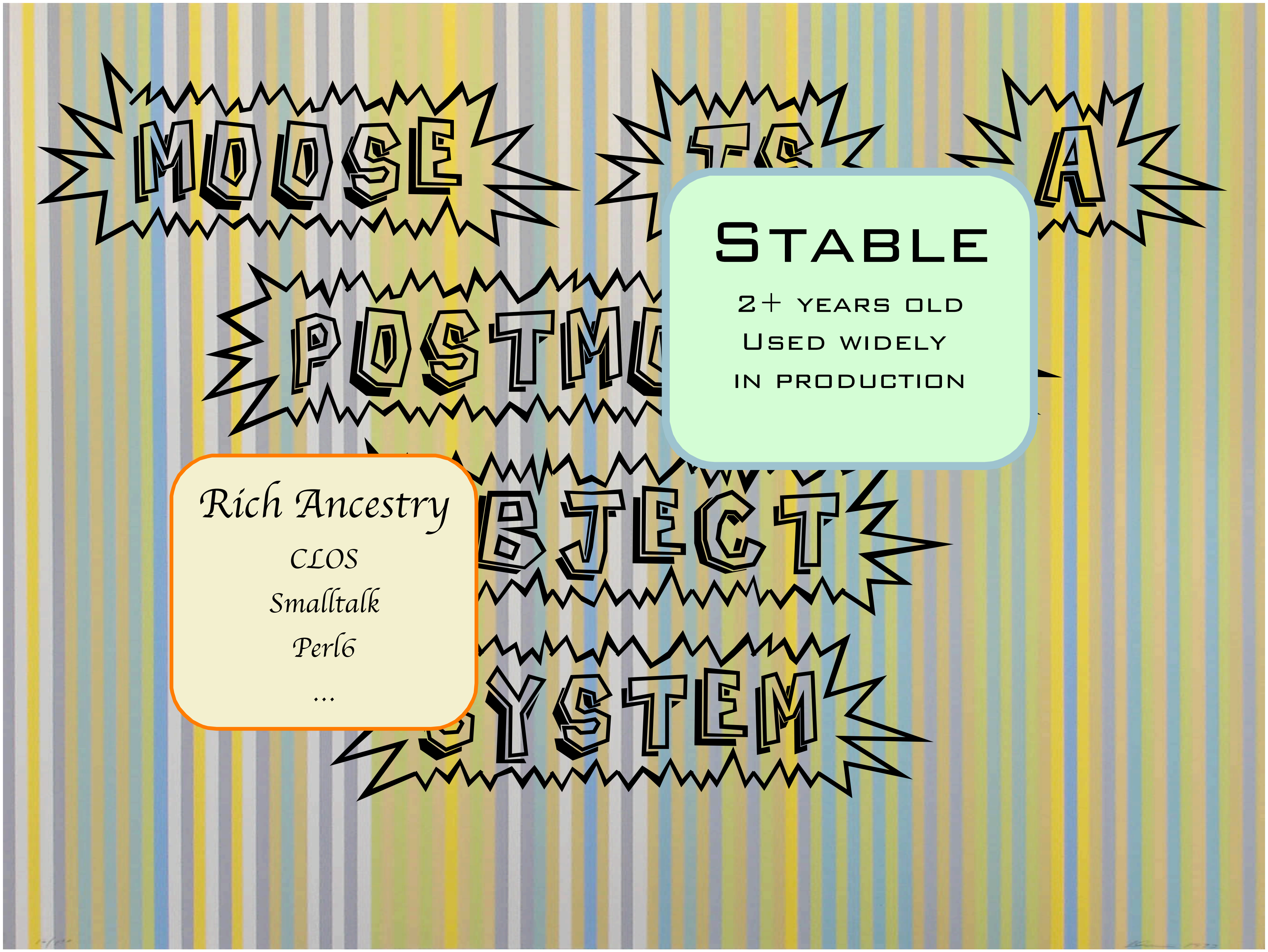


Stable

2+ years old

Used widely

in production



Rich Ancestry

CLOS

Smalltalk

Perl6

...

Stable

2+ years old

Used widely

in production

STABLE

2+ YEARS OLD

USED WIDELY

IN PRODUCTION

Rich Ancestry

CLOS

Smalltalk

Perl6

...

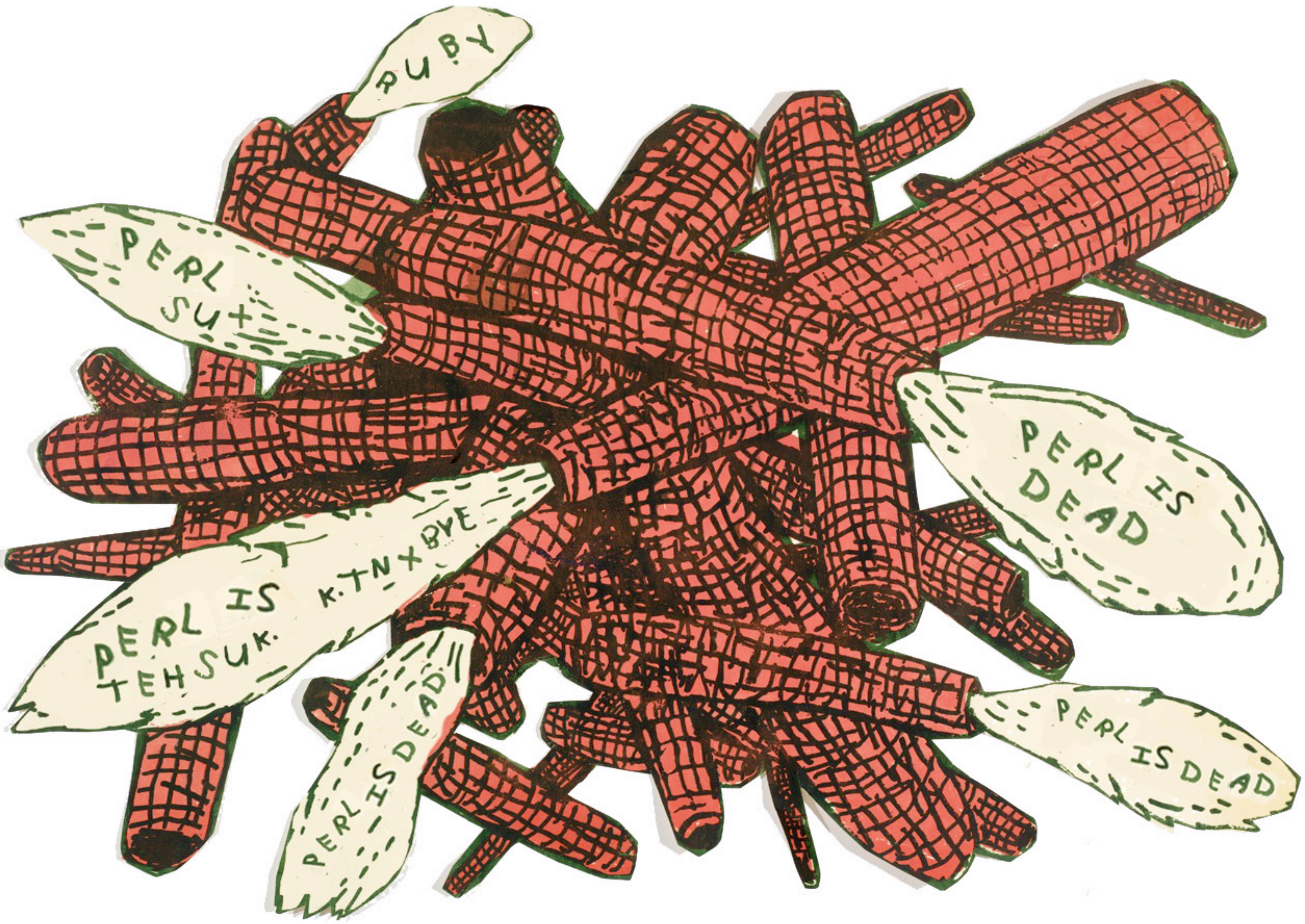
PERLISH

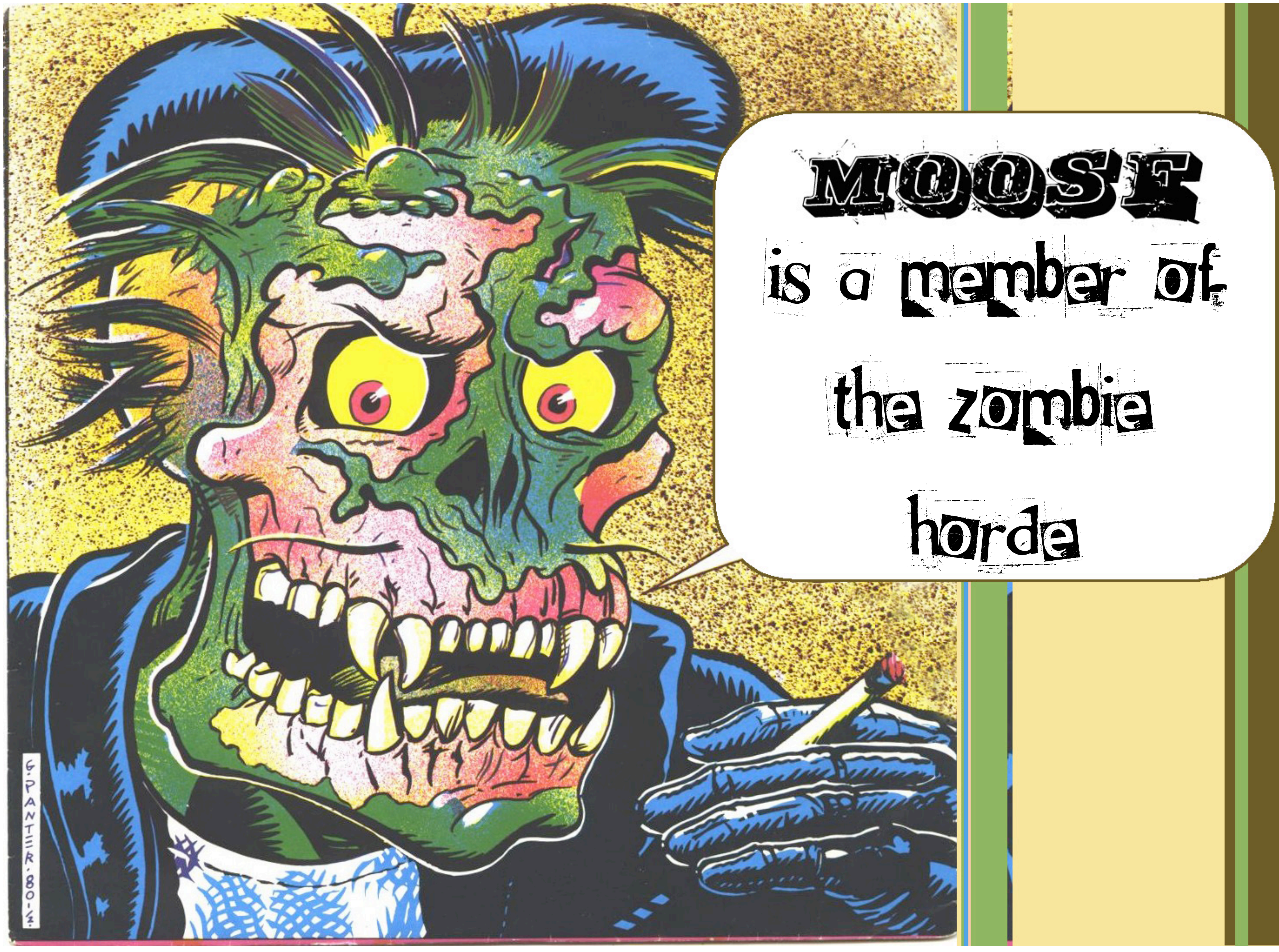
Plays well with CPAN
and vanilla perl 5 OOP

MOOSE IS NOT A TOY

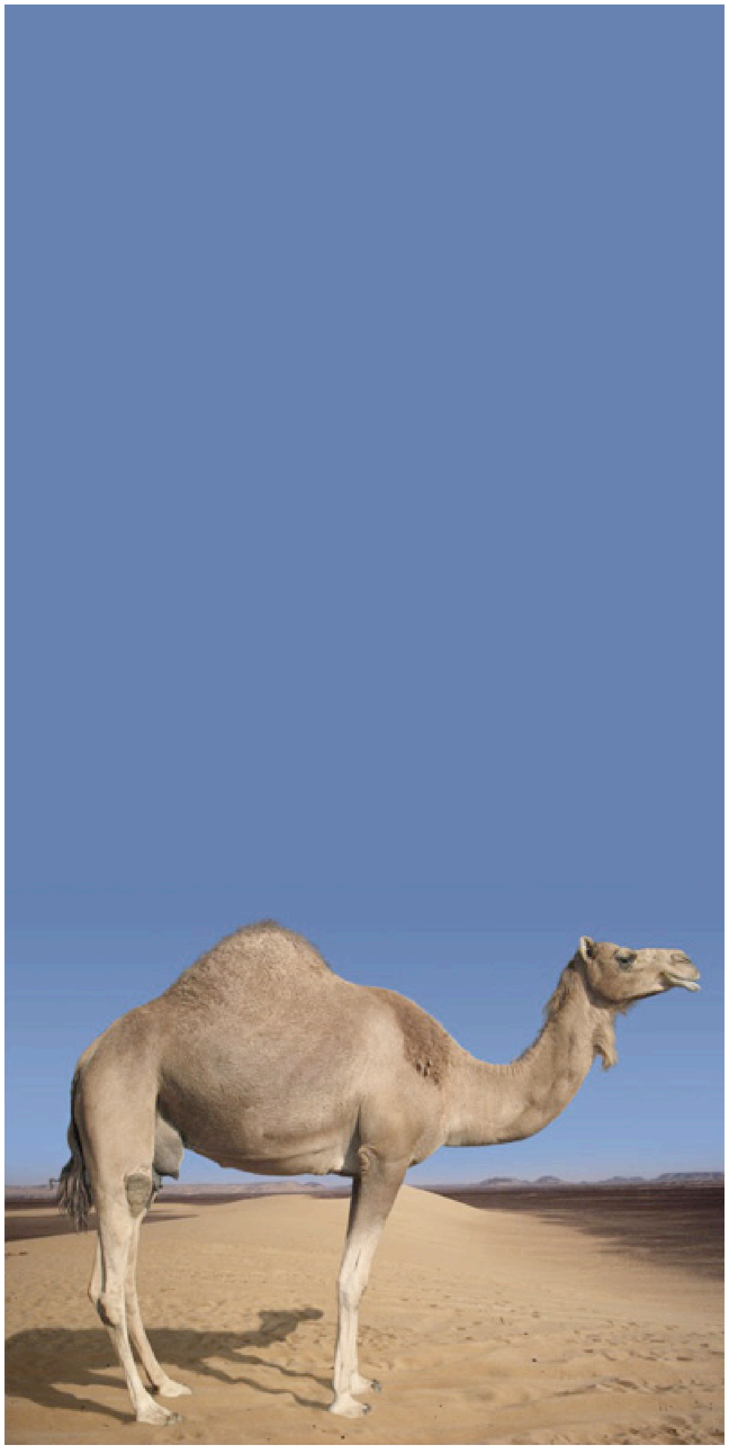
IT IS SERIOUS BUSINESS!

THE RUMOR ON THE TUBES IS THAT PERL IS DEAD




MOOSE is a member of the zombie horde
G. PANTER. 80

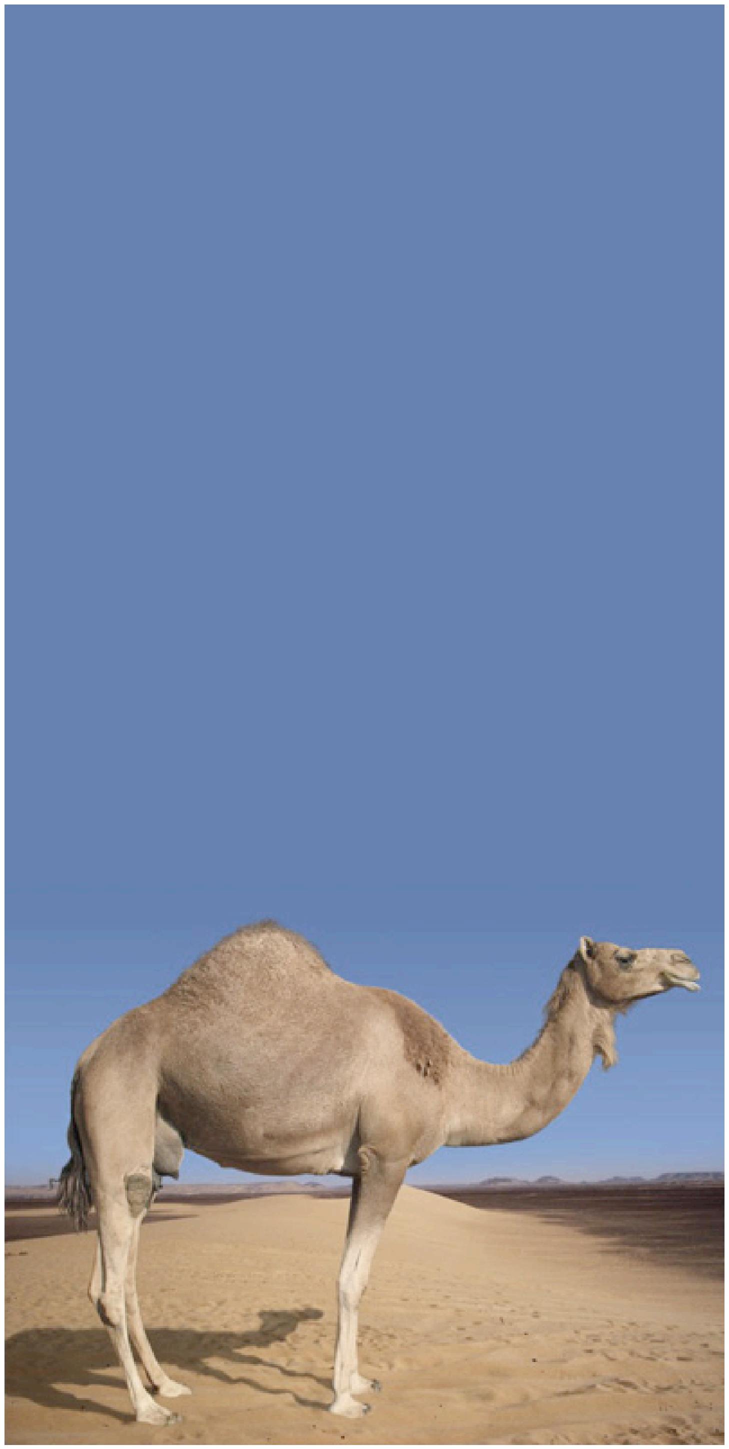
MOOSE IS AN AMPLIFIER



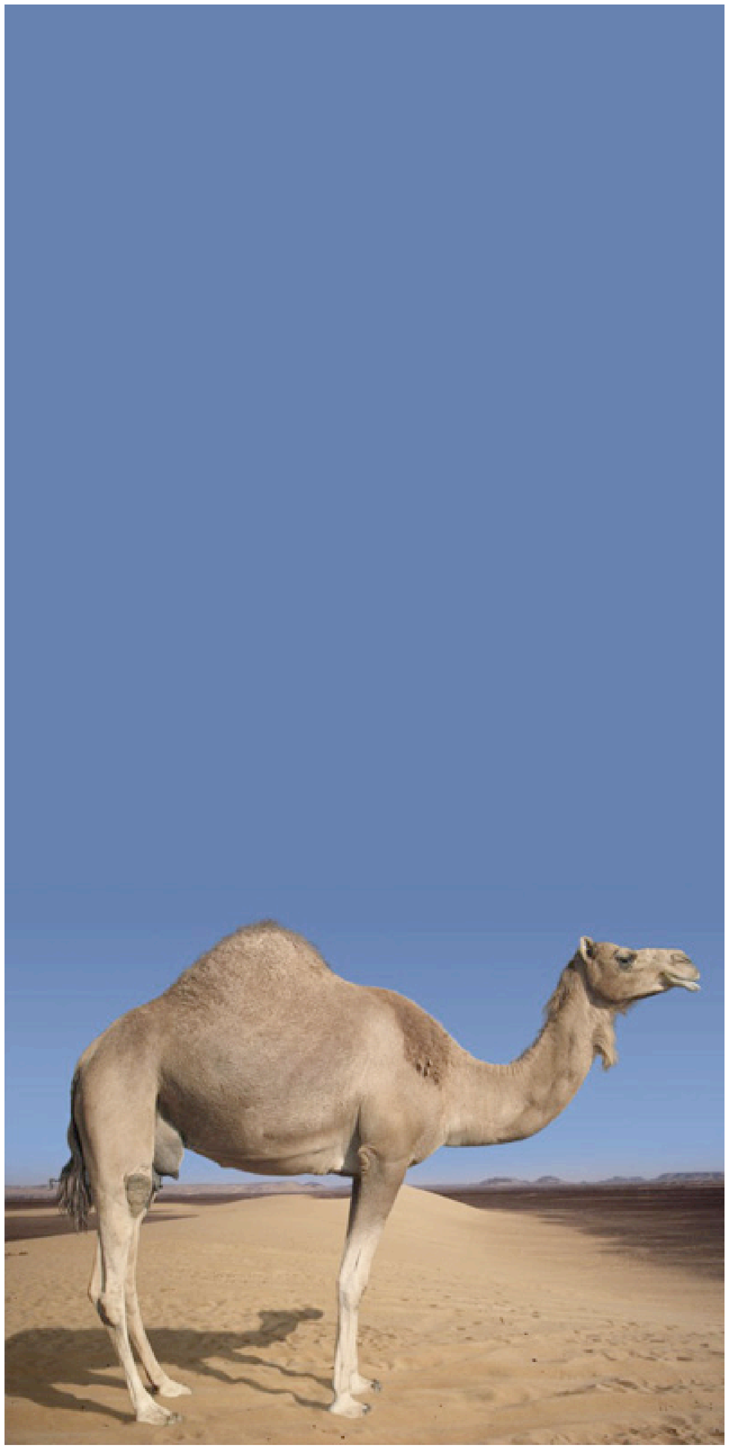
The Old Way

```
package Person;

use strict;
use warnings;

sub new {
    my ($class, %args) = @_;
    bless {
        name => $args{name},
        age  => $args{age} || 0,
    } => ref($class) || $class;
}

sub name {
    my $self = shift;
    $self->{name} = shift if @_;
    $self->{name};
}

sub age {
    my $self = shift;
    $self->{age} = shift if @_;
    $self->{age};
}

1;
```

The Old Way



```
package Person;

use strict;
use warnings;

use base 'Class::Accessor';

__PACKAGE__->mk_accessor(qw[name age]);

1;
```



The Old Way

```
package Person;

use strict;
use warnings;

use base 'Class::Accessor';

__PACKAGE__->mk_accessor(qw[name age]);

sub new {
    my ($class, $params) = @_;
    $params = {} unless defined $params;
    $params->{age} = 0;
    $class->SUPER::new($params);
}

1;
```

The Moose Way

```
package Person;
use Moose;

has ‘name’ => (is => ‘rw’);
has ‘age’  => (is => ‘rw’, default => 0);

1;
```

The Moose Way

```
package Person;
use Moose;
use Moose::Util::TypeConstraints;

use DateTime::Duration;

class_type 'DateTime::Duration';

coerce 'DateTime::Duration'
    => from 'Int'
        => via { DateTime::Duration->new(years => $_) }
    => from 'HashRef'
        => via { DateTime::Duration->new( %$_ ) };

has 'name' => (
    is       => 'rw',
    isa      => subtype('Str' => where { length $_ > 0 }),
    required => 1,
);

has 'age' => (
    is      => 'rw',
    isa     => 'DateTime::Duration',
    coerce  => 1,
    lazy    => 1,
    default => sub { DateTime::Duration->new },
);

1;
```

The Moose Way

```
package Person;
use Moose;
use Moose::Util::TypeConstraints;

use DateTime::Duration;

class_type 'DateTime::Duration';

coerce 'DateTime::Duration'
    => from 'Int'
        => via { DateTime::Duration->new(years => $_) }
    => from 'HashRef'
        => via { DateTime::Duration->new( %$_ ) };

has 'name' => (
    is       => 'rw',
    isa      => subtype('Str' => where { length $_ > 0 }),
    required => 1,
);

has 'age' => (
    is      => 'rw',
    isa     => 'DateTime::Duration',
    coerce  => 1,
    lazy    => 1,
    default => sub { DateTime::Duration->new },
);

1;
```

The Moose Way

```
package Person;
use Moose;
use Moose::Util::TypeConstraints;

use DateTime::Duration;

class_type 'DateTime::Duration';

coerce 'DateTime::Duration'
    => from 'Int'
        => via { DateTime::Duration->new(years => $_) }
    => from 'HashRef'
        => via { DateTime::Duration->new( %$_ ) };

has 'name' => (
    is       => 'rw',
    isa      => subtype('Str' => where { length $_ > 0 }),
    required => 1,
);

has 'age' => (
    is      => 'rw',
    isa     => 'DateTime::Duration',
    coerce  => 1,
    lazy    => 1,
    default => sub { DateTime::Duration->new },
);

1;
```

The Moose Way

```
package Person;
use Moose;
use Moose::Util::TypeConstraints;

use DateTime::Duration;

class_type 'DateTime::Duration';

coerce 'DateTime::Duration'
    => from 'Int'
        => via { DateTime::Duration->new(years => $_) }
    => from 'HashRef'
        => via { DateTime::Duration->new( %$_ ) };

has 'name' => (
    is       => 'rw',
    isa      => subtype('Str' => where { length $_ > 0 }),
    required => 1,
);

has 'age' => (
    is      => 'rw',
    isa     => 'DateTime::Duration',
    coerce  => 1,
    lazy    => 1,
    default => sub { DateTime::Duration->new },
);

1;
```

The Moose Way

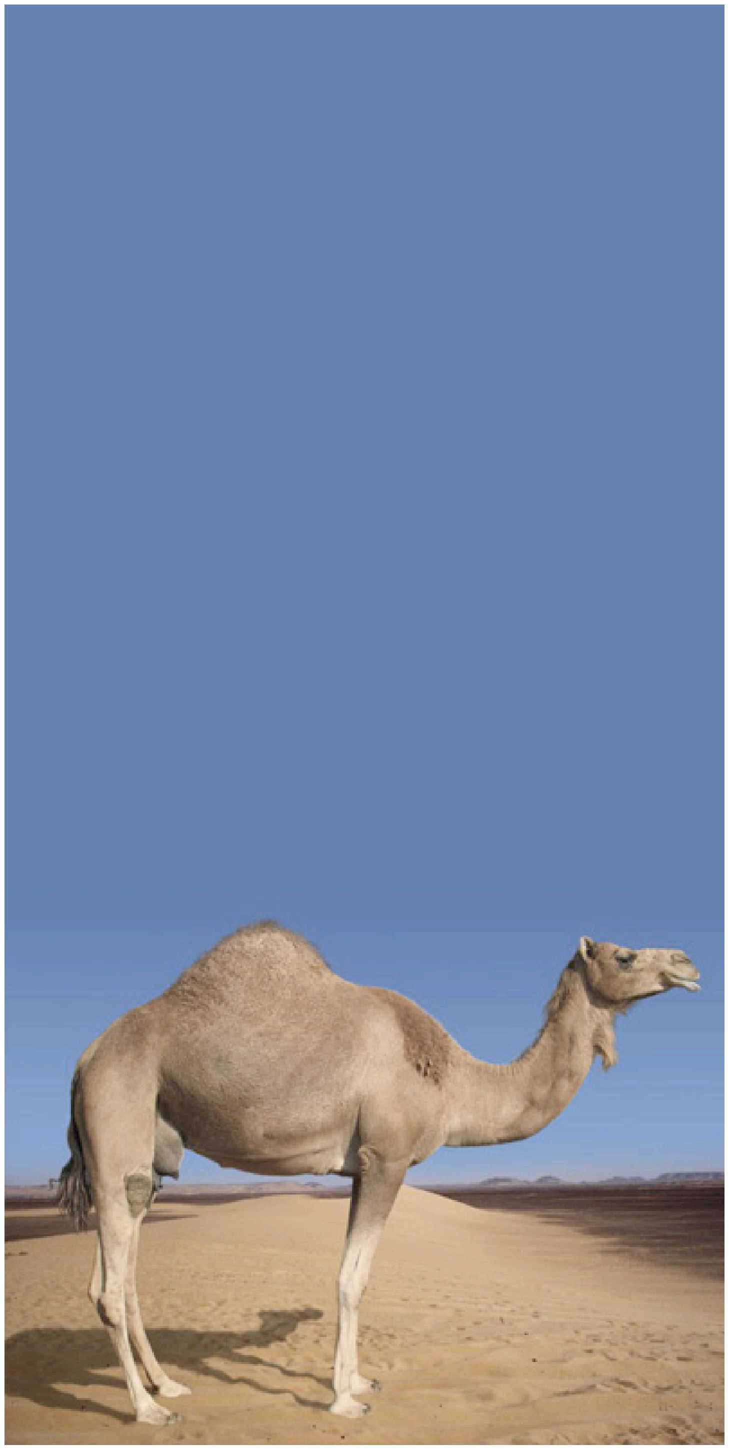
```
package Person;
use Moose;
use Moose::Util::TypeConstraints;

use DateTime::Duration;

class_type 'DateTime::Duration';

coerce 'DateTime::Duration'
    => from 'Int'
        => via { DateTime::Duration->new(years => $_) }
    => from 'HashRef'
        => via { DateTime::Duration->new( %$_ ) };

has 'name' => (
    is       => 'rw',
    isa      => subtype('Str' => where { length $_ > 0 }),
    required => 1,
);

has 'age' => (
    is      => 'rw',
    isa     => 'DateTime::Duration',
    coerce  => 1,
    lazy    => 1,
    default => sub { DateTime::Duration->new },
);

1;
```

The Moose Way

```
package Person;
use Moose;
use Moose::Util::TypeConstraints;

use DateTime::Duration;

class_type 'DateTime::Duration';

coerce 'DateTime::Duration'
    => from 'Int'
        => via { DateTime::Duration->new(years => $_) }
    => from 'HashRef'
        => via { DateTime::Duration->new( %$_ ) };

has 'name' => (
    is       => 'rw',
    isa      => subtype('Str' => where { length $_ > 0 }),
    required => 1,
);

has 'age' => (
    is      => 'rw',
    isa     => 'DateTime::Duration',
    coerce  => 1,
    lazy    => 1,
    default => sub { DateTime::Duration->new },
);

1;
```

The Moose Way

```
package Person;
use Moose;
use Moose::Util::TypeConstraints;

use DateTime::Duration;

class_type 'DateTime::Duration';

coerce 'DateTime::Duration'
    => from 'Int'
        => via { DateTime::Duration->new(years => $_) }
    => from 'HashRef'
        => via { DateTime::Duration->new( %$_ ) };

has 'name' => (
    is       => 'rw',
    isa      => subtype('Str' => where { length $_ > 0 }),
    required => 1,
);

has 'age' => (
    is      => 'rw',
    isa     => 'DateTime::Duration',
    coerce  => 1,
    lazy    => 1,
    default => sub { DateTime::Duration->new },
);

1;
```

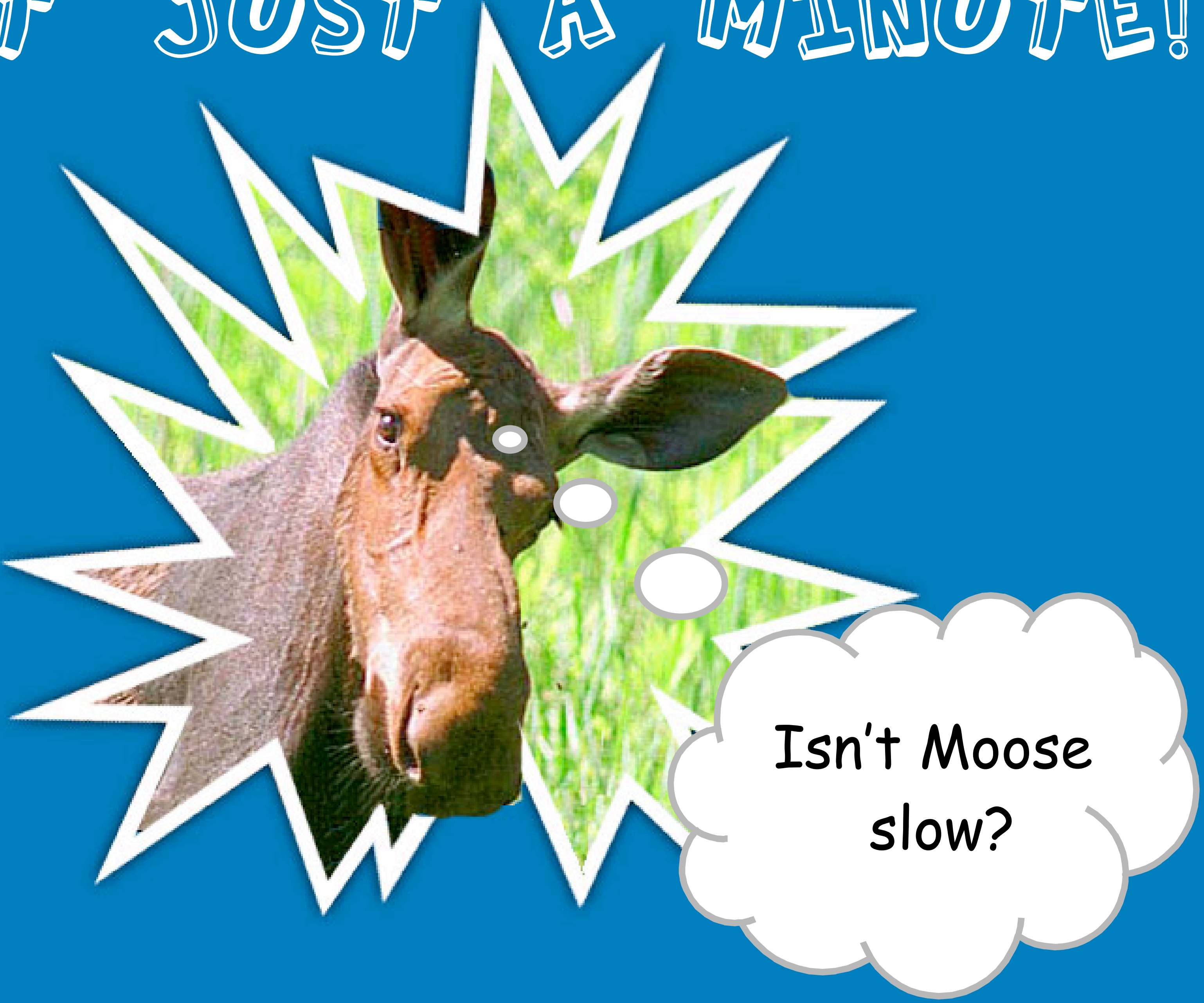


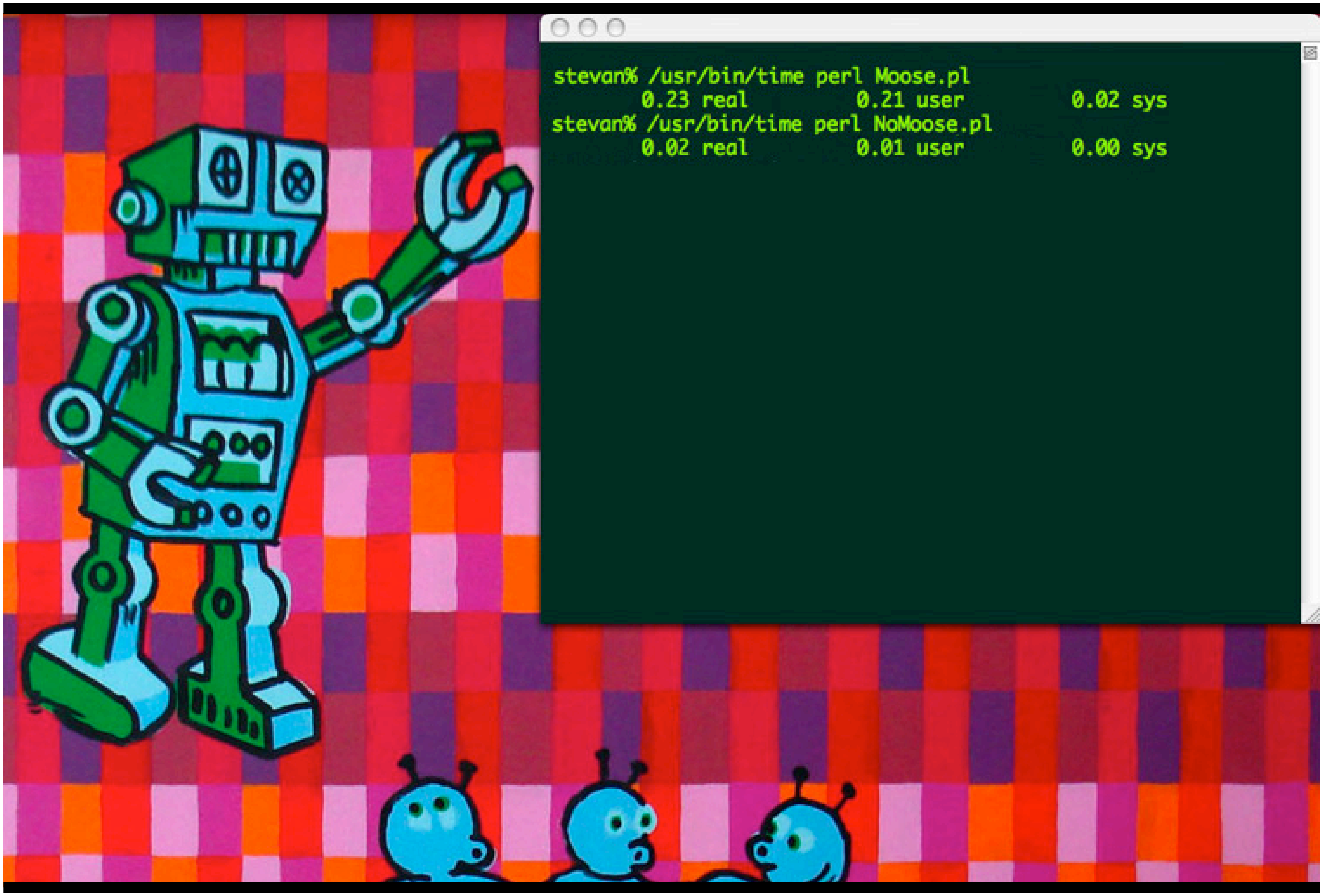
The Old Way

```
package Person;

use strict;
use warnings;

use DateTime::Duration;
use Scalar::Util 'blessed', 'looks_like_number';
use Carp 'confess';

sub new {
    my $class = shift;
    my %params;
    if (@_ == 1 && ref $_[0] eq 'HASH') {
        %params = %{$_[0]};
    }
    else {
        %params = @_;
    }
    (exists $params{name} && length $params{name} > 0)
        || confess "You must supply a name";
    if (exists $params{age}) {
        $params{age} = _coerce_date_time_duration($params{age});
    }
    return bless \%params => ref $class || $class;
}

sub _coerce_date_time_duration {
    my ($val) = @_;
    if (blessed $val) {
        return $val if $val->isa('DateTime::Duration');
        confess "A blessed value must be a DateTime::Duration object, not $val";
    }
    elsif (ref $val) {
        return DateTime::Duration->new( %$val ) if ref $val eq 'HASH';
        confess "We can only convert HASH refs to DateTime::Duration objects, not $val";
    }
    elsif (looks_like_number($val)) {
        return DateTime::Duration->new(years => $val);
    }
    else {
        confess "Cannot coerce $val into DateTime::Duration object";
    }
}

sub name {
    my $self = shift;
    if (@_) {
        my $name = shift;
        (length $name > 0)
            || confess "You must supply a name";
        $self->{name} = $name;
    }
    return $self->{name};
}

sub age {
    my $self = shift;
    if (@_) {
        $self->{age} = _coerce_date_time_duration(shift);
    }
    else {
        $self->{age} ||= DateTime::Duration->new;
    }
    return $self->{age};
}

1;
```

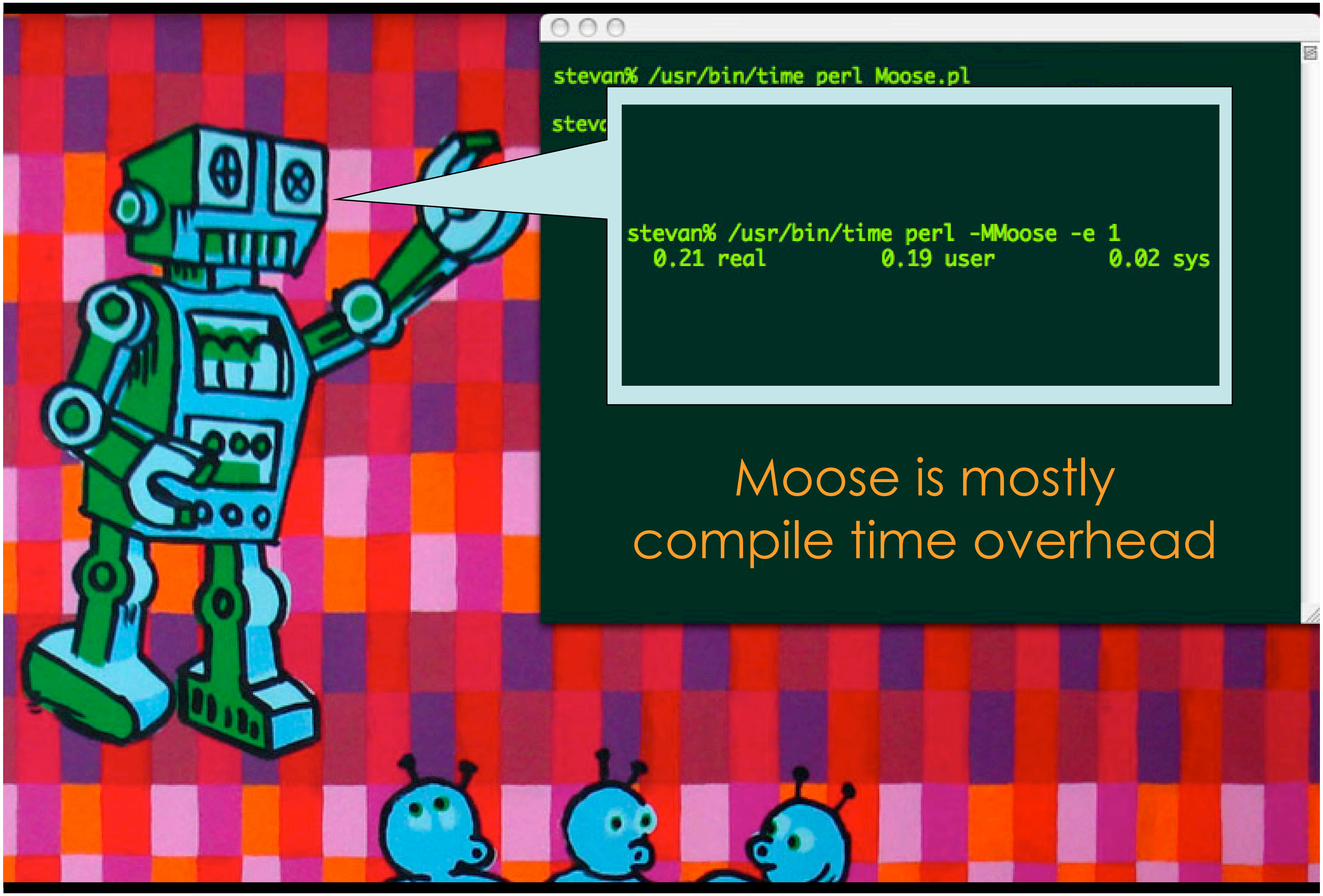
WAIT JUST A MINUTE!

WAIT JUST A MINUTE!



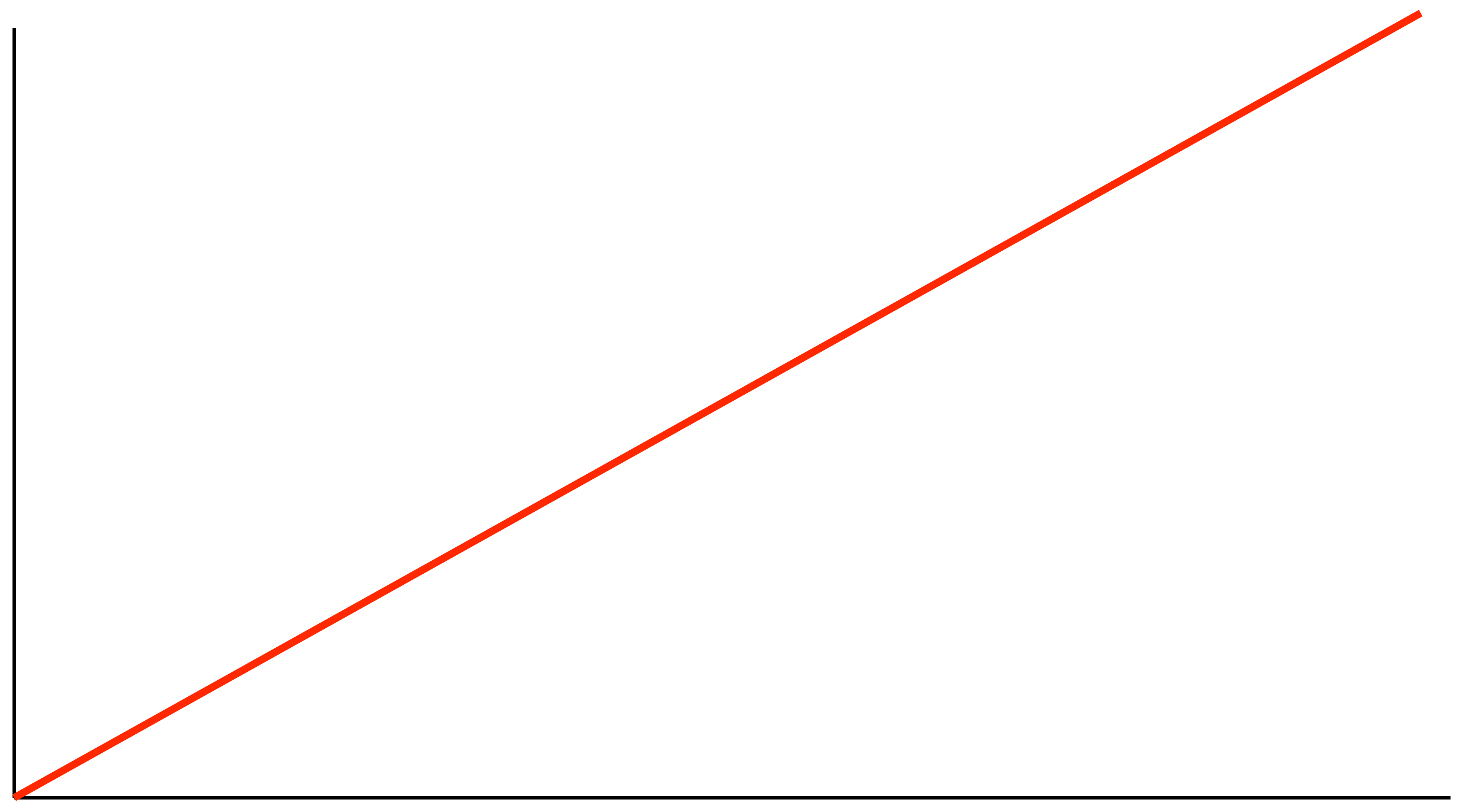


```
stevan% /usr/bin/time perl Moose.pl
        0.23 real         0.21 user         0.02 sys
stevan% /usr/bin/time perl NoMoose.pl
        0.02 real         0.01 user         0.00 sys
```

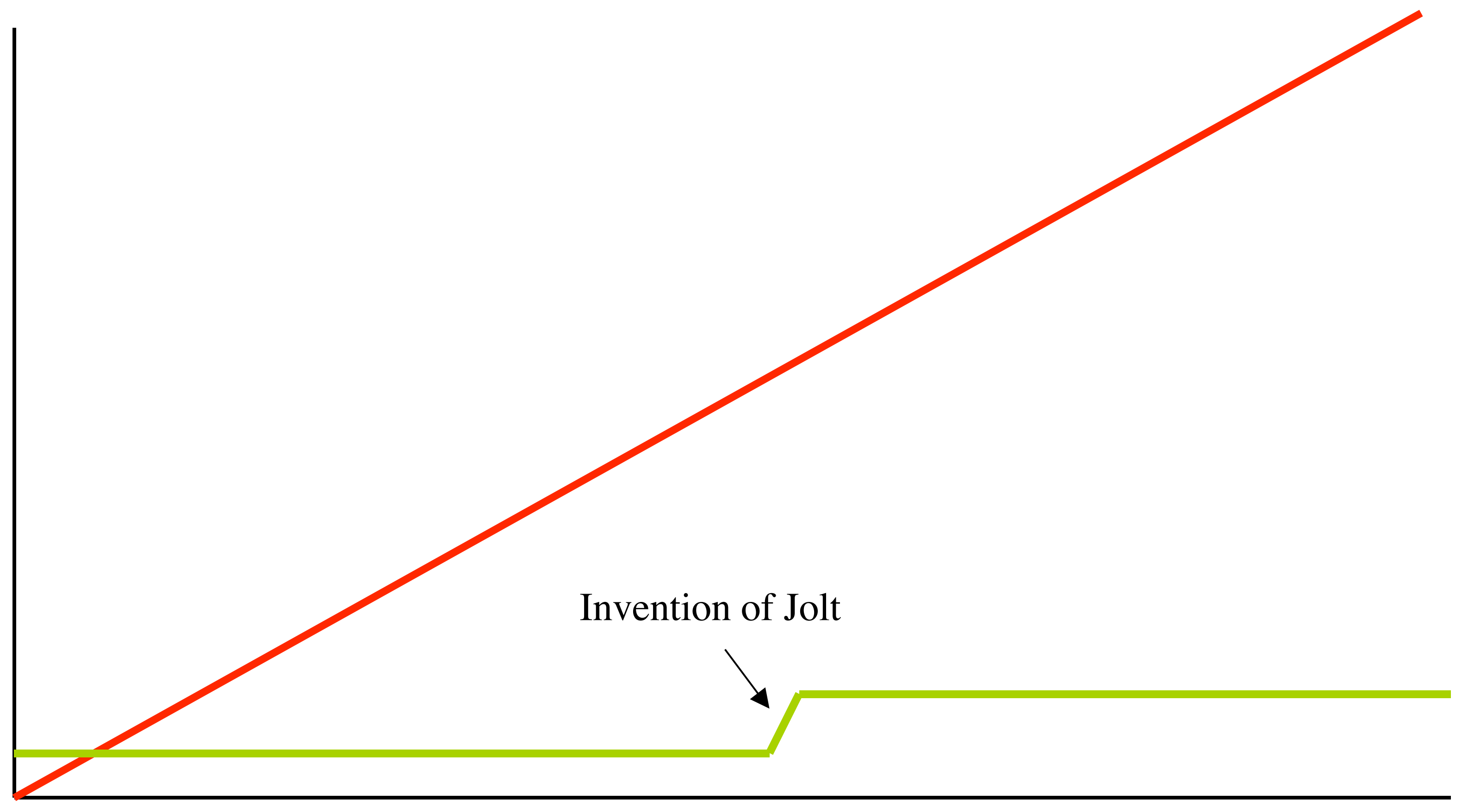


```
stevan% /usr/bin/time perl Moose.pl

stev

stevan% /usr/bin/time perl -MMoose -e 1
  0.21 real         0.19 user         0.02 sys
```

Moose is mostly compile time overhead

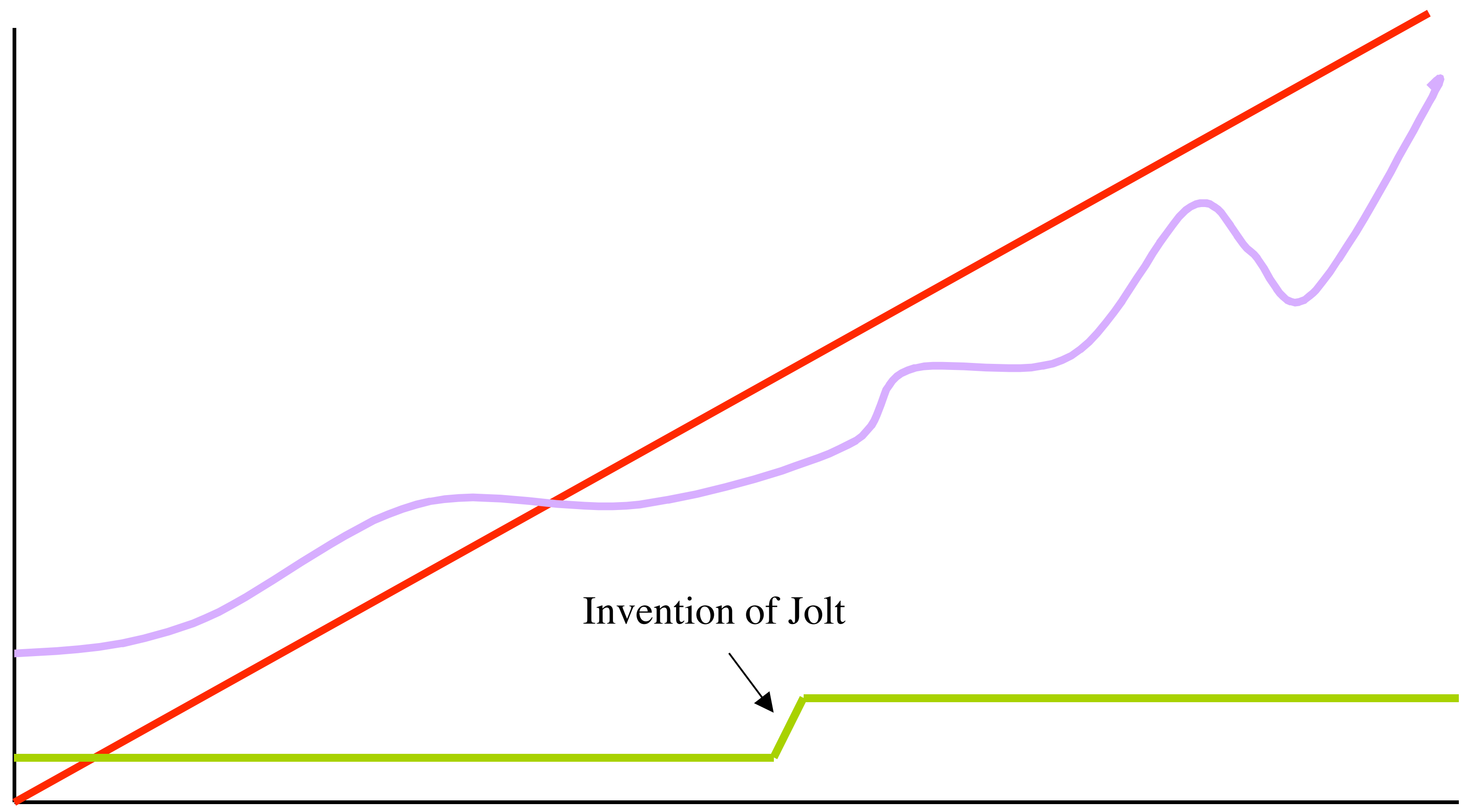


🟥 Moore’s Law



🟥 Moore’s Law

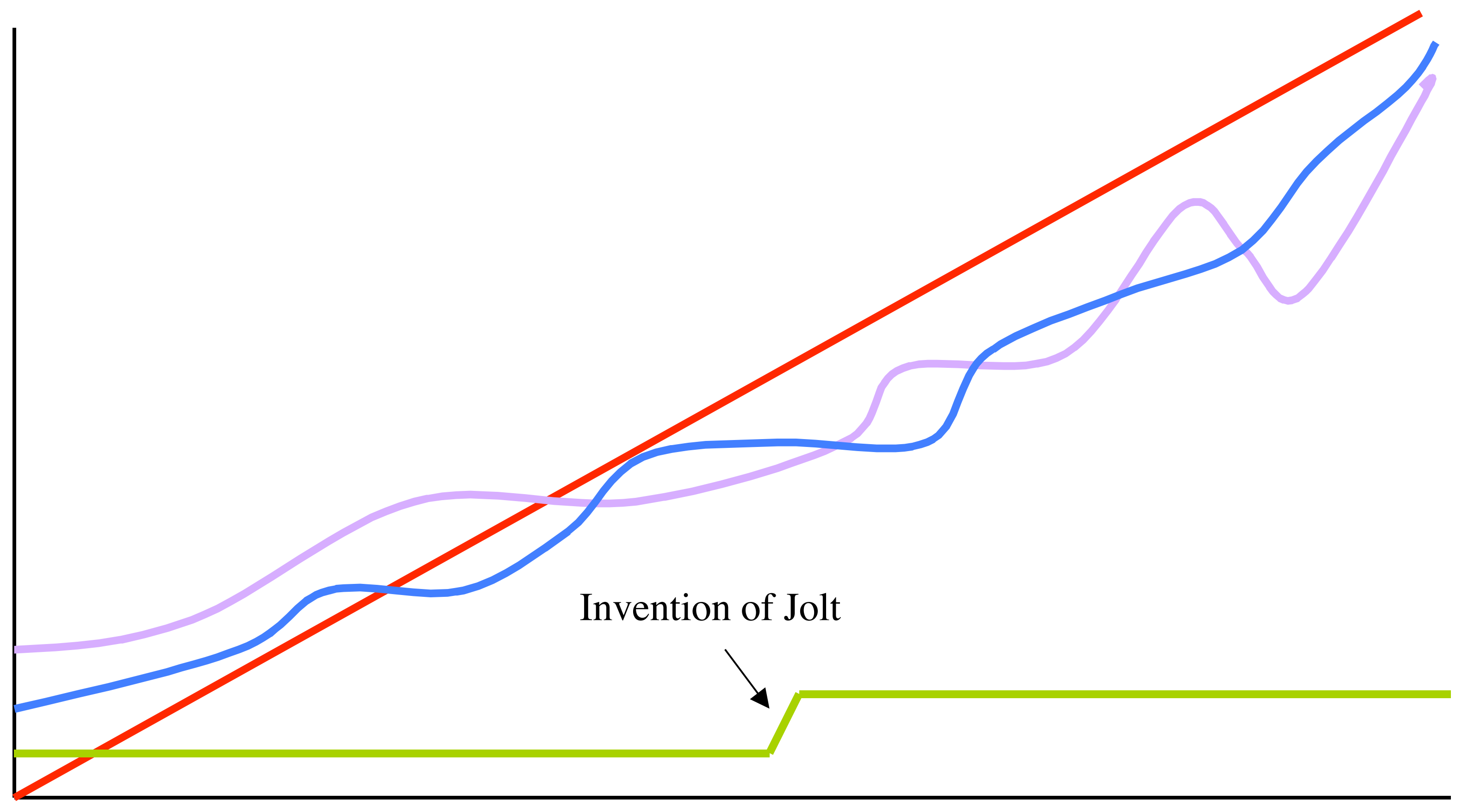
🟩 Developer Speed



🟥 Moore’s Law

🟩 Developer Speed

🟪 Developer Salaries

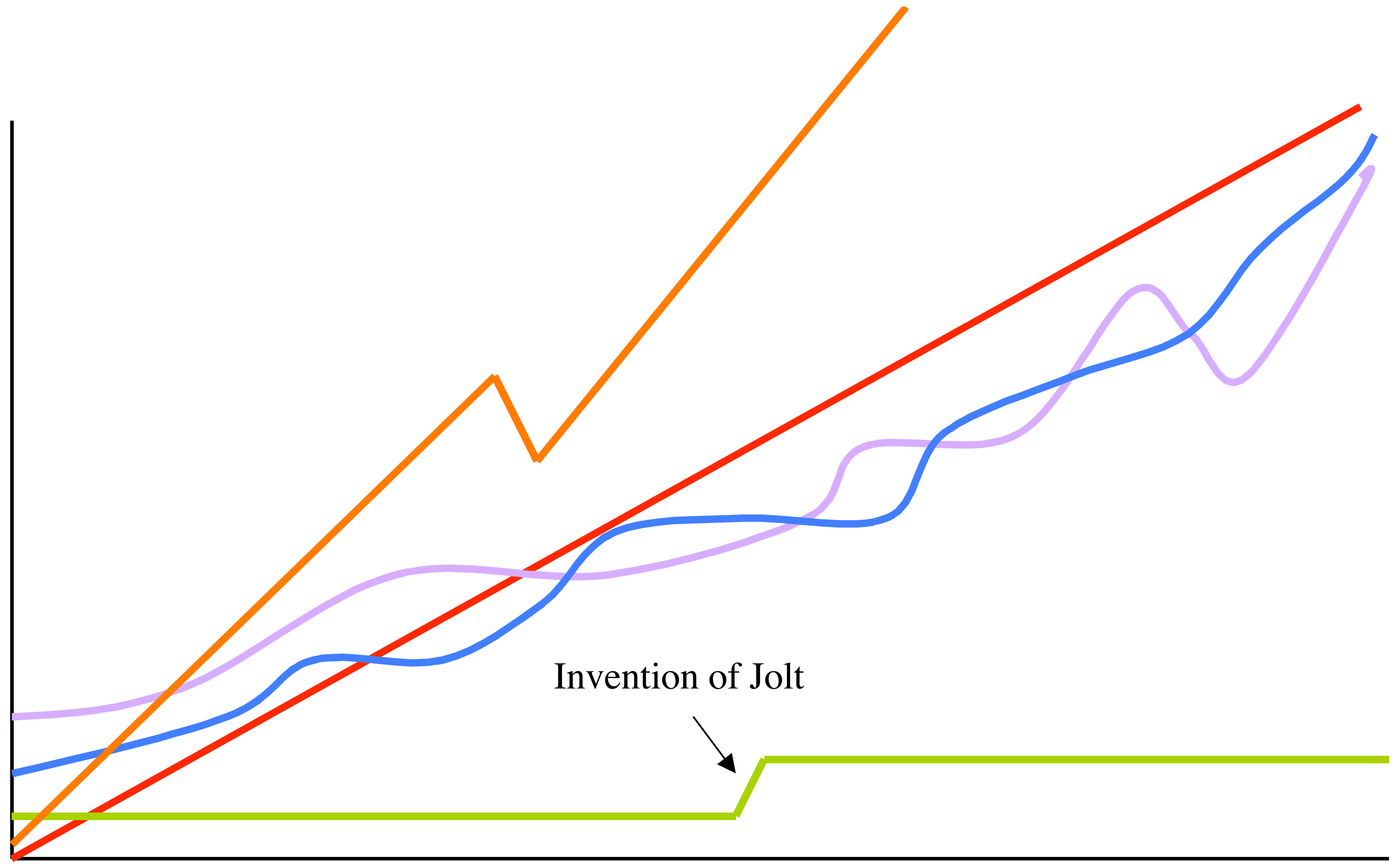


🟥 Moore's Law

🟦 Inflation

🟩 Developer Speed

🟪 Developer Salaries



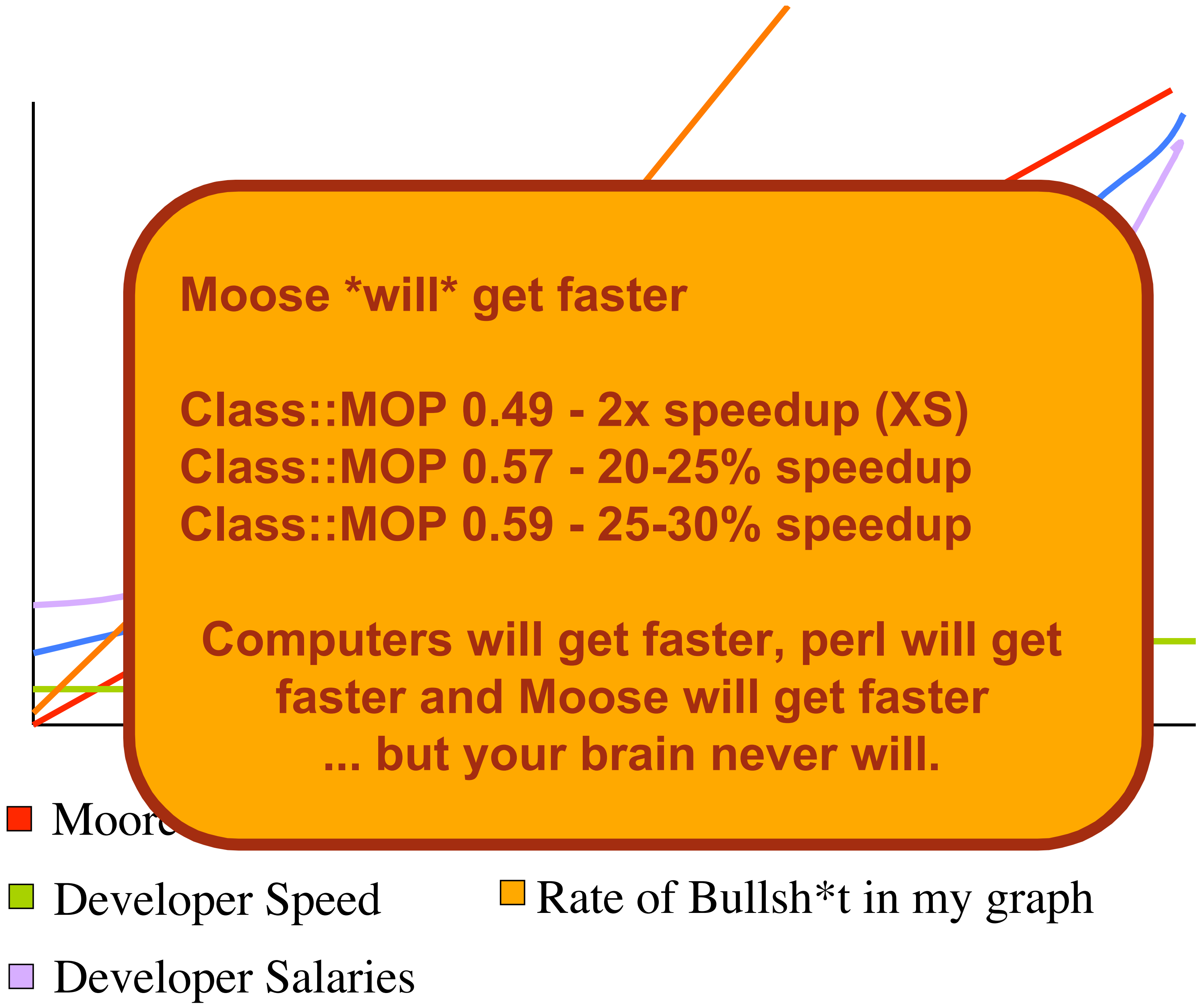
🟥 Moore’s Law

🟦 Inflation

🟩 Developer Speed

🟧 Rate of Bullsh*t in my graph

🟪 Developer Salaries



Moose *will* get faster

Class::MOP 0.49 - 2x speedup (XS)
Class::MOP 0.57 - 20-25% speedup
Class::MOP 0.59 - 25-30% speedup

Computers will get faster, perl will get faster and Moose will get faster ... but your brain never will.