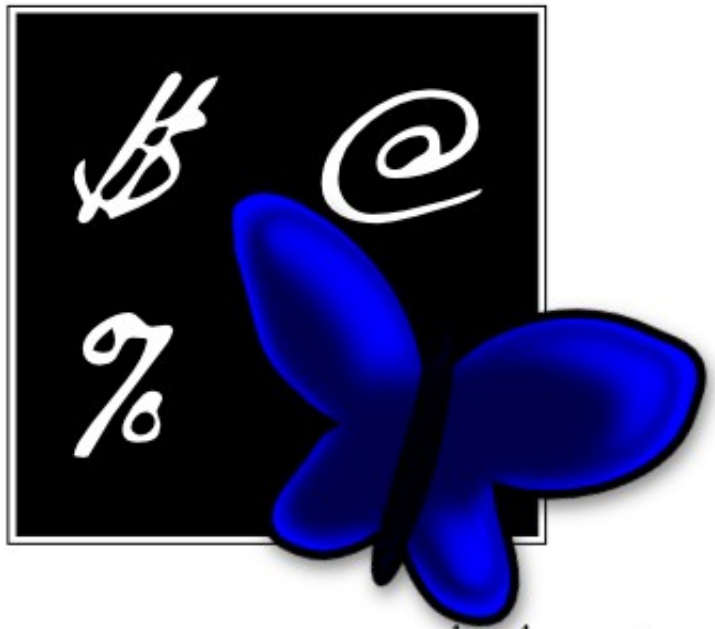
Take Advantage of Modern Perl

http://www.modernperlbooks.com/

http://onyxneon.com/books/modern_perl/

Modern Perl



by chromatic

ONYX NEON

Modern Perl is how the world's best Perl 5 programmers write great code.

You must understand the language

You must embrace the CPAN.

Essential Perl Philosophy

Context

You already understand this in English.

Sir, I **object** to that **object**!

Two Types of Context in Perl

What kind of thing do I expect?

How many are there?

Value Context

Do I want:

- A string?
- A number?
- An integer?

Amount Context

Do I want:

- No **things**?
- One **thing**?
- Several **things**?

Context is (Usually) Static

```
my $item = do_something();
do_something();
my @items = do_something();
```

Pay Attention to Punctuation

```
my ($item)     = do_something();
$hash{key}     = do_something();
@{ $arr_ref } = do_something();
```

Understand Operators

```
my $string    = $value . $value;
my $num       = $value + $value;
my $something = $value++;
```

The Noble Comma is an Operator

```
do_something(), do_something();
foo( do_something(),
     do_something() );
my %stuff = (
    bar => do_something(),
);
```

Dynamic Context

```
Sub do_something
{
    my $context = wantarray();
    ...
}
```

Implicit Variables

You already know how they work.

If you understand pronouns, you get it.

It is a Pronoun

Think *it* when you see $_ …

… and when you don't.

Read Into It

```
while (<$fh>)
{
    ...
}
```

Chomp It

```
while (<$fh>)
{
    chomp;
    ...
}
```

Regex It

```
while (<$fh>)
{
    chomp;
    s/cake/pie/g;
    ...
}
```

Loop It

```
for (@pancakes)
{
    nom( $_ );
}

nom( $_ ) for @pancakes;
```

Print It

```
say for @waffles;
```

Them is a Pronoun

Think them when you see @_ …

… and when you don't.

(*mostly*)

Shift It

```
sub a_great_method
{
    my $self = shift;
    …
}
```

Pop It

```
sub a_great_method
{
    my $self   = shift;
    my $unself = pop;
    …
}
```

… @ARGV is Them Too

```
die ”You don't know me!”
    unless @ARGV == 1;

my $path = shift;
main( $path );
```

Perldoc

Perl includes thousands of pages of documentation.

Documentation for Builtins

Your best friends on the command line:

```
$ perldoc -f whatever
$ perldoc perlop
```

Documentation for Ops

```
$ perldoc perlop
```

Documented FAQs

```
$ perldoc perlfaq
$ perldoc -q keywords
```

Read the TOC

```
$ perldoc perltoc
```

Read the Syn

```
$ perldoc perlsyn
```

Concision

Less structure, more meaning.

No More Manual Iteration

```
for (my $i = 0; $i < @foo; $i++)

for my $item (@foo)
```

No More Manual Searching

```
for my $item (@foo)
{
    next unless $item eq 'Whatever';
}

my $found = grep
    { $_ eq 'Whatever' } @foo;
```

No More Manual List Transformations

```
my @output;
push @output, $_ * 2 for @input;

my @output =
    map { $_ * 2 } @input;
```

Chunking

Learn to understand the *units* of Perl code.

Find the Operators

These are the *verbs* and control the behavior.

Find the Values

What *type* of value?

Are there special conversions?

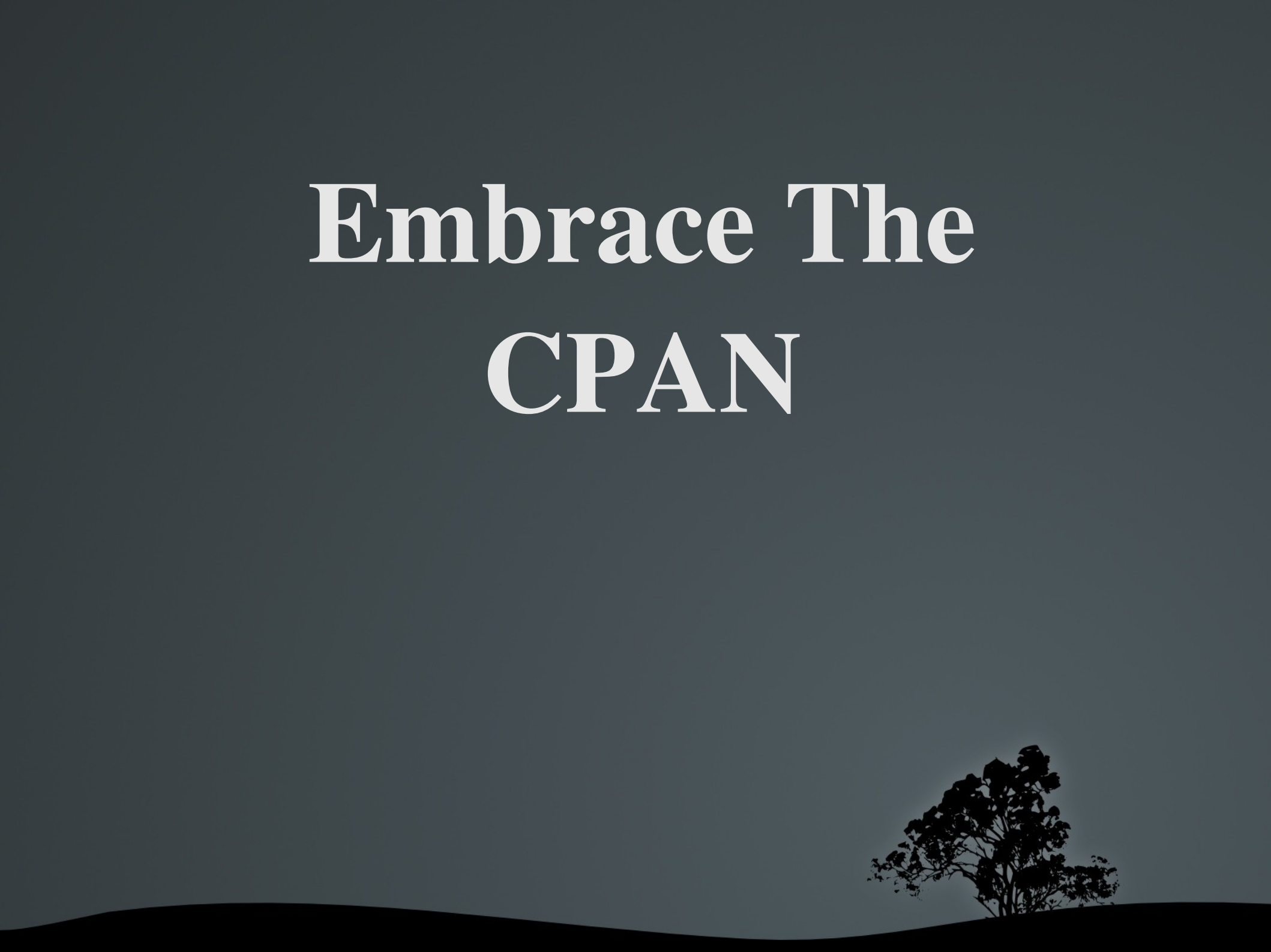
(Understand the Context)

Precedence, static context, and any dynamic context all affect chunks and interpretation.

Make Friends with B::Deparse

```
$ perl -MO=Deparse some_code.pl
```

Embrace The CPAN



Perl::Tidy and Perl::Critic

Reformat your code.

Critique your code.

Let your style evolve.

Where rEvolution Happens

The pressure of backwards compatibility.

Making CPAN Easier

- Autoconfiguration
- App::perlbrew
- App::cpanminus
- local::lib
- Task::Kensho & Strawberry Perl
- CPAN::Mini
- Reusable CPAN infrastructure

The Testing Revolution

- TAP, Test::Builder, Test::*, CPANTS, Kwalitee, and you.

Moose

Cheerful thievery and syncretism.

Moose Attributes

```
package Point
{
    use Moose;
    has 'x', is => 'rw', isa => 'Int';
    has 'y', is => 'rw', isa => 'Int';
}
```

Moose Subclassing

```
package Point::3D
{
    use Moose;
    extends 'Point';

    has 'z', is => 'rw', isa => 'Int';
    after 'clear' => sub {
        my $self = shift;
        $self->z(0);
    };
}
```

Moose Advice

Calling named methods Before, After, and Around pre-existing methods.

Moose Roles

Composable units of object behavior.

```
package Breakable
use Moose::Role;
has 'is_broken', is  => 'rw',
                 isa => 'Bool';

package Toy;
use Moose;
with 'Breakable';
```

Moose and the MOP

A well-designed API for building powerful, useful, extensible object systems.

See MooseX::*

Try::Tiny

```
try   { die "foo" }
catch { warn "caught error: $_" };
```

DBIx::Class

ORM built around Schemas, ResultSets, and Rows.

A Thousand More Flowers Bloom

Consensus of ecosystem, not solely syntax or mechanism.

Perl 5 is Modern and Powerful and Useful