
whoami


Tobias


FROGGS


Slangs


sub languages


syntax?


dialect


phonology


phonology


morphology


<foo bar baz>.join

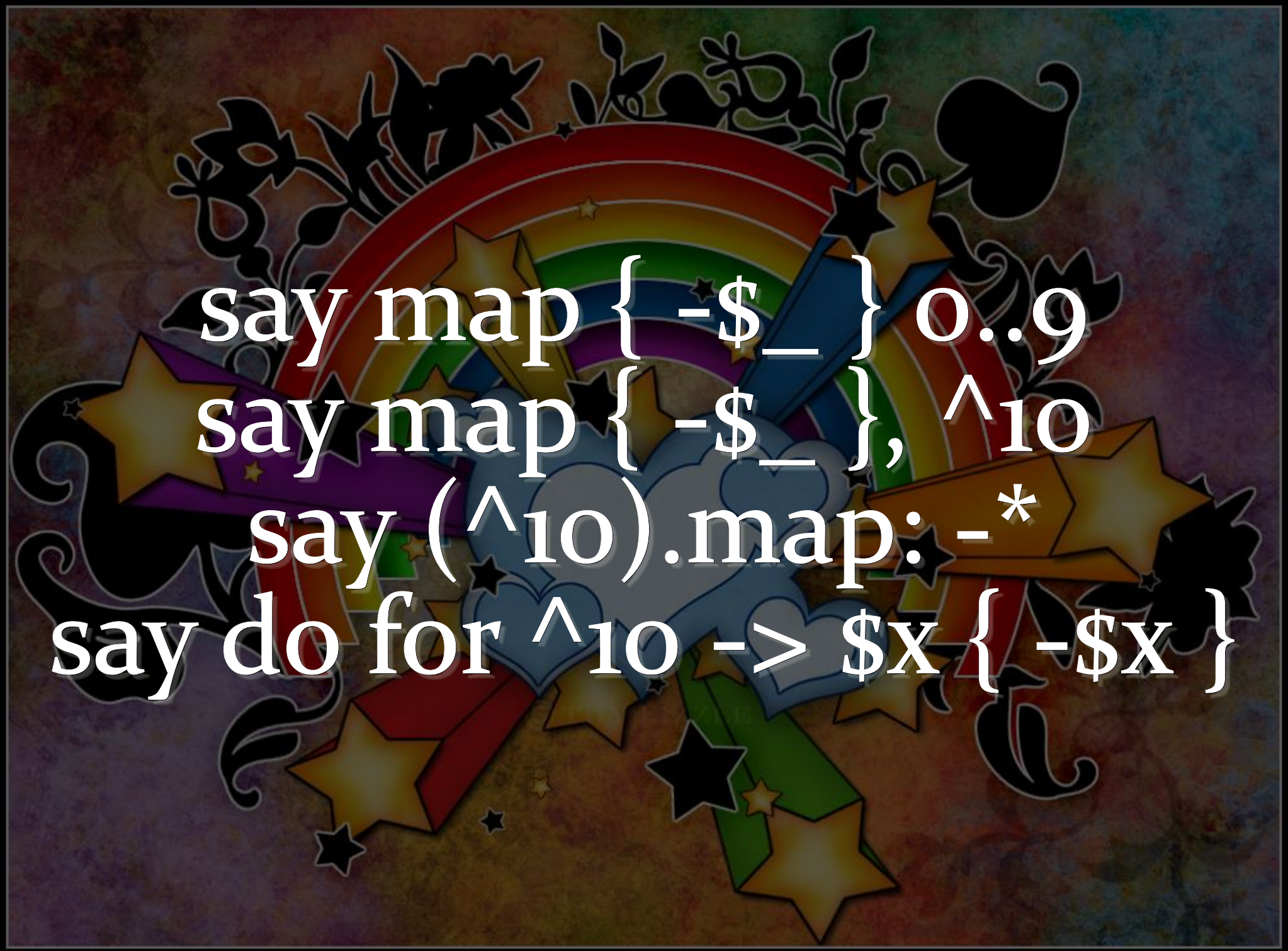

morphology


lexic


use Foo;


lexic ✓


syntax

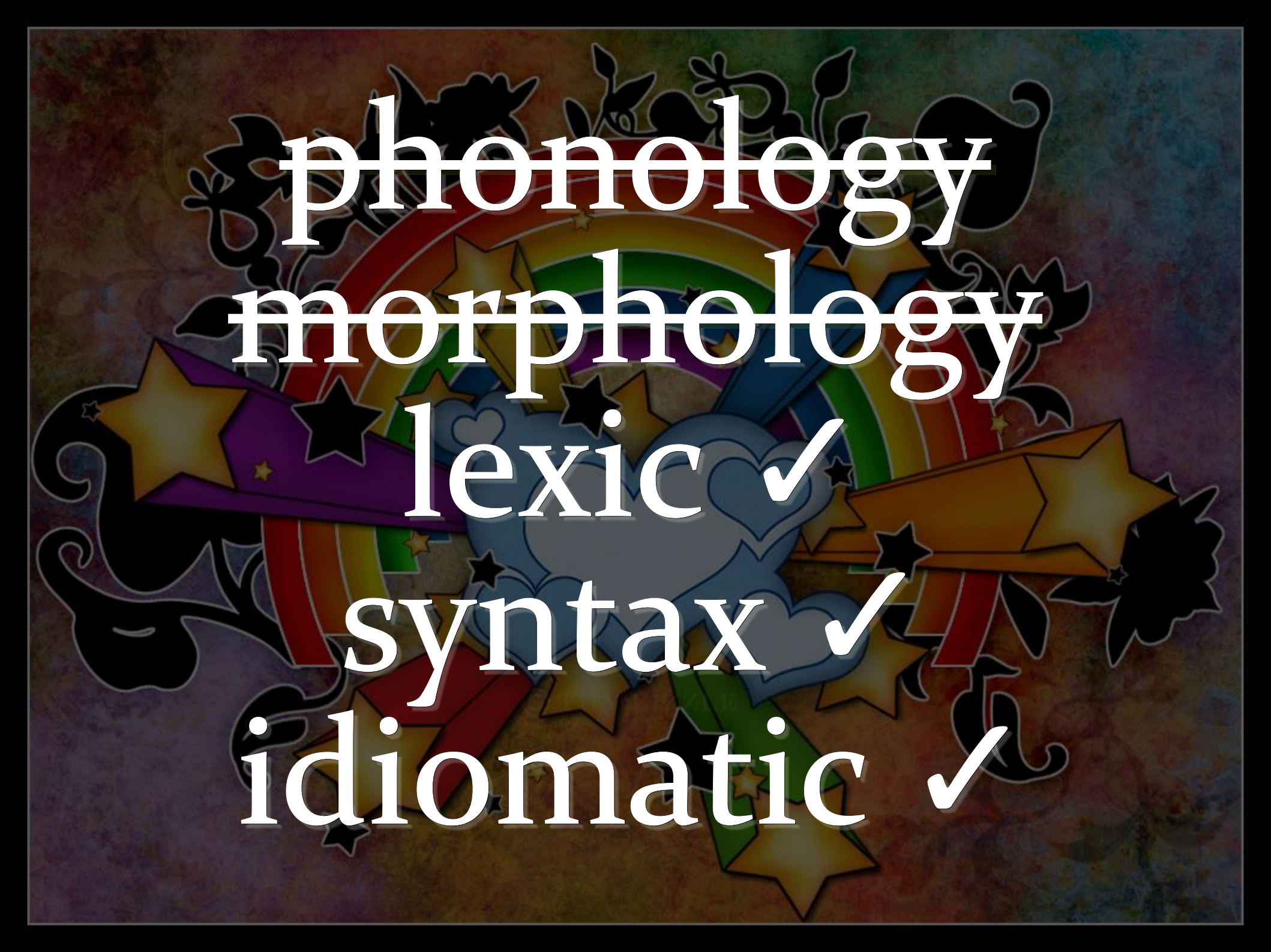

say map { -$_ } 0..9
say map { -$_ }, ^10
say (^10).map: -*
say do for ^10 -> $x { -$x }


syntax ✓

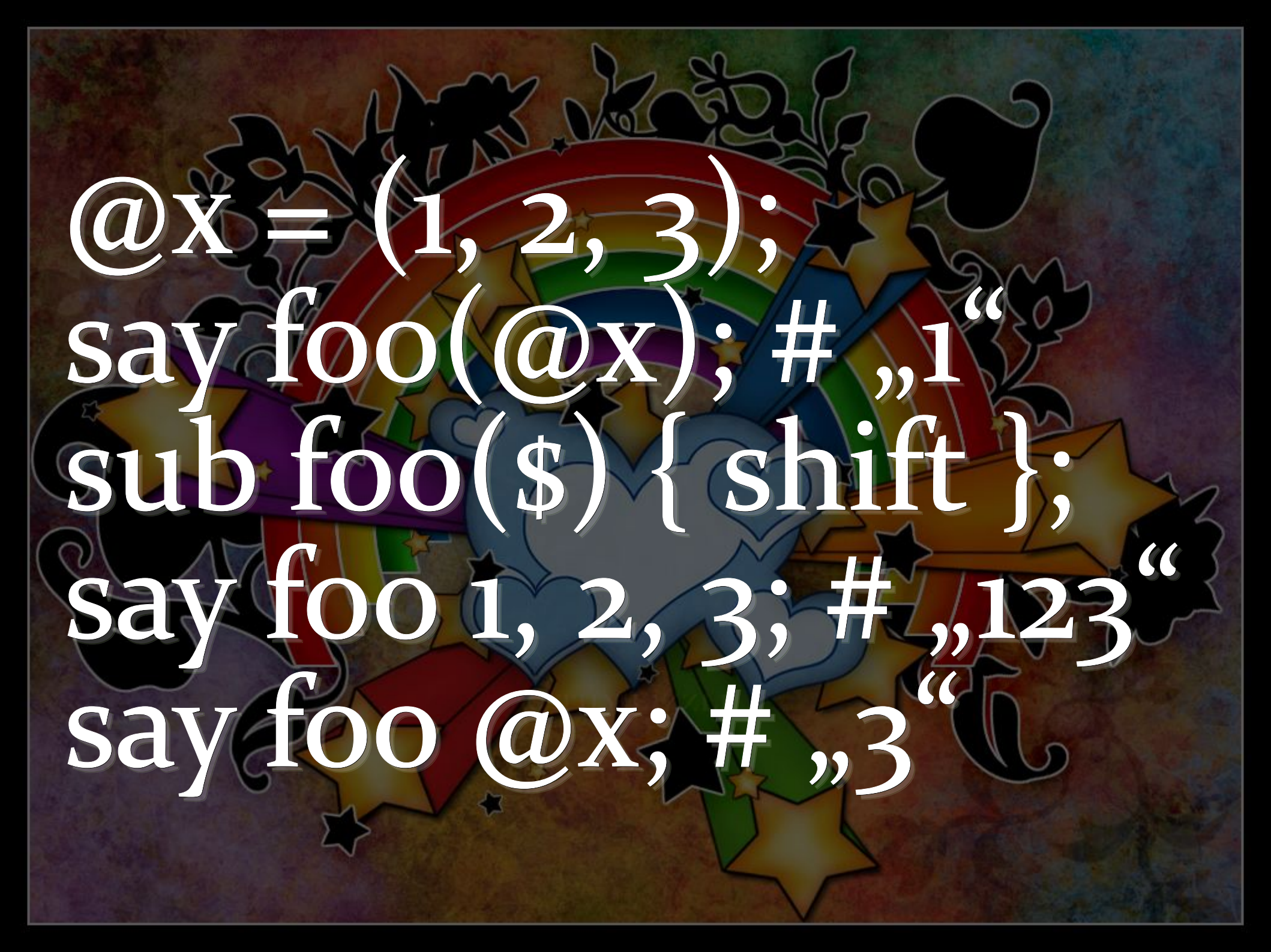

idiomatic

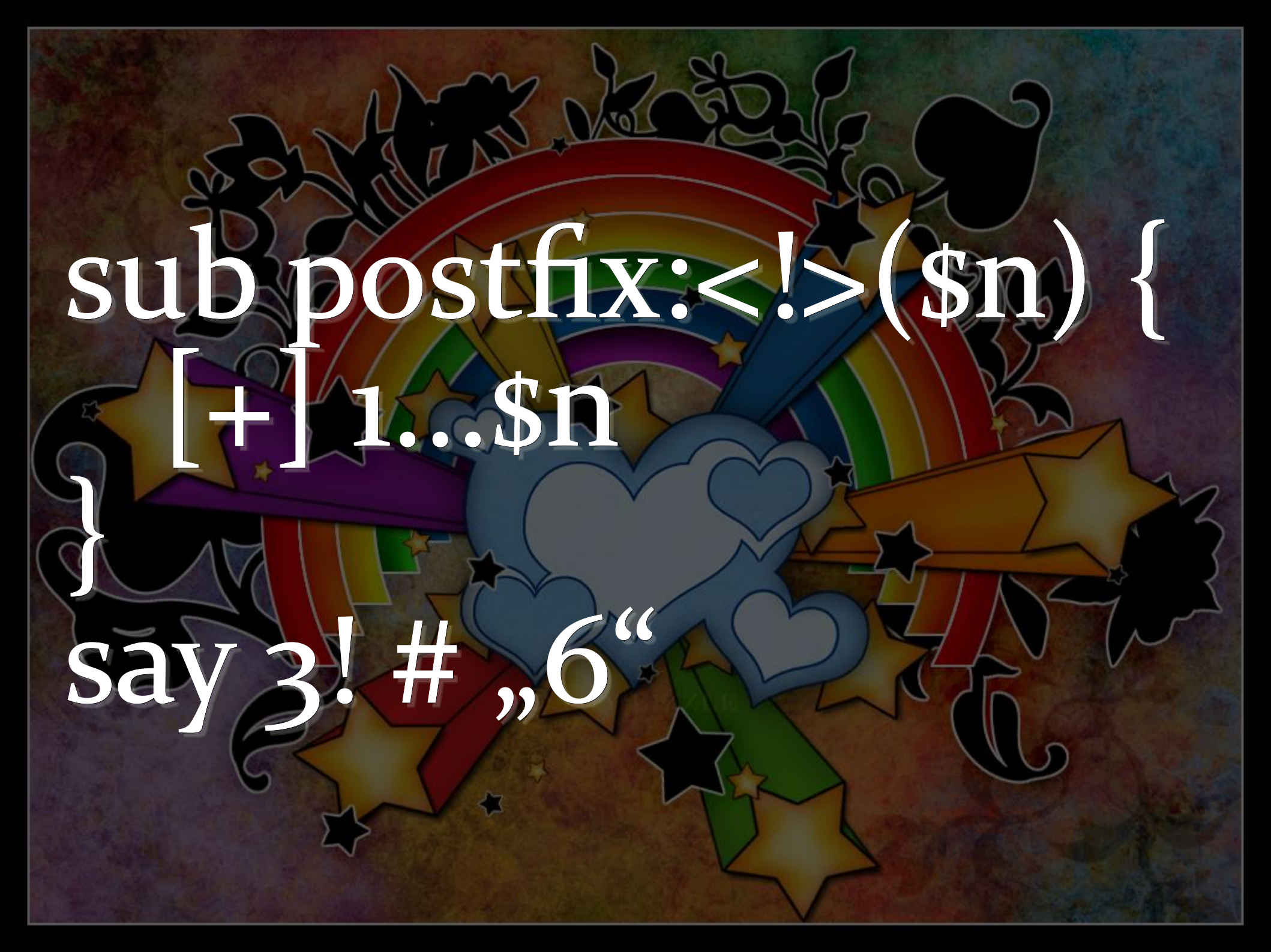

sub foo { my %opt = @_ }
sub foo(*%opt) { ... }
<?php ... ?>

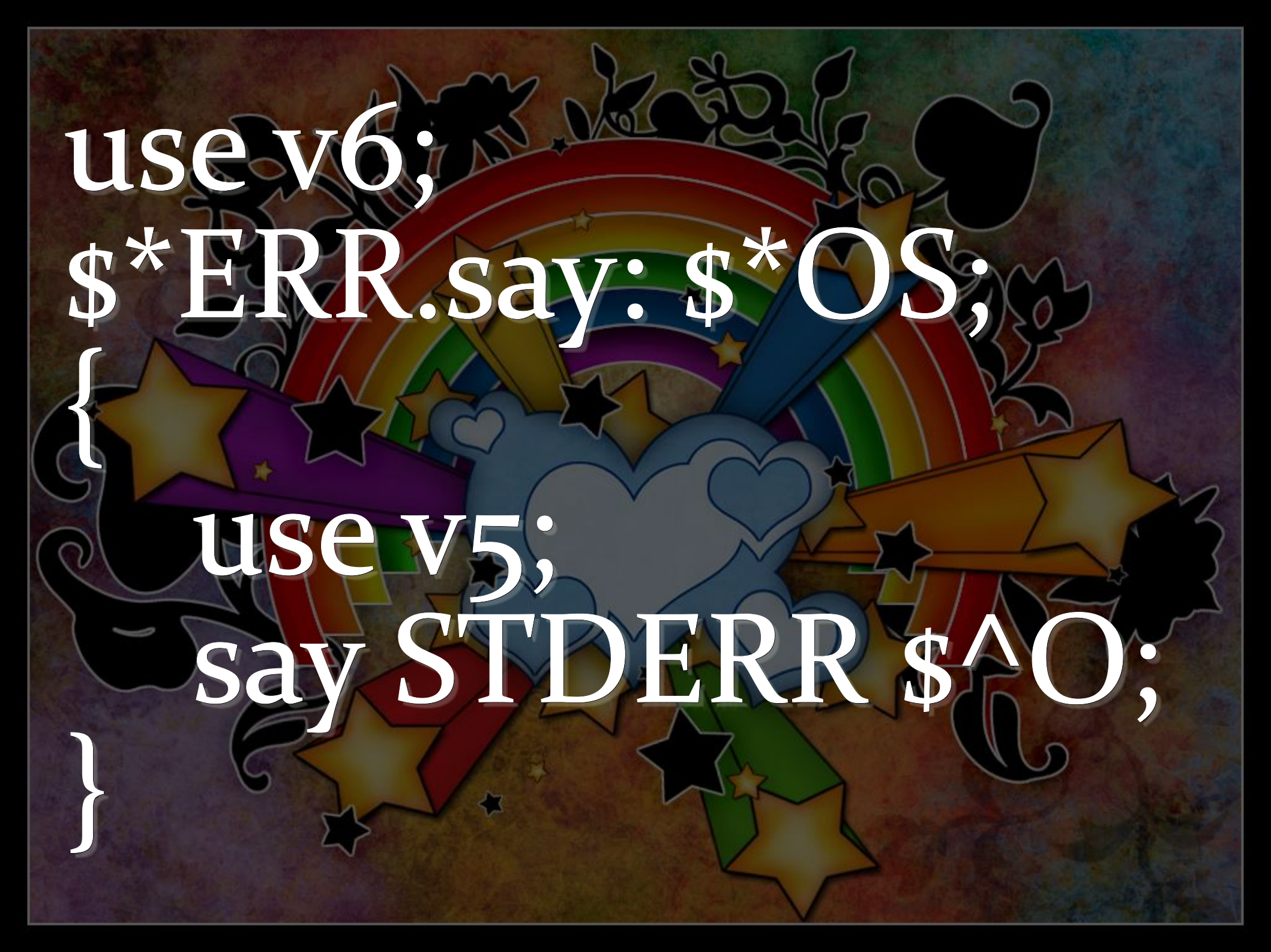

idiomatic ✓

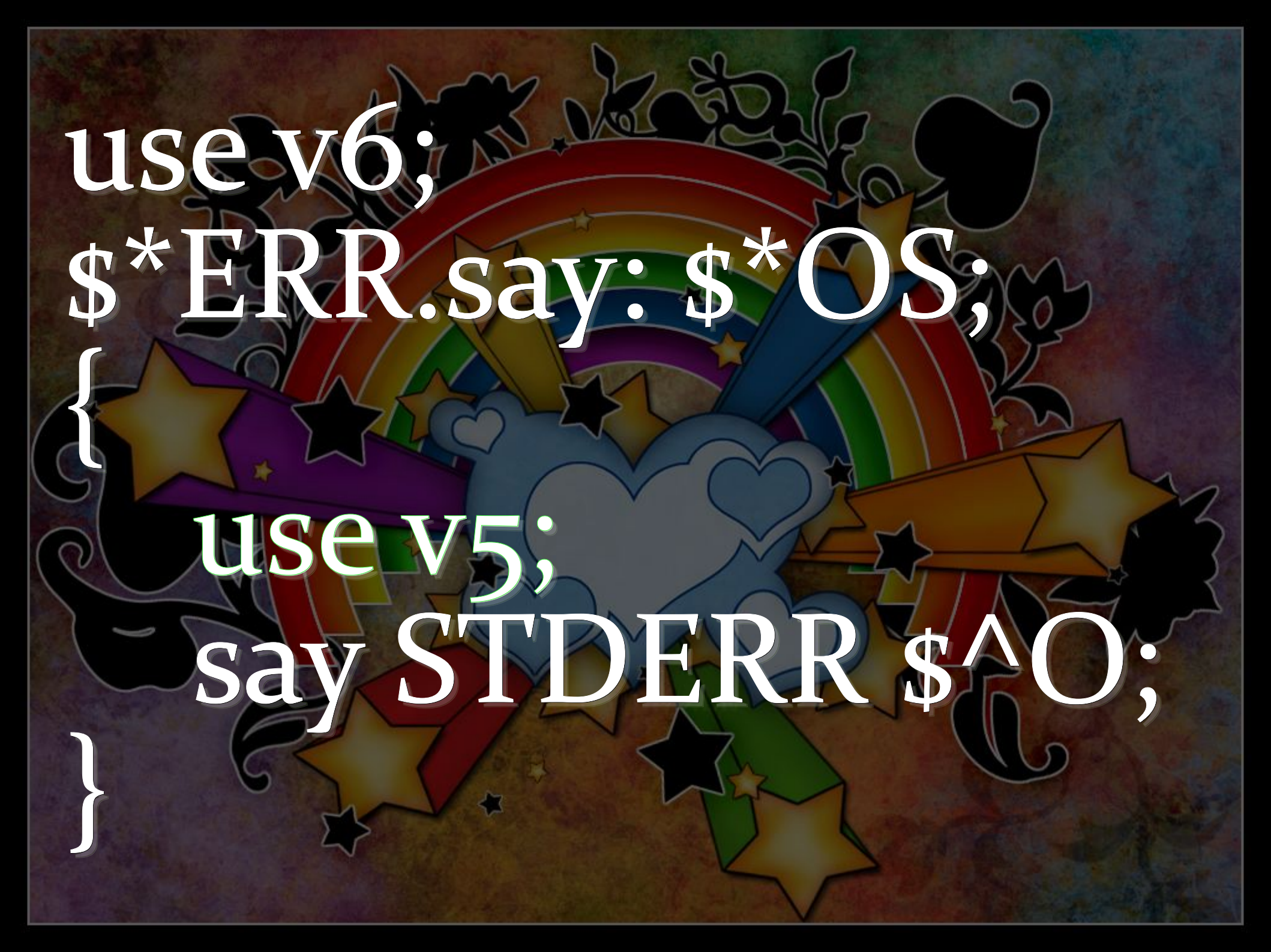

~~phonology~~
~~morphology~~
lexic ✓
syntax ✓
idiomatic ✓


I FORGOT HOW
TO SLANG


@x = (1, 2, 3);
say foo(@x); # „1“
sub foo($) { shift };
say foo 1, 2, 3; # „123“
say foo @x; # „3“

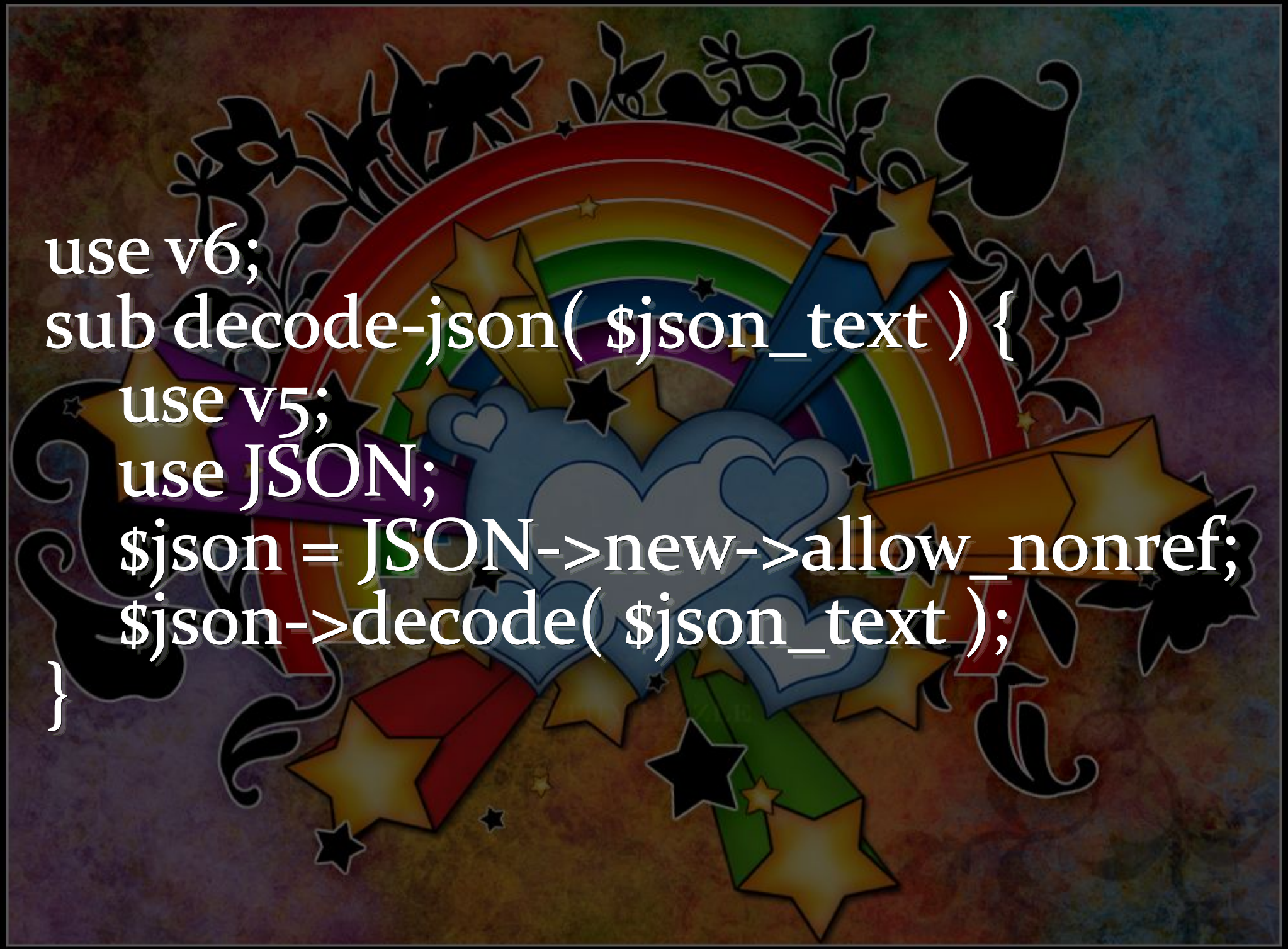

sub postfix:<!>($n) {
[+] 1...$n
}
say 3! # „6“

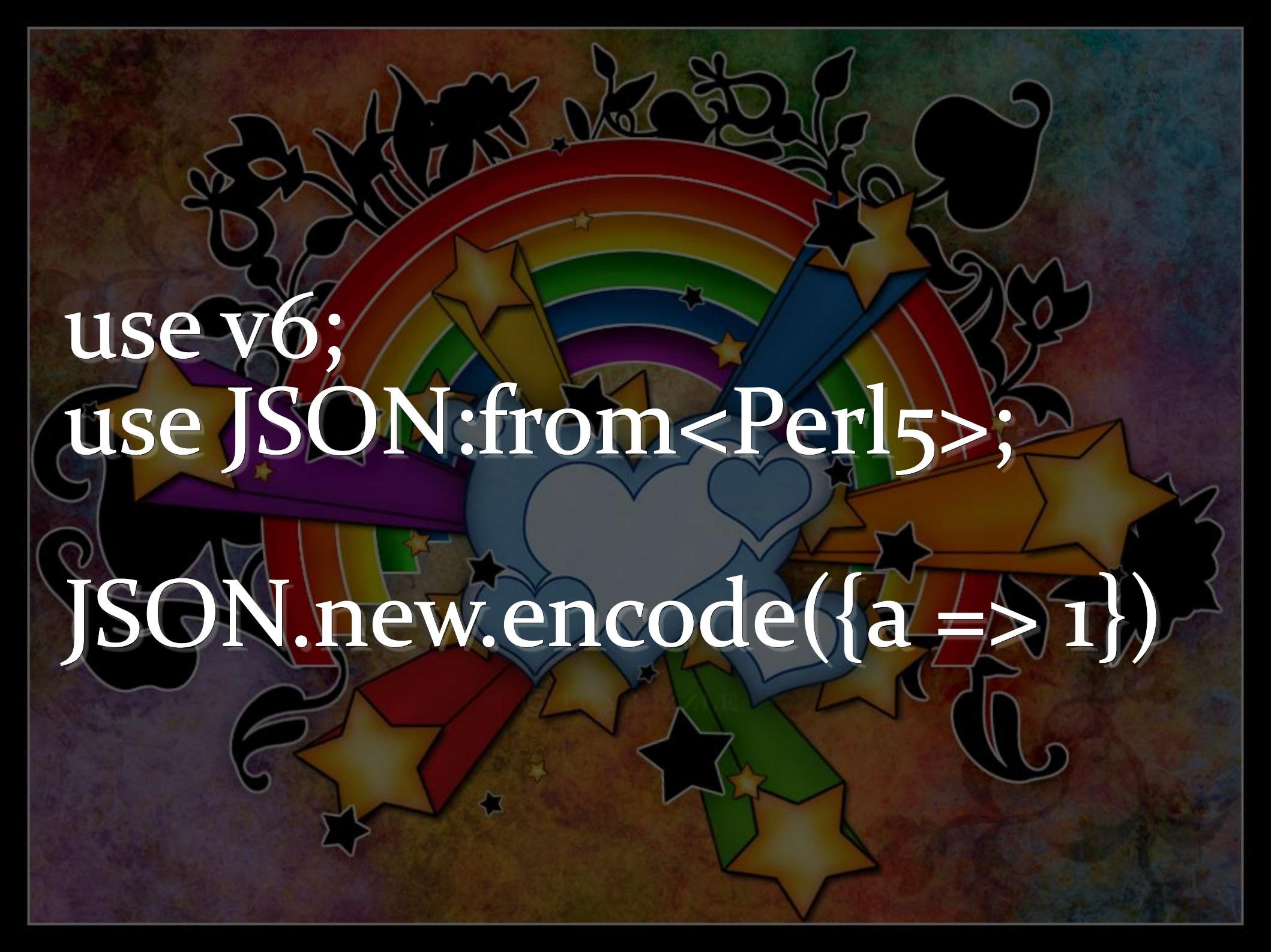

use v6;
$*ERR.say: $*OS;
{
use v5;
say STDERR $^O;
}

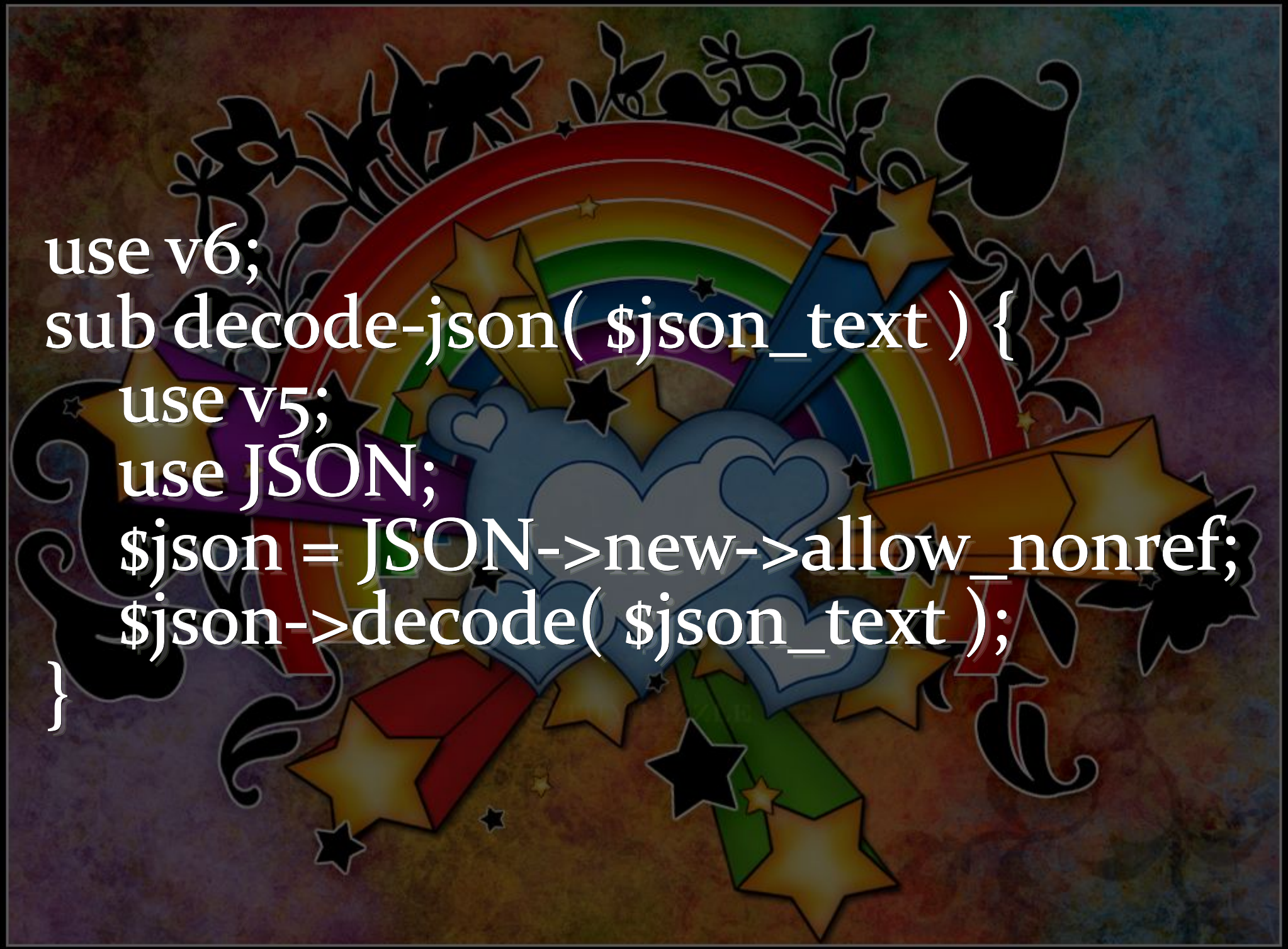

use v6;
$*ERR.say: $*OS;
{
use v5;
say STDERR $^O;
}

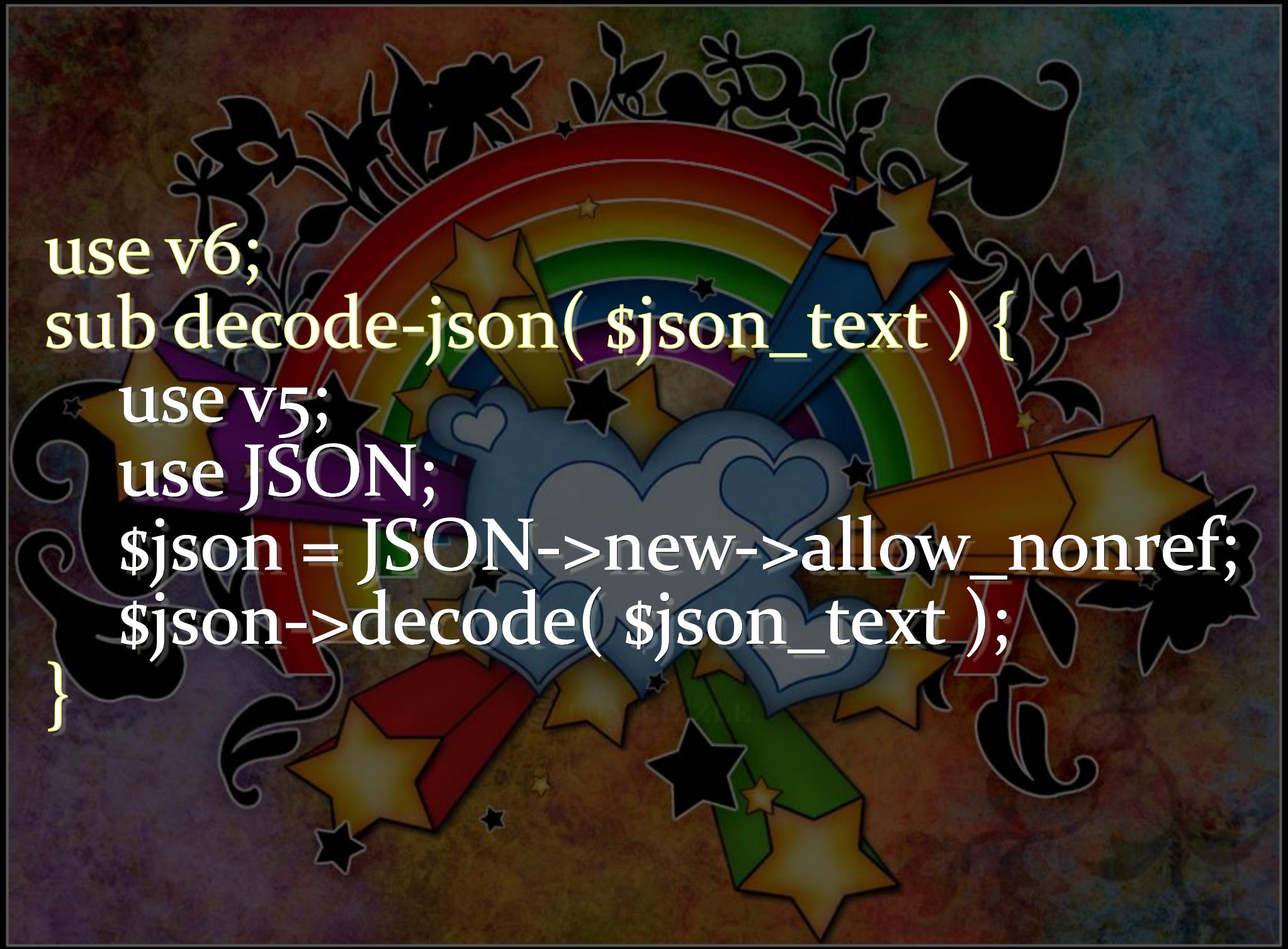

use v5;

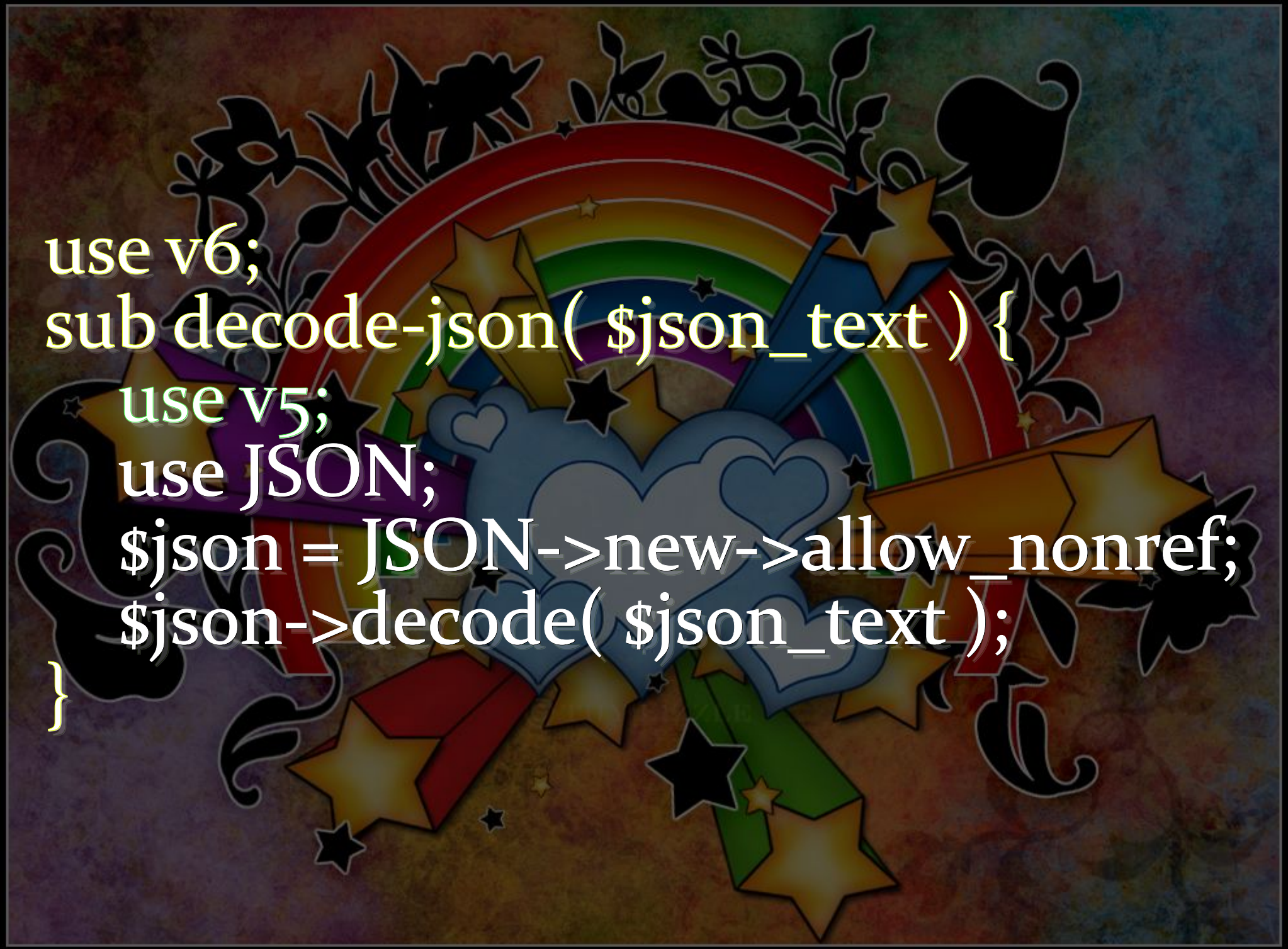
```
use v5;
sub is_valid {
    shift =~ m=^(((ht|f)tp(s?))\://)?
        (www.|[a-zA-Z].)
        [a-zA-Z0-9\-\.]+\.(com)(\:[0-9]+)*
        (/($|[a-zA-Z0-9\.\,\;\?\'\\\+&amp;amp;%\
        $#\~_\-]+))*$=i
}

{
    use 6.0.0;
    say ?is_valid( “http://github.com“ )
}
```

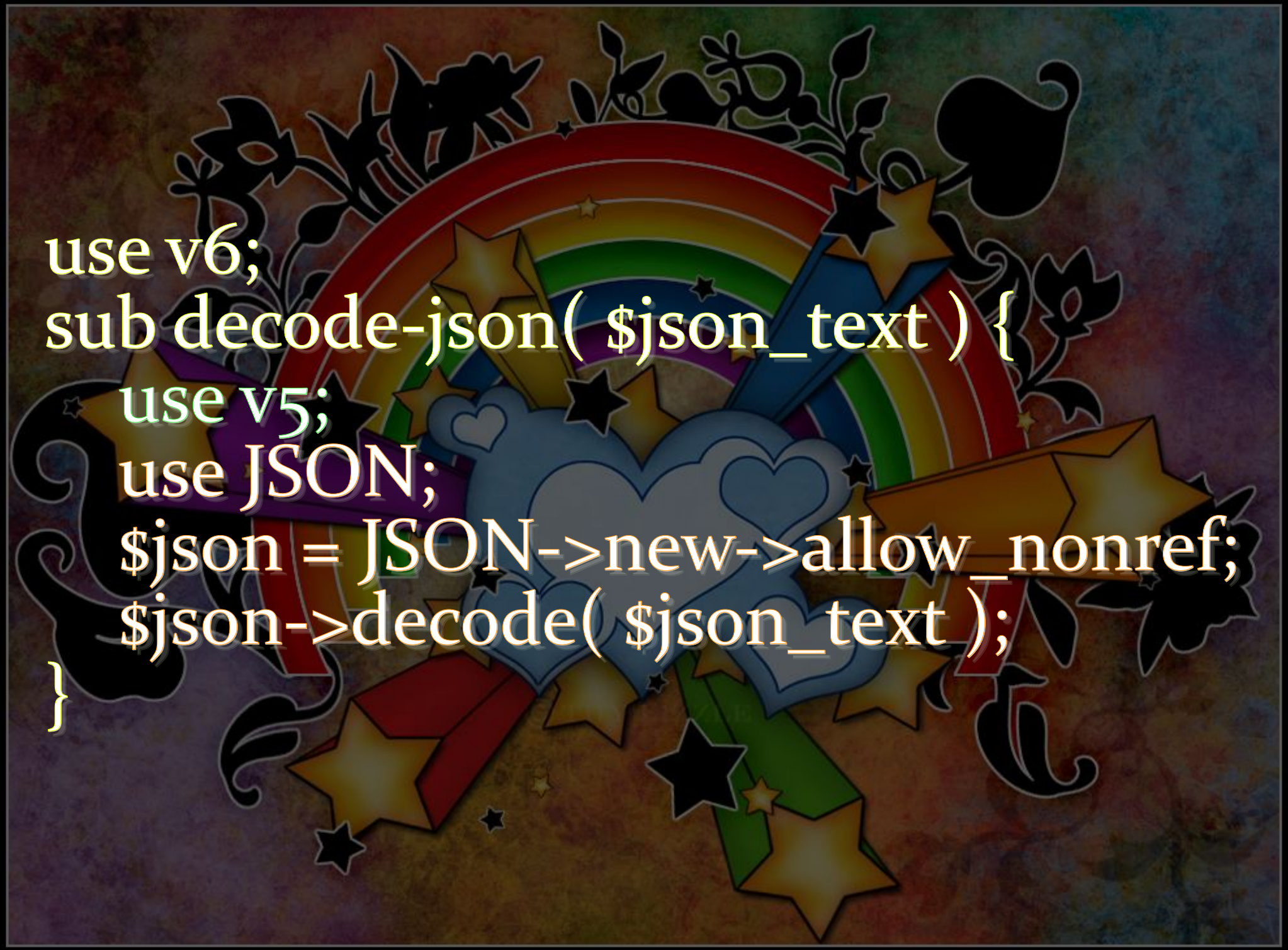

use v6;
sub decode-json($json_text) {
use v5;
use JSON;
$json = JSON->new->allow_nonref;
$json->decode($json_text);
}


use v6;
use JSON:from<Perl5>;
JSON.new.encode({a => 1})


use v6;
sub decode-json($json_text) {
use v5;
use JSON;
$json = JSON->new->allow_nonref;
$json->decode($json_text);
}


use v6;
sub decode-json($json_text) {
use v5;
use JSON;
$json = JSON->new->allow_nonref;
$json->decode($json_text);
}


use v6;
sub decode-json($json_text) {
use v5;
use JSON;
$json = JSON->new->allow_nonref;
$json->decode($json_text);
}

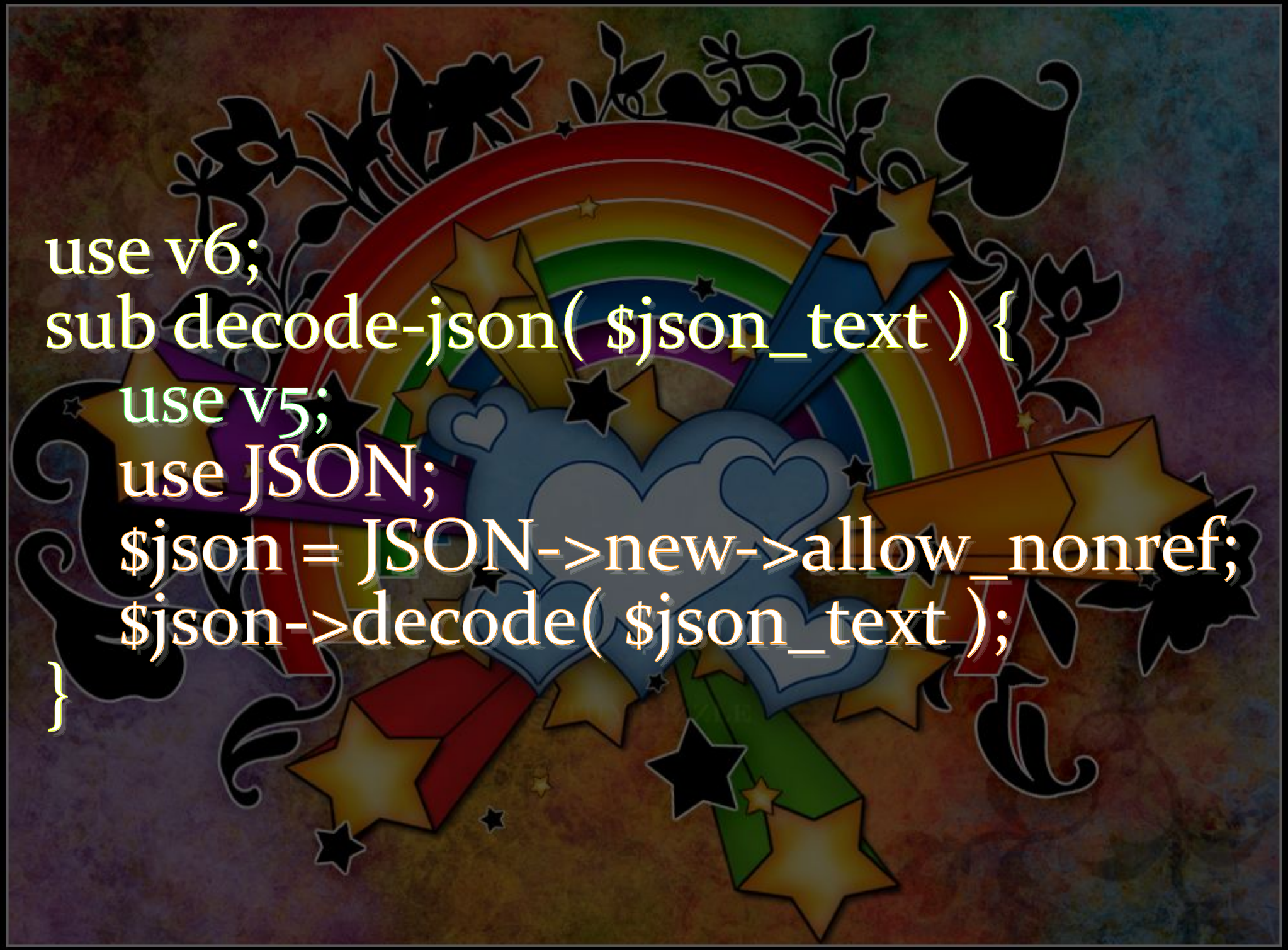

use v6;
sub decode-json($json_text) {
use v5;
use JSON;
$json = JSON->new->allow_nonref;
$json->decode($json_text);
}

```
token comp_unit {
    :my $*MAIN := “MAIN“;
}
token statement_control:sym<use> {
    :my $OLD_MAIN := ~$*MAIN;
    <sym> <.ws> [
        || <version>
        || <module_name> <version>?
    ]
    [ <?{ $*MAIN ne $OLD_MAIN }>
      <statementlist=.LANG($*MAIN,
                           'statementlist')> ]?
}
```

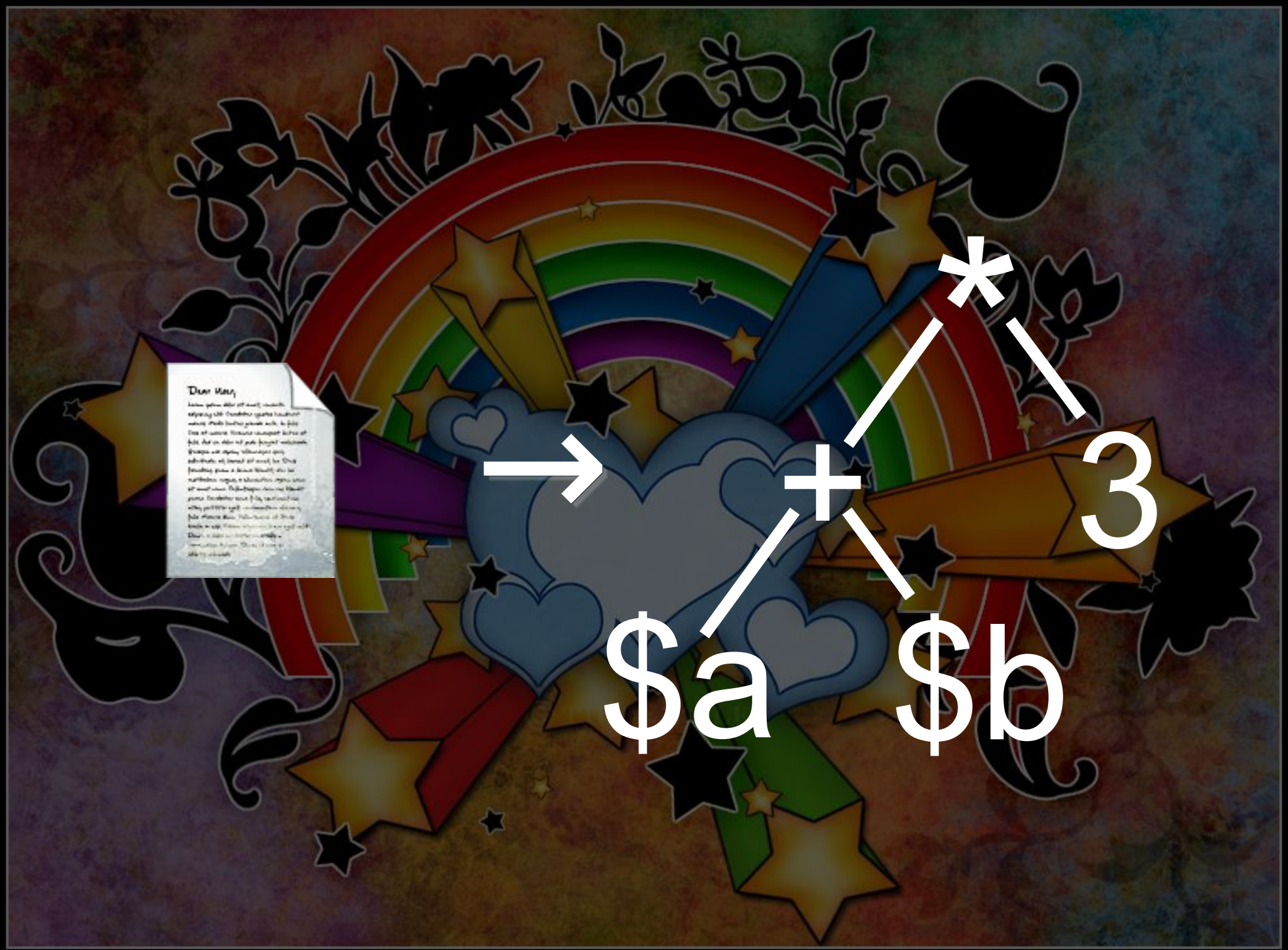
```
token comp_unit {
    :my $*MAIN := “MAIN“;
}
token statement_control:sym<use> {
    :my $OLD_MAIN := ~$*MAIN;
    <sym> <.ws> [
        || <version>
        || <module_name> <version>?
    ]
    [ <?{ $*MAIN ne $OLD_MAIN }>
      <statementlist=.LANG($*MAIN,
                           'statementlist')> ]?
}
```

```
grammar P5::Grammar {
    token statementlist { <statement>* }
}
class P5::Actions {
    method statementlist($/) {
        for $<statement> { [...] }
    }
}
sub EXPORT(*@a) {
    %*LANG<Perl5> := P5::Grammar;
    %*LANG<Perl5-actions> := P5::Actions;
    $*MAIN := “Perl5“;
}
```

```
grammar P5::Grammar {
    token statementlist { <statement>* }
}
class P5::Actions {
    method statementlist($/) {
        for $<statement> { [...] }
    }
}
sub EXPORT(*@a) {
    %*LANG<Perl5> := P5::Grammar;
    %*LANG<Perl5-actions> := P5::Actions;
    $*MAIN := "Perl5";
}
```

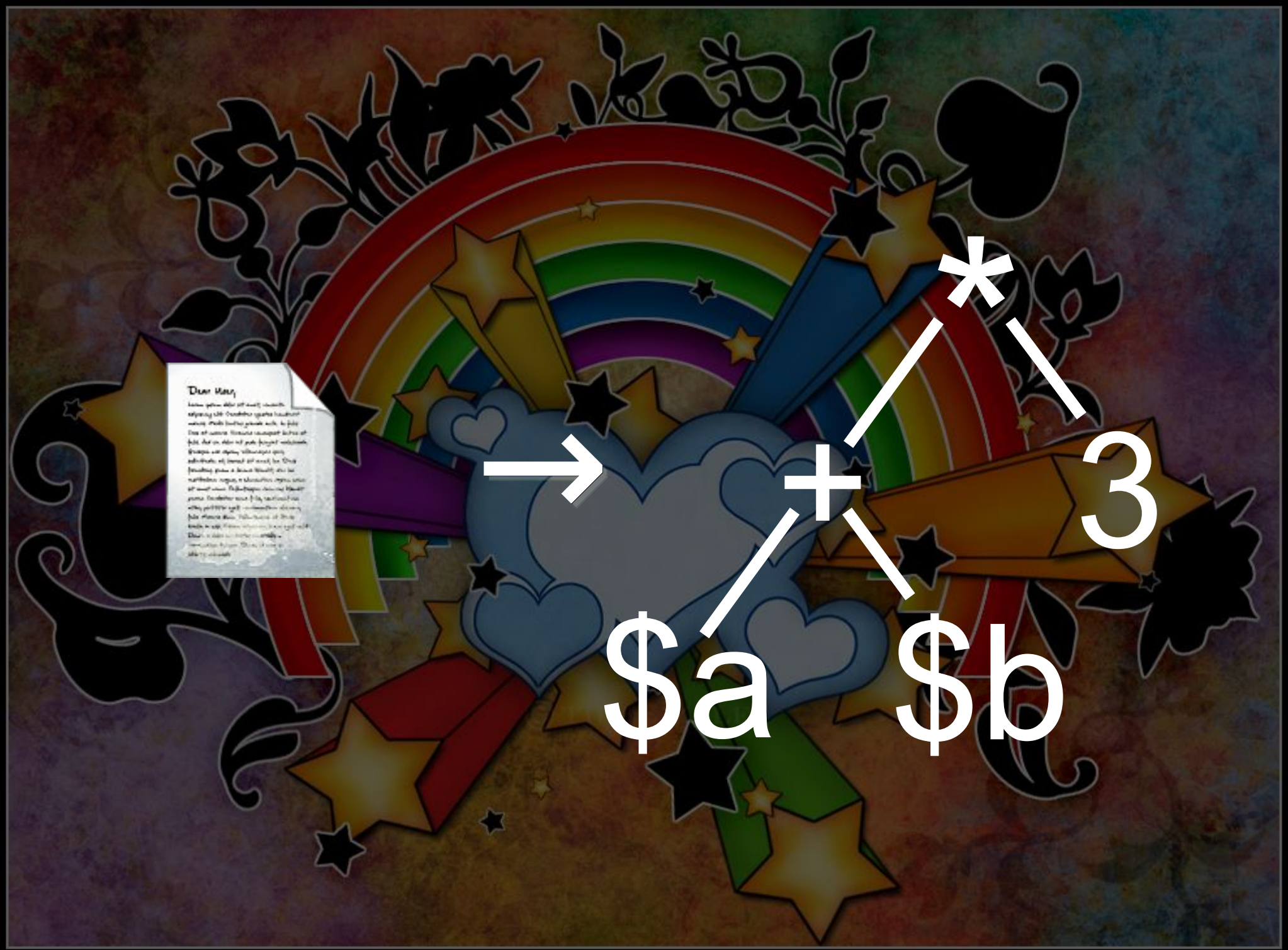

use v6;
sub decode-json($json_text) {
use v5;
use JSON;
$json = JSON->new->allow_nonref;
$json->decode($json_text);
}

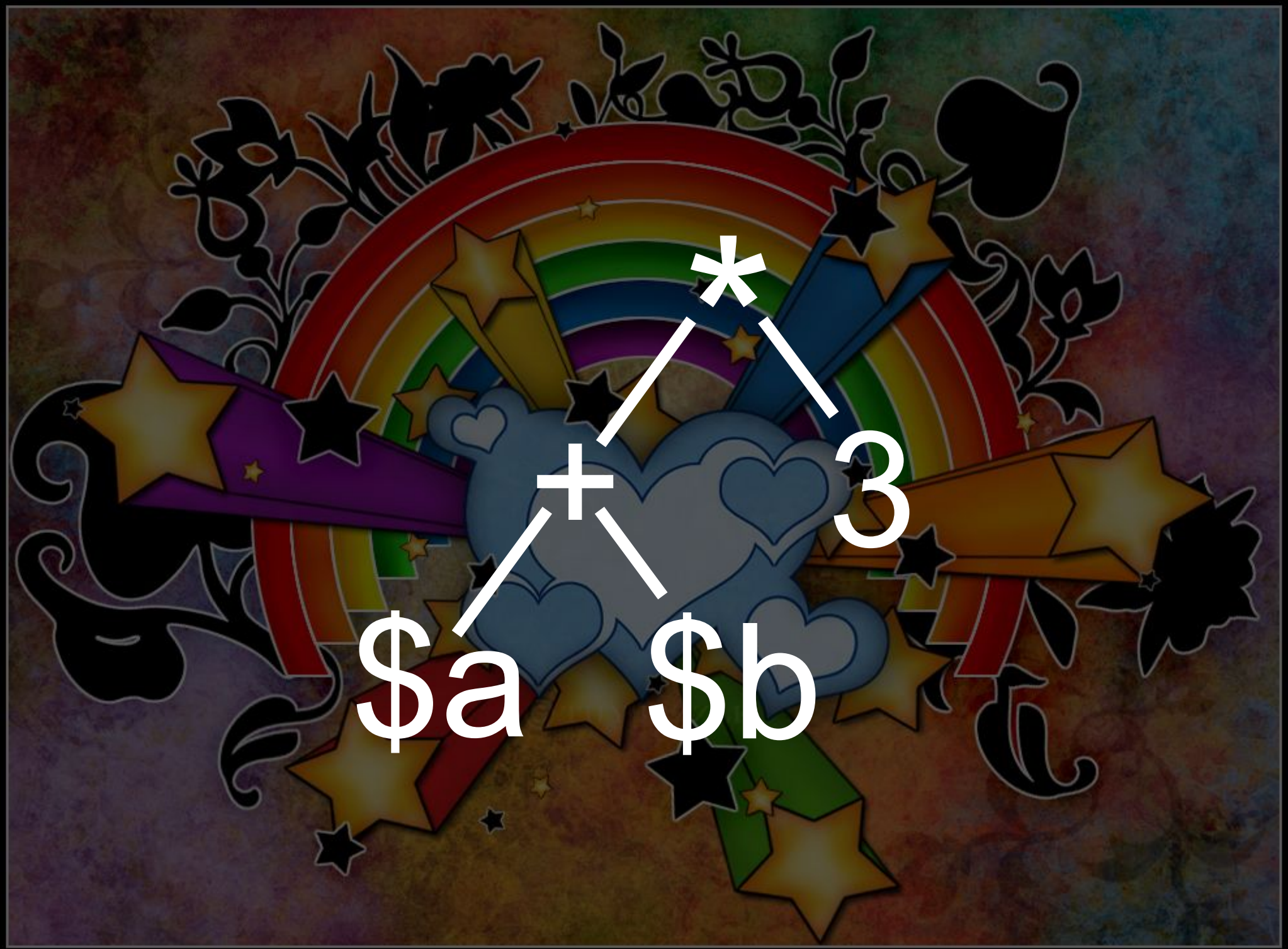

Parser?


*
+
3
$a
$b

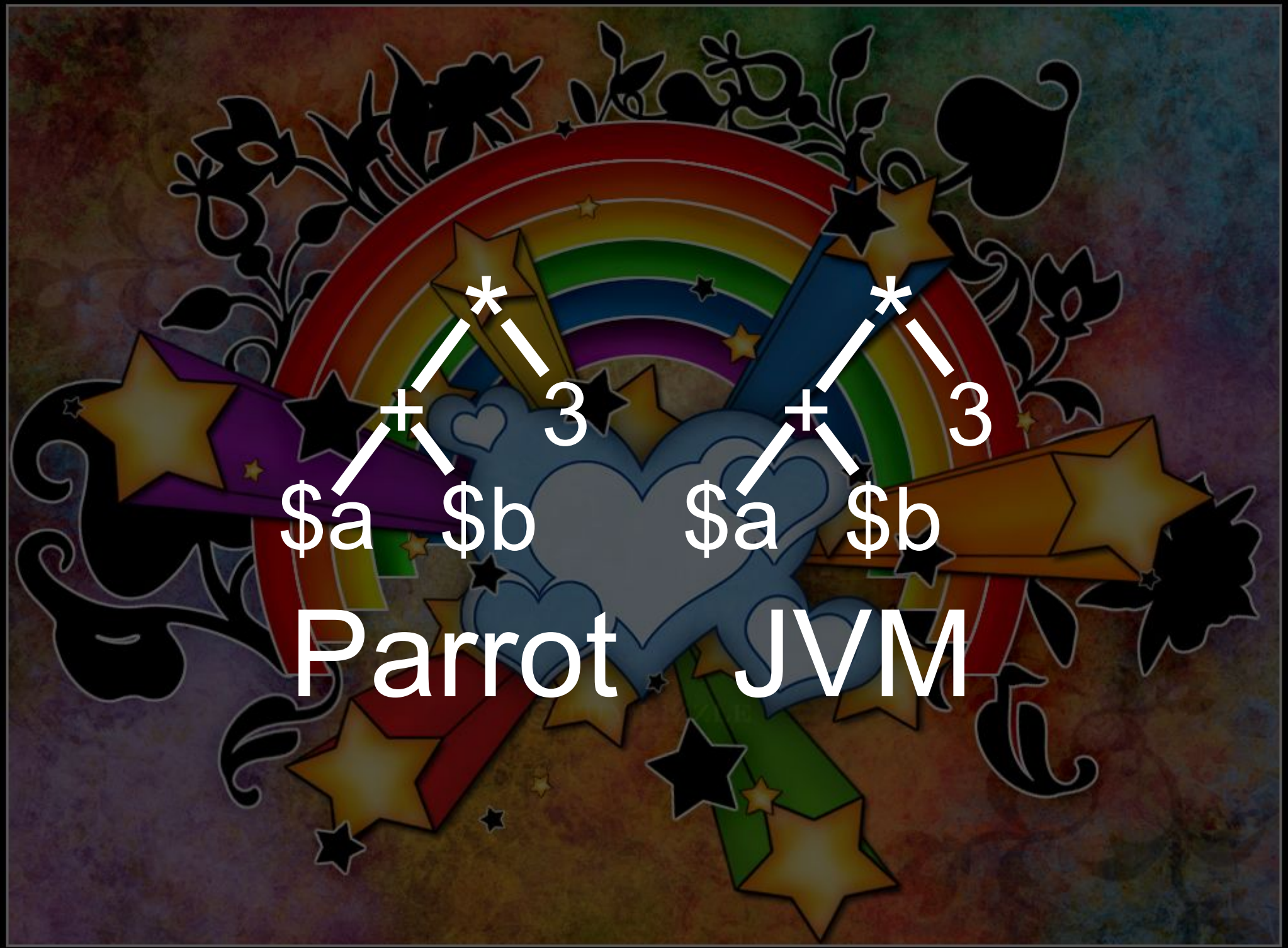
```
grammar P5::Grammar {
    token statementlist {
        <statement>*
    }
}
```

```
class P5::Actions {
    method statementlist($/) {
        for $<statement> { make .ast }
    }
}
```


*
+
3
$a
$b


*
+
3
$a
$b

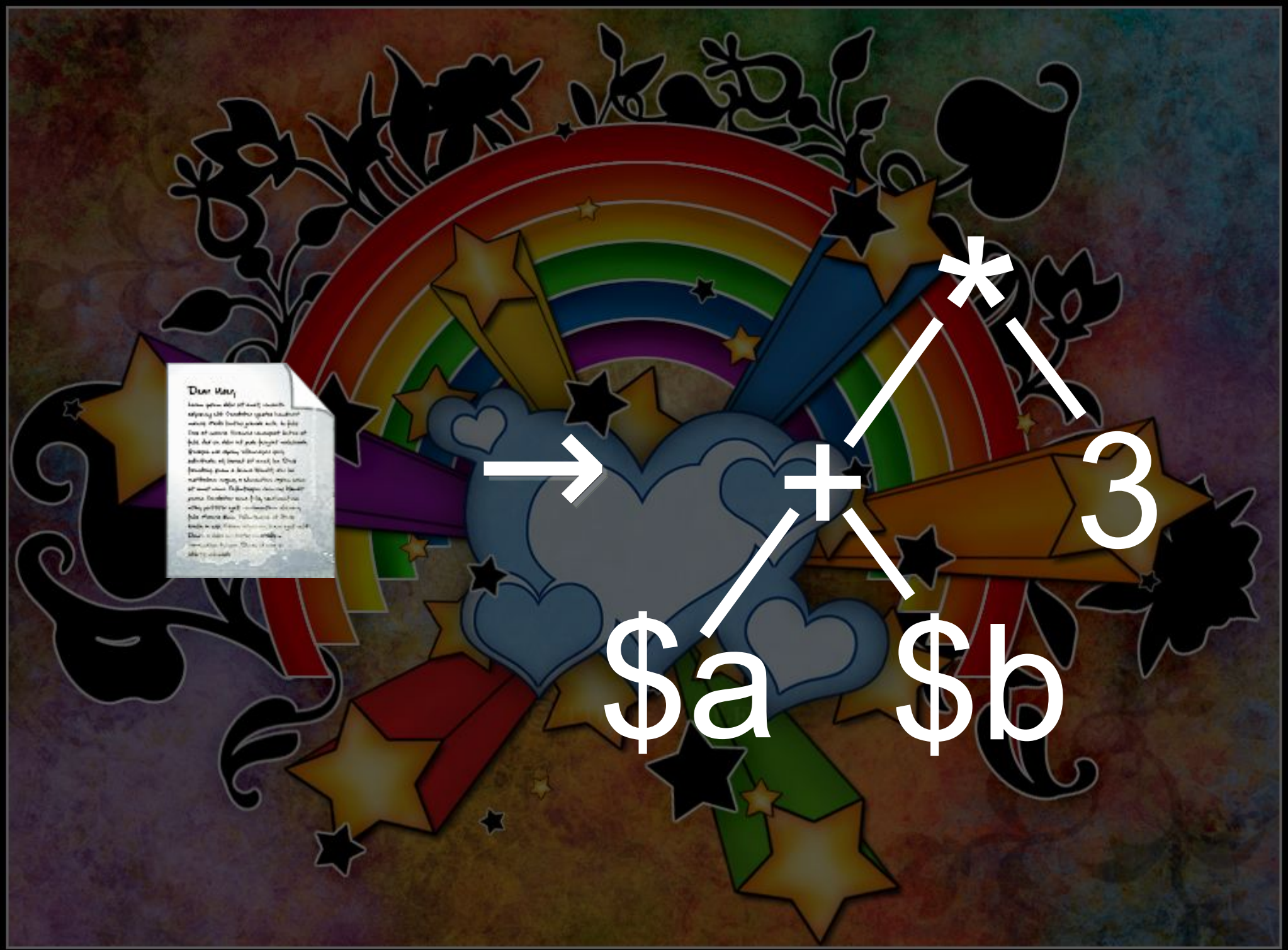

*
+
3
$a
$b
Parrot


*
+
3
$a
$b
*
+
3
$a
$b
Parrot
JVM

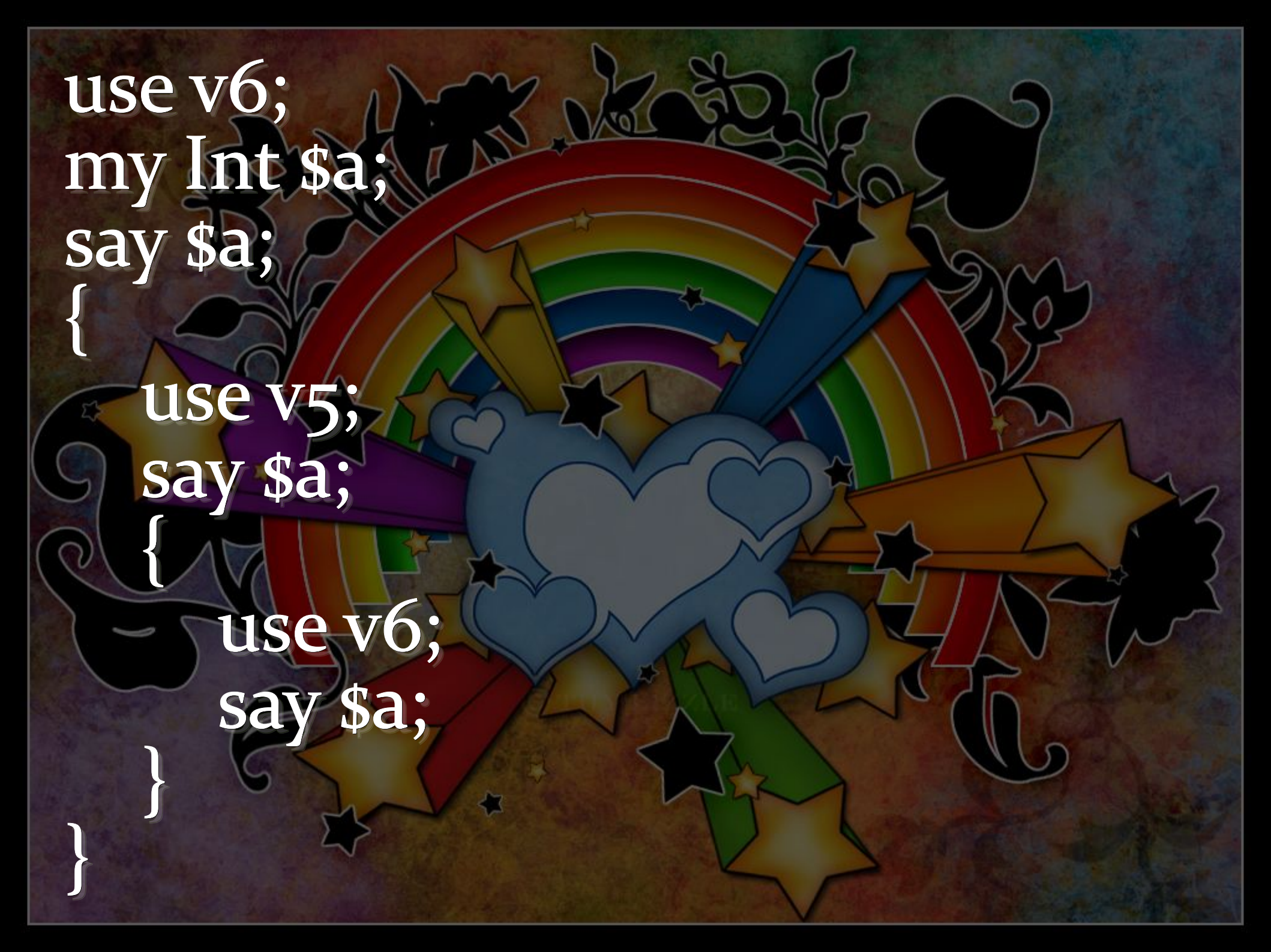

*
+
3
$a
$b
Parrot
*
+
3
$a
$b
JVM
*
+
3
$a
$b
MoarVM

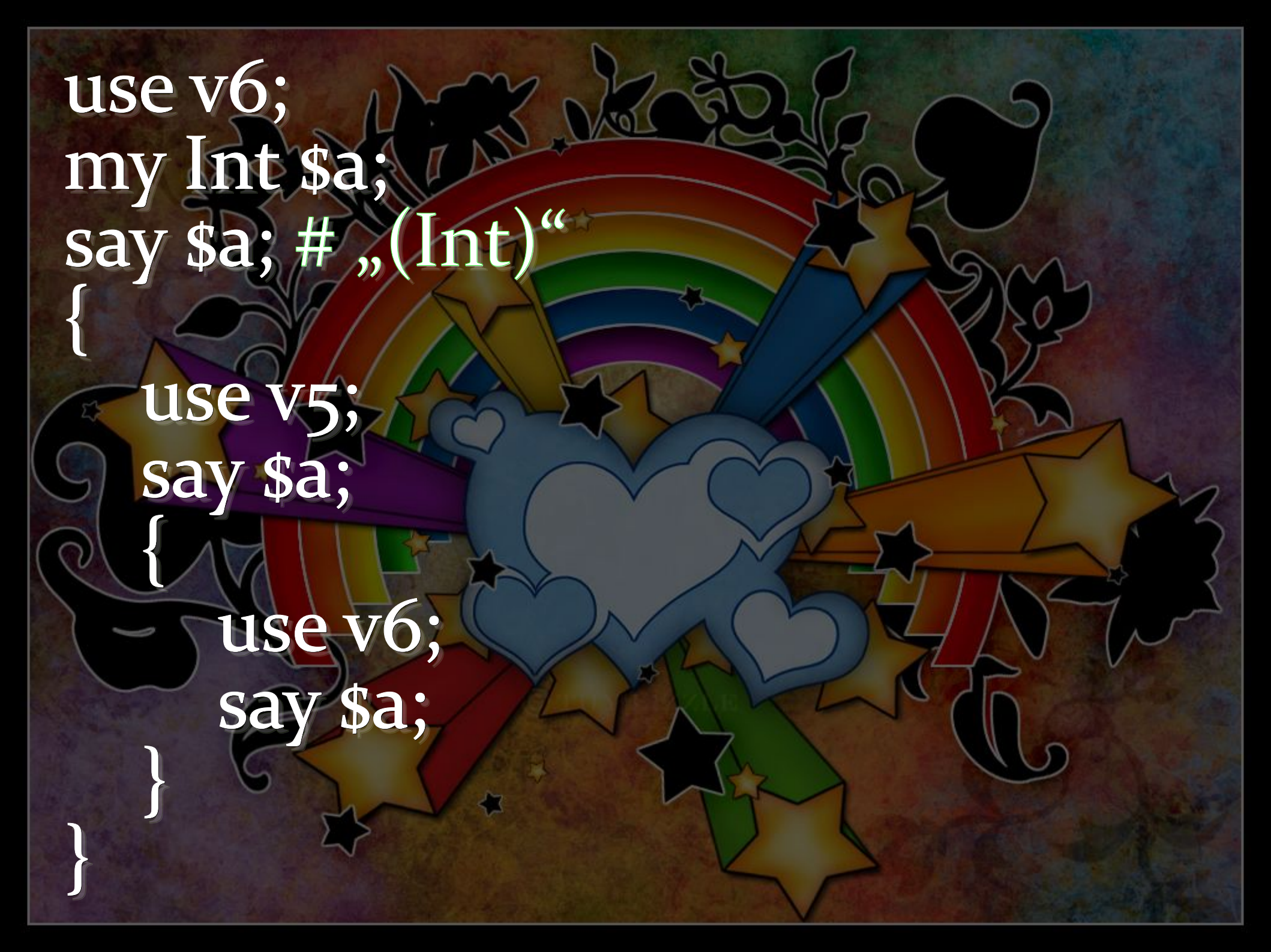

*
+
3
$a
$b
?

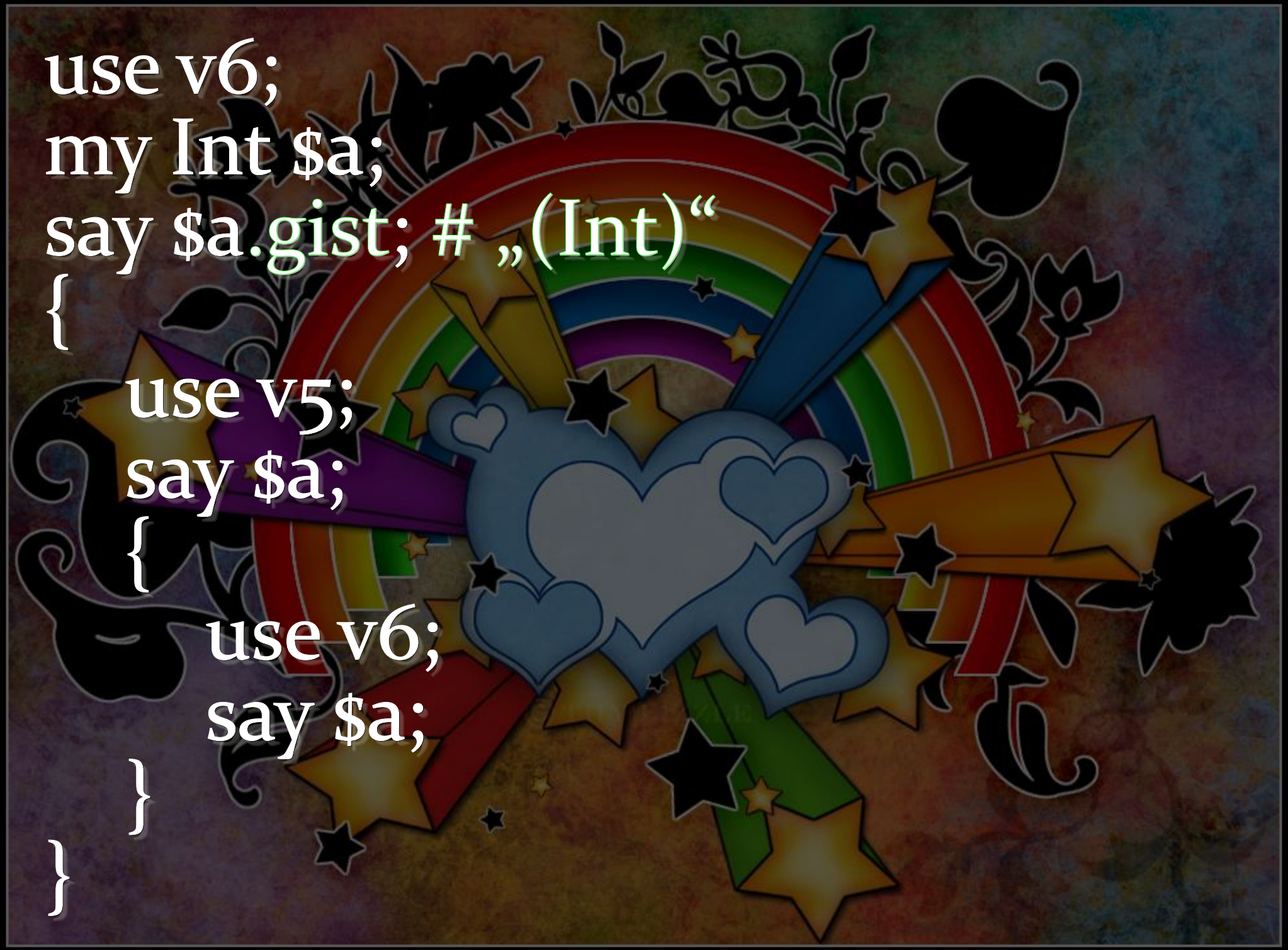

*
+
3
$a
$b
JavaScript

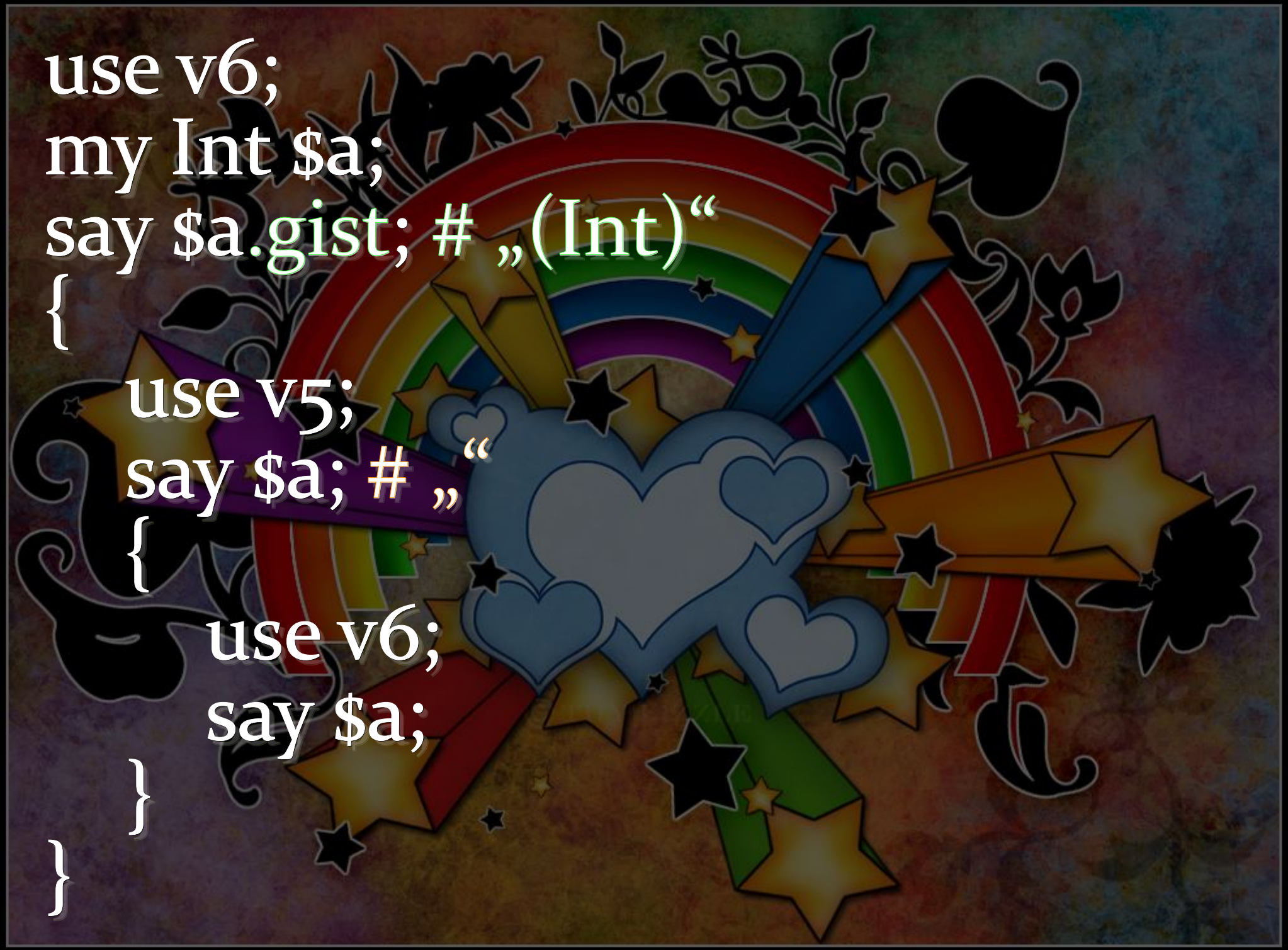

*
+
3
$a
$b

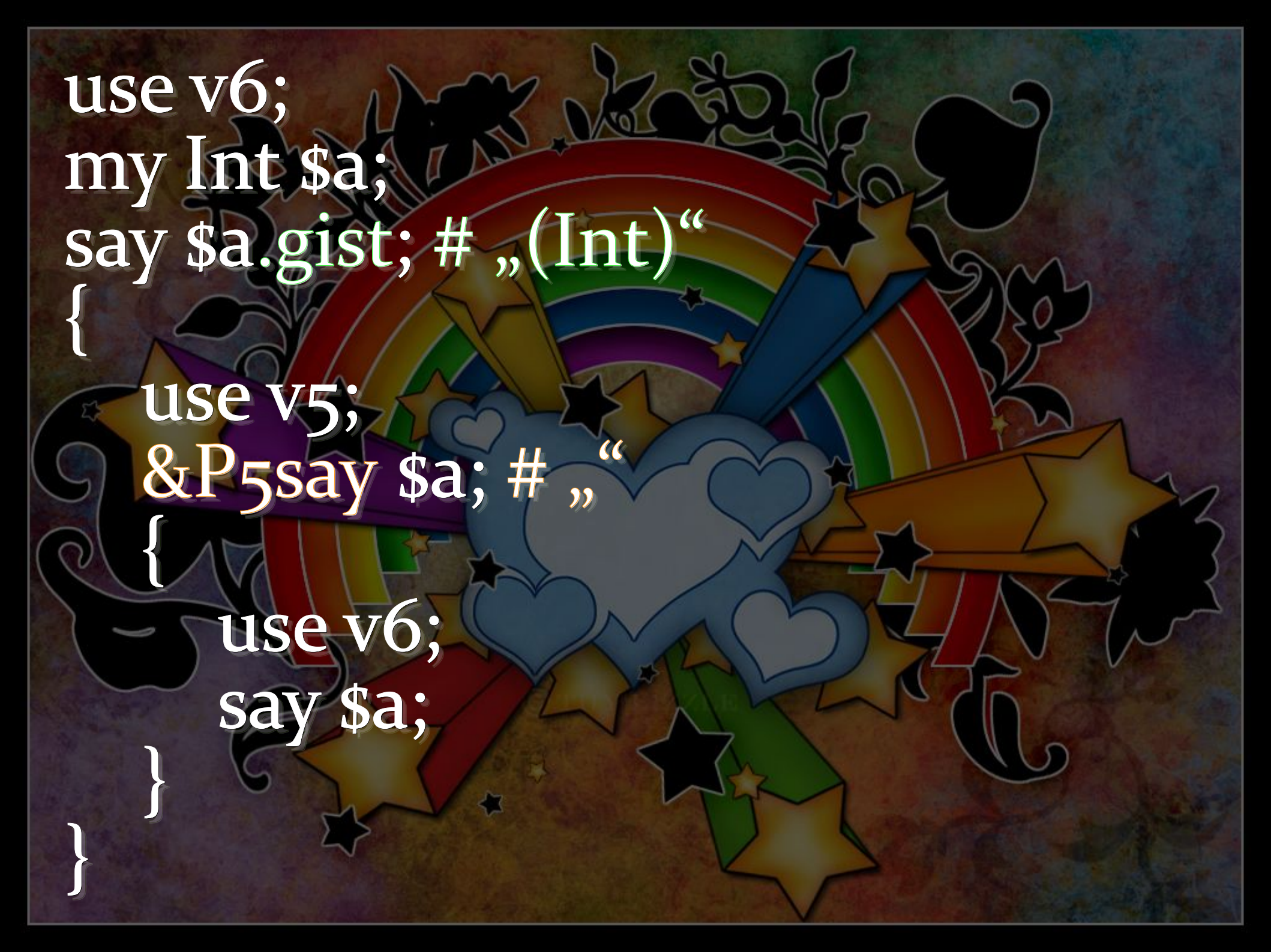

call
&say
&slurp
„hello“ $^O
…

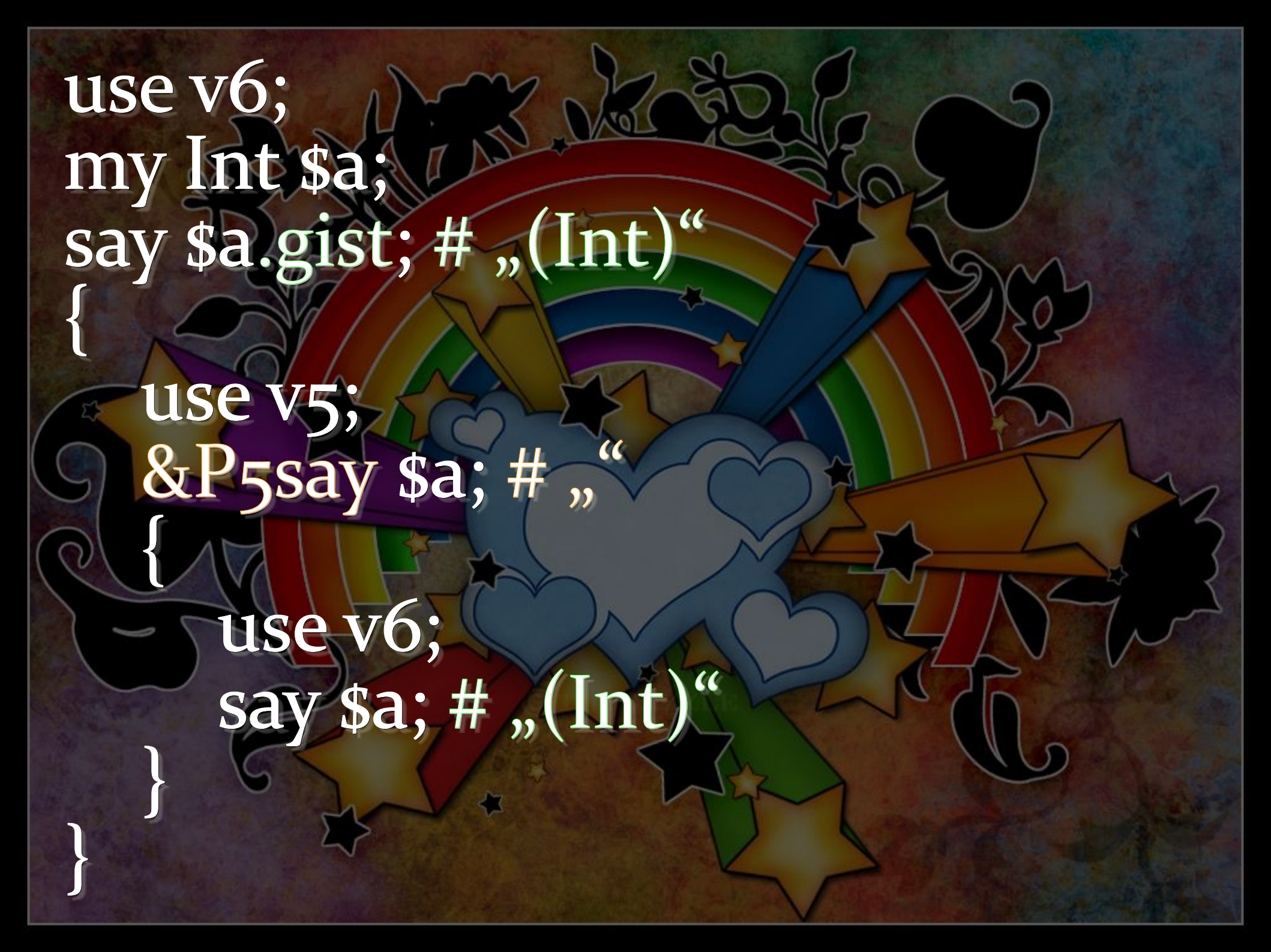

use v6;
my Int $a;
say $a;
{
use v5;
say $a;
{
use v6;
say $a;
}
}


use v6;
my Int $a;
say $a; # „(Int)“
{
use v5;
say $a;
{
use v6;
say $a;
}
}


use v6;
my Int $a;
say $a.gist; # „(Int)“
{
use v5;
say $a;
{
use v6;
say $a;
}
}

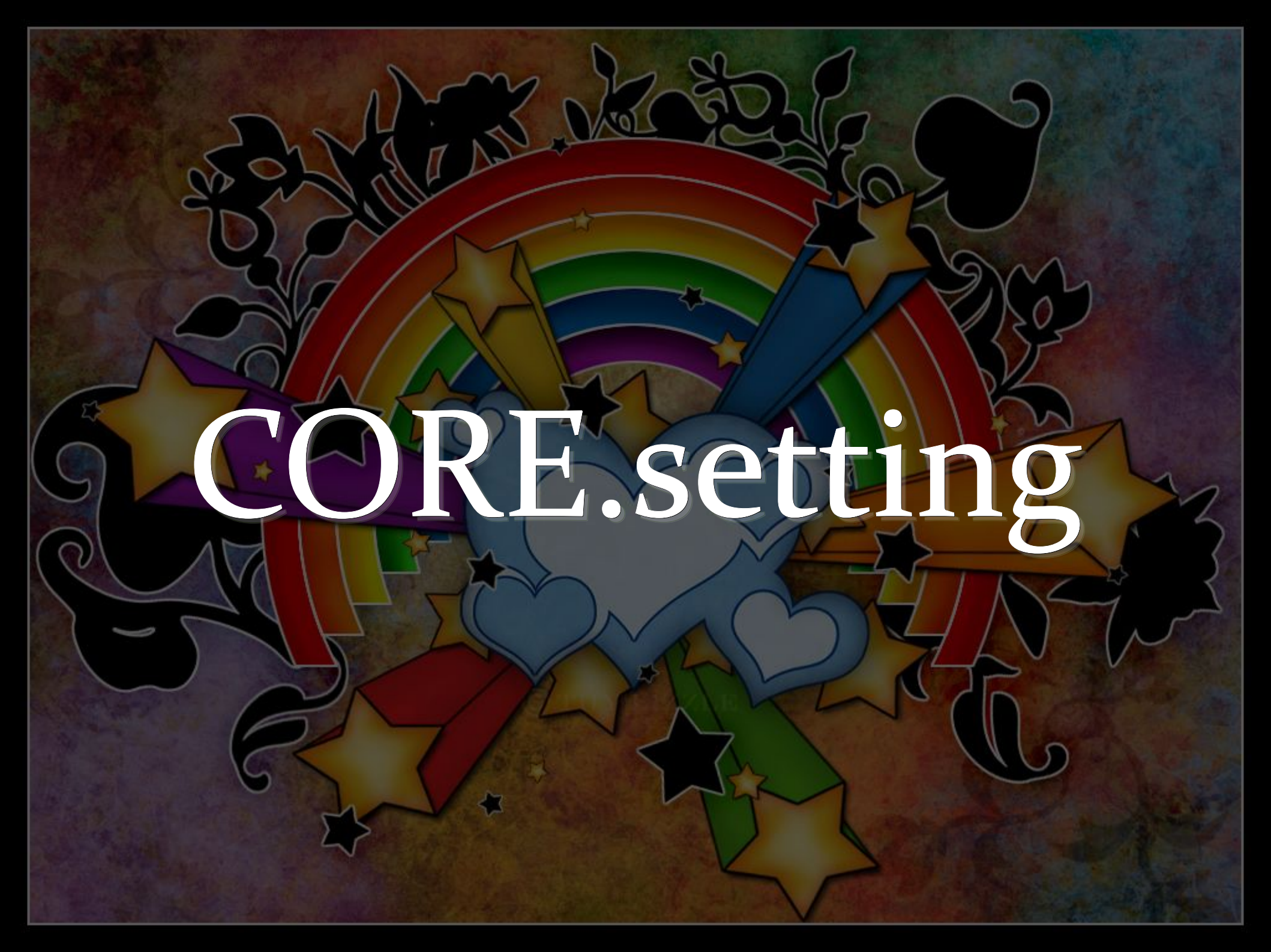

use v6;
my Int $a;
say $a.gist; # „(Int)“
{
use v5;
say $a; # „“
{
use v6;
say $a;
}
}


use v6;
my Int $a;
say $a.gist; # „(Int)“
{
use v5;
&P5say $a; # „“
{
use v6;
say $a;
}
}

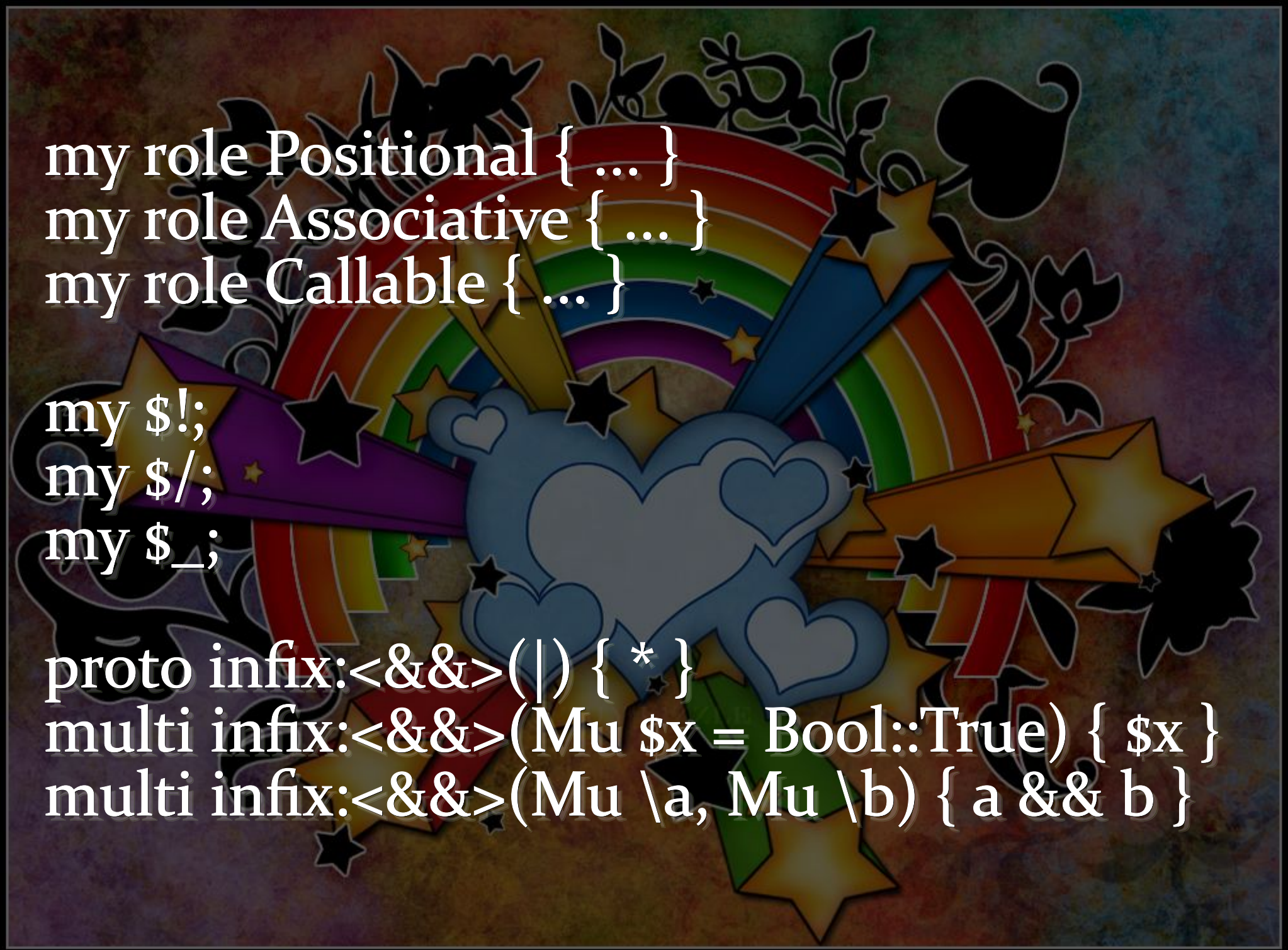

use v6;
my Int $a;
say $a.gist; # „(Int)“
{
use v5;
&P5say $a; # „“
{
use v6;
say $a; # „(Int)“
}
}

```
multi P5Str(Mu:U) {
    P5warn(:cat<uninitialized>,
        'Use of uninitialized value in string');
    ''
}
multi P5Str(Bool:D \s) { s ?? 1 !! '' }
multi P5Str(Int:D \s) { s.Int }
multi P5Str(Num:D \s) { s.Num }
multi P5Str(Parcel:D \s) { s.Int }
multi P5Str(Sub:D \s) {
  'CODE(' ~ s.WHERE.fmt('ox%X').lc ~ ')'
}
```

```
multi P5Str(Mu:U) {
    P5warn(:cat<uninitialized>,
        'Use of uninitialized value in string');
    ''
}
multi P5Str(Bool:D \s) { s ?? 1 !! '' }
multi P5Str(Int:D \s) { s.Int }
multi P5Str(Num:D \s) { s.Num }
multi P5Str(Parcel:D \s) { s.Int }
multi P5Str(Sub:D \s) {
  'CODE(' ~ s.WHERE.fmt('ox%X').lc ~ ')'
}
```


CORE.setting


20799


my role Positional { ... }
my role Associative { ... }
my role Callable { ... }
my $!;
my $/;
my $_;
proto infix:<&&>(|) { * }
multi infix:<&&>(Mu $x = Bool::True) { $x }
multi infix:<&&>(Mu \a, Mu \b) { a && b }


Perl5::Terms


1459


1459..*


conditions


if (EXPR) BLOCK
elsif (EXPR) BLOCK
…
else BLOCK


unless (EXPR) BLOCK
elsif (EXPR) BLOCK
...
else BLOCK


EXPR if EXPR
EXPR unless EXPR


loops

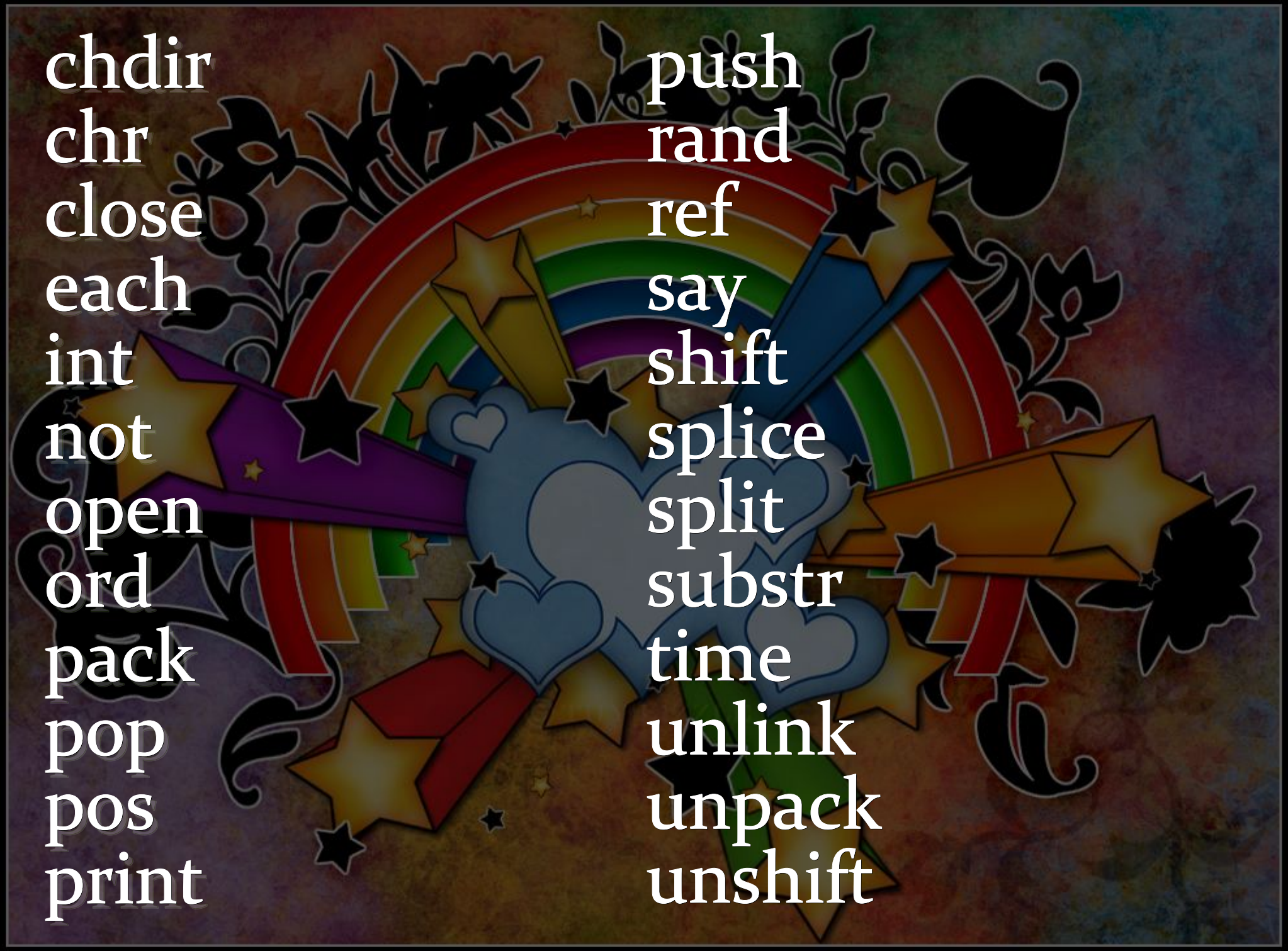

LABEL while (EXPR) BLOCK
continue BLOCK
LABEL until (EXPR) BLOCK
continue BLOCK

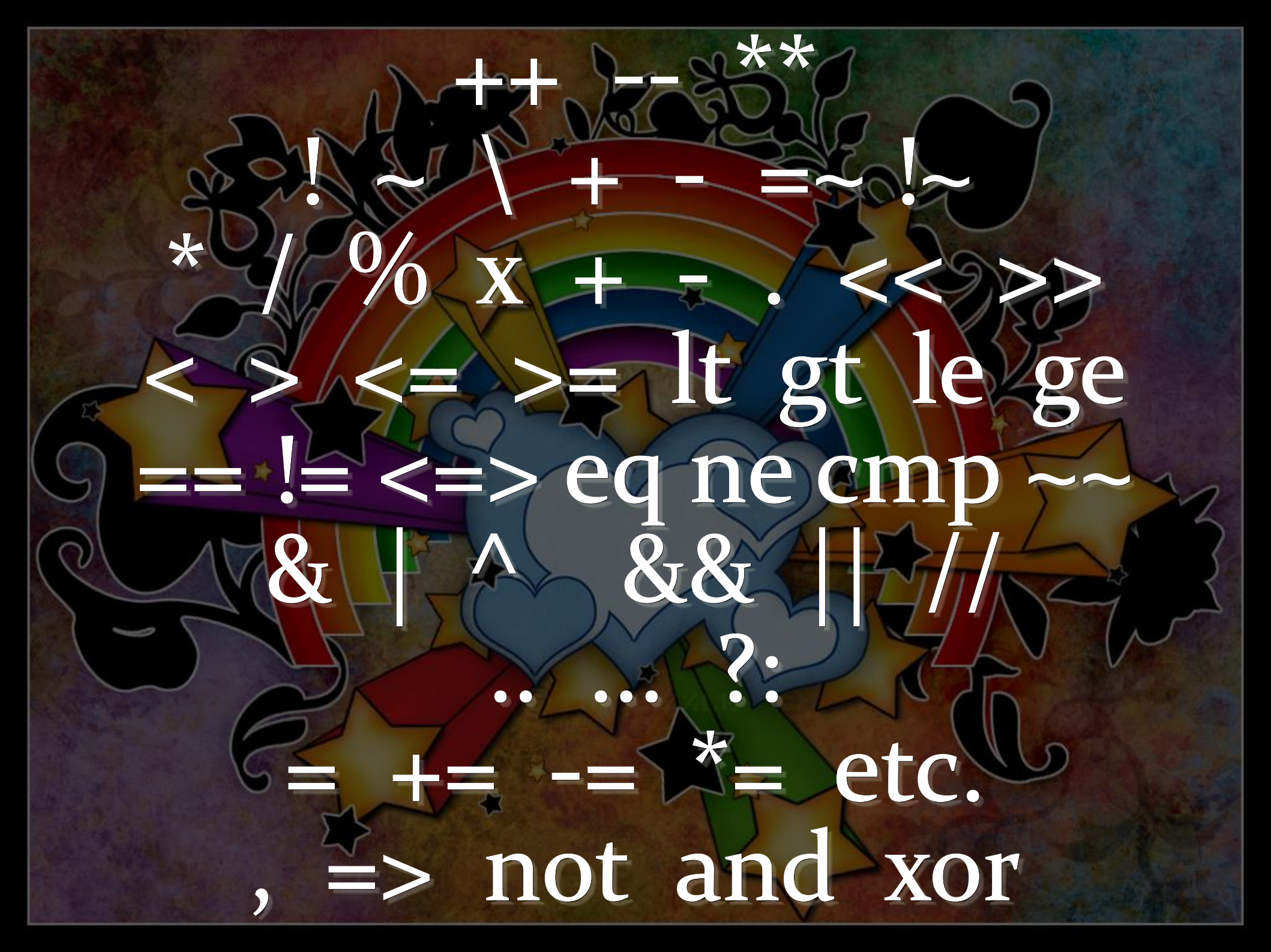

LABEL for (EXPR; EXPR; EXPR) BLOCK
LABEL for VAR (LIST) BLOCK continue BLOCK
LABEL foreach VAR (LIST) BLOCK continue BLOCK


LABEL BLOCK
continue BLOCK

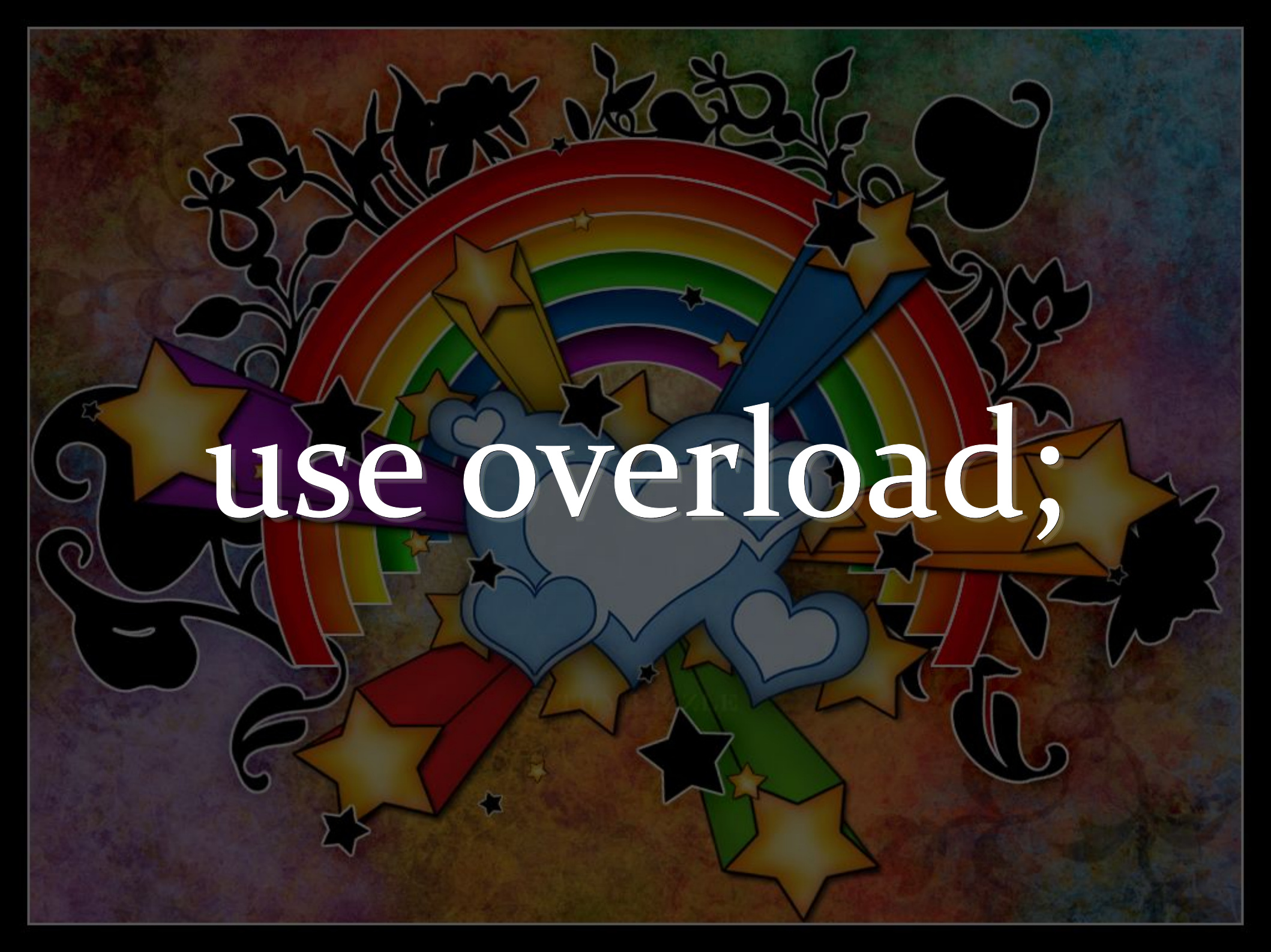

BEGIN BLOCK


use
require


chdir
chr
close
each
int
not
open
ord
pack
pop
pos
print
push
rand
ref
say
shift
splice
split
substr
time
unlink
unpack
unshift


++ -- **
! ~ \ + - =~ !~
* / % x + - . << >>
< > <= >= lt gt le ge
== != <=> eq ne cmp ~~
& | ^ && || //
.. ... ?:
= += -= *= etc.
, => not and xor


use overlord!


use overload;


use overload
'"" => sub {'('.ref(shift).')'}


sub foo($) { }

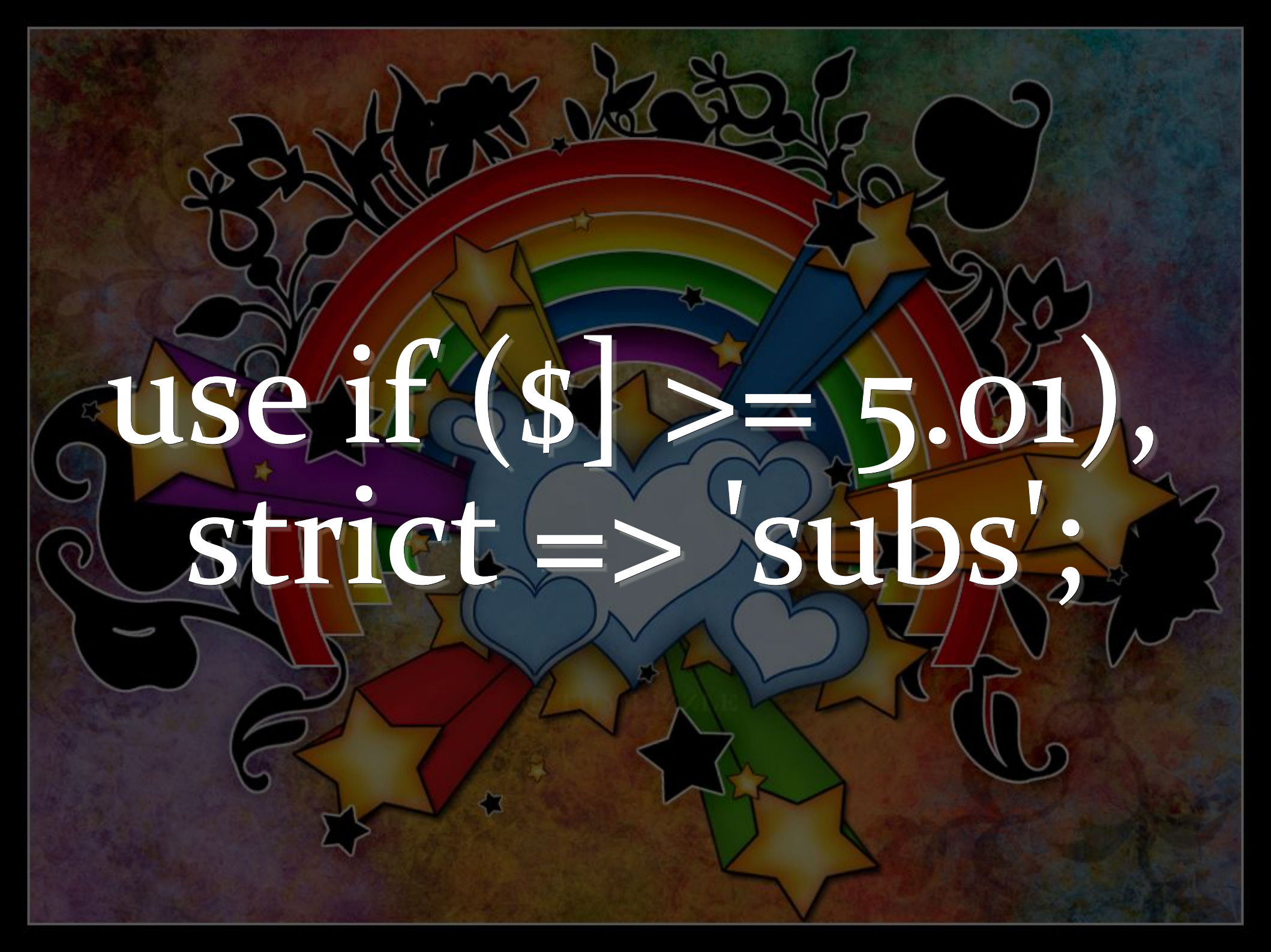

my $bar;
sub foo :lvalue { $bar }


use vars qw(%foo);

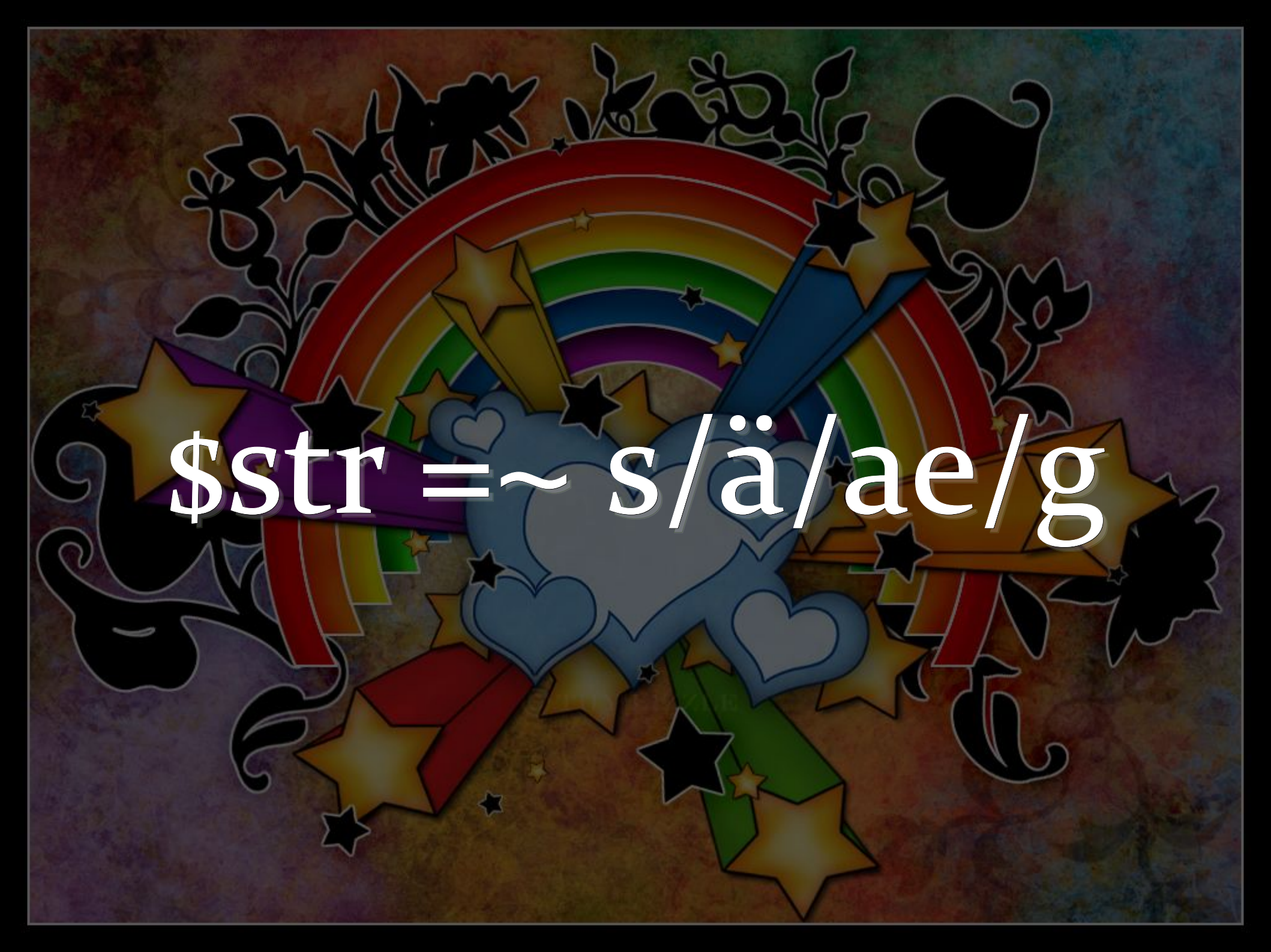

use warnings
qw(uninitialized);


use strict;


use Config;
use English;


use if ($] >= 5.01),
strict => 'subs';


sub bar {
goto &foo;
}


$str =~ s/ä/ae/g

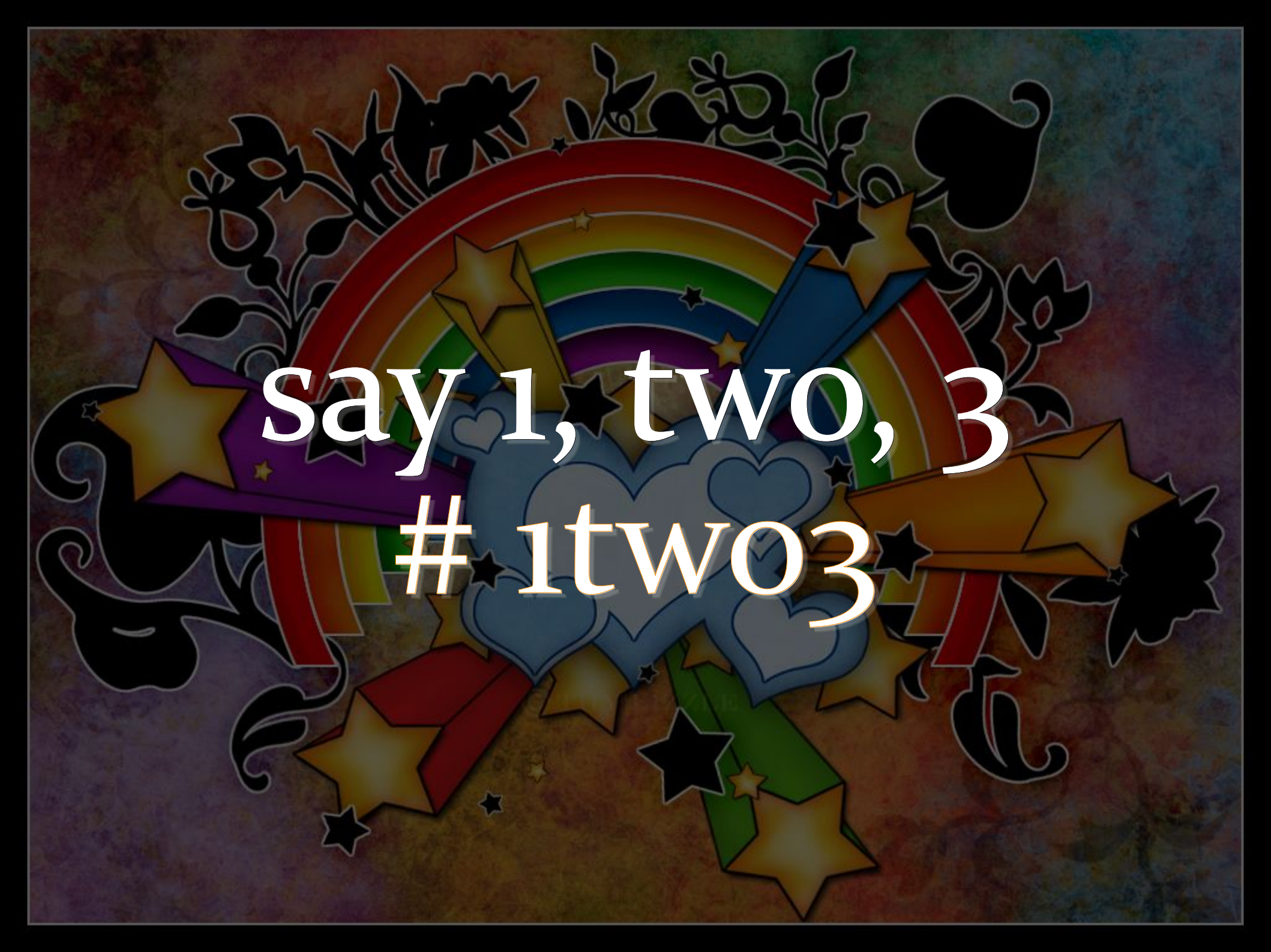

open FH '>', 'foo'

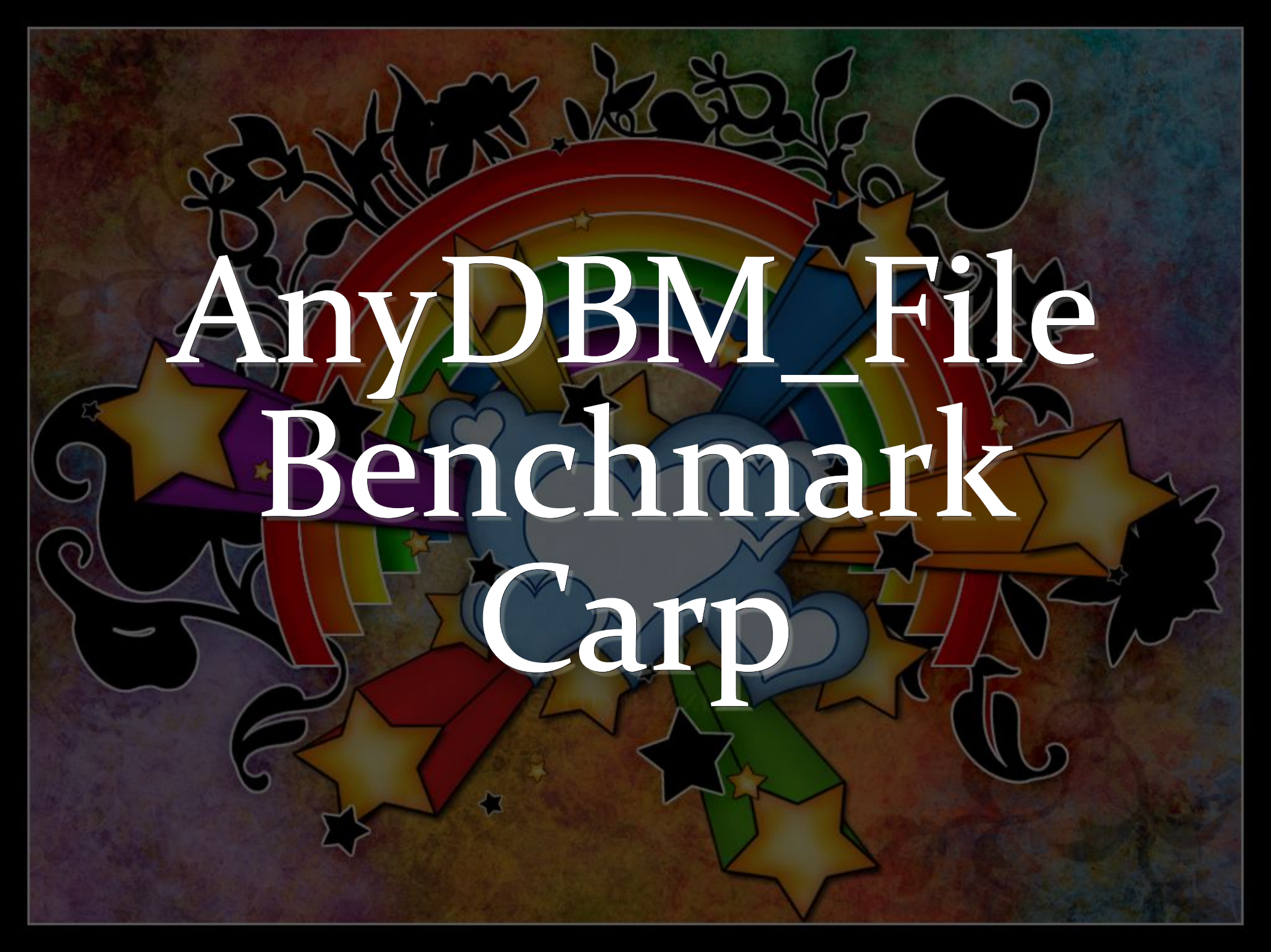

new SDL::Rect


:o(


next LABEL
last LABEL
redo LABEL


goto LABEL


say 1, two, 3
#1two3


AnyDBM_File
Benchmark
Carp


never?


XS


tests


8093


42906


>92000


FROGGS
usev5.wordpress.com
freenode/#perl6
FROGGS@cpan.org
github.com/FROGGS