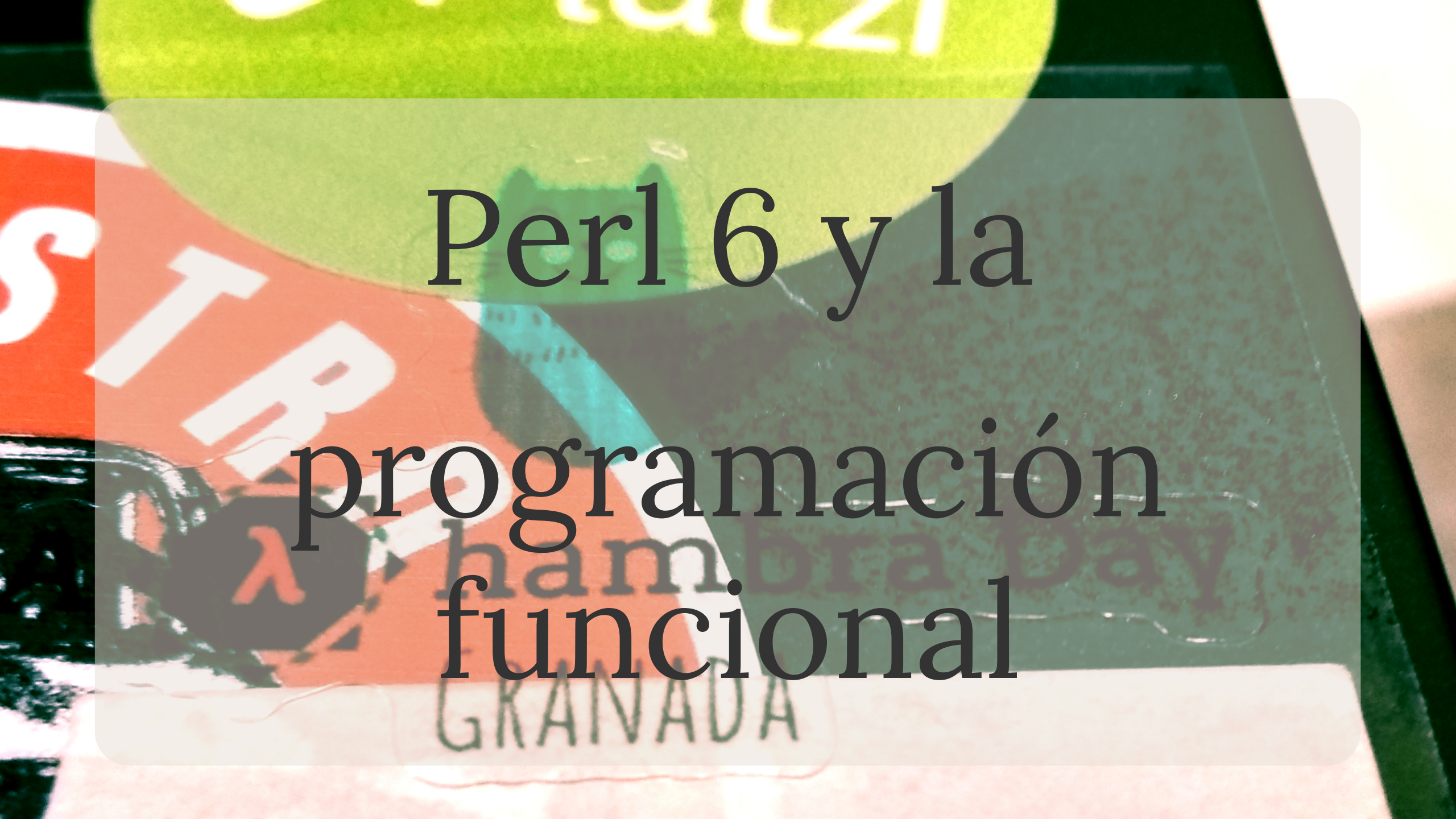


Perl 6 y la programación funcional


perl6.org

Perl 6 empezó en Haskell

Las mónadas me atrajeron a Perl 6

Procesando listas

```
(1..10).map( *² ).say;
```

```
(1..10).map( *² ).grep( * ≥ 12 ).say;
```

```
1..10 ==> map( *² ) ==> grep( * ≥ 12 ) ==> my @output;
@output.join( " ⬖ " ).put;
```

TIMTOWTDI

```
#!/usr/bin/env perl6
use v6;
((1..10) »**» 2 ).duckmap: → $x where * ≥ 12 { $x.say };
=output
16
25
....
```

Tipos

```
sub quita-no-mayúsculas( Str $cadena --> Seq ) {
    return $cadena.comb.grep: * ∉ 'A'..'Z';
}
say quita-no-mayúsculas( "LaTeX" );
say quita-no-mayúsculas( "Learn You a Perl 6" );
say &quita-no-mayúsculas.^name # Sub+{Callable[Seq]}
```

Typeclasses

```
sub þor( Int, Int --> Str ) {};
say &þor.^name; # Sub+{Callable[Str]}
say &þor.^mro;
#((Sub+{Callable[Str]}) (Sub) (Routine) (Block) (Code) (Any)
```

```
role CasiEq {
    method  casi-igual( \rhs --> Bool) {...}
    method  casi-diferente( \rhs --> Bool) {...}
}
class CasiEqInt does CasiEq {
    has Int $.n;

    method casi-igual ( \rhs --> Bool) { return True if abs($!
    method casi-diferente ( \rhs --> Bool) { True unless abs($
    method Numeric( --> Numeric:D ) { return $!n }
}
my $n1 := CasiEqInt.new( n => 1 );
my $n2 := CasiEqInt.new( n => 2 );
say "Casi " if $n1.casi-igual($n2);
```

O con protos/multi

```
proto sub infix:<≈> ( | --> Bool) {*}
```

```
multi sub infix:<≈> ( Int $lhs, Int $rhs --> Bool) {
        return True if abs( $lhs - $rhs) ≤ 1
}
```

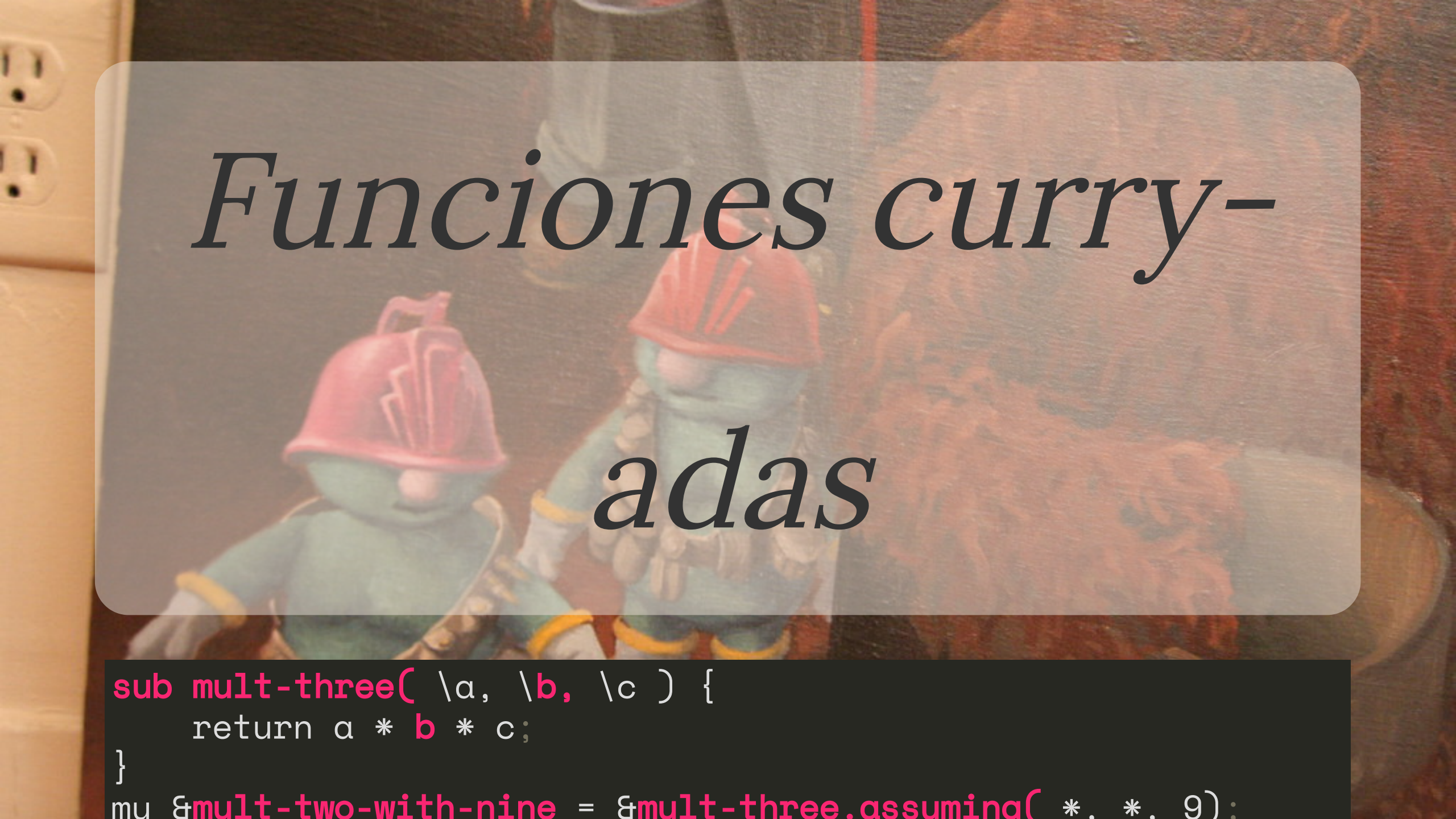
```
say "Casi " if 3 ≈ 2 ;
```

Emparejando

patrones

```
proto sub lucky (Int \a --> Str ) { *}
multi sub lucky ( 7 ) {  "LUCKY NUMBER SEVEN!" }
multi sub lucky ( $x where * ≠ 7 ) { "Sorry, you're out of luc

say lucky 7 ;
say lucky 33;
```



Funciones curry-adas

```
sub mult-three( \a, \b, \c ) {
    return a * b * c;
}
my &mult-two-with-nine = &mult-three.assuming( *, *, 9);
```

Funciones que devuelven funciones

```
sub apply-twice( &f, \a ) { f( f ( a ) ) }
say apply-twice( * + 3, 10 );
say apply-twice( * ~ " HAHA" , "HEY" );
say apply-twice( "HAHA " ~ *, "HEY" );
```

Lambda

```
say ((1..5) Z (5...1)).map( → (\a, \b) { (a*30+3)/b });
```

Composición de funciones

```
say map( { ( { - $_ } ∘ &abs)($_) },  [5,-3,-6,7,-3,2,-19,24])
```

Un functor empieza así

```
class Just {
    has &.a;
    method new( $a ) {
        return self.bless( a => {$a} );
    }

    method CALL-ME( |c ) {
        return &!a();
    }
}
```

Y continúa

```
multi sub maybe( &f, Nil ) { return Nil };
multi sub maybe( &f, Just $x ) { return Just.new( f( $x() ) ) }

my $treinta-y-tres = Just.new( -33 );

say maybe &abs, Nil;
say (maybe &abs,  $treinta-y-tres )();
```

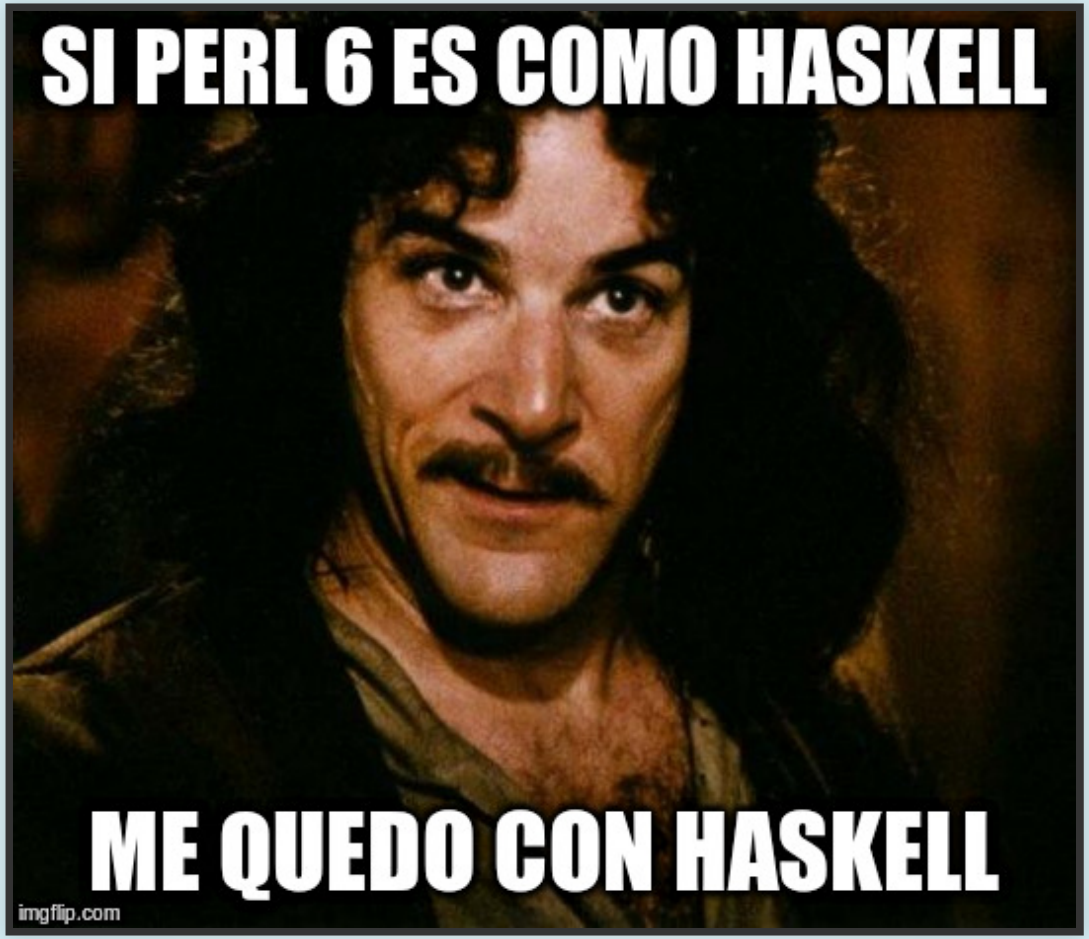
Functoreando

```
# fmap (++ " HEY GUYS IM INSIDE THE JUST") (Just "Something se
multi sub fmap( &f, Nil ) { return Nil };
multi sub fmap( &f, Just $x ) { return Just.new( f( $x() ) ) }
say fmap * ~ " HEY GUYS IM INSIDE THE JUST", Just.new: "Someth
```

Mónadas

```
{
  say "Write a couple of lines here →";
  my $line_from_user = getLine();
  my $echo = mbind($line_from_user, → $x { putStrLn($x) });
  my @actions = (getLine(), $echo);
  my $both = sequence_(@actions);
  $both();
}
```


[examples] first whack at modernization to STD standards 10 years ago
Added a small "Monad
[examples] first whack at modernization to STD standards
git-svn-id: http://svn.pugscode.org/pugs@27041 c213334d-75ef-0310-aa23-eaa082d1ae64
+340 −339
lwall committed on 9 Jun 2009
Upgraded it to curre
Added a small "Mo ad
Círculo completo
Upgraded it to current version 2 years ago
Added a small "Monads in Perl 6" example. 14 years ago
Upgraded it to current version 2 years ago
Added a small "Monads in Perl 6" example. 14 years ago
return {
my $line = $*IN.get;
$line;
};
}
# putStrLn :: String -> IO ()
# putStrLn = ...implemented by the compiler.
sub putStrLn(Str $x) { return { say $x; Nil
# Now some example code
# This is the one that gets run.
{
say "Write a couple of lines here ->";
my $line_from_user = getLine();
# Nothing is read yet.
my $echo = mbind($line_from_user, -> $x {
# Nothing is read or printed yet.


SI PERL 6 ES COMO HASKELL
ME QUEDO CON HASKELL
imgflip.com


No tendrás concurrencia basada en canales
... ni gramáticas

Y Camelia llorará

Muchas gracias

jj.github.io/fp-per16/eslibre.html

Código en git.io/p6lambda

Créditos

- Foto de Audrey Tang de Yoshinori Takesako
- Curris de Jeff Christiansen