

# Perl 5, Version 13

Updated for v5.13.7

David Golden  
dagolden@cpan.org  
[www.dagolden.com](http://www.dagolden.com)

# Perl 5, Version 13 development

- 5.13.0 - Apr 20, 2010 - Leon Brocard
- 5.13.1 - May 20, 2010 - Ricardo Signes
- 5.13.2 - Jun 22, 2010 - Matt S. Trout
- 5.13.3 - Jul 20, 2010 - David Golden
- 5.13.4 - Aug 20, 2010 - Florian Ragwitz
- 5.13.5 - Sep 19, 2010 - Steve Hay
- 5.13.6 - Oct 20, 2010 - Tatsuhiko Miyagawa
- 5.13.7 - Nov 20, 2010 - Chris Williams

# Hardest part of releasing Perl 5?

- Writing the perldelta.
- Seriously. Ask anyone who has written one.
- Pumpkings have resigned rather than write the perldelta!\*

\* I'm kidding.

# This talk is a perl`delta`

- A summary of the summaries
- Stuff you might actually use
- Internal stuff you shouldn't ignore

# Perl 5, Version 13, Subversion 0

- Faster `shift` without arguments
  - `my $self = shift; my $env = shift;`
  - 5% faster over `shift(@_)` on non-threaded perl
  - 25% faster on threaded
- On linux, assigning to `$0` sets process name for *ps*, *top*, etc.
  - `$0 = "my-app-foo";`

# Perl 5, Version 13, Subversion 1

- `given` returns a value

```
# old way in 5.12
```

```
my $type;
```

```
given ($string) {
```

```
    $type = undef      when undef;
```

```
    $type = 'digits'  when /^ \d+$/;
```

```
    $type = 'word'    when /^ \w+$/;
```

```
    $type = 'unknown';
```

```
};
```

# Perl 5, Version 13, Subversion 1

- `given` returns a value (but must use `do`)

```
# new way in 5.13.1
```

```
my $type = do {  
    given ($string) {  
        break      when undef;  
        'digits'   when /\d+$/;  
        'word'     when /\w+$/;  
        'unknown' ;  
    }  
};
```

# Perl 5, Version 13, Subversion 1

- Exceptions are less insane
  - When an exception is thrown in an `eval` block, `$_` will not be clobbered exiting the eval
  - Exceptions in DESTROY *always* warn, instead of only sometimes warning
  - Also, `warn()` takes objects just like `die()`
- Removed from core (available on CPAN)
  - `Class::ISA`, `Pod::Plainer`, `Switch`

# Perl 5, Version 13, Subversion 2

- Non-destructive substitution  
(*think 'r' for 'return'*)

```
($new = $old) =~ s/cat/dog/; # 5.12
```

```
$new = $old =~ s/cat/dog/r; # 5.13.2
```

```
@d = map { ($a=$_)=~s/::/-/g; $a } @m # 5.12
```

```
@d = map { s/::/-/gr } @m # 5.13.2
```

# Perl 5, Version 13, Subversion 2

- `package` block syntax

```
{ package Foo; ... }           # 5.12  
package Foo { ... }         # 5.13.2
```

```
{ package Foo 1.23; ... }      # 5.12  
package Foo 1.23 { ... }    # 5.13.2
```

# Perl 5, Version 13, Subversion 3

- New octal escape

- `\o{}` similar to `\x{}` and `\N{}`

```
"\033"          # ESC (the old way)
```

```
"\o{33}"        # ESC (new for 5.13.3)
```

```
"\o{23072}"     # SMILEY (new for 5.13.3)
```

- Old octal escapes were insane

- `"\777" ne "\0777"`

- `"\023072" eq "\023" . "072"`

- `qq{$stuff \10}` # octal

- `qr{$stuff \10}` # octal OR backref

# Perl 5, Version 13, Subversion 3

- `\N{NAME}` understands abbreviations  
"`\N{NBSP}`" # `\N{NO-BREAK SPACE}`
- Better documentation of **all** characters escapes
- Dual-life modules synchronized with CPAN

# Perl 5, Version 13, Subversion 4

- Iterative string appending on Win32 about 100x faster

```
$str .= stuff() for 1..1000000
```

- functions using prototypes `(*)`, `(;$)` or `(;*)` are parsed with higher precedence

```
sub foo(;$) {...}
```

```
foo $a < $b; # parsed as foo($a)
```

# Perl 5, Version 13, Subversion 4

- **srand( )** returns the seed so you can save it and call it again as **srand(\$seed)**
- **\N{NAME}** and **chardnames** now know about every Unicode character
- Fixed "Bizarre copy of ARRAY" warnings from Carp for modules overriding caller()

# Perl 5, Version 13, Subversion 5

- Deprecated qw() used as parentheses

```
for $x qw(a b c) { ... } # wrong
```

```
for $x (qw(a b c)) { ... } # right
```

- Fixes bug in smart-matching array slices

```
@foo{@what} ~~ @bar
```

# Perl 5, Version 13, Subversion 6

- New "u", "l", and "d" regex modifiers in (?...)  
`qr/(?u)\w/ # \w is Unicode`  
`qr/(?l)\w/ # \w is locale specific`  
`qr/(?d)\w/ # \w depends on matched string`
- The 'unicode\_strings' feature turns on (?u) semantics for a lexical scope  
`use feature 'unicode_strings';`
- Regular expressions retain their locale or unicode semantics when interpolated later

# Perl 5, Version 13, Subversion 6

- Stringification of regular expressions has changed to simplify handling of new modifiers

```
say qr/foo/i # was (?i-xs:foo)
```

```
say qr/foo/i # now (?^i:foo)
```

```
use feature 'unicode_strings';
```

```
say qr/\w/ # (?^u:\w)
```

# Perl 5, Version 13, Subversion 7

- Array and hash container functions (`push`, `pop`, `keys`, ...) work on references, too

```
push @$arrayref, @list;  
push $arrayref, @list;
```

```
push @{$obj->tags}, $new_tag;  
push $obj->tags, $new_tag;
```

```
for ( keys %{$hoh->{p}{q}} ) { ... }  
for ( keys $hoh->{p}{q} ) { ... }
```

# Perl 5, Version 13, Subversion 7

- New prototype '+' acts like '\$' but passes literal hashes or arrays by reference

```
sub mypush (+@) {  
    my $array = shift;  
    die "Not an array or arrayref"  
        unless ref $array eq 'ARRAY';  
    push @$array, @_;  
}
```

```
mypush @array, @stuff;
```

# Perl 5, Version 13, Subversion 7

- Full Unicode 6.0 support, except `\N{BELL}`, which will stay `u+0007 (BEL)` until Perl v5.16
- Set regular expression flags in a lexical scope (including "u", "l" or "d")

```
use re '/u';  
$str = "r\x{e9}sum\x{e9}" # résumé  
$chars = () = $str =~ /\w/g; # 6
```

# Perl 5, Version 13, Subversion 7

- Non-destructive transliteration, just like `s///r` from v5.13.2

```
(my $new = $old) =~ tr/a-zA-Z/ /cs;
```

```
my $new = $old =~ tr/a-zA-Z/ /csr;
```

# Perl 5, Version 13 future

- "Contentious changes freeze" - 20 Dec 2010
- "User-visible changes freeze" - 20 Feb 2011
- "Hard freeze" - 30 Mar 2011
- "5.14 target release date" - 20 Apr 2011