Modern Core Perl

Dave Cross
Magnum Solutions Ltd
dave@mag-sol.com

What We Will Cover

- Recent Perl releases
- 5.10
- 5.12
- 5.14
- 5.16

Perl Releases

- Perl 5 has moved to a regular release cycle
- Major release every year
  - In Spring
- Minor releases when required

Perl Version Numbers

- Even major numbers are production releases
  - 5.10, 5.12, 5.14
- Odd major numbers are dev releases
  - 5.9, 5.11, 5.13

Perl Support

- p5p provide support for current and previous major releases
  - Currently 5.12 and 5.14
- Further support may be available from distributors

Recent Perl Releases

- 5.10.0 – 2007 Dec 18
- 5.10.1 – 2009 Aug 22
- 5.12.0 – 2010 Apr 12
- 5.12.1 – 2010 May 16
- 5.12.2 – 2010 Sep 6
- 5.12.3 – 2011 Jan 21

Recent Perl Releases

- 5.14.0 – 2011 May 14
- 5.14.1 – 2011 Jun 16
- 5.12.4 – 2011 Jun 20
- 5.14.2 – 2011 Sep 26

Perl 5.10

- Released 18th Dec 2007
  - Perl's 20th birthday
- Many new features
- Well worth upgrading

New Features

- Defined-or operator
- Switch operator
- Smart matching
- say()
- Lexical $_

New Features

- State variables
- Stacked file tests
- Regex improvements
- Many more

Defined Or

- Boolean expressions “short-circuit”
- `$val = $val || $default;`
- `$val ||= $default;`
- What if 0 is a valid value?

Defined Or

- Need to check “definedness”
- ```
  $val = defined $val
              ? $val : $default;
  ```
- ```
  $val = $default
     unless defined $val;
  ```


Defined Or

- The defined or operator makes this easier
- `$val = $val // $default;`
- A different slant on truth
- Checks definedness
- Shortcircuit version too
- `$val //= $value;`

Switch Statement

- Switch.pm was added with Perl 5.8
- Source filter
- Parser limitations
  - Regular expressions
  - eval
- 5.10 introduces a build-in switch statement

Given ... When

- Switch is spelled “given”
- Case is spelled “when”
- Powerful matching syntax

Given Example

- ```
  given ($foo) {
      when (/^abc/) { $abc = 1; }
      when (/^def/) { $def = 1; }
      when (/^xyz/) { $xyz = 1; }
      default { $nothing = 1; }
  }
  ```



  ```

New Keywords

- Four new keywords
  - given
  - when
  - default
  - continue

given

- `given(EXPR)`
- Assigns the result of EXPR to $_ within the following block
- Similar to `do { my $_ = EXPR; ... }`

when

- `when (EXPR)`
- Uses smart matching to compare $_ with EXPR
- Equivalent to `when ($_ ~~ EXPR)`
- ~~ is the new smart match operator
- Compares two values and “does the right thing”

default

- default defines a block that is executed if no when blocks match
- default block is optional

continue

- continue keyword falls through to the next when block
- Normal behaviour is to break out of given block once the first when condition is matched
- Inverse of most other programming languages

continue

- ```
  given($foo) {
    when (/x/)
      { say '$foo contains an x';
        continue }
    when (/y/)
      { say '$foo contains a y' }
    default
      { say '$foo contains no x or y' }
  }
  ```



  ```

Smart Matching

- ~~ is the new Smart Match operator
- Different kinds of matches
- Dependent on the types of the operands
- See “perldoc perlsyn” for the full details
- Warning: Still under discussion

Smart Match Examples

- `$foo ~~ $bar; # == or cmp`
- `@foo ~~ $bar; # array contains value`
- `%foo ~~ $bar; # hash key exists`
- `$foo ~~ qr{$bar}; # regex match`
- `@foo ~~ @bar; # arrays are identical`
- `%foo ~~ %bar; # hash keys match`
- Many more alternatives

say()

- say() is a new alternative to print()
- Adds a new line at the end of each call
- `say($foo); # print $foo, “\n”;`
- Two characters shorter than print
- Less typing

Lexical $_

- $_ is a package variable
- Always exists in main package
- Can lead to subtle bugs when not localised correctly
- Can now use `my $_` to create a lexically scoped variable called $_

State Variables

- Lexical variables disappear when their scope is destroyed
- ```
  sub variables {
     my $x;

     say ++$x;
  }

  variables() for 1 .. 3;
  ```



  ```

State Variables

- State variables retain their value when their scope is destroyed
- ```
  sub variables {
      state $x;

      say ++$x;
  }

  variables() for 1 .. 3;
  ```



  ```

State Variables

- Like static variables in C
- Deprecating bugs
  - `my $x if 0;`

Stacked File Tests

- People often think you can do this
- `-f -w -x $file`
- Previously you couldn't
- Now you can
- Equivalent to
- `-x $file && -w _ && -f _`

Regex Improvements

- Plenty of regular expression improvements
- Named capture buffers
- Possessive quantifiers
- Relative backreferences
- New escape sequences
- Many more

Named Capture Buffers

- Variables $1, $2, etc change if the regex is altered
- Named captures retain their names
- `(?<name> ... )` to define
- Use new %+ hash to access them

Named Capture Example

```
while (<DATA>) {
    if (/(?<header>[\w\s]+)
         :\s+(?<value>.+)/x) {
      print "$+{header} -> ";
      print "$+{value}\n";
    }
}
```


Possessive Quantifiers

- `?+, *+, ++`
- Grab as much as they can
- Never give it back
- Finer control over backtracking
- `'aaaa' =~ /a++a/`
- Never matches

Relative Backreferences

- `\g{N}`
- More powerful version of `\1`, `\2`, etc
- `\g{1}` is the same as `\1`
- `\g{-1}` is the last capture buffer
- `\g{-2}` is the one before that

New Escape Sequences

- \h – Horizontal white space
- \v – Vertical white space
- Also \H and \V

Accessing New Features

- Some new features would break backwards compatibility
- They are therefore turned off by default
- Various ways to turn them on

Feature Pragma

- Turn new features on with the `feature` pragma
- `use feature 'say';`
- `use feature 'switch';`
- `use feature 'state';`
- `use feature ':5.10';`

Implicit Loading

- Two ways to automatically turn on 5.10 features
- Require a high enough version of Perl
- `use 5.10.0; # Or higher`
- -E command line option
- `perl -e 'say “hello”'`
- `perl -E 'say “hello”'`

Perl 5.12

- Released 12 April 2010
  - 5.12.4 20 June 2011
- Many new enhancements

5.12 Enhancements

- package NAME VERSION syntax
- ... operator
- Implicit strictures
- Y2038 compliance

5.12 Enhancements

- Smart match changes
- New modules
  - autodie
  - parent

package NAME VER

- Declare the version of a package in the package declaration
- `package My::Package 1.23;`
- Equivalent to
- ```
  package My::Package;
  our $VERSION = 1.23;
  ```



  ```

… Operator

- Called the “yada-yada” operator
- Used to stand in for unwritten code
- ```
  sub unimplemented {
     ...
  }
  ```
- Code compiles
- Throws an “unimplemented” exception when run


  ```

Implicit Strictures

- Requiring a version of Perl greater than 5.11 implicitly turns on use strict
- `use 5.12.0;`
- Is equivalent to
- ```
  use strict;
  use feature ':5.12';
  ```



  ```

Y2038 Compliance

- Core time functions are now Y2038 compliant

Smart Match Changes

- Some changes to Smart Match operator
- No longer commutative
- See new table in perlsyn
- Still in flux!

New Modules

- Some new modules in the standard distribution
- autodie
- parent
  - Better version of base.

Perl 5.14

- Released 14 May 2011
  - 5.14.2 26 Sept 2011
- Many new enhancements

5.14 Enhancements

- Non-destructive substitution
- Container functions accept references
- Package block
- New modules

Non-destructive substitution

- New /r option on s/// and tr///
- Copies input
- Acts on copy
- Original unmodifed
- ```
  $_ = 'cat';
  $new = s/cat/dog/r; # $_ remains 'cat'
  ```



  ```

Container functions accept references

- Array & hash functions used to require arrays or hashes
  - `push @array, $value`
  - `@keys = keys %hash`
- Even if you have a reference
  - `push @$arrayref, $value`
  - `@keys = keys %$hashref`

Container functions accept references

- Array & hash functions now accept references
  - `push $array_ref, $value`
  - `@keys = keys $hash_ref`
- Currently experimental

Package block

- Attach a code block to a package declaration
- `package MyPackage { ... }`
- Equivalent to
- `{ package MyPackage; ... }`
- Can also declare a version
- `package MyPackage 1.23 { ... }`

New Modules

- Many modules for parsing META files
- CPAN::Meta::YAML & JSON::PP
- CPAN::Meta
- CPAN::Meta::Spec & CPAN::Meta::History
- Module::Metadata

New Modules

- Other new modules
- HTTP::Tiny
- Perl::OSType
- Version::Requirements

Perl 5.16

- Due in spring 2012
- Currently in development at 5.15
  - 5.15.3 – 2011 Sep 21
  - Code freeze – 2011 Dec 20

Perl 5.16

- Look for changes in perldelta
  - perl5150delta
  - perl5151delta
  - perl5152delta
  - perl5153delta

Some Highlights

- CORE on all keywords
- Continue outside switch
- Breakpoints with filenames
- Remove Perl 4 *.pl

More Information

- perldoc perl5100delta
- perldoc perl5120delta
- perldoc perl5140delta

That's all folks

- Any questions?