Dude, where’s my flying car?

A whirlwind exposition of the Perl 6 language: its release status, some concrete syntactic examples, a historical overview, a real live demo, and current status and roadmap.
masak & jnthn

- started programming at 10
- learns a new language every year
- enjoys cooking, writing music, and beer
- programming since being 8 years old
- multi-paradigm programming wizard
- enjoys beer, mountains, and photography
We like Perl 6.

This talk is about why.
When will Perl 6 be released?

How about a production-ready Perl 6?

You know, an official release.

6.0.0?

You may have heard these things about Perl 6...

Will it ever be finished?

Perl 6 has “missed the boat”!

Perl 6 is vapourware!
Fun fact: we do make releases! 😊

But that’s not what people mean when they say “released”. 😞
Perl 6 is partway done. Some things are ready for use.

(In this talk and the next one, we’ll only talk about things that are implemented already. You’ll see that it’s quite a lot.)
We’re riding the wave of the adoption curve, inviting people as we go along:
“The future is already here — it's just not very evenly distributed.”

William Gibson
A few small language examples
Loops

for @students { ... }
for @students -> $student { ... }

for @tastes Z @foods -> $taste, $food { ... }
for @tastes X @foods -> $taste, $food { ... }

while $continue { ... }
until $quit { ... }

repeat while $continue { ... }
repeat until $quit { ... }

loop { ... }
loop (;;) { ... }

combine together like a zipper

combine together in all possible ways

test condition after first iteration

C-style loop
Subroutines

sub foo { say "OH HAI" }
foo();                      # OH HAI
foo;                        # OH HAI

sub bar($a, $b?) { say defined $b }
bar(1, 2);                  # True
bar(3);                     # False

sub baz($a, $b = 5) { say $b }
baz(1, 2);                  # 2
baz(3);                     # 5

sub greet($name, :$greeting = "Hello") {
    say "$greeting $name";
}
greet "jnthn";              # Hello jnthn
greet "kathy", :greeting(“你好”);      # 你好 kathy
class Point {
    has Real $.x;
    has Real $.y;

    method gist {
        "($.x, $.y)"
    }
}

my Point $p .=
    new(:x(3), :y(4));

say $p;  # (3, 4)

class Rectangle {
    has Point $.topleft;
    has Point $.bottomright;

    method gist {
        "$.topleft - $.bottomright"
    }
}

class SmoothRectangle is Rectangle {
    method gist {
        callsame() ~ " with web 2.0 corners"
    }
}
Subtypes and enums

\textbf{subset} EvenInt of Int where \{ $^n \% 2 \};

say 5 \textasciitilde EvenInt; \hspace{1em} \# False
say 8 \textasciitilde EvenInt; \hspace{1em} \# True

\textbf{sub} foo(EvenInt $e) \{ \ldots \}

\textbf{enum} Day <Sun Mon Tue Wed Thu Fri Sat>;
say +Fri; \hspace{1em} \# 5
say ~Fri; \hspace{1em} \# Fri
say Fri.kv; \hspace{1em} \# Fri 5

say 3 \textasciitilde Day; \hspace{1em} \# True
say 9 \textasciitilde Day; \hspace{1em} \# False
Operator overloading

sub postfix:<!>(n) { [*] 1..$n }

say 5!;
① Build ranks and suits
② Build a deck of cards
③ Build a table of card points
④ Draw a random hand of five cards
⑤ Print the hand and its total point sum
With Perl 5
my @suits = qw< ♣ ♦ ♥ ♠ >;
my @ranks = (2..10, qw< J Q K A >);

① Build ranks and suits
# concatenate each rank with each suit
my @deck;
for my $rank (@ranks) {
  for my $suit (@suits) {
    push @deck, "$rank$suit";
  }
}

② Build a deck of cards
③ Build a table of card points

```perl
my %points;
for my $rank (@ranks) {
  for my $suit (@suits) {
    my $score = $rank eq 'A' ? 11
      : $rank =~ /[JQK]/ ? 10
      : $rank;
    $points{"$rank$suit"} = $score;
  }
}
```
# grab five cards from the deck
my @hand;
for (1..5) {
    my $card = $deck[rand @deck];
    redo if grep { $_[0] eq $card } @hand;
    push @hand, $card;
}
# display my hand
say join ' ', @hand;

# tell me how many points it's worth
my $sum;
for $card (@hand) {
    $sum += $points{$card};
}
say $sum;
With Perl 6
my @suits = < ♠ ♦ ♥ ♣ >;
my @ranks = 2..10, < J Q K A >;

① Build ranks and suits

no need for qw any more;
<> is now a list quoter
# concatenate each rank with each suit
my @deck = @ranks X~ @suits;

the two for loops are gone; cross operator joins together elements in all possible ways

② Build a deck of cards
③ Build a table of card points

my %points = @deck Z ((2..10, 10, 10, 10, 11) xx 4);

no for loop; zip operator combines two lists
④ Draw a random hand of five cards

```perl
# grab five cards from the deck
my @hand = @deck.pick(5);
```

no for loop; built-in .pick method
# display my hand
say @hand;

# tell me how many points it's worth
say [+ %points{@hand};

Print the hand and its total point sum

no join; you get spaces for free

for loop folded into reduce operator
my @suits = < ♣ ♦ ♥ ♠ >;
my @ranks = 2..10, < J Q K A >;

# concatenate each rank with each suit
my @deck = @ranks X~ @suits;

my %points = @deck Z ((2..10, 10, 10, 10, 11) xx 4);

# grab five cards from the deck
my @hand = @deck.pick(5);

# display my hand
say @hand;

# tell me how many points it's worth
say [+] %points{@hand};
Overview of the history of Perl 6

- Project announced
- Apocalypses
- Synopses
- Exigeses

Phases:
- RFC phase
- Specification phase
- Implementation phase

Years:
- 2000
- 2001
- 2002
- 2003
- 2004
- 2005
- 2006
- 2007
- 2008
- 2009
- 2010
- 2011
- 2012

Tools:
- Pugs
- Rakudo
- Niecza

Today
Demo of a real application

IRC bot, running on Rakudo, using Github’s REST API to list issues
What's there today

Basic control structures, blocks, file IO, regexes, control flow, variables, constants, functions, etc

Classes

Roles

Subset types

Enums

Mixins

Operators

Reduction ops

Hyper ops

Cross ops

Zip ops

Introspection

Junctions

Phasers

Meta-Object Protocol

Lots of built-in types

Pod documentation

Multi dispatch

Advanced signature matching

Regexes

Grammars
What's we’re still working on

Macros

Performance

Native type stuff

Compile-time optimizations

Slangs

Some advanced regex constructs

Backend portability

Perl 5 interop
Perl 6 is partway done.

Some things are ready for use.

Is it finished, polished, production-hardened?

No.

But it’s worth checking out.
Try it out!
Welcome to Perl 6

Perl 6 is currently being developed by a group of volunteers. You can help too. The only thing you have to do is to be nice to all kinds of people (and don't use profanity) and someone will be glad to help you get started.

Community
- #perl6 IRC channel: system IRC client or chat live in your browser
- Perl 6 screencasts: Perl 6 on Rosetta Code, Blogs, Perl 6 wiki

»MORE«

Documentation
- Using Perl 6, an open-source Perl 6 book
- Perl 5 to 6, introduction for Perl 5 programmers

»MORE«

Specification
- Synopsis: official Perl 6 design documents
- STD.pm: official Perl 6 grammar

»MORE«

Compilers
- Rakudo, a compiler based on Parrot
- Niecze, a Perl 6 compiler for the CLR (mono/.NET)

»MORE«

Modules and Whatever
- Perl 6 modules directory
- Panda: a module installer for Rakudo

If you want to contribute to this page, look here for instructions.