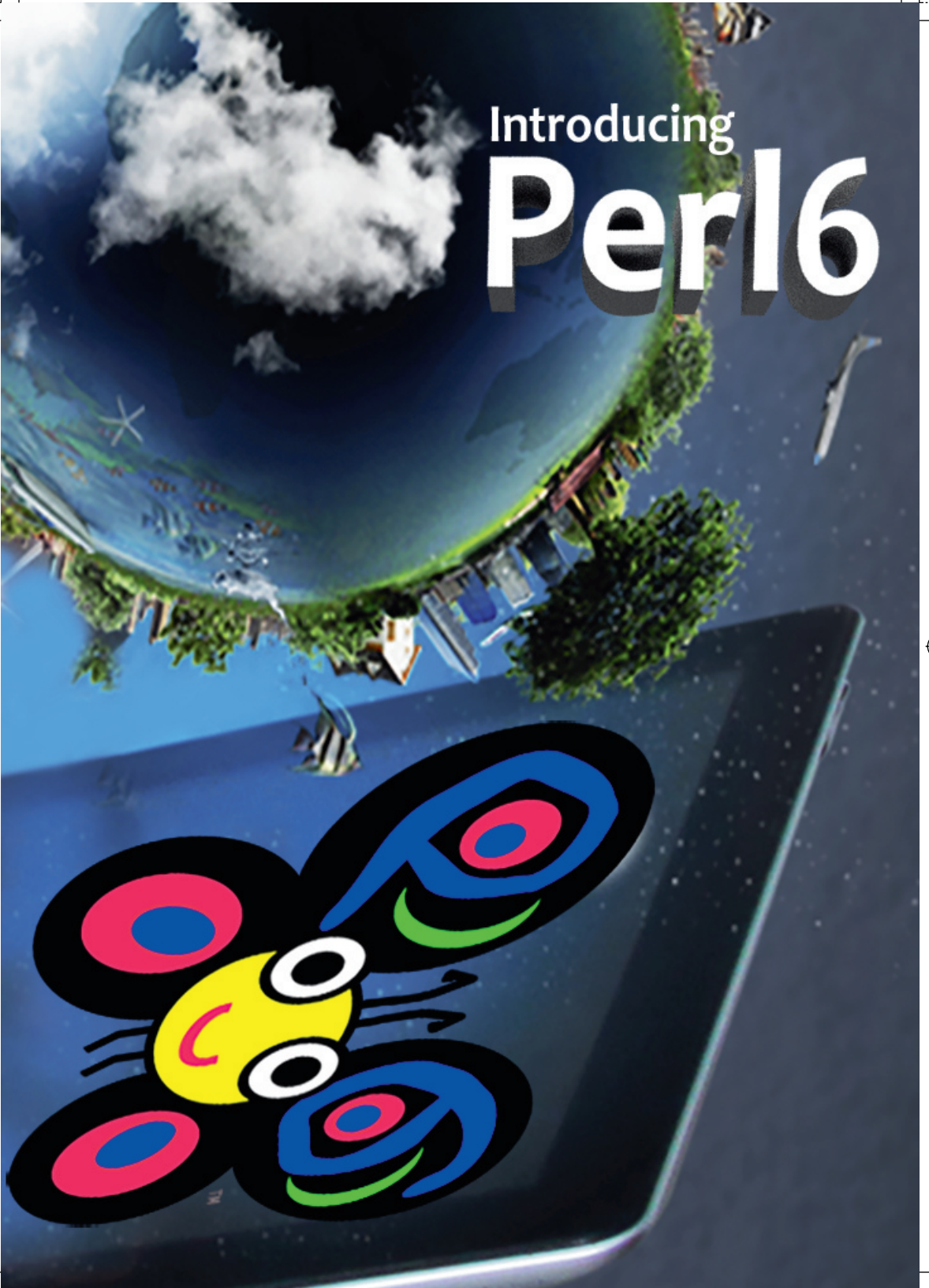




Introducing Perl6





Download & Use

perl6.org/getting-started
rakudo.org/how-to-get-rakudo

github.com/tadzik/rakudobrew

moarvm.org

Getting started
Suitable for most people,
download and installation
instructions.
The simplest way to install Perl6
is to use Rakudo brew
MoarVM

Useful places to start learning Perl6

perl6intro.com
learnxinyminutes.com/docs/perl6
rosettacode.org/wiki/Category:Perl_6

perl6advent.wordpress.com

p6weekly.wordpress.com

6guts.wordpress.com

strangelyconsistent.org

pl6anet.org

szabgab.com/perl6.html
blogs.perl.org/users/damian_conway

design.perl6.org

Perl6 Introduction
Perl6 Learn x in y minutes
Perl6 section of solving
problems with different
programming languages
Perl6 Advent calendar: every
day leading up to Advent, a
new interesting Perl6 example
Perl6 Weekly, with the latest
developments in Perl6
Perl6 Guts, the newest innards of
Perl6 explained by Jonathan
Worthington
Strangely Consistent, an
insightful blog by Carl Masak
Perl6 Planet, a collection of blogs
and articles
Screencasts about Perl6
If you ever have the chance to
attend a presentation by
Damian, enjoy!
Historical Perl6 Design Documents

Supported by

perlfoundation.org
enlightenedperl.org

The Perl Foundation
Enlightened Perl Organisation



[irc.freenode.net - #perl6](https://irc.freenode.net/#perl6)





Welcome to Perl6

After 15 years of design and development Perl6 has been released and being used in production. Perl6 is a supremely flexible language, adapting to your style of programming, whether that be quick one-liners for sysadmins, scripts to manage a database import, or the full stack of modules necessary to realise an entire website.

Perl6 enhances Perl's long-term appeal with a proper object system including roles, threading and multi-method dispatch. Perl6 has spent a long time coming to fruition and has learned from other programming languages building on their success and learning from the issues of the past. We believe Perl6 is a language that will last for decades as it has been conceived to adapt to future trends and is flexible in its usage with other languages.

We have collected here a list of some of the many advantages to using Perl6.

Rakudo and Perl6

Rakudo Perl6 on MoarVM (see Rakudo Star, rakudo.org) is currently the most advanced implementation of Perl6. Any implementation that passes the official test suite can call itself "Perl6". Other implementations (Niecza, Perlito, Pugs) are at various levels of maturity and completeness, see perl6.org/compilers/. Perl6 now runs on two virtual machines: MoarVM and JVM. MoarVM offers a wider support for specified features than Perl6 on JVM (work is in progress to add JavaScript as a virtual machine). MoarVM is solely developed for Perl6 and not (yet) used for other programming languages.

rakudo.org/how-to-get-rakudo





Perl6 Features

There are many reasons to learn Perl6 we have listed a few of our favourites.

General Features

- Perl6 is a clean, modern, multi-paradigm language; it offers procedural, object-oriented AND functional programming methodologies.
- Easy to use consistent syntax, using invariable sigils for data-structures.
- Perl6 is a very mutable language (define your own functions, operators, traits and data-types, which modify the parser for you).
- Adding a custom operator or adding a trait is as simple as writing a subroutine.
- Advanced error reporting based on introspection of the compiler/runtime state. This means more useful, more precise error messages.
- Multiple versions of a module can be installed and loaded simultaneously.
- System administration is simplified due to simpler update/upgrade policies.
- Runtime optimization of hot code paths during execution (JIT), by inlining small subroutines and methods.
- Runs on small (e.g. Raspberry Pi) and large multi-processor hardware.
- Garbage collection based: no timely destruction, so no ref-counting necessary. Use phasers for timely actions.
- Fewer lines of code allow for more compact program creation. Huffman-coding of names allows for better readability

Text-Processing

- Full grapheme based Unicode support, including Annex #29, meaning almost unparalleled excellent Unicode support.
- Extensible grammars for parsing data or code (which Perl6 uses to parse itself).
- Execute code at any time during parsing of a grammar, or when a certain match occurred.
- Regular expressions are cleaned up, made more readable, taken to the next level of usability, with a lot more functionality. Named regular expressions are made possible for ease of use.

Scoping

- Dynamic variables provide a lexically scoped alternative to global variables.
- Emphasis on composability and lexical scoping to prevent “action at a distance”. For example, imports are always lexically scoped.
- Easy to understand consistent scoping rules and closures.
- Phasers (like BEGIN/END) allow code to be executed at scope entry/exit, loop first/last/next and many more special contexts.

```
# Words from file
for 'dict.txt'.IO.words -> $word {
  say "$word probably rhymes with Perl"
  if $word ~~ /[ea?u|i] r1 $/;

  say "$word is a palindrome"
  if $word eq $word.flip;
}
```





Object Orientated Programming

- Powerful object orientation, with classes and roles (everything can be seen as an object). Inheritance. Subtyping. Code-reuse.
- Introspection into objects and meta-objects (turtles all the way down).
- Meta Object Protocol allowing for meta-programming without needing to generate/parse code.
- Subroutine and method signatures for easy unpacking of positional and named parameters, and data structures.
- Methods can be mixed into any instantiated object at runtime, e.g. to allow adding out-of-band data.

Typing

- Multi dispatch on identically named subroutines/methods with different signatures, based on arity, types and optional additional code.
- Compile time error reporting on unknown subroutines/impossible dispatch.
- Optional gradual type-checking at no additional runtime cost. With optional type annotations.
- Easy command-line interface accessible by MAIN subroutine with multiple dispatch and automated usage message generation.

Concurrency, Parallelism, Asynchrony

- High level concurrency model, both for implicit as well as explicit multi-processing, which goes way beyond primitive threads and locks. Perl6's concurrency offers a rich set of (composable) tools.
- Multiple-core computers are getting used more and more, and with Perl6 these can be used thanks to parallelism, both implicit (e.g. with the `>>` method) and explicit (`start { code }`). This is important, because Moore's Law is ending.
- Structured language support is provided to enable programming for asynchronous execution of code.
- Supplies allow code to be executed when something happens (like a timer, or a signal, or a file-system event, or gui events).
- The keywords `react/whenver/supply` allow easy construction of interactive, event driven applications.

Data-structures

- Junctions allowing easy checking of multiple possibilities, e.g. `$a == 1|3|42` (meaning is `$a` equal to 1 or 3 or 42).
- Lazy evaluation when possible, eager evaluation when wanted or necessary. This means, for example, lazy lists, and even infinite lazy lists, like the Fibonacci sequence, or all prime numbers.
- Lazy lists defined with a simple iterator interface, which any class can supply by minimally supplying a single method.
- Native data types for faster, closer to the metal, processing.
- Floating point math without precision loss because of Rats (rational numbers).
- Large selection of data-types, plus the possibility to create your own types.
- Multi-dimensional shaped and/or native arrays with proper bounds checking.
- Automatic generation of hyper-operators on any operator (system or custom added).

Interoperability

- Interfacing to external libraries in C/C++ is trivially simple with Native Call.
- Interfacing with Perl 5 (CPAN)/Python modules is trivially simple with `Inline::Perl5` and `Inline::Python`.
- Perl6 runs on a variety of back-ends. Currently MoarVM & JVM, JavaScript is in development, more may follow.
- The keywords `react/whenver/supply` allow easy construction of interactive, event driven applications.





Run Rakudo Perl6

To run a Perl6 program with Rakudo Perl6, include the installation directory in your system PATH variable and issue a command like:

\$ perl6 hello.pl (this must look familiar to any Perl developer).

If you invoke Rakudo Perl6 without an explicit script to run, it enters a small interactive mode that allows the execution of Perl6 statements from the command line.

```
# Custom operators
sub postfix:<!> (Int $n) {
    fail "Not a Natural Number in Factorial"
    if $n < 0;
    [*] 2..$n
}

use Test;
isa-ok (-1)!, Failure, "Factorial for -1 fails";
ok 0! == 1, "Factorial for 0";
ok 1! == 1, "Factorial for 1";
ok 5! == 120, "Factorial for a larger integer";
```

Modules for Rakudo Perl 6

A growing list of modules is published on modules.perl6.org. Of course, this can not be compared with the immense number of Perl5 modules on CPAN. Nevertheless, the number of modules grows every week and they make Perl6 more useful.

Especially useful is `Inline::Perl5`, enabling the use of virtually all Perl5 code within your Perl6 code, there is also `Inline::Python` which adds the same functionality for Python work and we will see other languages `Inline` to Perl6. If you are a Perl5 developer you can use Perl6 with `Inline::Perl6`, virtually all Perl6 code can be used within your Perl5 code.

Many important topics have been covered by Perl6 and more are under development to be added. The module management tools for Rakudo Perl6, **panda** or **zef**, make your life as a module user easier.

```
# Infinite Lazy Lists
my @fb = 0, 1, ** ... *;
say "Fibonacci number #8 is @fb[7]";
```



perl6.org/getting-started





Why the '6' in Perl6?

Perl6 developers don't need to keep Perl6 backward compatible to Perl5, and have changed and extended the language, and added new syntactic and semantic features that enable more power and expressiveness. This means a major break in syntactic and semantic compatibility from Perl5, thus the increase from 5 to 6. However, this does not mean that Perl5 is going away anytime soon. In fact, quite the opposite. Both Perl5 and Perl6 have very active developer communities who mold the languages.

Getting Involved

If you are inspired now and want to contribute to the Perl6 community, there are some resources available to you. A full list of these are on the inside cover of this booklet but we recommend you start with the perl6.org homepage.

IRC: the channel #perl6 on irc.freenode.net discusses all things Perl6. The people are very friendly and very busy developing Perl6. Keep an eye on this to stay up-to-date. The channel is logged, and you can read back to see what has been discussed: irclog.perlgeek.de/perl6/today

Mailing lists: send an email with subject 'subscribe' to:

perl6-announce-subscribe@perl.org

Announcements and news. Low traffic.

perl6-users-subscribe@perl.org

User questions and discussions regarding the Perl6 language and compilers.

perl6-language-subscribe@perl.org

For issues regarding the Perl 6 language specification.

perl6-compiler-subscribe@perl.org

For issues regarding various Perl6 compilers.

enlightenedperl.org

This document was produced by donations
to the Enlightened Perl Organisation





Come and Join the Party!

The official full first stable version, Perl6 version 6.c, was released on Christmas of 2015. The whole world can use it now and be a part of how awesome programming can be!

Perl6 is Perl5's sister language released after 15 years of development. Perl6 is a flexible language, adapting to your style of programming from quick one-liners for sysadmins to the full stack of modules necessary to realise an entire website.

It enhances Perl's long-term appeal with a proper object system including roles and multi-method dispatch.

Perl6 learns from other languages. For instance: taking threading from Java (simplified to a handful of methods); Foreign-function interfaces from Lisp make accessing libraries as simple as one line of code.

Regular expressions are now turned up to 11 with the introduction of Parser Expression Grammars, which let you tackle huge parsing tasks. Strictures and warnings are now automatic, cutting out huge swathes of potential errors.

Perl6's mottos remain the same as Perl: "Perl is different. In a nutshell, Perl is designed to make the easy jobs easy, without making the hard jobs impossible." and "There Is More Than One Way To Do It". Now with even more -Ofun added.

Look inside for a list of some of Perl6's many features, an introduction to the community and useful sites.



THE Perl FOUNDATION

