Raku Perl 6 DIWALI 6.d

Release Information for the Second Major Version of the Language

What is Raku?

Some community members believe "Perl 6" is a very confusing name: it uses a digit as part of the name and it conflicts with the name of a sister language "Perl". Some outsiders don't realize "Perl 6" is a brand new language—vastly different than "Perl"—and so avoid it, thinking they know what it is already. At the same time, some who love "Perl" come to "Perl 6" and feel tricked, because the language isn't the next release of that language, but an entirely different one.

To clear up the confusion, Larry Wall created a second name for the language, a "stage name" if you will. That name is "Raku". It can be used interchangeably with the original "Perl 6" name or even be combined with it to form "Raku Perl 6". Pick the one that works the best for you and use it consistently.

What is your preferred name for the language?

Table of Contents

What is being released?

The Language Specification

The Raku Perl 6 language is defined by its specification. This release brochure is for the second major version of that specification, version number 6.d, code named "Diwali".

Compiler Implementation

Compilers that implement the specification will switch to support 6.d Diwali version by default and follow their standard release schedule.

If you use the Rakudo compiler, the next version release process will start on November 17, 2018.

Compiler Distributions

Some implementations offer compiler distributions that include a compiler and some modules. Rakudo Star is one of such distributions and it's currently unknown if an out-of-schedule release of Rakudo Star will be with v6.d language as default.

Third Party Packages

Third-party packages will get updated following their standard release schedule.

Upgrade Info

Only a portion of all the language changes require you to do anything about it. The Version-Controlled Change will require you to adapt your code to the new behaviour or to request 6.c language in your file by adding use v6.c; pragma at the top of it. Deprecations require to switch away from the deprecated features, but they will continue to work for a couple of years or longer. The rest of the changes do not require you to do anything.

The ChangeLog on the pages that follow—where possible—gives details on the alternative code to use,

Types of Changes

The 6.d language changes can be broadly grouped into: (a) non-backwards compatible changes that are not available when a file requests an older language version; (b) non-conflicting changes; and (c) deprecations

Version-Controlled Changes

For non-backwards compatible changes, you can still get the old behaviour by asking to use 6.c language using a version pragma:

```
use v6.c;
my num $x;
say $x; # Output: NaN
```

Keep in Mind

Version pragmas must be the first thing in a file (preceeding comments and Pod are OK) and they only affect non-backwards compatible language changes, not all the 6.d changes.

Non-Conflicting Changes

These are changes that do not conflict with 6.c language and so will not impact your old code. The compilers are free to make these changes available even when 6.c language is enabled by the version pragma.

Deprecations

Some language features are no longer available in the new language version. Compilers that had them implemented should continue to support them, while possibly emitting a deprecation warning when 6.d language version is used.

Implementation-Specific Note

It is known that Rakudo compiler currently cannot emit version-dependent deprecation warnings. Due to this issue, deprecated features will be supported for a longer time period (until 6.e/6.f release).

ChangeLog (cont)

Introduction

This document lists changes in Perl 6.d (Diwali) language from Perl 6.c (Christmas) version. A particular implementation of the language may contain additional changes; please consult with the changelog for your implementation.

At the same time, a particular implementation may have had certain features already implemented during the 6.c version period. This ChangeLog concerns itself with new features added to the specification on a language level and not the status of their implementation in a particular compiler.

Scope and Target Audience

This ChangeLog is targeted towards language users, to help with preparation to use compilers supporting latest language version. Thus, it does not contain every minute change to the specification that occurred. Implementations wishing to ensure full compliance with the new version of the language specification should execute the test suite available at **https://github.com/perl6/roast/** and examine any failing tests.

There are new features that did not exist in 6.c language. For full details about them, please consult with the language documentation on **https://docs.perl6.org/**

Items in *Version-Constrolled Changes* section are protected by version pragma and older behaviours can be obtained by explicitly using use v6.c to request an older language version. All other changes do not conflict with the 6.c language version and implementations may choose to make them available even when an earlier language version is requested.

Version-Controlled Changes

- `&await` no longer blocks a thread while waiting
- `whenever` not in lexical scope of `react` throws
- `$*ARGFILES` inside `sub MAIN` is always fed by `$*IN` (see `IO::CatHandle` type if you need old behaviour)
- Constructs (literally that, with no space inside parentheses) `$()`, `@()`, and `%()` are no longer magical
- Variables with `:D`/`:U` type constraints default to type object of the constrained type
  (e.g. so you could use `.new` with them)
- `start` blocks in sink context attach exception handler
- Routines must use `return-rw` to return a `Proxy`, even if routine is marked as is `raw` or `is rw`
- Native num types default to `0e0` instead of `NaN`
- On subroutine names, the colonpair with key `sym` (e.g. `:sym<foo>`) is reserved

Deprecations

These methods are deprecated in 6.d language and will be removed in 6.e. Implementations may choose to emit deprecation warnings or to offer these methods for a longer period than 6.e release.

- The use of `'-'` (single hyphen) as a special path to `&open` to mean the special handles
  (`use IO::Special` objects instead)
- `IO::Handle.slurp-rest` (use `.slurp` instead)
- `Any.flatmap` (use combination of `.flat` and `.map` methods instead)
- `Cool.path` (use `.IO` instead)
- `Pair.freeze` (use `Pair.new` with decontainerized arguments instead)
- `Str.subst-mutate` (use `Str.subst` with `.=` method call assign metaop instead)
- `Rational.norm` (`Rational` types are required to be normalized on creation now)
- `IO::Path.child` (use `.add` instead)
- `&undefine` (assign `Empty`/`Nil` directly, instead)
- `:count` argument on `&lines`/`Str.lines` routines (use `.elems` call on returned Seq instead)
- `&is_approx` in Test.pm6 (use the very similar behaviour of `&is-approx` instead)

New Behaviors

- Improved custom handling of `sub MAIN` via new definable `&RUN-MAIN`, `&ARGS-TO-CAPTURE`, and `&GENERATE-USAGE` subs
- `QuantHashes`/`Map` in `%` variables and `List` in `@` variables can be declared with `is` trait (e.g. `my %h is Set`)
- New `<ww>` regex rule: match within word only
- Loops can produce a list of values from the values of last statements
- `last`/`redo` in a loop that collects its last statement values return `Empty` for the iterations they run on
- `.perl` can be called on consumed `Seq`s, multi-dimensional arrays, `Date`, and `CallFrame`
- `.gist` can be called on `Attribute`
- Numerous improvements to auto-generated `USAGE` message
- `is hidden-from-USAGE` trait to hide `sub MAIN` candidates from auto-generated `USAGE` message
- `Parameter.perl` includes introspectable defaults
- `%*ENV` values are allomorphic
- Trying to use variables `$;`, `$,`, `$.`, `$\`, `$(`, `$)`, `$<`, `$>`, `$/`, `$\`, `$[`, `$-`, `$+`, and `$@` throws `X::Syntax::Perl5Var`
- Default `Hash.of` returns a `Str(Any)` coercer type object
- Non-ASCII numerics can be used in `:42foo` colonpair shortcut
- `StrDistance` stringifies to its `.after` string
- More well-defined formatting of `Pod` tables
- `Enumeration.enums` returns a `Map`
- `.Range` on various integer types returns the range of values they support
- `min`/`max` routines also work on `Hash`es
- `Signature` literals can contain string/numeric literals as well as the invocant marker
- `List.invert` maps via a required `Pair` binding, resulting in potential type check failures
- `:exists` can be used with multi-dimensional associative subscripts
- Dynamically created lists can be used to define an enum
- `Junctions` can be used as a matcher in `.first`
- Native attributes can be used as bind targets in parameters
- `Proc` can work with `IO::Pipes` from other `Proc`s
- Typed arrays can be created with both `my SomeType @array` and `my @array of SomeType`
- Items with negative weights are removed when coercing a `Mixy` to `Setty`/`Baggy`
- `:nth` adverb on `m//` accepts a `Junction` as argument
- `CX::Warn` and `CX::Done` can be caught inside `CONTROL` phaser
- `next` can be used in `whenever`
- `require`'d symbols no longer transitively exposed
- Multi-dimensional access via `{…}`, similar to how it works with `[…]`
- Any open handles at `END` time get automatically closed
- On a cached `Seq`, the cached list is used when `&infix:<eqv>`, `.Slip`, `.join`, `.List`, `.list`, `.eager`, `.Array` and `.is-lazy` are called
- `IO::Handle.encoding` takes `Nil` to indicate switch to binary mode
- `is default` trait works with attributes
- Parameters with `is rw` trait are considered narrower in multi dispatch than those without it
- `.gist` of `Array`, `Blob`, and `Map` gets trimmed to 100 elements

ChangeLog (cont)

- New `for` statement modifiers `hyper for`, `race for`, and `lazy for`
- `for` loop automatically serializes `RaceSeq`/`HyperSeq`; use new for statement modifiers `hyper for`/`race for` to avoid
- `&infix:<does>` can be used with non-composable instances on RHS
- Numeric comparators can be used with `DateTime` objects
- `Pod` preserves the type of whitespace
- Defined semantics for `@-`, `%-` and `&-`sigilled constants

Math

- `Rational`s are always reduced on creation and remain immutable throughout their life
- `-Inf`, `Inf`, and `NaN` can be round-tripped through a `Rational` type by being represented as values `<-1/0>`, `<1/0>`, and `<0/0>` respectively. Zero-denominator `Rational`s are normalized to one of those three values
- Calling `.Int` on ±`Inf` and `NaN` throws
- Improved IEEE 754-2008 compliance in `Num` operators and math functions
- Negative zero `Num` (`-0e0`) gets correctly handled by all routines and syntactical constructs
- Stringification of `Num` type is required to be roundtrippable to the original `Num`
- Defined `Complex` exponentiation involving zeros
- Negative powers in `.expmod` are valid

Sets, Bags, Mixes (aka QuantHashes) and set operators

- Set operators can be used on any object, which will be coerced when needed
    - So no pre-coercion is needed or wanted
    - Set operators are at liberty to not create any `QuantHash` if they can perform the desired functionality without them
- Set operations on different types of `QuantHashes` will coerce to the most liberal form (`Set` ➜ `Bag` ➜ `Mix`)
- The set_precedes family of set operators (`(<+)`, ≼, `(>+)`, ≽) has been removed
    - Used to be a `Baggy` form of the subset operator
    - `QuantHash`es are upgraded to their most liberal form, so `(<=)`, ⊆, `(>=)`, ⊇ do the right thing
- `.classify-list` method is available on `Baggy` types
- `.categorize-list` method is available on `Baggy` types
- `.invert` method is available on core `QuantHash` types
- `.antipairs` method can be used on `QuantHash` types
- `QuantHash` types have `.new-from-pairs` and methods to convert one `QuantHash` type to another (e.g. `.Bag` method on `Set` type)
- `.hash` on `QuantHash` types does stringify keys

New Parameters and Arguments

- `Date.new` accepts a `:&formatter`
- `.first` can take `:kv`
- `unique` and `.repeated` can take `:&as` and `:&with`
- `&plan` in Test.pm6 can take `:skip-all`
- `&run`/`&shell` can take `:merge`
- `&note` can be called with no arguments
- `open` accepts `:$out-buffer`
- `IO::Path.resolve` can take `:completely`
- `IO::Path.parent` can take an `Int` indicating parent level
- `Proc::Async.new` slurps positional arguments
- `Signature.ACCEPTS` accepts non-`Signature`/`Capture` arguments
- `&EVAL` can take a `Blob`
- `Promise.keep`/`.break` can be called with no arguments
- `.sum` on native arrays can take `:wrap`
- `is required` trait can now take an argument indicating reason
- `IO::Socket::Async.listen` can bind to port `0` to ask OS for a free port
- `.encode` can take `:translate-nl`

New Routines and Operators

- New `atomicint` Unicode operators and ASCII alternatives that guarantee thread-safe, atomic operation: `&infix:<⚛=>`/`&atomic-assign`, `&prefix:<⚛>`/`&atomic-fetch`, `&prefix:<++⚛>`/`&atomic-inc-fetch`, `&postfix:<⚛++>`/`&atomic-fetch-inc`, `&prefix:<--⚛>`/`&atomic-dec-fetch`, `&postfix:<⚛-->`/`&atomic-fetch-dec`, `&infix:<⚛-=>`/`&infix:<⚛−=>`/`&atomic-fetch-sub`, and `&infix:<⚛+=>`/`&atomic-fetch-add`
- `&cas`: atomic compare and swap
- The `≤`, `≥`, and `≠` operators are Unicode operator alternatives to `<=`, `>=`, and `!=` respectively
- `&infix:<unicmp>`/`&infix:<coll>`: alternative behavior of `&infix:<cmp>`
- `TR///` operator: non-mutating version of `tr///`
- `submethod TWEAK`: similar to `BUILD`, except it's compatible with attribute defaults
- `&duckmap`: apply `&callable` on each element that behaves in such a way that `&callable` can be applied
- `&deepmap`: apply `&callable` on each element, descending into Iterables
- `&take-rw`: like `&take` but with a writable container
- `&indir`: execute code in a given `$*CWD`
- `&spurt`: see `IO::Path.spurt`
- `&prompt`: prompt user for input
- `uniprops`: multi-character version of `uniprop`
- `symlink`: create a file symlink
- `link`: create a file hardlink
- `.hyper`/`.race`: process a list of values in parallel
- `Seq.from-loop`: generate a `Seq` from a `Callable`
- `Str.uniparse`: parse one or more Unicode character names into the actual characters
- `Str.parse-base`: inverse of `Int.base` operation
- `IO::Path` provides `.ACCEPTS`, `.SPEC`, `.CWD`, `.Numeric`, `.add`, `.extension`, `.mode` and numerious file tests, `.parts`, `.sibling`, and `.spurt`
- `IO::Handle` provides `.READ`, `.WRITE`, `.EOF`, `.DESTROY`, `.readchars`, `.flush`, `.lock`, `.unlock`, `.out-buffer`, `.tell`, `.say`, `.slurp`, `.seek`, `.printf`, `.print-nl`, and `.watch`
- `IO::Pipe` provides `.proc`
- `Iterator` provides `.skip-one`, `.skip-at-least`, and `.skip-at-least-pull-one`
- `Mu.emit`: method form of `&emit`
- `&fails-like` in Test.pm6 module: allows testing for Failures
- `&bail-out` in Test.pm6 module: exit out of failing test suite
- `&is-approx` in Test.pm6 module: test a number is approximately like another
- `Buf` has `.allocate`, `.reallocate`, `.append`, `.push`, `.pop`, `.splice`, `.subbuf-rw`, `.prepend`, and `.unshift` methods
- `Range` supports `.rand`
- `Backtrace` has methods `.map`, `.flat`, `.concise`, and `.summary`
- `.classify-list` method is available on `Hash` types
- `.categorize-list` method is available on `Hash` types
- `Code.of`: returns the return type constraint
- `Code.line`/`.file`: returns the line/file of definition
- `Proc::Async` provides `.Supply`, `.ready`, `.pid`, `.bind-stdin`, `.bind-stdout`, and `.bind-stderr`

ChangeLog (cont)

- Proc.command/Proc::Async.command: the command we're executing
- Proc provides .signal, .pid, and .encoding
- Complex provides .cis, .reals, .ceiling, .floor, .round, .truncate, and .abs methods and can be compared with <=> (as long as the imaginary part is negligible)
- DateTime provides .offset-in-hours, .hh-mm-ss, and .Date
- DateTime can be compared with other DateTime objects using <=> operator
- Date provides .DateTime method
- &infix:<+>/&infix:<-> can be called with Duration, DateTime, and Real types
- Enumeration provides .Int, .pred, .succ, .kv, and .pair
- .Date can be called on an Instant
- Junctions can be created using Junction.new call
- List type has .to and .from methods
- Map type provides .Int method, returning the number of pairs
- Any.skip: skip values in a list
- Any.batch: more basic cousin of .rotor
- Mu.iterator: produce an Iterator for values in a list
- IO::Spec::* types provide .tmpdir, .extension, and .path
- Pair provides .ACCEPTS, .Pair, and .invert
- .Capture method is well-defined for all core types
- Defined semantics of .ACCEPTS on allomorphs
- Failure.self explodes unhandled Failures
- Thread.is-initial-thread: are we running in the initial thread?
- Match provides .Int and .actions
- IO::Socket::Async provides .socket-port and .peer-port
- Promise provides alternative constructors .kept and .broken
- WhateverCode provides .assuming
- WhateverCode and Block provide .cando
- .:<…> syntax for calling prefix operators as postfixes
- $*KERNEL provides .hostname
- Nil has .FALLBACK special method defined to return Nil

New Types

- atomicint: a native int sized to be usable with new atomic operators
- Lock::Async: a non-blocking mechanism for mutual exclusion
- Encoding::Registry: manage available encodings
- Encoding::Encoder: encoder for a specific encoding
- Encoding::Decoder: decoder for a specific encoding
- IO::CatHandle: use multiple read-only IO::Handles as if they were one
- Native str arrays
- Supplier::Preserving: cached live Supply factory
- Semaphore: control access to shared resources by multiple threads
- IO::Special: a path to special I/O device (e.g. STDOUT)
- Exceptions::JSON: an implementation of custom exceptions handler (can be used with PERL6_EXCEPTIONS_HANDLER env var)
- SeekType enum: values for use in IO::Handle.seek

New Variables

- $*USAGE: available inside MAIN subs and contains the auto-generated USAGE message
- %*SUB-MAIN-OPTS: settings for behaviour of sub MAIN
- %*SUB-MAIN-OPTS<named-anywhere>: allow named arguments to be placed at any position on the command line
- $*COLLATION: configures the four Unicode collation levels
- $*INIT-INSTANT: an Instant representing program startup time
- $*HOME: user's home directory, if one exists
- &*chdir: a Callable containing a variant of IO::Path.chdir that also sets process's current directory
- PERL6_TEST_DIE_ON_FAIL environmental variable: stop test suite on first failure
- PERL6_EXCEPTIONS_HANDLER environmental variable: specify custom exceptions handler class

Clarifications of Edge Case/Coercion Behaviour

- UInt smartmatches True with Int type object
- sink statement prefix explodes Failures
- Defined behaviour of permutations/combinations on 1- and 0-item lists and negative and non-Int arguments
- &val, Str.Numeric, and other Str numeric conversion methods fail when trying to convert Unicode No character group or synthetic numerics
- Synthetic numerics cannot be used in :42foo colonpair shortcut
- An Enumeration can now be used as a array shape specifier
- Numeric conversion of Str containing nothing but whitespace returns 0 now
- samemark with empty pattern argument simply returns the invocant
- .polymod can be used with lazy but finite lists of divisors
- .[*-0] index is defined
- Negative gaps in .rotor that are larger than the sublist throw
- Non-Int arguments to .rotor get coerced to Int
- .lines is defined when reading /proc files
- Defined behaviour of Thai numerals in postfix/prefix ++/-- on strings
- map inside sunk for is treated as sunk
- Sunk for loop sinks value of last statement's method call
- .Int on Bool objects returns an Int object
- splice can be used to extend an array
- classify works with Junctions
- .pairup on a type object returns an empty Seq
- .pairup always returns a Seq
- Synthetic codepoints are rejected from Date/DateTime constructors
- ⸨/⸩ pair can now be used as matching characters in quoting constructs
- .flat on an Array type object simply returns that type object
- Mixed-level classify on Hashes throws
- Junctions can be used to specify multiple keys to Hashes
- The Callable given to classify-list is now guaranteed to be executed only once per each item
- :delete on associative lookup on Hash type object returns Nil
- &is-deeply from Test.pm6 automatically .caches Seqs given as arguments and uses the returned Lists for testing
- Complex.new() gives <0+0i>
- Int.new is now guarantied to construct a new Int (rather than, say, re-use one from a constants cache)
- 1-arg versions of &infix:<=:=> and &infix:<eqv> are defined

ChangeLog (cont)

- Nil type now throws if directly or indirectly calling .BIND-POS, .BIND-KEY, .ASSIGN-POS, .ASSIGN-KEY, .STORE, .push, .append, .unshift, and .prepend
- Nil.ord returns an empty Seq
- Nil.chrs returns a "\0"
- Num.new coercers argument to Num
- infix:<Z>() returns an empty Seq
- .comb always returns a Seq
- Reduce with &infix:<+> with one item simply returns that item
- ()[0] returns Nil
- Regex smartmatching is allowed on (possibly-infinite) Seq
- Defined smartmatching with Range objects
- Set converted to a Mix/Bag no longer has Bool weights
- &infix:<gcd> is defined when one or more operands are 0
- Junctions autothread in defined routine
- sum can handle lists with Junctions in them
- Grammar.parse lets top regex backtrack
- The U+2212 MINUS SIGN [Sm] (−) is now supported by more constructs, such as Str.Numeric and &val
- Arity-1 &infix:<~> works with Blobs
- All Numeric literals are supported as value literals in signature
- \b and \B in regexes throw X::Obsolete
- True and False as value literals in signatures warn
- Return type of .sort and IO::Spec::Unix.path is always Seq
- Out-of-range .AT-POS on Range objects returns Nil
- Pair.AT-KEY for non-existent key returns Nil
- All Cool types provide .Rat/.FatRat coercers
- IO::Path filetests do not cache results of earlier test executions
- Seq eqv List as False based on type mismatch alone
- On arrays, Hashes, and QuantHashes, values from .kv, .values, and .pairs sequences are writable
- &infix:<∘>/&infix:<o> keep LHF's .of and RHS's .arity and .count
- Refined accepted arguments in regex operator adverbs (e.g. :in(…))
- Refined accepted combinations of arguments in IO::Handle.open
- IO::Path.Str does *not* include the value of .CWD attribute
- IO::Path type rejects paths with the nul byte ("\0") in them
- IO::Pipe's .path/.IO return an IO::Path type object
- IO::Path's .copy/.move fail if destination and original are the same
- dir-created IO::Paths' absoluteness is controlled by the invocant
- More-defined edge-case behaviour, Callable handling, .defined calling, and chaining of &infix:<andthen>, &infix:<orelse>, and &infix:<notandthen> operators
- Zen slicing of Seqs does not cache them
- List.Capture stringifies keys of any contained Pair objects
- &fail with handled Failure argument marks it as unhandled
- use lib accepts IO::Path objects
- Anchors ^, ^^, $, and $$ are valid in lookarounds
- Grammar.made supports type objects
- .isa supports subset type objects
- :delete can be used on lazy arrays
- &infix:<eqv> can work with certain cases of lazy arguments
- Dynamic lookup (::(…)) is restricted regex syntax and requires use MONKEY-SEE-NO-EVAL clearance
- Defined .Slip and .List on arrays with holes
- Promise.in/.at and Supply.interval work with zero and negative values
- Supply.interval minimum value is 0.001; lower values are treated as 0.001 and emit warnings
- Supply provides .Seq, .list, and .zip
- Can bind to native type attributes in build methods
- WhateverCode propagates use fatal
- say, note, put, print, and printf routines autothread Junctions
- IO::Handle.eof value changes accordingly when .seeking past end and back
- Defined .succ, .pred, and .Bool on allomorphs
- Defined .Bridge on core Numerics
- Defined .Numeric/.Real on type objects of core Numerics
- Defined Rational.Bool with respect to zero-denominator rationals
- say/note guaranteed to call .gist on subclasses of Str
- Defined Junction.Str returns a Junction
- Defined Junction.gist/.perl return a Str
- Map/Hash's .list/.cache return a List
- Defined .round's return type
- Defined Enumeration:D not does .ACCEPT an Enumeration:U

Miscellaneous

- The IO::ArgFiles type is just an empty subclass of IO::CatHandle
- Constraints on constants
    - Constraints are fully enforced
    - Attempting to parametarized type constraints on constants (i.e. using my Foo constant @int) throws X::ParametricConstant exception
- Pod =defn (definition list) directive is available
- Pod provides :numbered config key
- .^ver, .^auth, and .^name metamethods are available on module and are absent on a package, by design
- Fancy quotes ('…', "…", ｢…｣, and variants) are supported in qww<…>
- &infix:< > supports lookup of autogenerated Callables (e.g. &infix:<XX>)
- Using a named anon sub no longer produces redeclaration warnings
- Extended spec of ::?MODULE/$?MODULE variable
- sub MAIN can accept an Enumeration type constraint and where clause on arguments
- Type smiley constraints can be used on subsets
- start blocks and thunks get fresh $/ and $!
- R meta operator used with list-associative operators is defined
- Type coerces can be used in signature return type constraints
- &infix:<x>/&infix:<x> throw with -Inf/NaN repeat arguments
- Literal constructs put and put for throw, requiring use of parentheses
- Expanded specification coverage of Unicode routines and features
- Upgraded coverage to Unicode version 11
- $. method call syntax shorthand works with meta-methods

Policy Changes

Inclusion in Language Specification

It is now a policy that to be included in the language specification a "Proof of Viability" of a feature must be first successfully implemented in any largely-useful implementation of the language. The goal here is to avoid including language features that might be in conflict with some other feature.

Inclusion of New Features

To ensure we grow the language in a coherent manner, we're creating stricter requirements for the inclusion of new features in the language. The rough first draft of the policy is available in the specification's repository at

https://github.com/perl6/roast/blob/master/docs/New-Features-Policy.md

That document will likely see significant changes as we start applying it and refine the process to ensure the best outcome.

New Resources

We now post critical alerts related to the language on **https://alerts.perl6.org/** (see also `WWW::P6lert` / `p6lert` in ecosystem)

We launched a new subdomain **https://marketing.perl6.org/** The site contains marketing materials for the language as well as a way to make a request for new materials you need.

Authors

The following people contributed to this version of the language, including making documentation updates and contributing work to known open-source compiler implementations. The list is ordered alphabetically.

If you believe your name has been erroneously omited, please contact us (https://perl6.org/irc), and we'll update the primary copy of this list.

0racle, A. Sinan Unur, ab5tract, Adrian White, Ahmad M. Zawawi, Alberto Luaces, Aleks-Daniel Jakimenko-Aleksejev, Alex Chen, Alex Elsayed, Alex Schroeder, Alex Wander, Alexander, Alexey Melezhik, Alexis, Alexius Korzinek, allan, Altai-man, Amir Aharoni, andreoss, Andrew Ruder, Andrew Shitov, Andy Weidenbaum, Anthony Parsons, Antonio, Antonio Quinonez, antoniogamiz, Armand Halbert, Arne Skjærholt, Athos Ribeiro, Bahtiar `kalkin-` Gadimov, Bart Wiegmans, Bartosz Janus, bazzaar, Ben Davies, benji, Benny Siegert, Bill Barry, bitrauser, Brad Gilbert, Breno G. de Oliveira, Brent Laabs, brian d foy, Brian Duggan, Brian Gernhardt, Brian S. Julin, Brock Wilcox, Bruce Gray, bspecht, Cale, Carl Mäsak, CC, Cecilia, cfa, Christian Bartolomäus, Christian Sánchez, Christian Walde, Christopher Bottoms, chromatic, Claudio Ramirez, Clifton Wood, Coleoid, COMBORICO, coypoop, cpin, Cuon Manh Le, Curt Tilmes, cygx, Dabrien 'Dabe' Murphy, Dagfinn Ilmari Mannsåker, Dale Evans, Dan Book, Dan Kogai, Dan Miller, Dan Zwell, Daniel Dehennin, Daniel Green, Daniel Mita, Daniel Perrett, dariusantia, Dave Olszewski, Dave Rolsky, David Brunton, David H. Adler, David M. Cawthon, David Warring, dmaestro, Dmitri Iouchtchenko, Dominique Dumont, Donald Hunter, Douglas Jenkins, Douglas L. Schrag, Douglas Schrag, dugword, eater, Eckhart Arnold, Eike Frost, Elena Merelo, Elise, Elizabeth Mattijsen, Emeric54, Eric de Hont, Fernando Correa de Oliveira, Fernando Santagata, fireartist, flussence, Francis Grizzly Smit, Francois Perrad, francois@<redacted>, Fritz Zaucker, Fyodor Sizov, Gabor Szabo, Garrett Goebel, Geoffrey Broadwell, gerd, Gerd Pokorra, gotoexit, Greg Donald, Harrison Chienjo, holli-holzer, ianmcb, ijneb, ilyash-b, Innokenty Shniperson, Itsuki Toyota, İsmail Arılık, Jack Kuan, Jake Russo, James (Jeremy) Carman, Jan-Olof Hendig, Jared Miller, Jarkko Haapalainen, Jason Cole, Jeff Linahan, Jeffrey Goff, Jeremy Studer, Jim Davis, Jimmy Zhuo, jjatria, Joachim Durchholz, Joel, Joelle Maslak, John Gabriele, John Harrison, John Spurr, Jonas Kramer, Jonathan Beebe, Jonathan Scott Duff, Jonathan Stowe, Jonathan Worthington, Josh Soref, José Albert Cruz Almaguer, Juan Julián Merelo Guervós, Juerd Waalboer, Julien Simonet, Justin DeVuyst, karl yerkes, kjpye, KlappeZuAffeTot, Kris Shannon, Lance Wicks, Larry Wall, LE Manh Cuong, lefth, LemonBoy, Leon Timmermans, Lichtkind, LLFourn, Lloyd Fournier, Luca Ferrari, Lucas Buchala, Luis Balderas Ruiz, Luis F. Uceta, M, M. Faiz Zakwan Zamzuri, Maik Hentsche, Marc Chantreux, Marcel Timmerman, Mario, Mark Montague, Mark Rushing, Martin Barth, Martin Dørum Nygaard, Martin Ryan, Mathieu Gagnon, Matt Oates, Matthew Wilson, Matthias Bloch, mendel, Michael D. Stemle, Jr, Michal Jurosz, Mint, Moray, MorayJ, Moritz Lenz, mryan, Nacho Mas, Nadim Khemir, Naoum Hankache, Nat, Neil Shadrach, Nelo Onyiah, neuron, Nic, Nic Q, Nick Logan, Norbert Buchmueller, Nova Patch, nsvedberg, Nuno 'smash' Carvalho, okaoka, Oleksii Varianyk, Panu Ervamaa, parabolize, Patrick R. Michaud, Patrick Sebastian Böker, Patrick Sebastian Zimmermann, Patrick Spek, Patrick Zimmermann, Paul Cochrane, Paul Smith, Paula, Paweł Murias, Pepe Schwarz, Peter Stuifzand, Philippe Bruhat (BooK), Piers Cawley, Prakash Kailasa, Przemysław Wesołek, Páll Haraldsson, Rafael Schipiura, raiph, Randy Lauen, raydiak, Reini Urban, Reko98, ribbon-otter, Richard Hainsworth, Rob Hoelz, Robert Lemmen, Robert Newbould, Ronald Schmidt, ryn1x, Salvador Ortiz, Salve J. Nilsen, Sam Morrison, Sam S, Samantha McVey, sarna, seatek, Shlomi Fish, Siavash Askari Nasr, Simon Proctor, Simon Ruderich, Skarsnik, smls, Stefan Fischer, Stefan Seifert, Sterling Hanenkamp, Steve Bertrand, Steve Mynott, Stéphane Payrard, Sylvain Colinet, sylvarant, Tadeusz “tadzik” Sośnierz, thebooort, Thor Michael Støre, threadless-screw, thundergnat, Tim Smith, Timo Paulssen, Tobias Boege, Tobias Leich, Tom Browder, tomboy64, Tommy Stanton, Trey Harris, Tzu-Li "tison" Chen, Tzu-Li Chen, Valentin Anger, ven, Viacheslav Lotsmanov, vinc17fr, Vladimir Marek, Vynce Montgomery, VZ, Wenzel P. P. Peppmeyer, Will "Coke" Coleda, wukgdu, Zak B. Elep, Zoffix Znet, ZzZombo, 唐鳳, 陈梓立

The Future

Now that 6.d release is done, look forward to 6.e release in the future. It will likely occur sometime in 2020, but we're not in a rush.

If you have any questions about this or future language releases, you can inquire at our Help Chat at **https://perl6.org/irc** or on our mailing list at **perl6-compiler@perl.org**

Happy Diwali!