Perl6

Now it is released, is it worth looking at ?

Perl 6 introduces lots of new features many of which you will never need

Fortunately you can learn and use a subset of Perl 6 that meets your typical needs

Things we have in Perl5 which will just be better in Perl6

- better threading
- better garbage collection
- much better foreign function interface (cross-language support)
- full Unicode processing support
- string processing on various Unicode levels, including grapheme level
- a built-in switch statement
- Multiple versions of a module can be installed and loaded simultaneously
- System administration much simpler
- Fewer lines of code allow for more compact program creation
- Huffman-coding of names allows for better readability

New Perl 6 Features

- Extensible grammars for parsing data or code (which Perl 6 uses to parse itself)
- Subroutine and method signatures for easy unpacking of positional and named parameters, and data structures
- Multimethods
- coroutines
- continuations
- Currying
- Signatures
- Captures
- Exceptions
- Built in OO
- Functional programming primitives, {lazy and eager} evaluation
- junctions, autothreading, hyperoperators (vector operators)
- Advanced introspection and meta-programming
- language-level macros* Module aliasing and versioning, optional static/gradual typing
- Garbage collection
- Greatly-improved foreign function interface
- Optional Typing System
- Interfacing to external libraries in C / C++ is trivially simple with NativeCall.
- Interfacing with Perl 5 (CPAN) / Python modules is trivially simple with Inline::Perl5 resp. Inline::Python.*
- Perl 6 runs on a variety of back-ends

That's a lot of new stuff but what makes it worthwhile to change ?

In my opinion there are two features in Perl 6 which are not available in Perl5 which make it worth looking at :-

- 1. Enabling you to generate different backends
- 2. Using Grammars

Using different backends

One of the design features of Perl6 is to be able to compile Perl6 code using different backends enabling you to generate code to run on MoarVM, JVM or Javascript VM

Unfortunately only the MoarVM backend is working with the rakudo compiler at the moment but when it is workign you wil be able to produce java classes from perl6 code. To build Java classes, all you will need to do is build the jvm backend to produce the perl6-j compiler and then tell it to produce a class file for your perl6 code using the -target=classfile switch

To build the jvm backend it is easiest to build perl6 using rakudobrew which is similar to perlbrew but for perl6

i.e. git clone https://github.com/tadzik/rakudobrew.git

export PATH=\$PWD/rakudobrew/bin:\$PATH

rakudobrew init

rakudobrew build jvm

rakudobrew switch jvm

perl6-j --target=classfile --output=MyClass.class <your perl6 script>



Grammars are a powerful tool used to destructure text and often to return data structures that have been created by interpreting that text. For example, Perl 6 is parsed and executed using a Perl 6-style grammar.

The grammar to parse JSON is only 39 lines long

use v6; unit grammar JSON::Tiny::Grammar;

```
token TOP {\s^* < value > \s^*}
rule object { '{' ~ '}' <pairlist> }
rule pairlist { <pair> % \setminus, }
rule pair { <string> ':' <value> }
rule array { '[' ~ ']' <arraylist> }
rule arraylist { <value> * % [ \, ] }
proto token value {*};
token value:sym<number> { '-'?
                           [0|<[1..9]><[0..9]>*]
                           [\. <[0..9]>+]?
                           [ <[eE]> [\+|\-]? <[0..9]>+ ]? }
token value:sym<true> { <sym> };
token value:sym<false> { <sym> };
token value:sym<null> { <sym> };
token value:sym<object> { <object> };
token value:sym<array> { <array> };
token value:sym<string> { <string> }
token str { <-["\\\t\n]>+}
token str escape { <["\\/bfnrt]> | 'u' <utf16 codepoint>+ % '\u'}
token utf16_codepoint { <.xdigit>**4}
```