☺ A retrospective on Pugs ☺

Ingo Blechschmidt
`<iblech@speicherleck.de>`

Abstract. "Hi. Today I have started working on specifying and implementing Featherweight Perl 6 (FP6), a side-effect-free subset of Perl 6." Audrey Tang used these words to unveil the Pugs project in February of 2005. Initially conceived as an implementation of a small subset of Perl 6 in Haskell, the project quickly grew to contain a full-fledged compiler and interpreter for Perl 6 and attracted a large and diverse community.

The talk will give a subjective survey of the history of Pugs. We will pay particular attention to the special manner with which Audrey led the project and what the philosophy "-O*fun*" meant to the developers. We'll also discuss which parts of Pugs were absorbed into other implementations of Perl 6 and which influence Pugs had on the Perl and Haskell communities.

About me. I contributed to Pugs as a school student in 2005, at first by porting modules and writing tests, then gradually also by writing Haskell code and later by implementing a JavaScript backend. Audrey and the unique spirit in the Pugs community had a strong and lasting influence on me (exposing me to Haskell, category theory, and a beautiful way of tending communities); I look back on very exciting and fun days.

Warning. The account is mostly from memory and not properly researched. Try not to trust it! Also note that the timeline covers only the year 2005 and that *the code excerpts are edited for legibility*, i. e. shortened at a few places and not reproduced verbatim. Pull requests are very much welcome; in fact, in good old Pugs spirit you will get full write access to the repository so you can merge them yourself.

`https://github.com/iblech/talk-pugs-retrospective`

1. A glimpse of Perl 6

2. Timeline of Pugs development
   - The beginning
   - The early days
   - Further highlights
   - The end

3. Pugs's unique culture
   - Test-driven development
   - Transparency
   - Optimizing for fun

4. Lasting contributions of Pugs

A glimpse of Perl 6

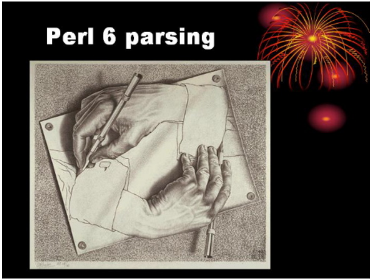
```
if all(@age) > 42 or $name eq "Curry"|"Schoenfinkel" {
    say "All good!";
}

for 17..41 -> $i {
    ...;
}

say "I was compiled ", time - BEGIN { time },
    " seconds ago.";

say @foo.map:{ $_**2 }.sort:{ abs($^a) <=> abs($^b)
```

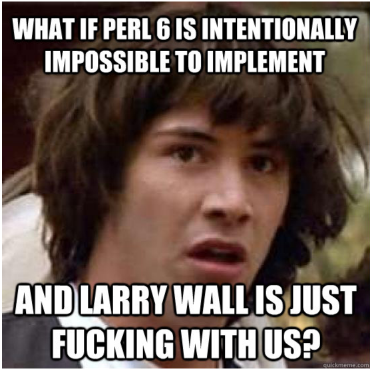
A glimpse of Perl 6

```
class Fido {
    is Dog;

    has Str $name;
    has Person $owner is rw;

    method bark (Int $times) {
        say "Wuff!" for 1..$times;
    }
}
```

A glimpse of Perl 6

closures • anonymous types • roles and traits • named arguments • expressive subroutine and method signatures • a strong meta object system • macros • state variables • cleaned up regular expressions • named regexes for easy reuse • grammars for parsing • lazy lists • junctions of values • optional type annotations (gradual typing) • powerful run-time multi dispatch • lexical imports


Perl 6 parsing


WHAT IF PERL 6 IS INTENTIONALLY IMPOSSIBLE TO IMPLEMENT
AND LARRY WALL IS JUST FUCKING WITH US?
quickmeme.com

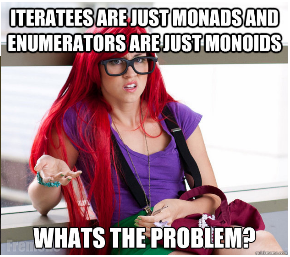
The beginning

> **“Hi. Today I have started working on specifying and implementing Featherweight Perl 6 (FP6), a side-effect-free subset of Perl 6.”**
>
> – Audrey Tang, February 2nd, 2005, link

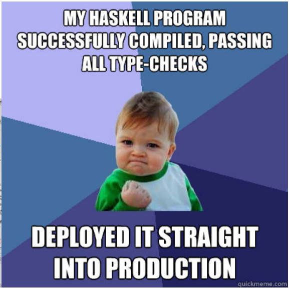
2001	Apocalypse 1, first Perl 6 design document
2004	Many Apocalypse and Synopse documents
2005	Active discussion on perl6-language@perl.org
2005	Pugs, providing the first usable implementation

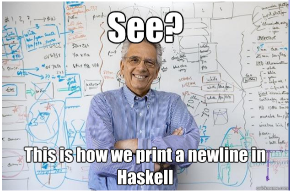
2005	Facebook
2005	YouTube
2008	GitHub

Haskell?

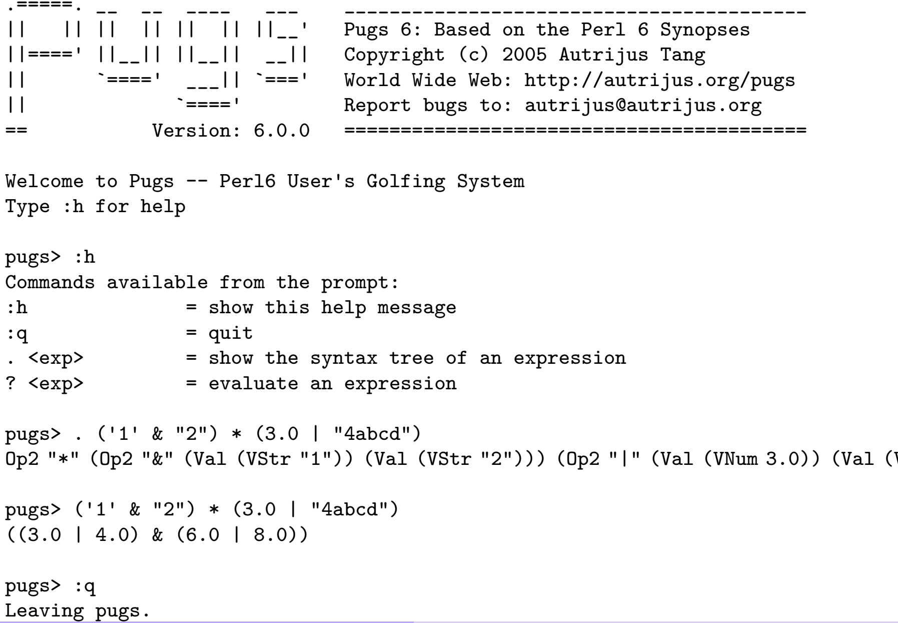

ITERATEES ARE JUST MONADS AND ENUMERATORS ARE JUST MONOIDS
WHATS THE PROBLEM?
quickmeme.com

HASKELL MOTHERFUCKER
DO YOU SPEAK IT?
Troll.me


MY HASKELL PROGRAM SUCCESSFULLY COMPILED, PASSING ALL TYPE-CHECKS
DEPLOYED IT STRAIGHT INTO PRODUCTION
quickmeme.com


See?
This is how we print a newline in Haskell
quickmeme.com

Screenshot



```
.=====. __  __  ____   ___    -----------------------------------------
||   || ||  || ||  || ||__'   Pugs 6: Based on the Perl 6 Synopses
||====' ||__|| ||__||  __||   Copyright (c) 2005 Autrijus Tang
||      `===='  ___|| `==='   World Wide Web: http://autrijus.org/pugs
||             `===='         Report bugs to: autrijus@autrijus.org
==           Version: 6.0.0   =========================================

Welcome to Pugs -- Perl6 User's Golfing System
Type :h for help

pugs> :h
Commands available from the prompt:
:h              = show this help message
:q              = quit
. <exp>         = show the syntax tree of an expression
? <exp>         = evaluate an expression

pugs> . ('1' & "2") * (3.0 | "4abcd")
Op2 "*" (Op2 "&" (Val (VStr "1")) (Val (VStr "2"))) (Op2 "|" (Val (VNum 3.0)) (Val (V

pugs> ('1' & "2") * (3.0 | "4abcd")
((3.0 | 4.0) & (6.0 | 8.0))

pugs> :q
Leaving pugs.
```

- "As I'm finding my way through TaPL and ATTaPL today, it occurs to me that I should implement a real language as an exercise; that real language turns out to be Perl 6. So it begins …"

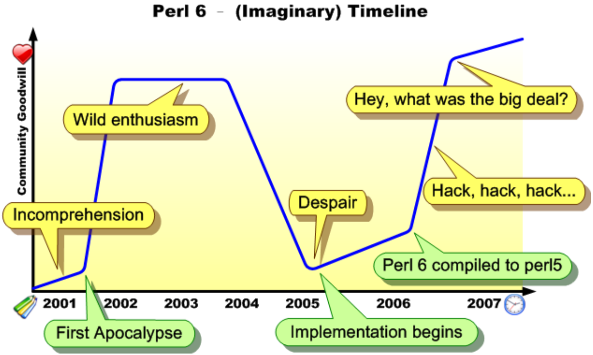
  – Audrey Tang, February 1st, 2005, link

- TaPL refers to *Types and Programming Languages*, a computer-science book on implementing programming languages.
- Audrey quickly dropped her initial plan to focus on a side-effect-free subset (on day 3, to be precise).
- There was an implementation before Pugs, a tiny project sitting in the Parrot source tree.
- Audrey's first post contained a language semantics question. This would be the beginning of continuing refinements of the specification, made possible by the existence of a useful Perl 6 platform which allowed people to play with Perl 6.

- "This last year, we were starting to lose our sense of fun in the Perl community. Though we tried to be careful about not making promises, everyone knew in their hearts that five years is an awfully long time to wait for anything. People were getting tired and discouraged and a little bit dreary.

  Then [Audrey Tang] showed up."

  – Larry Wall, August 2005 (OSCON), link



#haskell on 2009-04-17, discussion about (AT)TaPL, link:

⟨**edwardk**⟩ shapr recommended it to audreyt and audrey went off and wrote pugs to work through the processes in the book

So, thank you shapr!

The early days

Day 4	“Parsec; 90 % of operators implemented!”	see code
Day 8	“Pugs 6.0.2; Turing Completeness”	see code
Day 12	“Refactoring”	see code
Day 13	“Continuations”	see code
Day 14	“All tests successful”	see code
Day 20	“6.0.8”	see code
Day 23	“Test.pm flies!”	see code

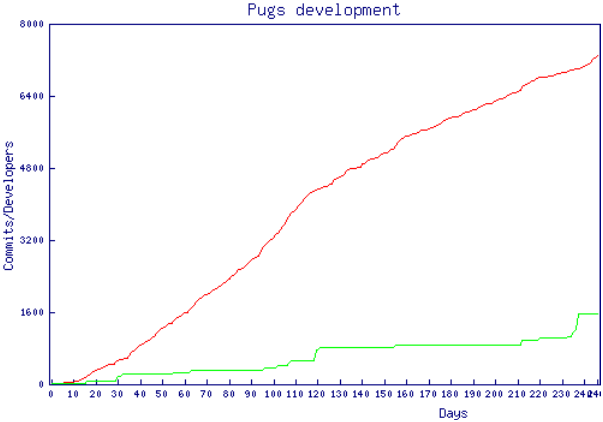
skip details

As can be seen by the journal headlines, Pugs was extremely fast-moving. In the early days, a Pugs build of a day ago was considered to be vastly outdated. This had two reasons:

- Firstly, Audrey was an astoundingly productive hacker, rapidly implementing major features and attracting a large community of fellow *lambdacamels* (totaling more than 200 contributors).

- Secondly, Haskell's unique features rendered it a very fine language for implementing other languages. It allowed both for quick prototyping and easy refactoring.

See the logs of #perl6 for 2005-03-02 (link) for an interesting chat between Audrey and chromatic, among other things about Pugs's beginnings and Haskell's strengths. Highly recomended reading.



- This graph was made at the release of Pugs 6.2.10. The red curve shows the absolute number of commits, the green one shows the number of committers – multiplied by 10.

- This graph is by far not the nicest graph in the history of graphs.

Day 4: “Parsec; 90 % of operators implemented!”

```
-- src/AST.hs at revision 7
data Exp
    = App String [Exp]
    | Syn String [Exp]
    | Prim ([Val] -> Val)
    | Val Val
    | Var String SourcePos

data Val
    = VUndef
    | VBool  Bool
    | VList  [Val]
    | VNum   Double
```

Continue slides …

Day 4: “Parsec; 90 % of operators implemented!”

```haskell
-- src/Parser.hs at revision 7
parseVar = lexeme $ do
    sigil <- oneOf "$@%&"
    name  <- many1 (alphaNum <|> char '_')
    return $ Var (sigil:name)
```

```haskell
-- src/Prim.hs at revision 7
op2 :: Ident -> Val -> Val -> Val
op2 "*" = op2Numeric (*)
op2 "~" = op2Str (++)

op2Str f x y = VStr $ f (vCast x) (vCast y)
```

Continue slides …

- To read the Haskell snippets, one needs to know two peculiarities of Haskell's syntax. Firstly, function application is written by juxtaposition, without parentheses – "`f x y`" instead of "`f(x, y)`". Secondly, a dollar sign is like a big opening parenthesis. The corresponding closing parenthesis is implicitly at the end of the line or block – that is "`foo $ a long expression`" instead of "`foo (a long expression)`".

- An English reading of the presented parser snippet goes as follows. "First, try to read one of the characters `$@%&`. Then try to read many, at least one, alphanumerical characters (or underscores). If that worked, prepend the sigil to the variable name (using the `:` operator), and pack the result into an abstract syntax tree."

- We used the magic of *parser combinators* (in the form of the Parsec library) to build the parser. Their central idea is to provide higher-order functions (like `many1`) to construct complex parsers from simple building blocks (like `alphaNum` or `char`). This style allows for parsers which are easy to read, write, and maintain. Also, the parsing rules may dynamically depend on arbitrary runtime values; this flexibility is necessary to support user-defined operators and BEGIN blocks executing at parse time. An important difference to the regular expressions of Perl 5 is that while parsing, one can build a complex syntax tree.

Day 8: “Pugs 6.0.2; Turing Completeness”

User-defined subroutines, variable binding, …

```haskell
-- src/Eval.hs at revision 7
reduce env@Env{ cxt = cxt } exp@(Syn name exps)
    | name `isInfix` ";"
    , [left, right] <- exps
    , (env', exp)   <- runStmt "Any" env  left
    , (env'', exp)  <- runStmt cxt   env' right
    = (const env'', exp)
    | name `isInfix` ":="
    , [Var var _, exp] <- exps
    , (fenv, Val val)  <- reduce env exp
    = (combineEnv fenv var val, Val val)
```

Continue slides …

- In line 5, the code to the left of the semicolon operator is executed in the (Perl) context `Any` and in the environment `env`.

- This results in some changed environment `env'` and a value `exp` (which is ignored).

- In line 6, the code to the right of the semicolon operator is executed in the context `cxt` of the whole expression, resulting in a further updated environment `env''` and a value `exp` (this shadows the earlier declarations of `exp`).

Day 12: “Refactoring”

eval(), rand(), …

```
-- src/Prim.hs at revision 19
op1 "rand" e = \(x :: VNum) -> VNum $
    unsafePerformIO $ getStdRandom
        (randomR (0, if x == 0 then 1 else x))
```

Continue slides …

Day 13: “Continuations”

```haskell
-- src/Monads.hs at revision 23
data Env = Env { envContext :: Cxt
               , envPad     :: Pad
               , envEval    :: Exp -> Eval Val
               , envCC      :: Val -> Eval Val
               , envBody    :: Exp
               , envID      :: Unique
               }
```

```haskell
-- src/AST.hs at revision 27
type Eval a = ContT Val (ReaderT Env IO) a
```

Continue slides …

Day 14: “All tests successful”

```
# t/01basic.t at revision 32
use v6;

say "1..2";
say "ok 1 # Welcome to Pugs!";

sub cool { fine($_) ~ " # We've got " ~ toys }
sub fine { "ok " ~ $_ }
sub toys { "fun and games!" }

say cool 2;  # and that's it, folks!
```

Continue slides …

Day 20: “6.0.8”

Hashes, pairs, many IO primitives, …

```
sub { $_ ?? $_ * &?SUB($_ - 1) :: 1 }.(10).say
# 3628800
```

Continue slides …

Day 23: “Test.pm flies!”

First Perl 6 module, more than 400 unit tests, …

```
module Test-0.0.1;
use v6;

my $loop = 0;

sub ok (Bool $cond, Str ?$desc) is export {
    my $ok  := $cond ?? "ok " :: "not ok ";
    my $out := defined($desc)
        ?? (" - " ~ $desc)
        :: "";
    $loop++;
    say $ok, $loop, $out;
}
```

Continue slides …

Further highlights

Day 47	“Perl 5 regular expressions landed.”	see code
Day 50	Compiling to Parrot	
Day 69	Autovivification, tied magic, slice assignment	see code
Day 85	BEGIN blocks	see code
Day 87	“STM: Atomic power!”	see code
Day 88	“A shiny new monad.”	see code
Day 99	“Full named rules!”	see code
Day 100	“OO support landed!” (a first sketch)	
Day 107	User-defined operators, Inline::Pugs	see code
Day 108	svnbot (announcing commits to #perl6)	see code
Day 109	gather/take	see code
Day 111	Hyper operators	see code

Further highlights, cont'd

Day 113	"Pugs runs CPAN modules!"	see code
Day 117	evalbot (using a new safe mode)	
Day 128	Academic paper for the Haskell Workshop 2005, link (rejected, but still recommended reading)	
Day 162	"Say hello to P5ugs!"	
Day 164	"Perl 6 compiled to… JavaScript!"	see code
Day 166	"Mandel.p6 on JavaScript."	
Day 177	"JS backend 43% pass"	
Day 193	"JSAN and CPAN, here we come!"	
Day 219	Perl 6 on JavaScript passes 91%.	

Timeline only covers 2005 and has several highlights omitted.

skip details

Day 47: “Perl 5 regular expressions landed.”

```
"(balanced)" ~~ rx:perl5{^(\()?[^()]+(?(1)\))$};
# bool::true

"(balanced" ~~ rx:perl5{^(\()?[^()]+(?(1)\))$};
# bool::false
```

Continue slides …

“Today I asked on #perl6 which one should I do first: Perl 5 regex (via PCRE), or a Perl 6 to C compiler (via Template Haskell)? pjcj suggested that the former is more practical, so I went with it. I’m glad to report that it now works, with full `$/` as arrayref and `$0` `$1` `$2` etc capturing magic.”

Day 69: Tied %*ENV

```haskell
-- src/AST.hs at revision 1750
instance Hash.Class IHashEnv where
    fetch _ = do
        envs <- liftIO getEnvironment
        return [ (k, VStr v) | (k, v) <- envs ]
    fetchVal _ key = tryIO undef $ do
        str <- getEnv key
        return $ VStr str
    storeVal _ key val = do
        str <- fromVal val
        liftIO $ setEnv key str True
```

Continue slides …

Day 85: BEGIN blocks

```haskell
-- src/Pugs/Parser.hs at revision 2380
ruleClosureTrait :: RuleParser Exp
ruleClosureTrait = rule "closure trait" $ do
  name  <- tryChoice $ map symbol $ words "BEGIN END"
  block <- ruleBlock
  let (fun, names) = extract (block, [])
  let code = VCode { subName = name, subFun = fun }
  case name of
    -- Notice the bug here? Yes, you in the audience!
    "END"   -> return $
      App "&unshift" [Var "@*END"] [Syn "sub" code]
    "BEGIN" -> do
      rv <- unsafeEvalExp fun
      return rv
```

Continue slides …

- The parser desugared END blocks to something like `unshift @*END, { ... }`. The runtime system took care of running the blocks stored in `@*END` at the end of program execution.

- By calling `unsafeEvalExp` in the BEGIN branch, the block was executed immediately, at parse time.

- So what's the bug in the END branch?

Day 87: “STM: Atomic power!”

```
my ($x, $y) = (1, 2);
async { atomic { $x = $y * 10; $y = $x * 10 } };
async { atomic { $x = $y * 10; $y = $x * 10 } };
async { atomic { $x = $y * 10; $y = $x * 10 } };
atomic { $x = $y * 10; $y = $x * 10 };
say "$x, $y";  # always "20000000, 200000000"
```

Continue slides …

- Shared Transactional Memory (STM) is an exciting technology for writing composable threadsafe code without having to worry about race conditions.
- It was easy to expose Haskell's STM support to the user, so we did it (as an unspecced extension).

Day 88: “A shiny new monad.”

⟨obra⟩ What the hell changed in the last 12 hours with pugs?

⟨obra⟩ I mean it smoked 30 % faster.

⟨audreyt⟩ oh. yeah. I rewrote the monad

Continue slides …

Day 99: “Full named rules!”

Perl 6 rules: Named regexes that form a formal grammar.
Capture objects can be full-fledged abstract syntax trees.

```
grammar Perl6 {
    token ident    { <alpha> \w+ }

    token name     { <ident> [ '::' <ident> ] * }

    token sigil    { '$' | '@' | '&' | '%' }

    token variable { <sigil> <name> }
}
```

Continue slides …

Day 107: User-defined operators

```
sub prefix:<Σ>  (@x) { [+] *@x   }
sub postfix:<!> ($x) { [*] 1..$x }
say Σ [1..5!];  # 7260
```

```
-- src/Pugs/Parser.hs at revision 3388
tightOperators :: RuleParser [[Operator Char Env Exp]]
tightOperators = do
  infixOps <- currentInfixOps
  return $
    [ leftOps $ " " ++ unwords infixOps ++ " "
    , leftOps  " >>+<< + - ~ +| +^ ~| ~^ ?| "
    , listOps  " & "
    , listOps  " ^ | "
    , ...
    ]
```

Continue slides …

Day 108: svnbot

⟨**svnbot6**⟩ r3131 | clkao++ | for svnbot.p6:
⟨**svnbot6**⟩ r3131 | clkao++ | * Increase karma for
⟨**svnbot6**⟩ r3131 | clkao++ | author that commits.

…

⟨**osfameron**⟩ ooo, you get a ++ every time you commit!

⟨**Juerd**⟩ No, for every line in the commit message.
So flood away!

Continue slides …

Day 108: svnbot

The bot used a module Net::IRC (link), which at the time emulated true OO:

```
# ext/Net-IRC/lib/Net/IRC.pm at revision 2850
module Net::IRC-0.03;

sub new_bot (Str $nick, Str $host, ...) {
    my $self;
    ...;
    return $self = {
        connect => {...},
        join    => -> Str $channel {...},
        ...,
    };
}

# Usage: my $bot = new_bot(...); $bot<run>();
```

Continue slides …

Day 109: gather/take

```
my @foo = gather {
    for @data {
        take ... if ...;
        take ... if ...;
    }
};
```

Continue slides ...

Day 109: gather/take

```haskell
-- src/Pugs/Prim.hs at revision 3524
op1 "gather" = \v -> do
    av  <- newArray []
    evl <- asks envEval
    symTake <- genSym "@?TAKE" (MkRef av)
    enterLex [symTake] $ evl (App (Val v) [] [])
    fmap VList $ readIVar av

op1 "take" = \v -> do
    lex <- asks envLexical
    arr <- findSymRef "@?TAKE" lex
    op2 "push" (VRef arr) v
```

```perl
my @foo = gather {
    for @data {
        take ... if ...;
        take ... if ...;
    }
};
```

Continue slides ...

- In plain English: A `gather` call creates an internal lexical variable `@?TAKE`. Calling `take` pushes to this array.

Day 111: Hyper operators

```
(1,2,3) >>-<< (4,5,6) # (-3, -3, -3)

-<< (4,5,6)           # (-4, -5, -6)

sub postfix:<!> { [*] 1..$_ }
(4, 5, 6) >>!         # (24, 120, 720)

[+]<< ([1,2], [3,4])  # (3, 7)

[//] @foo             # first defined value in @foo
```

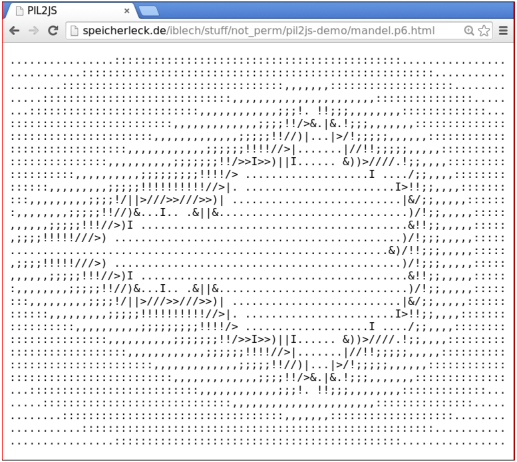
Continue slides …

Day 113: “Pugs runs CPAN modules‼”

```
# Perl 6 code, runnable in Pugs.
use Digest::SHA1--perl5;

my $cxt = Digest::SHA1.new;
$cxt.add('Pugs!');

say $cxt.hexdigest;
# 66db83c4c3953949a30563141f08a848c4202f7f
```

Continue slides …

Day 164 and others: Perl 6 on JavaScript


PIL2JS
speicherleck.de/iblech/stuff/not_perm/pil2js-demo/mandel.p6.html

```
................::::::::::::::::::::::::::::::::::::::::::::................
...........::::::::::::::::::::::::::::::::::::::::::::::::::::::...........
........::::::::::::::::::::::::::::::::::,,,,,,,:::::::::::::::::::........
.....:::::::::::::::::::::::::::::,,,,,,,,,,,,,,,,,,,,,,:::::::::::::::.....
...::::::::::::::::::::::::::,,,,,,,,,,,,;;;!. !!;;;,,,,,,,,:::::::::::::...
:::::::::::::::::::::::::,,,,,,,,,,,,,;;;;!!/>&.|&.!;;;,,,,,,,::::::::::::::
::::::::::::::::::::::,,,,,,,,,,,,,;;;;;!!//)|...|>/!;;;;;,,,,,,::::::::::::
::::::::::::::::::,,,,,,,,,,,,;;;;;;!!!!//>|.......|//!!;;;;;,,,,,::::::::::
:::::::::::::::,,,,,,,,,,;;;;;;;!!/>>I>>)||I...... &))>////.!;;,,,,:::::::::
::::::::::,,,,,,,,,,;;;;;;;;;!!!!/> ...................I ..../;;,,,,::::::::
::::::,,,,,,,,,;;;;;!!!!!!!!!!//>|. .......................I>!!;;,,,,:::::::
:::,,,,,,,,,;;;;!/||>///>>///>>)| ..........................|&/;;,,,,,::::::
:,,,,,,,,;;;;;!!//)&...I.. .&||&.............................)/!;;,,,,,:::::
,,,,,,;;;;;!!!//>)I .........................................&!!;;,,,,,:::::
,;;;;!!!!!///>) ............................................)/!;;,,,,,,:::::
..........................................................&)/!!;;;,,,,,:::::
,;;;;!!!!!///>) ............................................)/!;;,,,,,,:::::
,,,,,,;;;;;!!!//>)I .........................................&!!;;,,,,,:::::
:,,,,,,,,;;;;;!!//)&...I.. .&||&.............................)/!;;,,,,,:::::
:::,,,,,,,,,;;;;!/||>///>>///>>)| ..........................|&/;;,,,,,::::::
::::::,,,,,,,,,;;;;;!!!!!!!!!!//>|. .......................I>!!;;,,,,:::::::
::::::::::,,,,,,,,,,;;;;;;;;;!!!!/> ...................I ..../;;,,,,::::::::
:::::::::::::::,,,,,,,,,,;;;;;;;!!/>>I>>)||I...... &))>////.!;;,,,,:::::::::
::::::::::::::::::,,,,,,,,,,,,;;;;;;!!!!//>|.......|//!!;;;;;,,,,,::::::::::
::::::::::::::::::::::,,,,,,,,,,,,,;;;;;!!//)|...|>/!;;;;;,,,,,,::::::::::::
:::::::::::::::::::::::::,,,,,,,,,,,,,;;;;!!/>&.|&.!;;;,,,,,,,::::::::::::::
...::::::::::::::::::::::::::,,,,,,,,,,,,;;;!. !!;;;,,,,,,,,:::::::::::::...
.....:::::::::::::::::::::::::::::,,,,,,,,,,,,,,,,,,,,,,:::::::::::::::.....
........::::::::::::::::::::::::::::::::::,,,,,,,:::::::::::::::::::........
...........::::::::::::::::::::::::::::::::::::::::::::::::::::::...........
................::::::::::::::::::::::::::::::::::::::::::::................
```

```
use jsan:Number.Roman <to_roman>;
say to_roman(42);   # XLII
```

Continue slides …

- PIL2JS was written in Perl 5, in a style heavily influenced by Haskell. It used the Haskell part of Pugs to parse (and run BEGIN blocks) of Perl 6 code and translated the resulting PIL code (Pugs Intermediate Language) to JavaScript. Together with a runtime library in JavaScript, this allowed Perl 6 code to be run in the browser.

- Having the advantage of a later start, with more details of Perl 6's container binding semantics understood, it was the first backend to pass those tests. (See `https://github.com/perl6/roast/blob/master/S03-binding/arrays.t` for some of the binding features of Perl 6.)

- The first version was hacked together on a weekend and passed most of the basic "sanity tests", two days after putter (Mitchell N. Charity) started working on a PIL interpreter in Perl 5.

- Amazingly, a compiled version of the testsuite survived ten years of life: Run the tests yourself at `http://speicherleck.de/iblech/stuff/not_perm/pil2js-demo/`. For the bulk of the tests, in the `t/` subdirectory, you have to add the entry `31.15.67.4 m19s28.vlinux.de` to your `/etc/hosts`.

- JSAN was an attempt to bring a CPAN-like platform to JavaScript. The JavaScript backend was able to use JSAN modules.

- Foreshadowing node.js: “It is entirely possible that the JavaScript backend may prove to be the most important one. […] Indeed, if JavaScript2 does survive the standardization process, it is entirely possible that it may become the next Ruby, because writing programs that run at both client and server side is a strong motivation – the same reason to keep Pugs targetable to multiple backends.”

  – Audrey Tang, October 30th, 2005, link

The end

Beginning of 2007 Active development of Pugs stalls.

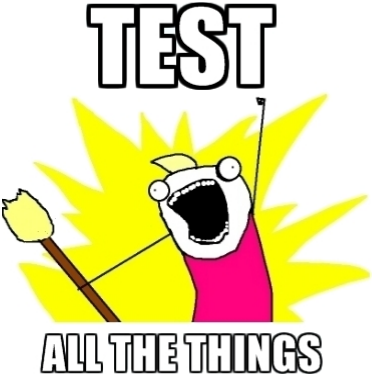
– discussion only verbally –

- For school reasons, I withdrew from Pugs development before its active phase ended. Therefore I'm not qualified to report on the reasons for its end and deem it unresponsible to speculate in public.

- Note that Pugs is still kept compatible with respect to new GHC releases.

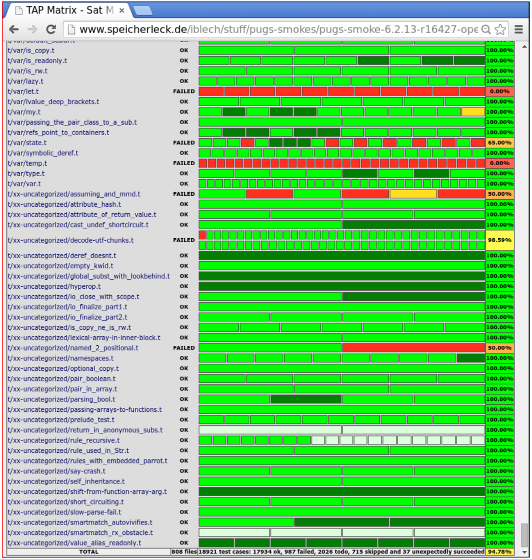
Test-driven development

- Tests as to do lists.
- Tests as bug reports.
- Tests as specification.



- One could upload test results to a public *smoke server*.

- The neat visualization (using Test::TAP::HTMLMatrix, created specifically for Pugs) was interlinked with the design documents.



TAP Matrix - Sat M ×
www.speicherleck.de/iblech/stuff/pugs-smokes/pugs-smoke-6.2.13-r16427-ope

t/var/is_copy.t	OK		100.00%
t/var/is_readonly.t	OK		100.00%
t/var/is_rw.t	OK		100.00%
t/var/lazy.t	OK		100.00%
t/var/let.t	FAILED		0.00%
t/var/lvalue_deep_brackets.t	OK		100.00%
t/var/my.t	OK		100.00%
t/var/passing_the_pair_class_to_a_sub.t	OK		100.00%
t/var/refs_point_to_containers.t	OK		100.00%
t/var/state.t	FAILED		65.00%
t/var/symbolic_deref.t	OK		100.00%
t/var/temp.t	FAILED		0.00%
t/var/type.t	OK		100.00%
t/var/var.t	OK		100.00%
t/xx-uncategorized/assuming_and_mmd.t	FAILED		50.00%
t/xx-uncategorized/attribute_hash.t	OK		100.00%
t/xx-uncategorized/attribute_of_return_value.t	OK		100.00%
t/xx-uncategorized/cast_undef_shortcircuit.t	OK		100.00%
t/xx-uncategorized/decode-utf-chunks.t	FAILED		98.59%
t/xx-uncategorized/deref_doesnt.t	OK		100.00%
t/xx-uncategorized/empty_kwid.t	OK		100.00%
t/xx-uncategorized/global_subst_with_lookbehind.t	OK		100.00%
t/xx-uncategorized/hyperop.t	OK		100.00%
t/xx-uncategorized/io_close_with_scope.t	OK		100.00%
t/xx-uncategorized/io_finalize_part1.t	OK		100.00%
t/xx-uncategorized/io_finalize_part2.t	OK		100.00%
t/xx-uncategorized/is_copy_ne_is_rw.t	OK		100.00%
t/xx-uncategorized/lexical-array-in-inner-block.t	OK		100.00%
t/xx-uncategorized/named_2_positional.t	FAILED		50.00%
t/xx-uncategorized/namespaces.t	OK		100.00%
t/xx-uncategorized/optional_copy.t	OK		100.00%
t/xx-uncategorized/pair_boolean.t	OK		100.00%
t/xx-uncategorized/pair_in_array.t	OK		100.00%
t/xx-uncategorized/parsing_bool.t	OK		100.00%
t/xx-uncategorized/passing-arrays-to-functions.t	OK		100.00%
t/xx-uncategorized/prelude_test.t	OK		100.00%
t/xx-uncategorized/return_in_anonymous_subs.t	OK		100.00%
t/xx-uncategorized/rule_recursive.t	OK		100.00%
t/xx-uncategorized/rule_used_in_Str.t	OK		100.00%
t/xx-uncategorized/rules_with_embedded_parrot.t	OK		100.00%
t/xx-uncategorized/say-crash.t	OK		100.00%
t/xx-uncategorized/self_inheritance.t	OK		100.00%
t/xx-uncategorized/shift-from-function-array-arg.t	OK		100.00%
t/xx-uncategorized/short_circuiting.t	OK		100.00%
t/xx-uncategorized/slow-parse-fail.t	OK		100.00%
t/xx-uncategorized/smartmatch_autovivifies.t	OK		100.00%
t/xx-uncategorized/smartmatch_rx_obstacle.t	OK		100.00%
t/xx-uncategorized/value_alias_readonly.t	OK		100.00%
TOTAL	808 files	18921 test cases: 17934 ok, 987 failed, 2026 todo, 715 skipped and 37 unexpectedly succeeded	94.78%

Transparency

- Audrey’s journal (link):
  documenting progress,
  spreading excitement
- Public IRC logs
- svnbot, announcing new commits
- “Private code = dead code.” “url?”

Pugs’s unique culture

~~-O2~~

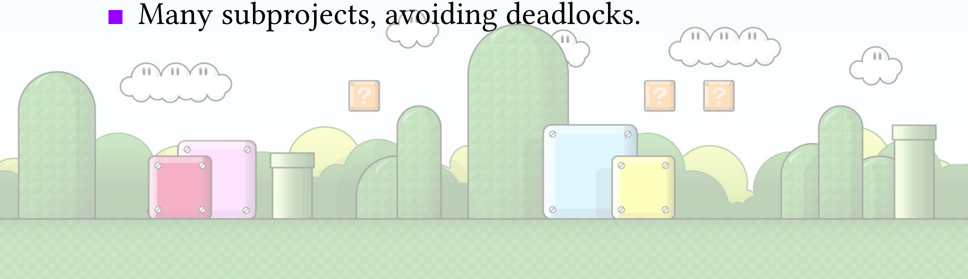
Pugs's unique culture

~~-O~~s

Optimizing for fun

-Ofun!

- Liberal granting of commit bits.
- Forgiveness > permission.
- “Imagineering”, sketching ideas with code.
- Many subprojects, avoiding deadlocks.



Optimizing for fun

-Ofun!

- Liberal granting of commit bits.
- Forgiveness > permission.
- “Imagineering”, sketching ideas with code.
- Many subprojects, avoiding deadlocks.

Audrey single-handedly bootstrapped a diverse and tight-knit community of lambdacamels, sharing a common vision and thoroughly enjoying their time.
Thank you, Audrey. ♡

- "Patches are boring, commits are fun."
- The diverse community was very welcoming. Trolls were hugged and turned into committers.
- Somebody would come to #perl6 and mention that $FEATURE does not work yet. Audrey would send them a commit bit, ask to create a test exemplifying the missing feature, and often implement the feature till the next day (or hour (or quarter of an hour)).
- People worked on the Haskell parts of the interpreter and the compiler, on unit tests, on porting and creating modules, on examples and documentation, on various backends (Perl 5 and JavaScript) written in Perl 5, on fun side projects (such as understanding type inference algorithms), and many other things.
- Audrey's slides *-Ofun: Optimizing for Fun* and Geoff Broadwell's article on O'Reilly Network are highly recommended reading.

ONE DOES NOT SIMPLY
TROLL #PERL6
imgflip.com

- “One of my goals of this project is to keep it dual-cultured. So the source tree is managed with both svk and darcs; the build system requires both Perl5 and GHC; I will submit my Apocrypha series of design documents as monthly articles to both Perl.com and The Monad Reader; the project info is on both CPAN and the Haskell Wiki; etc, etc.”

  – Audrey Tang, February 6st, 2005, link

- “In other news, Pugs was mentioned on The Haskell Sequence today. Indeed, I have noted that a significant part of questions asked in #haskell are from camelfolks. Conversely, we saw a large influx from lambdafolks to #perl6 as well. Lots of knowledge transfer is happening, which makes me really happy.”

  – Audrey Tang, February 24th, 2005, link

- Post on Lambda the Ultimate, a programming languages weblog:

  “I’m a great fan of theoretical concepts like arrows, but at the same time I’m a self-employed programmer interested in solving my clients’ problems.

  Pugs is notable in that it profitably uses recent developments such as GADTs and Template Haskell for an implementation of Perl6.

  I recently became a regular on the #perl6 irc channel and soon after joined the list of committers.

  In just a few days I’ve seen a lot. I’ve seen enthusiastic members of the Perl community learning Haskell. I’ve seen myself learning Perl. I’ve also seen how daily Perl programmers work with abstractions like monad transformers. I’ve seen how some structures are easy to extend for programmers new to both the Pugs codebase and Haskell.”

  – Shae Errison (shapr), April 5st, 2005, link

#haskell on 2013-03-23, link:

⟨**edwardk**⟩ I found Haskell first through the existence of pugs.

⟨**elliott**⟩ perl, well known for attracting category theorists

⟨**edwardk**⟩ well, i was mining category theory at the time too.

⟨**shachaf**⟩ Pugs, well known for getting people into Haskell.

Edward Kmett is well-known in the Haskell community. One of his hobbies is infusing Haskell's equivalent of CPAN with generalized abstract nonsense packages inspired by category theory, which somehow end up as dependencies for such mundane things as web frameworks.

- Other fun aspects of Pugs's culture: academic paper reading, game development, code obfuscation, Lord of the Rings poetry, other poetry (Larry was a mariner), videos (Fear and Loathing in Pugsland), hackathons (among others, with Elizabeth Mattijsen – now, of course, a prolific Rakudo contributor), …

```
-- src/Main.hs at revision 1
{-# OPTIONS -fglasgow-exts #-}

{-
    The Main REPL loop.

    A ship then new they built for him
    Of mithril and of elven-glass
    With shining prow; no shaven oar
    Nor sail she bore on silver mast;
    The Silmaril as lantern light
    And banner bright with living flame
    To gleam thereon by Elbereth
    Herself was set, who thither came...
-}
```

Lasting contributions of Pugs

- Renewed interest in Perl 6.
- Major refinements of the specification.

Lasting contributions of Pugs

- Renewed interest in Perl 6.
- Major refinements of the specification.
- **The test suite.** Approximately 20 000 unit tests.

Lasting contributions of Pugs

- Renewed interest in Perl 6.
- Major refinements of the specification.
- **The test suite.** Approximately 20 000 unit tests.
- The culture.
- Publicity for Haskell.

Lasting contributions of Pugs

- Renewed interest in Perl 6.
- Major refinements of the specification.
- **The test suite.** Approximately 20 000 unit tests.
- The culture.
- Publicity for Haskell.
- **Moose.**

#perl6 on 2005-07-20:

⟨**stevan**⟩ geoffb: I think “moose” is a private joke between nothingmuch and himself :)

#perl6 on 2006-03-06:

⟨**stevan**⟩ audreyt: I have to run (dinnertime), but I have to show you Moose.pm soon, it is my (Class::MOP based) answer to Spiffy :)

stevan is Stevan Little, nothingmuch is Yuval Kogman. Moose originated in Stevan’s work on the Perl 6 metamodel (which governs objects, classes, and metaclasses).

“It came out of the Pugs project, which is a project that in about 2005 started. It was Audrey Tang who had decided that she wanted to implement Perl 6 in Haskell. So it was sort of a very fun project. Audrey coined the term “-Ofun”, optimized for fun.

And a lot of the goal of the project was to get some juice flowing back into the Perl 6 community and really get a working or a semi-working implementation so people could play with it.

One of the things that I did in that project was to prototype the object system for Perl 6. I read over the Apocalypse 12, read up on a number of different object systems in different languages such as Smalltalk, CLOS, which is the Common Lisp Object System. Objective C’s object runtime, Ruby, Python, all those things. We were doing a lot of research at the time and we tried to put a lot of that stuff, a lot of the good ideas, into the Perl 6 object system.

And then basically as the Pugs project started to peter out, I found myself going back to my work code, which was your basic vanilla Perl 5 OO. And I really craved all the features that I had been prototyping.

So months here and months there, I fiddled around and I finally came up with a module called Class::MOP which is basically the basis on which Moose sits. And so a couple months after Class::MOP, we released Moose and sort of got running from there.”

– Stevan Little, December 2010, link

This *lambdacamel* illustration is by Carina Willbold. Feel free to use it in your talks! (CC BY)