Markdown on Parrot (40')

François Perrad
francois.perrad@gadz.org
PAUSE ID : PERRAD
Parrot SVN : fperrad

Markdown by Example

```
# Header 1

Some inline markup like _italics_, **bold**, and
   `code()`

* Bullet lists are easy too
- Another one
+ Another one

> Blockquotes are like quoted text in email replies
>> And, they can be nested

Here is a link to [Parrot](http://www.parrot.org)
```

What are the motivations ?

- PCT (Parrot Compiler Toolkit)
  - Subset of Perl 6
    - PGE : Perl Grammar Engine
    - NQP : Not Quite Perl
  - Show its power
  - Learning it (by doing)
- Lightweight Markup Languages
  - It’s about text transformation
  - Like compilation

Why choose Markdown ?

- A lot of Lightweight Markup Languages
  - See http://en.wikipedia.org/wiki/Lightweight_markup_language
- An initial implementation in Perl
- Ports in others languages
- An official syntax
  - See http://daringfireball.net/projects/markdown/syntax
- An official test suite
- An leg implementation
  - ie. close to PGE

Architectural Pattern

- Classic implementations
  - s/p1/r1/ | s/p2/r2/ | ...

- Markdown on Parrot : a Compiler
  - Parse against a grammar
  - Build a tree (AST)
  - Visit it (ie. generate an output)

A Very Small Sample

```
$ cat sample.text
Parrot *speaks your* language

$ parrot markdown.pbc sample.text
<p>Parrot <em>speaks your</em>
  language</p>
```

Grammar fragments (PGE)

```
token Emph {
    | <EmphStar> {*}              #= EmphStar
    | <EmphUI> {*}                #= EmphUI
}

token EmphStar {
    <.OneStarOpen> [ <!OneStarClose> <Inline> ]* <OneStarClose>
    {*}
}

token OneStarOpen { <!StarLine> '*' <!Spacechar> <!Newline> }

token OneStarClose { <!Spacechar> <!Newline> <Inline> <!
   StrongStar> '*' }

token Spacechar { ' ' | \t }

token Newline   { \n }
```

Parse output

```
$ parrot markdown.pbc --target=parse sample.text
"parse" => PMC 'Markdown;Grammar' => "Parrot *speaks your* language\r\n\r\n" @ 0 {
   <Block> => ResizablePMCArray (size:1) [
      PMC 'Markdown;Grammar' => "Parrot *speaks your* language\r\n\r\n" @ 0 {
         <Para> => PMC 'Markdown;Grammar' => "Parrot *speaks your* language\r\n\r\n
            <Inlines> => PMC 'Markdown;Grammar' => "Parrot *speaks your* language" (
               <_Inline> => ResizablePMCArray (size:5) [
                  PMC 'Markdown;Grammar' => "Parrot" @ 2 {
                     <Inline> => PMC 'Markdown;Grammar' => "Parrot" @ 2 {
                        <String> => PMC 'Markdown;Grammar' => "Parrot" @ 2
                     }
                  },
                  PMC 'Markdown;Grammar' => " " @ 8 {
                     <Inline> => PMC 'Markdown;Grammar' => " " @ 8 {
                        <Space> => PMC 'Markdown;Grammar' => " " @ 8
                     }
                  },
                  PMC 'Markdown;Grammar' => "*speaks your*" @ 9 {
                     <Inline> => PMC 'Markdown;Grammar' => "*speaks your*" @ 9 {
                        <Emph> => PMC 'Markdown;Grammar' => "*speaks your*" @ 9 {
                           <EmphStar> => PMC 'Markdown;Grammar' => "*speaks your*" (
                              <Inline> => ResizablePMCArray (size:2) [
                                 PMC 'Markdown;Grammar' => "speaks" @ 10 {
                                    <String> => PMC 'Markdown;Grammar' => "speaks" (
                                 }
```

Action fragments (NQP)

```
method Emph($/, $key) {
    make $/{$key}.ast();
}

method EmphStar($/) {
    my $mast := Markdown::Emphasis.new();
    for $<Inline> {
        $mast.push( $_.ast() );
    }
    $mast.push( $<OneStarClose><Inline>.ast() );
    make $mast;
}

method String($/) {
    make Markdown::Word.new( :text( $/.Str() ) );
}
```

AST output

```
$ parrot markdown.pbc --target=past sample.text
"past" => PMC 'Markdown;Document'  {
    [0] => PMC 'Markdown;Para'  {
        [0] => PMC 'Markdown;Word'  {
            <text> => "Parrot"
        }
        [1] => PMC 'Markdown;Space'  {
            <text> => " "
        }
        [2] => PMC 'Markdown;Emphasis'  {
            [0] => PMC 'Markdown;Word'  {
                <text> => "speaks"
            }
            [1] => PMC 'Markdown;Space'  {
                <text> => " "
            }
            [2] => PMC 'Markdown;Word'  {
                <text> => "your"
            }
        }
        [3] => PMC 'Markdown;Space'  {
            <text> => " "
```

Visitor fragments (PIR)

```
.sub 'html' :method :multi(_, ['Markdown'; 'Emphasis'])
    .param pmc node
    $S1 = self.'html_children'(node)
    $S0 = "<em>"
    $S0 .= $S1
    $S0 .= "</em>"
    .local pmc code
    new code, 'CodeString'
    set code, $S0
    .return (code)
.end

.sub 'html' :method :multi(_, ['Markdown'; 'Word'])
    .param pmc node
    $S1 = node.'text'()
    $S0 = escape_xml($S1)
    .local pmc code
    new code, 'CodeString'
    set code, $S0
    .return (code)
.end
```

Test suite fragment (Perl 5)

```
use Parrot::Test tests => 1;

language_output_is( 'markdown', <<'CODE',
   <<'OUT', 'sample' );
Parrot *speaks your* language

CODE
<p>Parrot <em>speaks your</em> language</p>
OUT
```

Some Metrics

- 103 commits since Sept 2008
- Grammar (PGE) : 650 lines
- Actions (NQP) : 450 lines
- HTML Visitor & glue (PIR) : 900 lines
- 21 different Markdown nodes
- 56 tests + official test suite

Rakudo integration

```
#!/usr/bin/perl6
…
   my $markdown = q{
Title
=====
Some text (could be useful for a Wiki).

   };

   say eval($markdown, :lang<markdown>);
```

Status of Markdown on Parrot

- Basic features : OK
- Advanced features : **KO**
  - Implies a full rewrite of the grammar
- Integration with Rakudo : OK
- Code available on :
  - http://github.com/fperrad/markdown/
- An unexpected use of PCT
- TDD => ready for refactoring