New Parser

Motivation

- Parser now is not well structured, hard to extend
- Parser now is slow, because it does reduction/conversion by regular expression replacement multiple times over the whole text
- Parser is buggy by design

The easy way

The easy way would be to implement the Text_Wiki PEAR package.
Pro:

- Very well structured
- Very flexible
- Can be used to produce other output like other wiki dialects, Latex, ... via renderers
- Maintained by PEAR == used by a larger set of users and under some quality assessment (toggg)

Contra:

- does multipass reduction (rescans the input several times for every regex pattern defined)

toggg: (what is nothing to compare to the current 1.9 parser)

toggg: example from Justin Patrin (thanks : a multi wiki converter
(needs some complement to really do the whole tw syntax, but very clean)

The hard way

Make a lexer (tokenizer), a parser and renderer like you learned in computer science (i've never learned ...). I'd start with ideas of the lexer.
The lexer should be a state machine, but we should avoid to get input byte by byte. A pre-tokenizer could hack the input by whitespace. We should try to find out, what is the fastest way.
The parser should be connected to the lexer via a "pipe". Everytime a lexer finds a token, he can pass it to the parser. The parse tree should be build by php arrays.
Also the renderers can be connected to the parser. It should be possible to connect multiple renderers to a parser, what should be easy.

Pro:

- All pro's of PEAR))Text_Wiki((plus speed.

Contra:

- hard work

If i start do create a parser "the hard way", i think i'll use pear's coding styles and make it LGPL as the))Text_Wiki((PEAR package. I'll probably make it compatible to their renderers.

redflo