Evolving wikitext: Embracing incrementalism

S.Subramanya Sastry & C.Scott Ananian

Wikimedia Foundation
Please add your ideas and suggestions to the etherpad:

https://etherpad.wikimedia.org/p/wikimania2019wikitext
Parsing Team Mission

- **Input**: Advance wikitext as a language
  - Easier to write, faster to parse, less error prone
- **Output**: Make wikitext content easier to analyze
  - Expose wikitext semantics in well-specified output
- **Parsers**: Unify parsers
  - Same parser for reads as well as edits
Today’s focus

- **Input:** Advance wikitext as a language
  - Easier to write, faster to parse, less error prone
- **Output:** Make wikitext content easier to analyze
  - Expose wikitext semantics in well-specified output
- **Parsers:** Unify parsers
  - Same parser for reads as well as edits
Past experience
Example (2012)

- No breaking change (new feature): Lua templating engine
  - Nothing to break
  - Templates gradually adopted Lua
  - Wikitext-based templates still around!

( Predates current Parsing Team, but still relevant )
Example (2017)

- **Minor breaking change**: Language Converter fixes
  - Cleaned up edge cases and fixed longstanding bugs
  - Searched dumps to identify pages that could break
  - Community-led effort to fix these pages
  - ~3 months from start to finish
Example (2018)

- Big breaking change: Tidy → RemexHtml
  - HTML4 → HTML5 transition
  - Took ~3 years from start to finish
  - Could have been done faster with tighter planning but not that much more
  - Lots of QA tooling + Linting tools to aid editors
Making changes to wikitext is hard!
Constraints

- Huge corpus of revisions on wikimedia wikis
- Established workflows of editors
- All the 3rd party wikis and their content
- Wikitext-based tools (bots, gadgets, etc.)
What we know

- Big breaking changes are hard
- Significant syntax changes are especially difficult
- QA and change management tools very important!
- Hard to roll out changes quickly
- Easier to add new features than change existing ones
Incremental changes add up!
Changes in the pipeline
Some proposals

- Heredoc syntax for template uses
- Balanced templates
- Refinement / Generalization: Typed templates
- Parsing scopes for sections, lists, tables, talk page comments, etc.
- Your ideas!
Improving template uses with heredoc syntax
Example

{{tablestart|class="shiny"}}
| Hello || wiki = x
{{tableend}}

{{table|class="shiny"|{{!}}{{!}} Hello {{{!}}}{{{!}}}} wiki &#61; x}}
Example

```
{{tablestart|class="shiny"}}
| Hello | wiki = x |
{{tableend}}
```

```
{{table|class="shiny"|<<<
| Hello | wiki = x |
>>>}}
```
Benefits

● Introduces a nested parsing scope
  ○ Reduces need for escaping and makes template args easier to read, especially long args
  ○ Makes it easier to generate well-balanced output
  ○ Syntax can be useful more broadly
Improving template semantics
Templates today

- Generate wikitext fragments, not well-formed output
- Can interact with page wikitext in unexpected ways
- Implications
  - **Usability**: Hard to reason about consistently for humans
  - **Tooling**: Makes it difficult for tools to manipulate a wiki-page
  - **Performance**: Independent parsing of page chunks is “not possible”
foo {{tpl}} bar baz

foo will be bolded. Is bar going to be bolded?

Depends ...

Yes if {{tpl}} is {{1x|a}} or {{1x|"a"}} or {{1x|<b>}}, for ex.

No if {{tpl}} is {{1x|"a"}} or {{1x|"""a"""}} or {{1x|</b>}}, for ex.

The 1x template just prints its parameters
'''foo {{tpl}} bar''' baz

foo will be bolded. Is bar going to be bolded?

Let us say {{tpl}} was {{1x|'''a'''}}

Will a be bolded?

No!

Not hypothetical - editors on multiple wikis encountered something similar during Tidy replacement while fixing Linter-flagged wikitext issues.
foo {{tpl}} bar baz

foo will be bolded. Is bar going to be bolded?

Let us say {{tpl}} was {{1x|'''a'''}}

Expectation: bar and a would both be bold!

Reality: No! Only possible if wikitext had independent parsing / DOM scopes without non-local effects
Balanced templates
Draft proposal

● Templates declare how to “balance” output HTML
  ○ inline HTML, block HTML, table-cell, etc.
  ○ Opt-in!

● Parser enforces semantics at use sites
  ○ Correctness: Errors don’t leak out
  ○ Performance: When a template is edited, its output can be updated
    in pages without an expensive reparse of all those pages
Typed templates
Draft proposal

- Generalization of previous idea
  - `block, inline, table-cell` can be considered output types
  - Expand beyond HTML notion
    - string, CSS, structured data, etc.
  - Maybe expand to abstract types with which you associate other resources like javascript, styles, editing hooks, domain types, etc.
  - Might lead to template arguments beyond strings
A bunch of other changes along the way
Other ideas?

- Parsing scopes: expand “balanced template” idea
  - Apply to sections, lists, paragraphs, talk page threads, talk page replies …
- Your ideas here!
  - [https://etherpad.wikimedia.org/p/wikimania2019wikitext](https://etherpad.wikimedia.org/p/wikimania2019wikitext)
And we get to “wikitext 2.0”, one step at a time!
THANK YOU!