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 can be hard!
Constraints

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

- Hard to go to a “wikitext 2.0”
  - It becomes a big all-or-nothing gamble
- Better to make “incremental” changes one at a time
  - Changes build on each other
  - Learn and evolve a repeatable change process
Incremental changes add up over time!
Changes in the pipeline
Some proposals

- Heredoc syntax for template uses
- Balanced templates
  - Refinement / Generalization: Typed templates
- Parsing scopes for page fragments:
  - Sections, lists, tables, talk page comments, etc.
Improving template uses with heredoc syntax
{{tablestart|class="shiny"}}
| Hello || wiki = x
{{tableend}}

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

<table>
<thead>
<tr>
<th>Hello</th>
<th>wiki = x</th>
</tr>
</thead>
</table>

{{tablestart|class="shiny"}}

<table>
<thead>
<tr>
<th>Hello</th>
<th>wiki = x</th>
</tr>
</thead>
</table>

{{tableend}}

{{table|class="shiny"}}

<table>
<thead>
<tr>
<th>Hello</th>
<th>wiki = x</th>
</tr>
</thead>
</table>

<<<

<table>
<thead>
<tr>
<th>Hello</th>
<th>wiki = x</th>
</tr>
</thead>
</table>

>>>
Benefits

- Introduces a nested parsing context / 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 opt-in
  - Parser treats output as DOM, not wikitext
  - ⇒ all tags are closed within the template output

- No syntax changes
  - Article authors unaffected; only template authors affected
Draft proposal

- Templates declare how to “balance” output HTML
  - inline HTML, block HTML, table-cell, etc.
  - Parser enforces semantics at use sites
    - If inline, all block tags are stripped
    - Other such fixes. Ex: `<a>-inside-<a>` scenarios
''foo {{tpl}} bar'' baz

Let us say {{tpl}} declares balance: inline

Will bar be bolded?
YES! Always .. no matter what {{tpl}} returns

Let us say {{tpl}} generates ''a''
Will a be bolded?
YES! Always no matter what the page has
Benefits

- Independent parsing:
  - Article page & templates are decoupled
- 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: *string, CSS, structured data, etc.*
  ○ Maybe expand to abstract types with which you associate other resources like javascript, styles, editing hooks, domain types, etc.

● Implications:
  ○ Will lead to template arguments beyond strings
Other ideas?
Draft proposal

- Parsing scopes: apply “balancing” notion beyond templates to other page fragments
  - sections, lists, paragraphs, talk page threads, talk page replies …
- Main benefit:
  - Markup errors are contained to the fragment
  - Potential for performance enhancement
Your ideas!

Some possibilities to slot your ideas:
- Semantic changes
- New syntax
- Syntactic sugar for existing syntax
- Syntactic sugar for boiler-plate code

Add here: https://etherpad.wikimedia.org/p/wikimania2019wikitext
And that is how we get to “wikitext 2.0”. One step at a time!
THANK YOU!
Questions?