Updates from the Wikitext Parsing world

C. Scott Ananian
Content Transform Team
SMWCon 2023, Dec 12, 2023

WIKIMEDIA
FOUNDATION

End goal

Use Parsoid for all wikitext processing

WIKIMEDIA
FOUNDATION

End goal

Use Parsoid for all wikitext processing

Then improve wikitext!

WIKIMEDIA
FOUNDATION

Why Parsoid?

- Core parser cannot support Parsoid clients (VE, etc.)
- Parsoid's annotated HTML provides more semantic information (for editing, bots, gadgets, etc)
- Two parsers not tenable and hamstrings future feature work
- Long-term transition to manipulation of balanced DOM trees & composition of rich typed fragments

Initial focus: Start transitioning the Wikimedia cluster in 2024.

WIKIMEDIA
FOUNDATION

Parsoid for readers

- Parsoid has been in use beneath VisualEditor for about 10 years now (also CX, DT, Flow, Mobile Apps, Kiwix, etc), but:
  - Original implementation language incompatible with core
  - Original caching infrastructure incompatible with core
  - Page metadata and “uneditable” features of the page generally not supported (example: category metadata, section edit links)

- Hence: the Parser Unification project

WIKIMEDIA
FOUNDATION

Some milestones

Functionality

✓ Equivalent ParserOutput, ParserOptions interfaces

↣ Build a robust and cacheable post-processing pipeline

Rendering

✓ Support editing products, migrate from HTML4 Tidy to a HTML5 “tidier”

✓ Migrate core parser to use Parsoid HTML for media wikitext

↣ Use Parsoid for “read views”, starting with opt-in & Discussion Tools

WIKIMEDIA
FOUNDATION

Sneak Peek of the Future

- There is now an opt-in preference to use Parsoid to render articles on WMF projects.
  - Enable it on your own wikis using [[Extension:ParserMigration]]
- Not everything works yet: see [[mw:Parsoid/Parser_Unification/Known_Issues]]
- Help us find & document the *other* stuff that doesn't work yet
  - Especially related to SMW which might not be on our radar

WIKIMEDIA
FOUNDATION

Demo Time!

WIKIMEDIA
FOUNDATION

The near future

- ParserOutput unification
  - "Write-only" interface to ParserOutput that allows flexible composition
  - `::appendJsVars`, `::appendExtensionData`, etc
  - Removing `ParserOutput::getText()`
  - Moving flags from `::getText([...options…])` to `ParserOptions`

WIKIMEDIA
FOUNDATION

The near future, part 2

- MediaWiki\OutputTransform
  - More robust support of “post cache” transforms
  - Including support for caching these!
  - More support for DOM-based transformations
    (as opposed to regex manipulations of HTML as text)

EDIT flavor → READ flavor → YOUR OTHER flavors

WIKIMEDIA
FOUNDATION

Looking further

- Improvements to transclusion syntax and structure
  - Templates, parser functions, magic words, …
  - Typed/balanced results, typed/protected arguments
  - Standardizing on `{{ }}` syntax (removing `[[ ]]` hooks)
  - Page context?
- Shifts in parsing model
  - Restructured caching, more post-processing, more variant flavors
  - Separate “parse” and “compose” steps
  - Multiple async “compose” steps (proposed)
    - Abstract Wikipedia, Graph extension
- Scribunto?

WIKIMEDIA
FOUNDATION

Composition of typed fragments

- Treat transclusions as independently computed cacheable fragments
- Provide richer “output” types that plug together well
  - DOM tree, attribute pairs, structured data?
- Provide richer “parameter” types w/ good editor support
  - [[Extension:TemplateData]] is moving in this direction

WIKIMEDIA
FOUNDATION

Semantic MediaWiki Impacts

(that we know of)

WIKIMEDIA
FOUNDATION

Semantic MediaWiki Impacts

- Depending on a linear parse order will be discouraged
  - Page-order dependencies in the Variables extension
  - Metadata composition in ParserOutput
- Extending the `[[....]]` link syntax will be deprecated
  - Use `<ext>` for API stability or `{{#parserFunction}}`
  - `T204370`: uniform `{{#...}}` syntax
- Parser Function API likely to be extended/improved
- Likely other updates needed? Help us find issues early.

WIKIMEDIA
FOUNDATION

Postprocessing

- Fragments are independent, but there is often a need to do some global reconciliation
  - Simple example: sequentially numbering references
  - Try to avoid it: use content-based hashes, CSS numbering, etc
- We are building a robust “post-composition” pipeline for this
  - Try to keep it fast, but we expect it will be cacheable
- Although a lot of “sequential” behavior can be emulated this way, the code to do so usually looks different.

WIKIMEDIA
FOUNDATION

Link syntax

- SMW hooks `InternalParseBeforeLinks` to replace `[[...]]` syntax “before” the MW legacy parser sees it
  - Parsoid does not have parser stages like this
- Two solutions: (T76278)
  - Introduce a specific hook to allow look-aside on `[[...]]` syntax specifically (no current plan to do this)
  - Use the alternative `{{#...}}` syntax (much preferred; I believe this is current community consensus)
- Is linter/migration help needed?

Variables extension (T250963)

- The widely-used Variables extension uses the deprecated `InternalParseBeforeSanitize` hook.
  - Parsoid does not have parser stages like this
- It also introduces a central read/write store in the Parser, with constructs affecting all following wikitext
  - This conflicts with incremental/async parsing goals
  - But it’s not unique in that: citations, LanguageConverter, etc, also have had that property (but we’re trying to fix that)

Variables extension (T250963)

- Also: `#var_final`
- Some possible solutions:
  - Lint away `#var_final` – or make everything `#var_final`
  - Do all/most work in the final postprocessing phase
  - Forced linear parsing (T282499)
  - Come up with alternative/focused means to cache Cargo queries (is this the primary use case?)
    - https://www.mediawiki.org/wiki/Extension:Variables#Alternatives
    - Marijn van Wezel at Wikibase Solutions has a very interesting approach!

WIKIMEDIA
FOUNDATION

Separation of Presentation and Data

- Instead of interleaving computation with wikitext and formatting, split off the computation into its own thing
- Evaluate the page as presentation “in the context of data”

https://gitlab.wikibase.nl/community/arrayfunctions/

WIKIMEDIA
FOUNDATION

One page, three titles

In [[MainArticle]]:

```
{{MainArticlePresentation|{{#invoke:MainArticleData}}}}
```

In [[Template:MainArticlePresentation]]:

```
<table>
{{#af_foreach:{{{1}}} | key | value |
<tr><td>{{{key}}}</td><td>{{{value}}}</td></tr>
}}
</table>
```

In [[Module:MainArticleData]]:

```
return mw.af.export({
  “Row 1” = “First”,
  “Row 2” = “Second”,
})
```

WIKIMEDIA
FOUNDATION

Real-World Concerns

- Embedding certain wikitext constructs in transclusion arguments can be tricky.
  - Heredoc arguments should help (T114432)
- Memoizing computed data was necessary
  - Fragment composition framework tries to generalize this to allow broader memoization and reuse
- Passing structured data between transclusions
  - Richer types for parameters and results would help

WIKIMEDIA
FOUNDATION

Looking forward: Parser Function API

- Parser function API is currently quite primitive
    - Eg, no support for named or numbered arguments!
- Currently area of active work, but is hoped that we can introduce “soon” a declarative API consistent with extension tags/magic words/template data, that can allow richer input and output types (at least: “raw text” input and “DOM tree” output)

WIKIMEDIA
FOUNDATION

The Road Ahead

We’re excited about the transition to Parsoid read views on WMF projects starting in 2024, and the opportunities it will unlock: new transformation pipelines, new syntax, new composition mechanisms, & more flexible caching.

We expect to continue to work with the SMW community and others to enhance wikitext’s ability to encode meaning.

Let’s discuss!

WIKIMEDIA
FOUNDATION