The Long And Winding Road To Making Parsoid The Default MediaWiki Parser

S. Subramanya Sastry (Subbu), Parsing Team
Wikimedia Tech Talk, February 2019

Parsing Team Mission

Mission since 2016:

- **Parser “Input”**: Advance wikitext as a language
  - Easier to write, faster to parse, less error prone
- **Parser “Output”**: Make wikitext content easier to analyze
  - Expose wikitext semantics in well-specified output
- **Parsers**: Unify parsers ← **Focus of today’s talk**
  - Same parser for reads as well as edits

WIKIMEDIA
FOUNDATION

Talk Overview

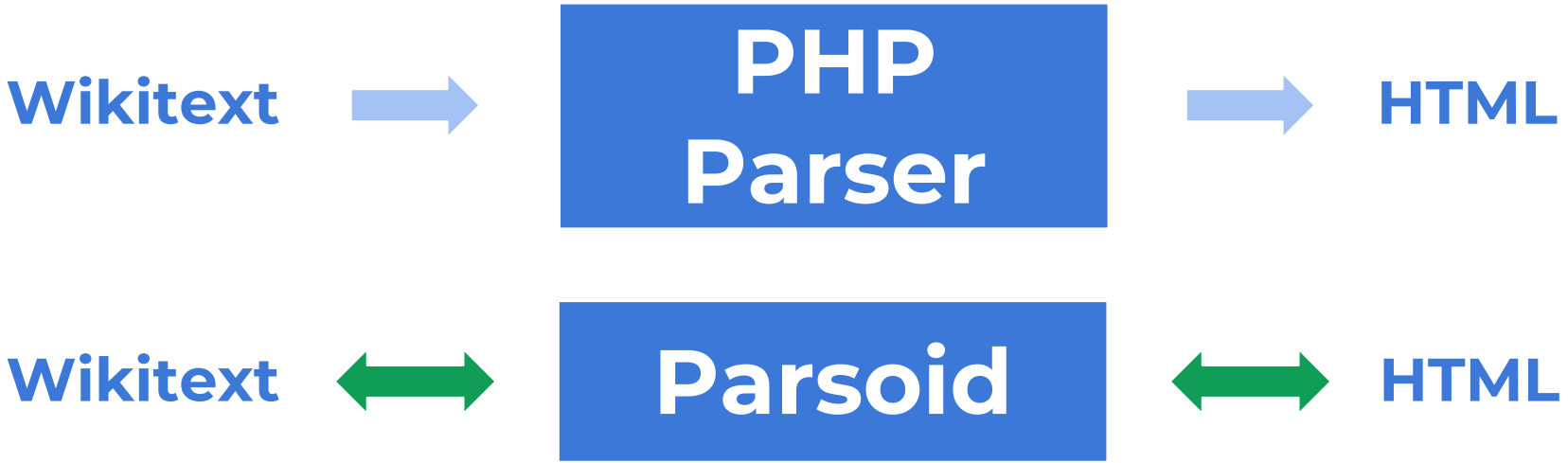
- **Part 1: Background**
  - **First things first: what are the two parsers?**
  - Setting the stage: what is the core 2-parser issue?
  - Some details: HTML, DOM, wikitext, Parsoid approach
- Part 2: Roadmap
  - How did we get here? 2011 - 2018
  - Why are we porting Parsoid to PHP?
  - How do we get to the end of this road from here?

WIKIMEDIA
FOUNDATION

First things first

WIKIMEDIA
FOUNDATION

Wikitext “parsers”



WIKIMEDIA FOUNDATION

PHP Parser: default parser (since 2003)

Used by: **Desktop view**, **Mobile web**, iOS app, Action API

WIKIMEDIA FOUNDATION

Parsoid: alternate parser (since 2012)

Used by: VisualEditor, 2017 Wikitext Editor, Flow, Content Translation, Android App, Linter Extension, Kiwix Offline Reader, Google, REST API

Written in Node.js; runs as a separate service

WIKIMEDIA FOUNDATION

Unifying parsers

All roads lead to Parsoid

- Legacy (PHP) parser cannot support Parsoid clients
- Parsoid's annotated HTML provides more information
- Two parsers not tenable and hamstrings future work

  ⇒ Make Parsoid the default MediaWiki parser

- Moving towards that since 2015
  - It has been a ***long and winding road*** …

WIKIMEDIA
FOUNDATION

Parsoid to MediaWiki ...

The long and winding road
That leads to your door
Will never disappear
I've seen that road before
It always leads me here
Lead me to your door

Imagine the Beatles singing for you here! :-)

WIKIMEDIA
FOUNDATION

Long and winding road ...

The long and winding road.jpg from commons by Leo-setä; cc-by-2.0

WIKIMEDIA FOUNDATION

The long and winding road (26438547342).jpg from commons by davebloggs007; cc-by-2.0

Long and winding road ...

The_Long_And_Winding_Road_-_panoramio.jpg from commons by Tevfik Teker; CC Attribution 3.0 Unported

WIKIMEDIA FOUNDATION

The long and winding road (37599809975).jpg from commons by davebloggs007; cc-by-2.0

Setting the stage with Aladdin's genie

WIKIMEDIA
FOUNDATION

Aladdin's genie …

LLW_Aladdin_genie.jpg from commons by Jerry Dakin; cc-by-2.0 (Legoland Windsor)

Wish: Port MediaWiki to Node.js, integrate Parsoid, no services!

Q: All code in Node.js. How many parsers will we need today to support our products?

A: ~~One~~ Two …

WIKIMEDIA FOUNDATION

Aladdin's genie …

LLW_Aladdin_genie.jpg from commons by Jerry Dakin; cc-by-2.0 (Legoland Windsor)

Wish: Port Parsoid to PHP, integrate with core, no services!

Q: All code in PHP. How many parsers will we need today to support our products?

A: ~~One~~ Two …

WIKIMEDIA FOUNDATION

Aladdin's genie …

LLW_Aladdin_genie.jpg from commons by Jerry Dakin; cc-by-2.0 (Legoland Windsor)

Wish: Port Parsoid to PHP, fix Parsoid's output and feature differences including fixing extensions

Q: How many parsers … ?

A: One … Parsoid (PHP)!

WIKIMEDIA FOUNDATION

Aladdin's genie …

LLW_Aladdin_genie.jpg from commons by Jerry Dakin; cc-by-2.0 (Legoland Windsor)

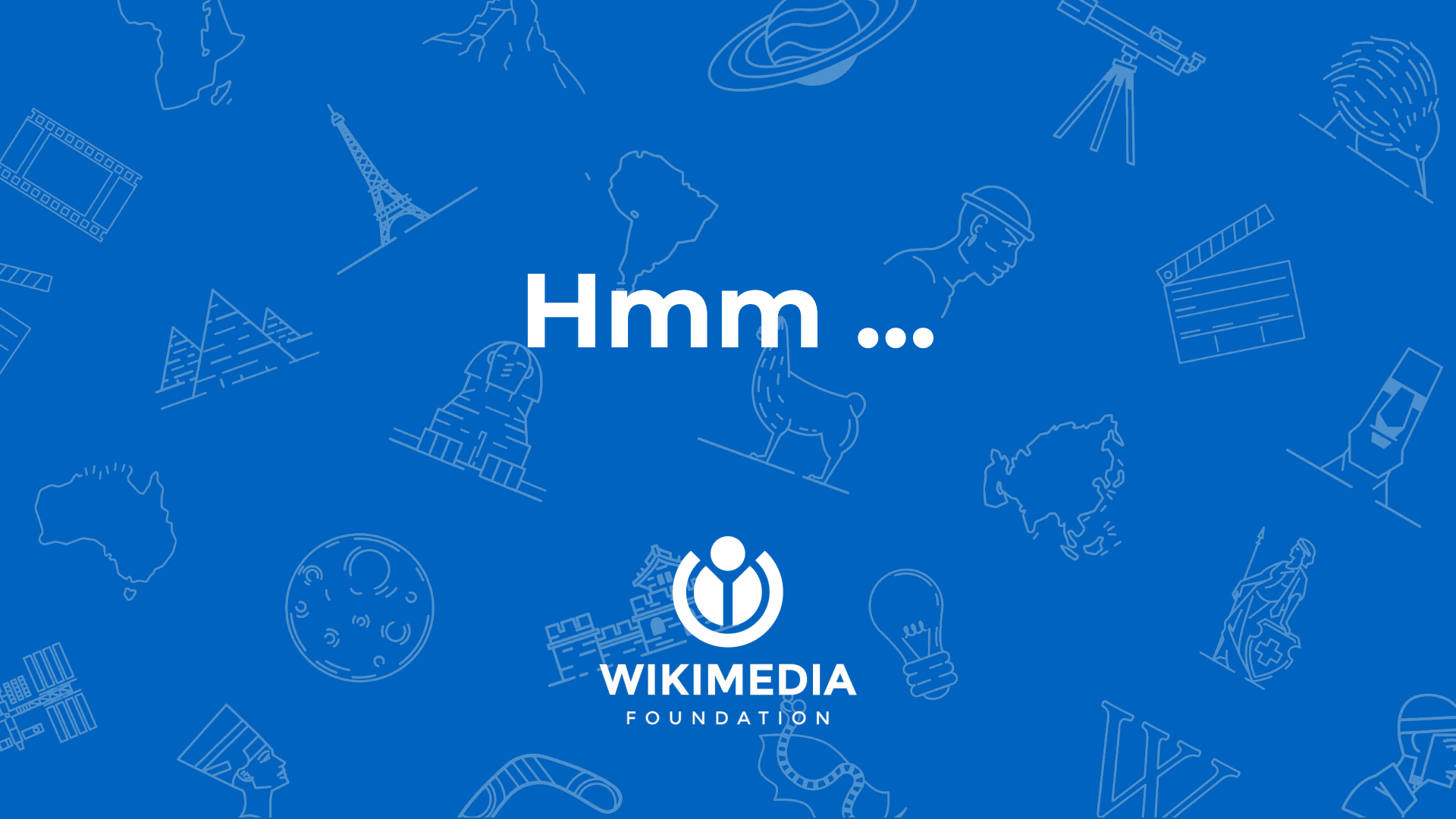
An alternate wish: Port a subset of MediaWiki core + exts to Node.js, fix Parsoid's output and feature differences including fixing extensions

Q: How many parsers …?

A: One … Parsoid (JS)!

WIKIMEDIA FOUNDATION

Hmm ...



Core two-parser issue

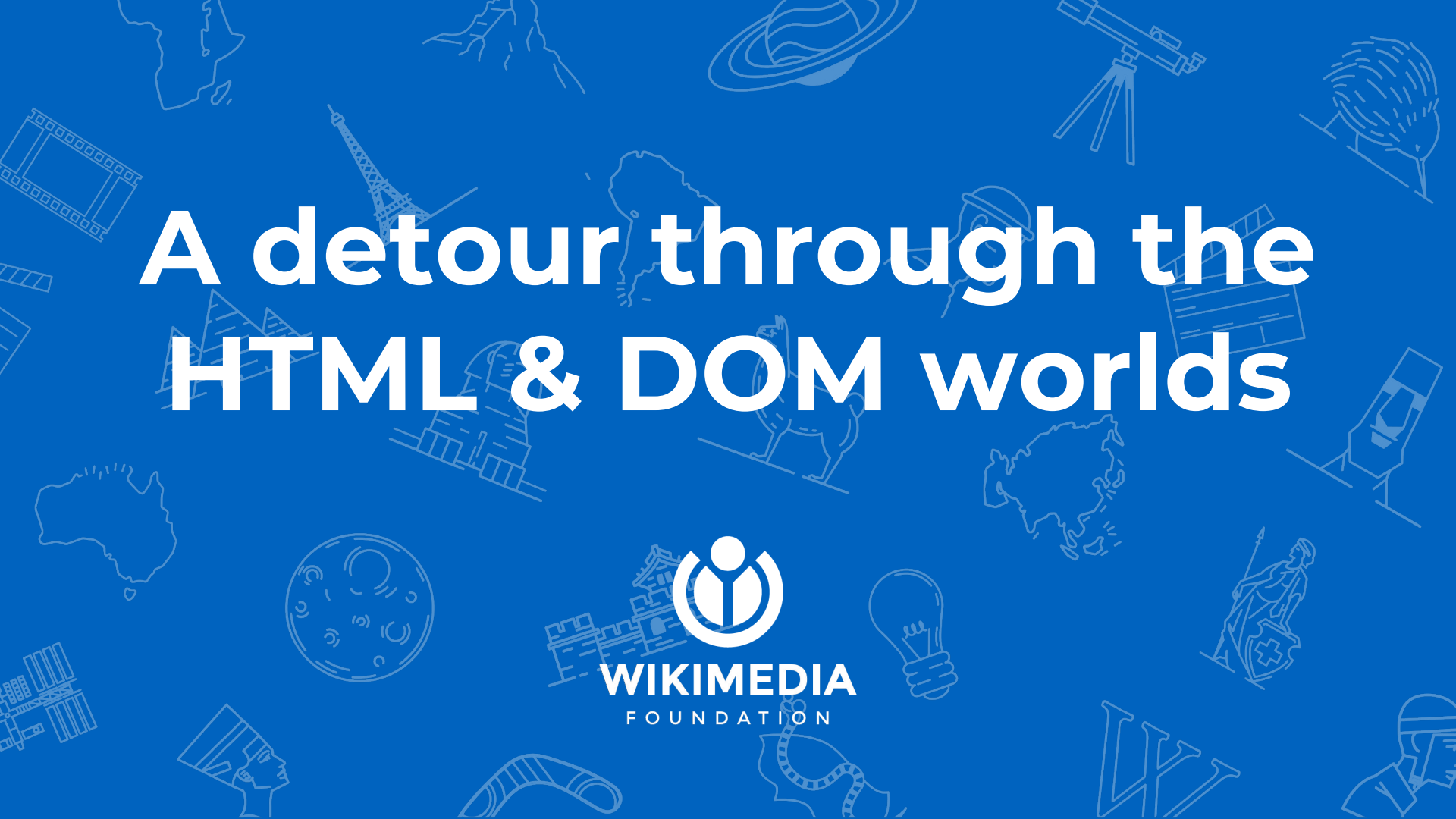
- Not language (PHP vs Node.js)
- Not service architecture (SOA vs monolithic)
- Wikitext processing:
  - string-based vs structured repn. (tokens, DOM)

Language & service architecture matter as constraints on the parser

- They affect the “twistiness and length of the road” but are not the primary reason why we have two parsers

WIKIMEDIA
FOUNDATION

A detour through the HTML & DOM worlds



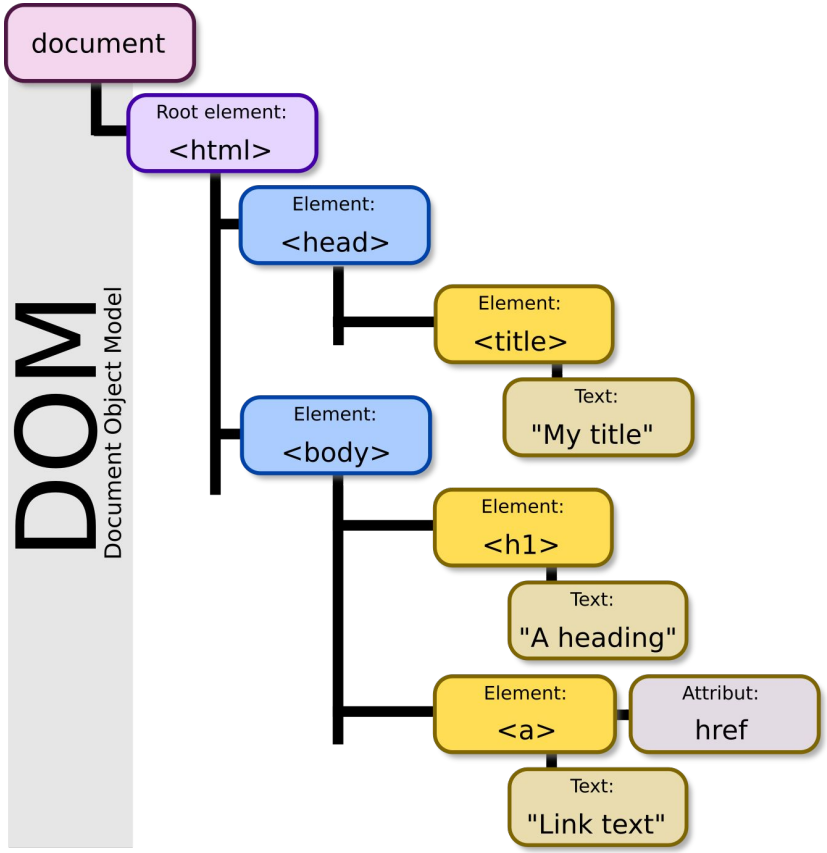
HTML & DOM

- HTML documents have structure and semantics
  - The semantics are defined by the HTML5 and DOM web standards
  - DOM = Document Object Model
  - All browsers build a DOM for web pages after parsing HTML

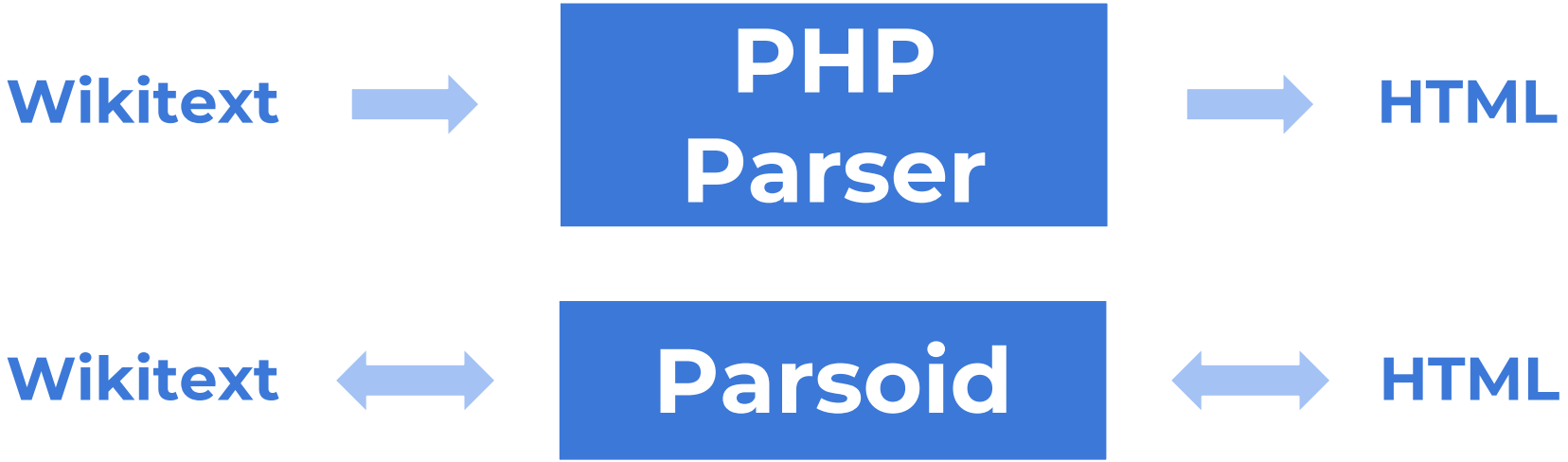
```
<html>
<head><title>My title</title></head>
<body>
<h1>A heading</h1>
<a href=”...”>Link text</a>
…
</body>
</html>
```

Query & manipulate the document

- *Give me all headings*
- *Give me the second <section>*
- *Give me the 1st column in the table with id ‘Badminton Tournaments’*
- *Replace the 1st list item in the ‘Greatest Band’ list with ‘Beatles’*



DOM-model.svg from commons by Birger Eriksson; CC-by-SA-3.0



What structure does input have?

Can you map output to input structures?

Output has DOM structure

WIKIMEDIA FOUNDATION

Why does this matter?

- Wikitext is the authoritative representation of a wikipage
  - HTML is a non-canonical representation
  - You cannot answer queries or manipulate the page via HTML without a mapping between input and output formats

Q: *Give me the infobox content and its parameters.* **A:** *No*

Q: *Give me all citations that come from templates.* **A:** *No*

Q: *Can you replace this infobox with the other infobox?* **A:** *No*

You can do this if you parse and manipulate wikitext instead!

WIKIMEDIA
FOUNDATION

Wikitext semantics

- Do wikitext constructs behave like DOM trees? No.
  - Wikitext has no spec and hence no formal semantics
  - Behavior emerges out of legacy PHP parser's implementation
  - The legacy parser doesn't think in terms of DOM constructs at all -- it is primarily concerned about efficiently constructing the output HTML string
- Templates inherit the same behavior
  - Lot of templates are about constructing the output HTML string
  - Meaning emerges from conventions and practices on wikis

```
'''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|}}, for ex.

No if {{tpl}} is {{1x|'"a}} or {{1x|""a"}} or {{1x|}}, 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

Wikitext state of affairs

- Non-local effects exist in wikitext

- 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”

Parsoid's approach

Assumes DOM structure as required to make this work

Compute a mapping between faux-DOM (input) and real-DOM (output)

Output has DOM structure

WIKIMEDIA
FOUNDATION

Examples

- Extension output, link content, figure captions are treated as “independent documents” (DOM fragments)
- Handles scenarios where template output doesn’t map to a DOM tree
  - `{{1x|'''a'''}}` maps to a DOM tree normally
  - But, for `'''foo {{1x|'''a'''}} bar''' baz`, the template-generated DOM tree expands to `'''foo {{1x|'''a'''}} bar'''` (note `baz` isn’t included)

Exercise: Create a page with this wikitext and try editing in VE.

- Adds `<section>` wrappers around wikitext sections, but can handle cases where the wikitext section doesn’t map to a DOM tree.

WIKIMEDIA FOUNDATION

Parsoid Sales Pitch

- Parsoid demonstrated two big ideas:
  - Round-trip conversion to/from wikitext is practical
  - Can map output DOM node to input wikitext string
- Assumes DOM structure as required to make it work
  - It is a statistical gamble that mostly works out
  - Has a lot of hacks and fallbacks to handle scenarios where DOM semantics don’t apply.

Parsoid Sales Pitch

"Parsoid deals with wikitext so you don't have to"

- Parsoid has a (versioned) spec for its output
  - **Read use cases**: Clients can use the spec to query the output DOM and glean info about the input wikitext document.
    - Google stopped querying MediaWiki and now uses Parsoid's output
  - **Edit use cases**: Clients (like VE, CX) can manipulate DOM structures and rely on Parsoid to map it back to wikitext faithfully.

WIKIMEDIA
FOUNDATION

Talk Overview

- Part 1: Background
  - First things first: what are the two parsers?
  - Setting the stage: what is the core 2-parser issue?
  - Some details: HTML, DOM, wikitext, Parsoid approach
- **Part 2: Roadmap**
  - **How did we get here? 2011 - 2018**
  - Why are we porting Parsoid to PHP?
  - How do we get to the end of this road from here?

Roadmap: 10K foot view

- **Parsoid till 2015**
  - Genesis: 2011-2012 (Feasibility unknown)
  - Getting established: 2013 - 2015 (Here to stay)
- 2015: Oh no! Two parsers (and two of other things)!
- Unification path: 2015 onwards
  - Two-pronged approach: Move both parsers towards each other
  - Participate in ***conversations to bring clarity to architecture constraints*** ⇒ Platform Evolution; Port Parsoid to PHP

WIKIMEDIA
FOUNDATION

Rewind to 2011

Wikimedia Engineering's key future-facing priority for 2011-2012 is creating a rich text editing environment, backed by a revamped, normalized, more consistent wikitext parser.

(From https://www.mediawiki.org/wiki/Parser_2011)

WIKIMEDIA
FOUNDATION

Rewind to 2011

- A “new parser” was planned May ‘11 during the Berlin hackathon along with wikitext.next, VisualEditor
- Work on the “new parser” started Nov ‘11 as part of the VisualEditor project. It was renamed Parsoid in Feb ‘12

Structured representation (AST / DOM) at the core

Parsoid in early 2012

- Core design in place (still used in 2019)
  - Parse to tokens + transform tokens + build DOM + transform DOM
  - Output HTML5 with annotations to expose wikitext info
    - Link types, extensions, template information, meta-information (categories, etc.)

- Early plan:
  - Prototype in Node.js and port to C++ as a PHP extension in core
  - Self-contained with fallbacks to core functionality only for extensions

Parsoid by end-2012

- Rapid development to first release in Dec 2012
  - C++ port dropped, Node.js impl deployed ⇒ we got a parsing service
  - In-Parsoid preprocessing dropped ⇒ Parsoid relied on MediaWiki API for expanding templates to wikitext
- Worked well & lot of compatibility problems to resolve to make Parsoid feasible ⇒ we stuck with this architecture
  - Led to all the subsequent debates about SOA, 3rd party support, etc.

Parsoid 2013 - early 2015

- VE support stabilized by 2015, new clients appeared
  - Flow, Content Translation, MCS, Google, Kiwix, Linter prototype
- Dealing with ramifications of a different wikitext pipeline and processing model
- HTML → WT: wikitext escaping and template formatting
- Helping 3rd party users + Debian packaging
- Became the poster child in the services debate!

WIKIMEDIA
FOUNDATION

Parsoid 2013 - early 2015

- Ramifications of a different wikitext processing model
  - Templates that produce strings that have no DOM equivalent
    - `{{1x|<div>}}`; Infobox styling templates that also emit content; Table cell style templates that also emit something else; table start & end templates; ...
  - Demarcating template boundaries while respecting DOM structures
    - `'''foo {{1x|'''a'''}} bar'''` `baz`
    - `foo` `{{1x|'''a'''}}` `bar''' baz`
  - Realizing need for Parsoid-specific versions for extensions
  - Bad markup and needing to roundtrip back without dirty diffs
    - **“Fostered content”** markup errors are a prominent one

WIKIMEDIA FOUNDATION

Fostered content!

- Misplaced content in tables
- HTML5 spec defines error handling

```
<table>x<tr>y<td>z</td></tr></table>
```

renders as

```
xy<table><tr><td>z</td></tr></table>
```

x and y are adopted by the table's parent

⇒ they have a foster parent

- Content is reordered!
- Causes dirty diffs if not detected and accounted for
- Tables are common on wikis
- Especially problematic when templates cause this
- Source of a bunch of complexity in Parsoid and many early bug reports related to "dirty diffs"

WIKIMEDIA FOUNDATION

Parsoid 2015

- Parser unification conversations start
- Addressing Parsoid ⇔ MediaWiki interaction issues
  - Reducing request volume via ParsoidBatchAPI extension
  - Addressing timeout issues between services and retry cascades
- Stabilizing Parsoid (performance, deployment, QA)
  - 8 Parsoid incidents between 2013-2015 (4 in 2015)
  - 1 incident between 2016 and none in 2017 and 2018

WIKIMEDIA
FOUNDATION

Parsoid 2015 - 2016

- Planning to use Parsoid for reads as well as edits
  - Charted options for how to unify the parsers
  - Catalogued all the ways Parsoid was incompatible with PHP parser
    - **Step 1: upgrade core to HTML5 by replacing html4-tidy**
- Speculative planning for wikitext 2.0
  - RFCs / proposals written for: balanced templates, heredoc syntax for long template args, wikitext 2.0, typed templates, etc.
  - In retrospect, a bit premature but we are well-prepared for next steps. :-)

Roadmap: 10K foot view

- Parsoid till 2015
  - Genesis: 2011-2012 (Feasibility unknown)
  - Getting established: 2013 - 2015 (Here to stay)
- 2015: Oh no! Two parsers (and two of other things)!
- **Unification path: 2015 onwards**
  - Two-pronged approach: Move both parsers towards each other
  - Participate in ***conversations to bring clarity to architecture constraints*** ⇒ Platform Evolution; Port Parsoid to PHP

WIKIMEDIA
FOUNDATION

Two-pronged approach

- Move PHP Parser towards Parsoid
  - Migrate core towards HTML5: Tidy → RemexHtml
  - Move media output to Parsoid-style semantic markup (WIP)
  - ...
- Move Parsoid towards PHP parser
  - Language variant support (WIP)
  - Redlinking (done)
  - ...

2015 - 2018: Replace Tidy

- 1 year became 3 years - see blog post for overview
  - Built RemexHtml, a pure PHP HTML5 parser
  - Built a number of custom QA tools to figure out impact on wikis
    - Spun up 2 VMs (one running Tidy, another RemexHtml) with 60K pages from 40 wikis
    - Revamped TestReduce + VisualDiff tools to diff screenshots of all those pages -- generated quantitative data
    - Built UprightDiff, a video-motion based screenshot diffing tool to eliminate diff noise
  - Initial testing revealed potential for a lot of disruption on wikis
    - Required editors to fix pages and templates to mitigate impact

WIKIMEDIA
FOUNDATION

2015 - 2018: Replace Tidy

- Wikis had come to depend on Tidy-specific behavior
  - We added some Tidy-compatibility code to handle some issues
  - But, required wikitext fixes in many other scenarios
- Assisting editors
  - Added linting to Parsoid to “precisely” identify wikitext that needed fixing
  - Built Linter extension to expose this to editors on all wikis
  - Built ParserMigration extension to let editors test and verify fixes
  - 18 months of community engagement efforts (including preparation)

2015 - 2018: Replace Tidy

- Community engagement
  - Wiki pages with info, Linter help pages, regular announcements
  - Wikis given a 1-year window to make the fixes
  - Continuous engagement on wikis providing help around fixes
  - QA tool to run weekly screenshot diffs (Tidy vs Remex) on 75K live pages on 60 wikis -- quantitative data to monitor progress
  - Progressive switch over 9 months including on early adopter big wikis to discover additional Tidy-specific problems -- **final switch in July 2018**
  - Tidy support has been removed from MediaWiki 1.33

Legacy constraints

- Prescient thoughts from 2011 hackathon:
  - ***Tim S**: there are a lot of features that people count on that rely on the current regime*
  - ***Neil K**: a simpler parser seems possible, but it becomes impossibly distant when we insist on not breaking anything?*
- Use Tidy replacement process as a template for future
  - Leverage Parsoid's wikitext linting abilities
  - Linter + ParserMigration to assist editors to fix pages where required
  - Community engagement

Roadmap: 10K foot view

- Parsoid till 2015
  - Genesis: 2011-2012 (Feasibility unknown)
  - Getting established: 2013 - 2015 (Here to stay)
- 2015: Oh no! Two parsers (and two of other things)!
- **Unification path: 2015 onwards**
  - Two-pronged approach: Move both parsers towards each other
  - Participate in ***conversations to bring clarity to architecture constraints*** ⇒ Platform Evolution; Port Parsoid to PHP

WIKIMEDIA
FOUNDATION

The services debates

- Till 2016:
  - no clarity around future of MediaWiki architecture
  - ⇒ unclear constraints around parser unification
- 2017 - 2018:
  - Debates, position papers, conversations, working groups, summits
  - ⇒ Led to the Platform Evolution Program in 2018
  - *Parser Unification* with *Porting Parsoid to PHP* as a first step became a part of that program.

WIKIMEDIA
FOUNDATION

Talk Overview

- Part 1: Background
  - First things first: what are the 2 parsers?
  - Setting the stage: what is the core 2-parser issue?
  - Some details: HTML, DOM, wikitext, Parsoid approach
- **Part 2: Roadmap**
  - How did we get here? 2011 - 2018
  - **Why are we porting Parsoid to PHP?**
  - How do we get to the end of this road from here?

Let’s bring back the genie slides!

1-parser scenario wishes

LLW_Aladdin_genie.jpg from commons by Jerry Dakin; cc-by-2.0 (Legoland Windsor)

Wish 1: Port Parsoid to PHP, fix Parsoid's output and feature differences including fixing extensions

Wish 2: Port a subset of MediaWiki core + exts to Node.js, fix Parsoid's output and feature differences including fixing extensions

Maybe other wishes ...

WIKIMEDIA FOUNDATION

Why Wish 1 makes sense

Assumption: current Parsoid service arch needs fixing

- **Limited scope** (port Parsoid codebase to PHP)
  - vs. unknown scope: Port a “subset” of MediaWiki core + exts to Node.js
- Ability to **leverage core code** to bridge feature gaps
  - vs. writing / porting more code to Node.js
  - Ex: redlinking, language variants (could have leveraged core code)
  - Ex: i18n and l10n support for extensions

WIKIMEDIA
FOUNDATION

Side-effect benefits

- Non-wikimedia wikis get a simpler install in some cases
  - VE + wikitext linter come for free by default
- Potentially simpler codebase (no async)
- Expand exposure to Parsoid codebase during the port
- A bit more clarity about other services once Parsoid exits the services debate

WIKIMEDIA
FOUNDATION

Talk Overview

- Part 1: Background
  - First things first: what are the two parsers?
  - Setting the stage: what is the core 2-parser issue?
  - Some details: HTML, DOM, wikitext, Parsoid approach
- **Part 2: Roadmap**
  - How did we get here? 2011 - 2018
  - Why are we porting Parsoid to PHP?
  - **How do we get to the end of this road from here?**

Milestones to the end

- Finish ongoing projects
  - Finish Parsoid/PHP port + take Parsoid/JS out of service
    - Another 4-6 calendar months?
  - Finish updating legacy PHP parser media output to match Parsoid
    - Might require some gadgets & bots to be updated
    - 1-2 person months of effort?
  - Fix known bugs (already in phabricator)
    - Effort needs to be estimated
  - Finish implementing language variants support
    - Effort depends on how we scope this work, but 1-3 person months

WIKIMEDIA
FOUNDATION

Milestones to the end

- Identify any other Parsoid feature gaps
  - Bridge with core code where possible
  - Example: i18n support in extensions
- Improve Parsoid performance
  - Parsoid does much more work than the current PHP parser and is slower
  - This will require some serious performance work including throwing hardware at the problem at some point

Milestones to the end

- Focus on making Parsoid the default on Wikimedia wikis
  - Establish regular visual diff QA runs to identify uncaught issues
    - Code already in please, just need to kick it off to run maybe once a week
  - Analyze results and file Parsoid bugs or identify any wikitext changes required on wikis (similar to Tidy replacement)
    - Depends on results of QA runs
  - Finalize **new parser hooks API** + migrate over Wikimedia extensions
    - A few extensions will probably require some non-trivial changes

Likely 18-24 months from now but sooner depending on dev. resources

WIKIMEDIA
FOUNDATION

Parser hooks

- Parsoid's internals are different
  - ⇒ hooks named after PHP Parser internals won't work
    - `ParserBeforeStrip`, `ParserAfterStrip`, `ParserBeforeTidy`, `ParserAfterTidy`, and likely other before/after hooks
- The new hooks API won't reference parser internals
  - Currently, extensions in Parsoid use the following hooks and accept either their content model (wikitext) or DOM as input:
    - `toDOM`, `fromDOM`, `wt2htmlPostProcessor`, `html2wtPreProcessor`, `lintHandler` (new ones will be added to mimic some existing hooks)
  - Cite, Gallery, Poem, Nowiki, Pre have been implemented with this

Milestones to the end

- Focus on MediaWiki next
  - Publish new parser hooks API for MediaWiki
  - Publish docs around fixing extensions + wikipages
  - Follow deprecation process to deprecate legacy parser

- … the end … ?

WIKIMEDIA
FOUNDATION

Who's (been) involved (so far)?

- **Arlo Breault, C. Scott Ananian, Shannon Bailey, Subbu Sastry,** Kunal Mehta, **Tim Starling,** *Brion Vibber, Gabriel Wicke, Marc Ordinas i Llopis, Mark Holmquist* (all work(ed) on Parsoid)
- **Brad Jorsch, Gergő Tisza** (helping with PHP port currently)
- *Trevor P, James F, Roan K* (via VE as first guinea pigs)
- GSoc and Outreachy interns (5) and other code contributors
- "VE", "Services", CL, SRE, Cloud Services, Management (all in different ways at different times)
- Wikimedia Community (via Tidy-related fixes!)

WIKIMEDIA
FOUNDATION

… the end …