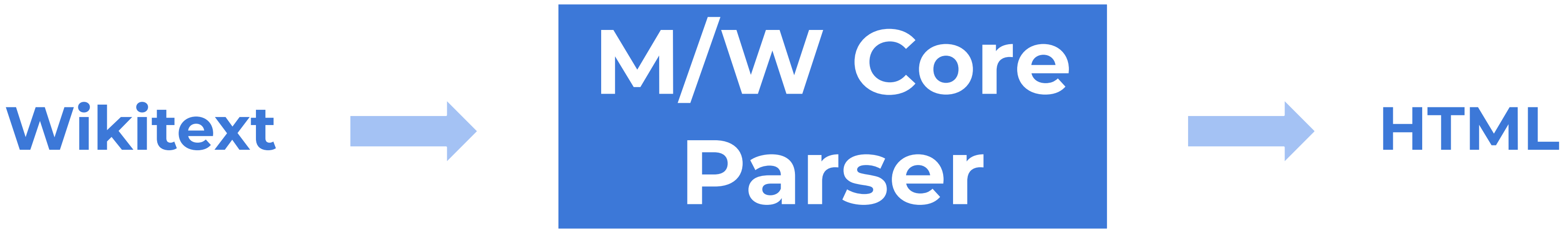
Retargeting extensions for Parsoid

S. Subramanya Sastry (Subbu), Parsing Team
Wikimedia Tech Talk, August 2020

WIKIMEDIA
FOUNDATION

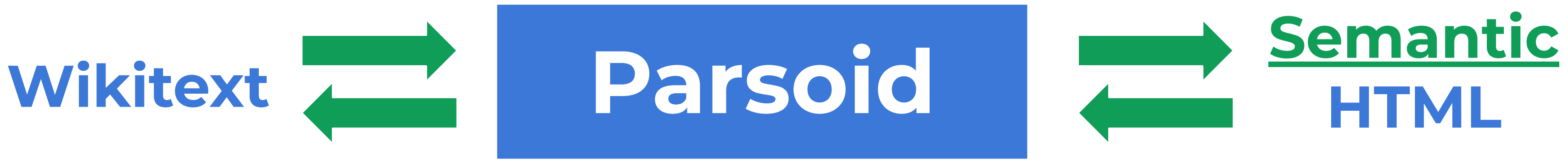
First things first!
Background context

WIKIMEDIA
FOUNDATION



M/W Core Parser: default parser (since 2003)

Clients: Desktop view, Mobile web, Action API



Parsoid: alternate parser since 2012

Used by: VisualEditor, 2017 Wikitext Editor, Discussion Tools, Mobile Apps, Flow, Content Translation, Wikitext Linting, Kiwix Offline Reader, Google, REST API

Javascript (Node.js) codebase 2011-2019; Ported to PHP in 2019

WIKIMEDIA
FOUNDATION

Use Parsoid everywhere

- Core parser cannot support Parsoid clients
- Parsoid's annotated HTML provides more information
- Two parsers not tenable and hamstrings future work
- Parsing Team goal:
  - **Make Parsoid the default MediaWiki wikitext engine**
  - Initial focus: Start transitioning on the Wikimedia cluster late 2021

WIKIMEDIA
FOUNDATION

Implications for exts

Bad news:

- All parser extensions will need to be updated

Good news:

- Most extensions are probably minimally affected

WIKIMEDIA
FOUNDATION

Specifically …

Extension affected if it satisfies at least one of these:

- Has a **parser hook** listener
- Uses **Parser.php** methods / properties

Restricted focus for 2020/21: extensions deployed on Wikimedia wikis

WIKIMEDIA
FOUNDATION

Parser-Extension model differences

WIKIMEDIA
FOUNDATION

What is different?

- Core parser hooks refer to parsing stages
  - `ParserBeforeStrip`, `ParserAfterStrip`
  - `ParserBeforeTidy`, `ParserAfterTidy`, `ParserAfterParse`
  - `InternalParseBeforeLinks`, `BeforeParserFetchFileAndTitle`
- Parsoid's pipeline is different
  - It has very different pipeline stages
    - **wt** → **html**: tokenizer, 17 token passes, DOM builder, 21+n DOM passes
    - **html** → **wt**: DOM builder, 2+n DOM passes, Serializer
  - Pipeline keeps changing over time ⇒ **not exposed to extensions**

What is different?

- Core parser exposes sequential processing
- Parsoid does not expose processing order
    - Parsoid (JS) had out-of-order async processing ⇒ *your extension tags could be processed out-of-order*
    - While disabled now, Parsoid reused content from a previous parse ⇒ *your extension will not be invoked on that content*

⇒ We cannot expose / guarantee specific processing order

WIKIMEDIA
FOUNDATION

What is different?

- Core parser treats everything as strings
    - Parsoid treats extension output as documents
- Core parser exposes its methods & objects directly
    - Parsoid prefers keeping implementation details opaque
    - Parsoid provides an API object to extensions instead

WIKIMEDIA
FOUNDATION

Implications

Different pipeline stages + Parsoid doesn’t expose them

⇒ Far fewer hooks + transformation hooks (i.e. WHAT not WHEN)

No processing order guarantees

⇒ Cannot maintain global ordered state in extensions (ex: counters)

No direct access to Parser object

⇒ Use a Parser API object

WIKIMEDIA FOUNDATION

Today's talk focus

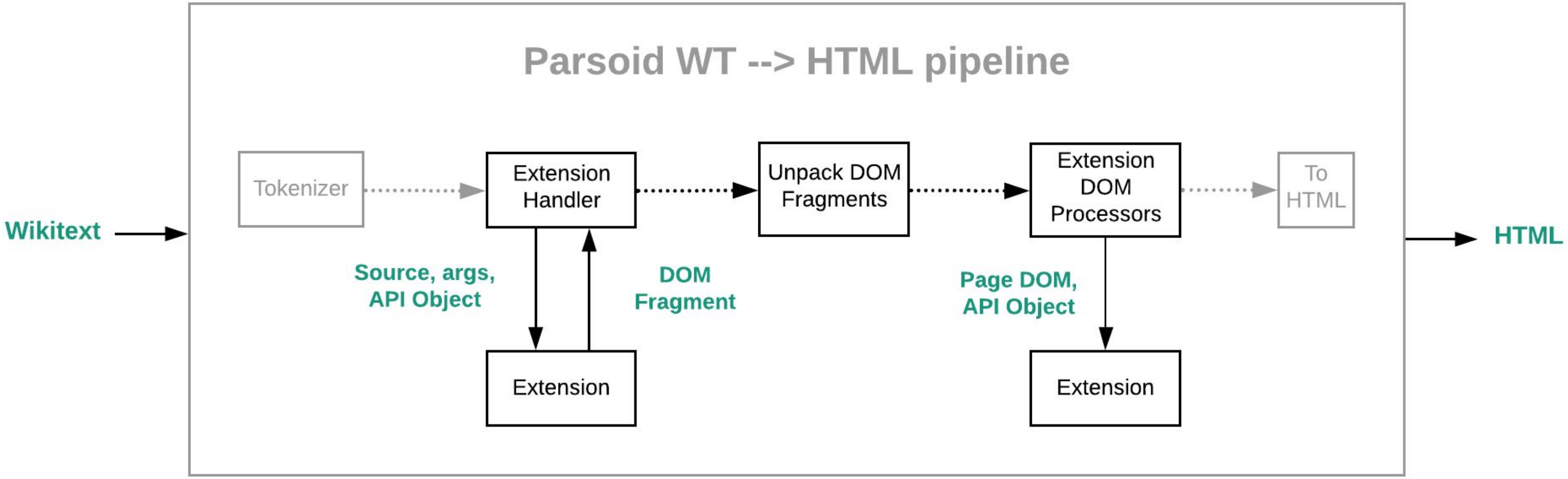
- Tag Extensions (`<ext … >...</ext>`)
  - Parsoid supports content-handler extensions too

- wikitext → HTML conversions
  - Parsoid supports HTML → wikitext in extensions too
  - Get in touch if you want to update your extensions for this mode

WIKIMEDIA
FOUNDATION

Tag extensions: Overview

WIKIMEDIA
FOUNDATION



WIKIMEDIA FOUNDATION

Hooks

- Places to hook in wt → html direction
  - One localized transformation hook: (**source, args**) → **DOM**
  - One global DOM processing hook: **DOM** → **DOM**
  - **Not shown in diagram**: One wikitext linting hook (if your extension handles wikitext)
  - Maybe others in the future (depending on use case)

WIKIMEDIA
FOUNDATION

Extension Output

- DOM is annotated with `typeof` & `data-mw` attributes
    - `<div typeof="mw:Extension/Poem" data-mw="{attrs: { … }, … }">..</div>`
- DOM is tunneled through pipeline
    - Placeholder node represents the DOM output until it is unpacked ⇒ DOM is unmodified by intervening passes
        - Similar to strip-state mechanism that exts. explicitly manage currently
        - Extensions don't need to do anything special

Extension registration

- extension.json adds **`ParsoidModules`** for config
  - One of 2 options
    - Inline JSON config
    - ObjectFactory declaration
  - ObjectFactory decl. should provide **`ExtensionModule`** interface impl.
    - Interface has one method: `getConfig()`

WIKIMEDIA
FOUNDATION

Example config

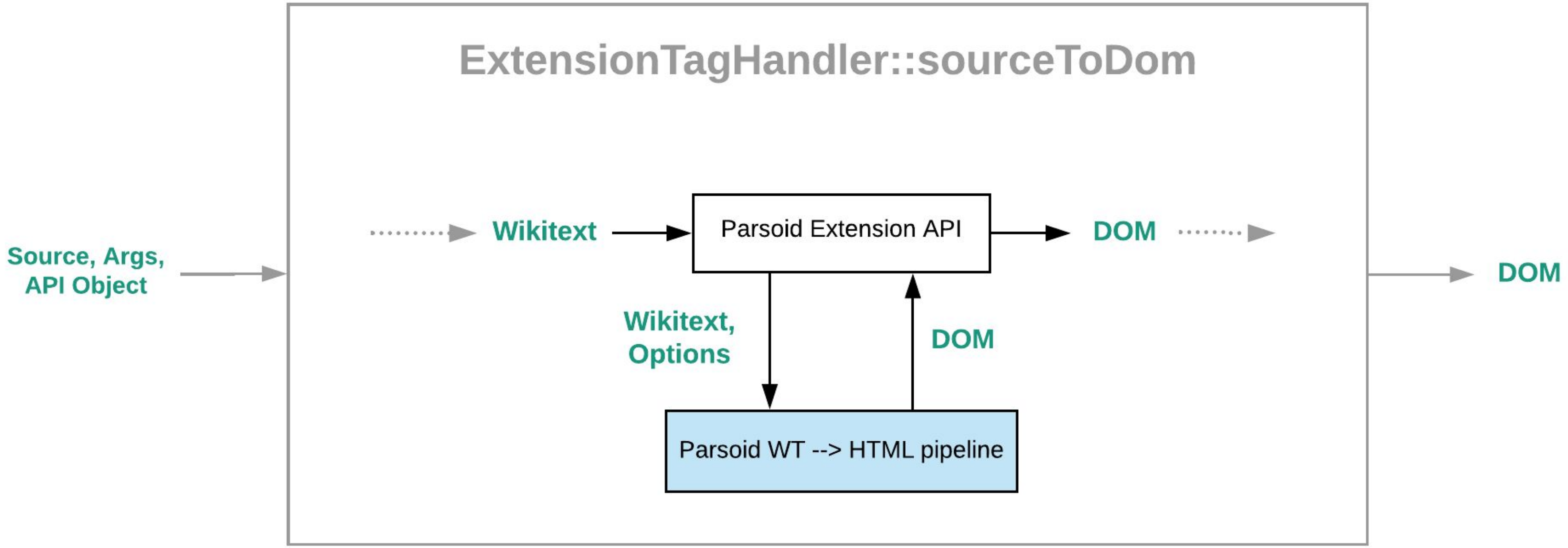
```
{
      'name' => 'Cite',
      'tags' => [
            [
                  'name' => 'ref', ‘handler’ =>Ref::class,
                  'options' => [ 'wt2html' => [ … ], 'html2wt' => [ … ]  ],
            ],
            [ … one for <references> as well with html2wt options … ]
      ],
      'domProcessors' => [ RefProcessor::class ],
      …
}
```

Extends **ExtensionTagHandler** class

Extends **DOMProcessor** class

ExtensionTagHandler

- Declares transformation hooks with dummy impls
  - `sourceToDom($api, string $src, array $args): DOM*`
  - `domToWikitext($api, DOMDocument $dom, …): string`
  - `lintHandler($api, $dom, $defaultLintHandler)`
  - `.....`
- Expect **`sourceToDom`** will be implemented
- **`domToWikitext`** optional
  - Parsoid provides default handling


ExtensionTagHandler::sourceToDom
Source, Args, API Object
Wikitext
Parsoid Extension API
DOM
Wikitext, Options
DOM
Parsoid WT --> HTML pipeline
DOM

Observations

- Use the API object to process wikitext
- Minimal control over parsing pipeline
  - You cannot run specific pipeline stages
  - You can specify output type / embedding context
    - Currently, **inline** or **block**
    - Additional output type / embedding contexts might be available in the future
- Output DOM has all applicable passes run
  - DOM passes that only apply to top-level Page DOM are skipped

WIKIMEDIA
FOUNDATION

Underlying principle

Wikitext should behave uniformly no matter where it shows up

- All deviations should have some conceptual grounding
  - Ex: embedding context type (CSS, HTML attribute, inline / phrasing content, table cell, etc.) introduces output constraints
  - No arbitrary subsets - https://phabricator.wikimedia.org/T192037

WIKIMEDIA
FOUNDATION

ParsoidExtensionAPI

- Categories of API methods today:
    - Wikitext → DOM; DOM → wikitext (multiple methods)
    - **HTML → DOM; DOM → HTML** (vs. native DOM / library methods)
    - Methods that deal with extension args
    - get* methods (title, page URI, config objects, etc.)
    - ExtensionTag methods (query properties about `<ext … >` usage)
    - A few others (some transitional and may go away)

WIKIMEDIA
FOUNDATION

Examples

WIKIMEDIA
FOUNDATION

Extension tag types

- Don’t wrap wikitext: `nowiki`, **`pre`**, `syntaxhighlight`, **`rawhtml`**
  - `$output = genDOM($input)`
- Thin wrapper over wikitext: **`ref`**
  - `$output = parseWT($input)`
- Process content as more-or-less-wikitext: **`poem`**
  - `$output = postProcessDOM(parseWT(mangle($input)))`
- Content has wikitext snippets that are processed separately: **`gallery`**
  - `$output = buildDOM(LOOP(parseWT(mangle($frag))))`

WIKIMEDIA
FOUNDATION

RawHTML extension

```
class RawHTML extends ExtensionTagHandler implements ExtensionModule {
    public function getConfig(): array {
        return [ 'name' => 'RawHTML',
            'tags' => [ [ 'name' => 'rawhtml', 'handler' => self::class ] ]
        ];
    }
    public function sourceToDom(ParsoidExtensionAPI $api, $src, $args) {
        return $api->htmlToDom($src); // returns DOM*
    }
}
```

<pre>

WIKIMEDIA
FOUNDATION

Pseudocode

```
function sourceToDom(ParsoidExtensionAPI $api, $txt, $args): DOM* {
    $doc = $api->htmlToDom(''); // Empty doc
    $pre = $doc->createElement('pre');
    $api->sanitizeArgs($pre, $args);
    $txt = decodeWtEntities(trimLeadingNL(stripNoWikis($txt)));
    $pre->appendChild($doc->createTextNode($txt));
    DOMCompat::getBody($doc)->appendChild($pre);  // libxml fixes; T215000
    return $doc;
}
```

WIKIMEDIA
FOUNDATION

<ref>

WIKIMEDIA
FOUNDATION

<ref> Example

Sample Wikitext

```
Foo <ref>''AB'' and '''CD'''</ref>
  and bar and baz.

== References ==

<references />
```

Rendered Output

Foo [1] and bar and baz.

References

1. ^ *AB* and **CD**

WIKIMEDIA
FOUNDATION

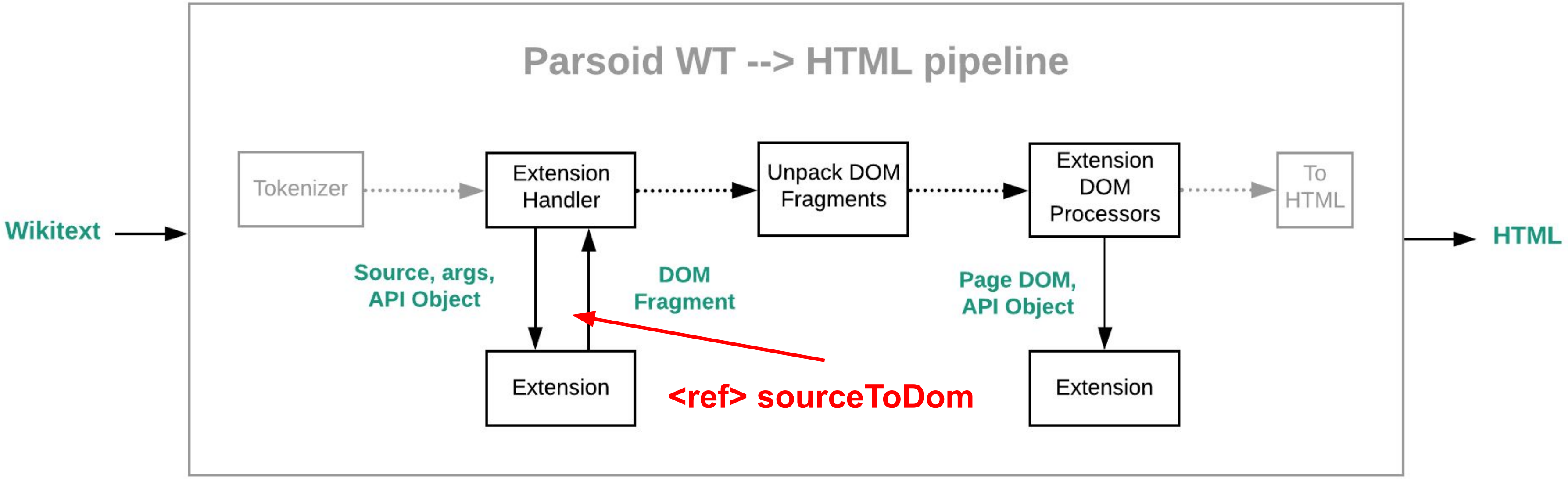
<ref> sourceToDom

```
function sourceToDom(ParsoidExtensionAPI $api, $txt, $args): DOM* {
    ... some checks to detect ref-in-ref scenarios ...
    return $api->extTagToDOM($args, $txt, [
        'wrapperTag' => 'sup', // DOM is wrapped in <sup> tag
        'parseOpts' => [
            'context' => 'inline', // No paragraphs, No "indent-pre"
            'extTag' => 'ref', 'extTagOpts' => [ 'allowNestedRef' => … ],
        ]
    ]);
}
```

WIKIMEDIA
FOUNDATION

Wait a minute …

- That handler returned DOM of <ref>’s content
- How does that content migrate to the references section?
- What happened to numbered ref links?



WIKIMEDIA FOUNDATION

<ref> sourceToDom

- **Reminder**: this is the local transformation hook
- Cannot reliably count in the right order to generate links
- Does not have access to the final DOM

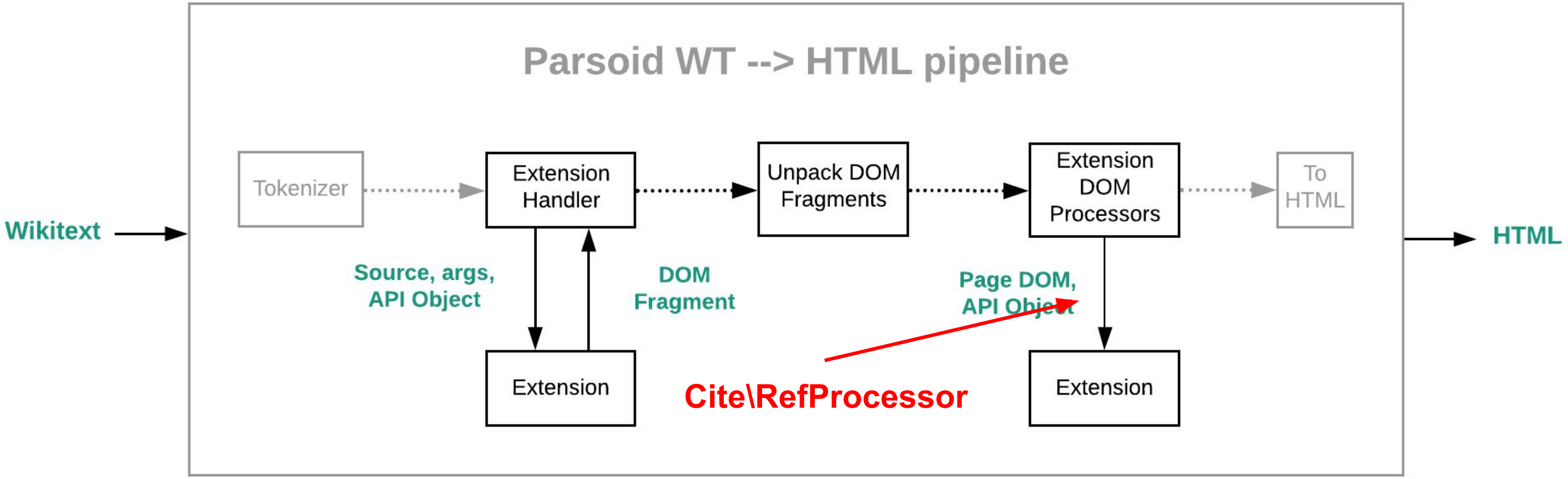
Cite config from before

```
{
        'name' => 'Cite',
        'tags' => [
                [
                        'name' => 'ref', ‘handler’ =>Ref::class,
                        ...
                ],
                ...
        ],
        'domProcessors' => [ RefProcessor::class ],
        ...
}
```

Extends **ExtensionTagHandler** class;
Implements local transformation hook

Extends **DOMProcessor** class;
Implements global DOM processing hook

WIKIMEDIA
FOUNDATION



WIKIMEDIA FOUNDATION

Cite\RefProcessor

TIP: Parsoid adds ext. output to DOM. Page DOM is your global state object.

- `RefProcessor::wtPostProcess`
  - Has access to output of all `<ref>` tags via the Page DOM
  - Walks the tree (depth-first) **in-order** and harvests `<ref>` content
  - Generates the `<references>` section in required format
  - Migrates the `<sup>` content to the `<references>` section
  - Updates `<sup>` at `<ref>` sites with links to the `<references>` section

Effectively restructures the DOM

WIKIMEDIA
FOUNDATION

DOMProcessor class

- `wtPostProcess($api, $root, $opts, $atTopLevel)`
  - Invoked by Parsoid when the full page is constructed
  - Almost at “the end” when most, but not all, information is in the DOM
    - MediaInfo updates, Link annotation (external, red, disambig), LangConverter, Heading ids, other DOM fixups, section wrapping haven’t run yet
  - For extensions that might need all info, we might introduce a new DOM processor hook (maybe **`finalizeDoc($api, $root)`**)?
- `htmlPreProcessor($api, $root)`
- `Maybe others … ?`

WIKIMEDIA
FOUNDATION

Mapping extension functionality between Core parser & Parsoid

WIKIMEDIA
FOUNDATION

Mapping: Parser hooks

- `ParserFirstCallInit:`
  - Register tag handlers directly in config
- `ParserBeforeTidy, ParserAfterTidy, ParserAfterParse:`
  - Use `wtPostProcess` DOMProcessor hook or if necessary, we can provide a `finalizeDoc` DOMProcessor hook
- `ParserClearState:`
  - Should not be needed - let us know if you have a use case for this
- `ParserBeforeStrip, ParserAfterStrip:`
  - Should not be needed - let us know if you have a use case for this

WIKIMEDIA
FOUNDATION

Mapping: Parser hooks

- `ParserLimit*:`
  - Unaffected. Will be refactored into meta-parsing functionality.
- `InternalParseBeforeLinks:`
  - Will not support (link syntax heavily overloaded in wikitext)
  - Use `wtPostProcess` DOM hook if you want to update links in any way
  - Alternatively, use different syntax (ex: parser functions)

Watch mw:Parsoid/Extension_API for complete mapping between hooks

WIKIMEDIA
FOUNDATION

Mapping: Parser API

Replacements for `parse, internalParse, startExternalParse, recursiveTagParse, recursiveTagParseFully`

- `extTagToDOM`: use when tag wraps wikitext (ex: <ref>)
- `extArgToDOM`: use when you need to process an arg as wikitext (ex: <gallery> caption)
- `renderMedia`: *what it says on the tin* (ex: <gallery>, <imagemap>)
- `wikitextToDOM`: use when none of the above meet your needs
  - **ParsoidExtensionAPI uses this internally for all the above 3 API methods.**
- May provide other flavours in the future

WIKIMEDIA
FOUNDATION

To reiterate ...

- No control in Parsoid over how much parsing happens
  - `recursiveTagParse, internalParse` in current Parser API return "half-parsed HTML" whereas other methods return "fully-parsed HTML"
- wt2html options provide some semantic control
  - But, cannot turn on/off pipeline stages OR run stages selectively
- Dealing with special wikitext semantics
  - Mangle input as necessary (ex: `<poem>, <gallery>`)
  - Use DOM post processing as necessary (ex: `<poem>, <ref>`)

WIKIMEDIA
FOUNDATION

Mapping: Parser API

- Will expose `ParserOutput` object via the API
- Will expose `setFunctionHook` for declaring parser fns
    - Callback will get `ParsoidExtensionAPI`, not parser.
- Will augment `ParsoidExtensionAPI` with additional methods as necessary based on discovery and feedback

Watch mw:Parsoid/Extension_API for complete mapping between API methods

WIKIMEDIA
FOUNDATION

Mapping: Strip Markers

- Strip markers: used to tunnel output through parser stages
- Not needed in Parsoid
  - Extension output always tunneled through ⇒ output doesn’t go through additional processing
  - StripState related hooks and methods don’t exist in Parsoid
  - If found necessary, will introduce equivalent functionality

WIKIMEDIA
FOUNDATION

Status

WIKIMEDIA
FOUNDATION

Extensions

- Tag extensions:
  - In production: Cite, Poem, Gallery, Nowiki, Pre
  - In gerrit: ImageMap
  - Incomplete skeletons: LST, Translate
  - Next in line: `<indicator>`
- ContentHandler Extensions:
  - In production: JSON
- ParserTests extensions:
  - In use: RawHtml, StyleTag

WIKIMEDIA
FOUNDATION

Status: Hooks + API

- In late draft stage
  - As we discover unsupported uses, we will:
    - Add new hooks
    - Update ParsoidExtensionAPI with new functionality
  - Parser.php interface being narrowed
    - Make more methods private!
    - Deprecate lots of things!
- ParserTests & CI support will land in a couple weeks

WIKIMEDIA
FOUNDATION

Docs, next steps, …

WIKIMEDIA
FOUNDATION

Next steps for us

- Will continue outreach and soliciting feedback
  - Presented early draft @ EMWCon in April 2020
  - Solicited feedback internally July 2020
  - This talk is next step in process
  - **Next:** wikitech-l, mediawiki-l, TechCom RFC

Will do our best to not break things unnecessarily

WIKIMEDIA
FOUNDATION

Next steps for you!

- Learn more, dive into the details, provide feedback
- Start updating your extensions now!
    - Best way to figure out what is missing, what is easy, what is hard

Help us migrate MediaWiki to Parsoid rendering for Wikimedia wikis next year!

WIKIMEDIA
FOUNDATION

Learn more: Look at code

- `Wikimedia\Parsoid\Ext:`
  - `ExtensionModule`, `ExtensionTagHandler`, `DOMProcessor`, `ParsoidExtensionAPI`
  - Helper classes
- `Wikimedia\Parsoid\Core:`
  - `DOMSourceRange`, various exception classes
- `Wikimedia\Parsoid\Utils\DOMCompat`
  - Work around broken PHP DOM API

Learn more: docs, etc.

- https://mediawiki.org/wiki/Parsoid/Extension_API
  - **Discuss / leave questions on the Talk page**
- Look at Parsoid's implementations for Poem, Pre, Cite, etc.
- Look at Parsoid docs for the Ext/ namespace @ https://doc.wikimedia.org/Parsoid-PHP/master/
- Parsoid HTML spec @ https://www.mediawiki.org/wiki/Specs/HTML
- Find us at:
  - IRC: #mediawiki-parsoid
  - Email: parsing-team@wikimedia.org

WIKIMEDIA
FOUNDATION

Thanks!
Questions?

WIKIMEDIA
FOUNDATION

Backup slides

WIKIMEDIA
FOUNDATION

Extensions & Parser Tests

WIKIMEDIA
FOUNDATION

Parser Tests

- Add **`html/parsoid`** section w/ expected Parsoid output
  - Only needed if output differs
- ParserTests support multiple test modes per test
  - wt → HTML, wt → HTML → wt, HTML → wt, HTML → wt → HTML
  - Manual HTML edit tests (specify HTML edits, and expected wikitext)
  - Automated HTML edit tests
  - You can enable specific test modes per test
  - Use `{$ext}ParserTests-knownFailures.json` to track expected failures

WIKIMEDIA
FOUNDATION

Example

```
!! test
Poem with class
!! wikitext
<poem class="hiho">
hi ho
</poem>
!! html/php
<div class="poem hiho">
<p>hi ho
</p>
</div>
!! html/parsoid
<div class="poem hiho" typeof="mw:Extension/poem" about="#mwt3"
    data-mw='{"name":"poem","attrs":{"class":"hiho"},"body":{"extsrc":"\nhi ho\n"}}'><p>hi ho</p></div>
!! end
```

WIKIMEDIA
FOUNDATION

DOM Processor ordering

- How are DOM processors from multiple exts ordered?
  - Ordering problem not unique to Parsoid
  - Present wherever there are multiple listeners for the same event / hook that might operate on the same data
  - We have some ideas for Parsoid but nothing that is ready yet

WIKIMEDIA
FOUNDATION

<poem>

WIKIMEDIA
FOUNDATION

Pseudocode

```
function sourceToDom(ParsoidExtensionAPI $api, $txt, $args): DOM* {
    $mTxt = $this->mangle($txt); // process :, newlines, ----, nowikis
    return $api->extTagToDom($args, $mTxt, [
        'wrapperTag' => 'div', // DOM is wrapped in <div> tag
        'parseOpts' => [ 'extTag' => 'poem'  ],
        'processInNewFrame' => true, // mangled $mTxt is different from $txt
        'clearDSROffsets' => true    // mangled $mTxt => DSRoffsets incorrect
    ]);
}
```

More ...

- Poem extension treats `<nowiki>` blocks differently
  - Unlike normal wikitext, newlines inside becomes `<br>`s
  - Changing newlines to `<br>` in `mangle(..)` won't work because `<nowiki>` will escape them!
  - Poem extension registers a DOM processor to deal with this
- Processor finds the `<nowiki>`s and fixes newlines
  - `typeof="mw:Extension/$extName"` attr. present on ext. wrappers
  - Processor looks for matching `typeof` to identify nowiki blocks
  - Replaces newlines inside them with `<br>` tags

WIKIMEDIA
FOUNDATION

<gallery>

WIKIMEDIA
FOUNDATION

Pseudocode

```
$doc = … ; // construct gallery scaffolding
foreach ($line in $txt) {
    $mLine = makeImageWikitext($line); // [[File:..|..|..]]
    $imgDOM = $api->wikitextToDom($mLine, [
        'parseOpts' => [ 'extTag' => 'poem', 'inlineContext' => true ],
        'processInNewFrame' => true,
        'shiftDSRFn' => function($dsr) { return updated $dsr; }
    ]);
    … Process $imgDOM and add to $doc …
}
```

WIKIMEDIA
FOUNDATION