TECHNISCHE FAKULTÄT

Department Informatik

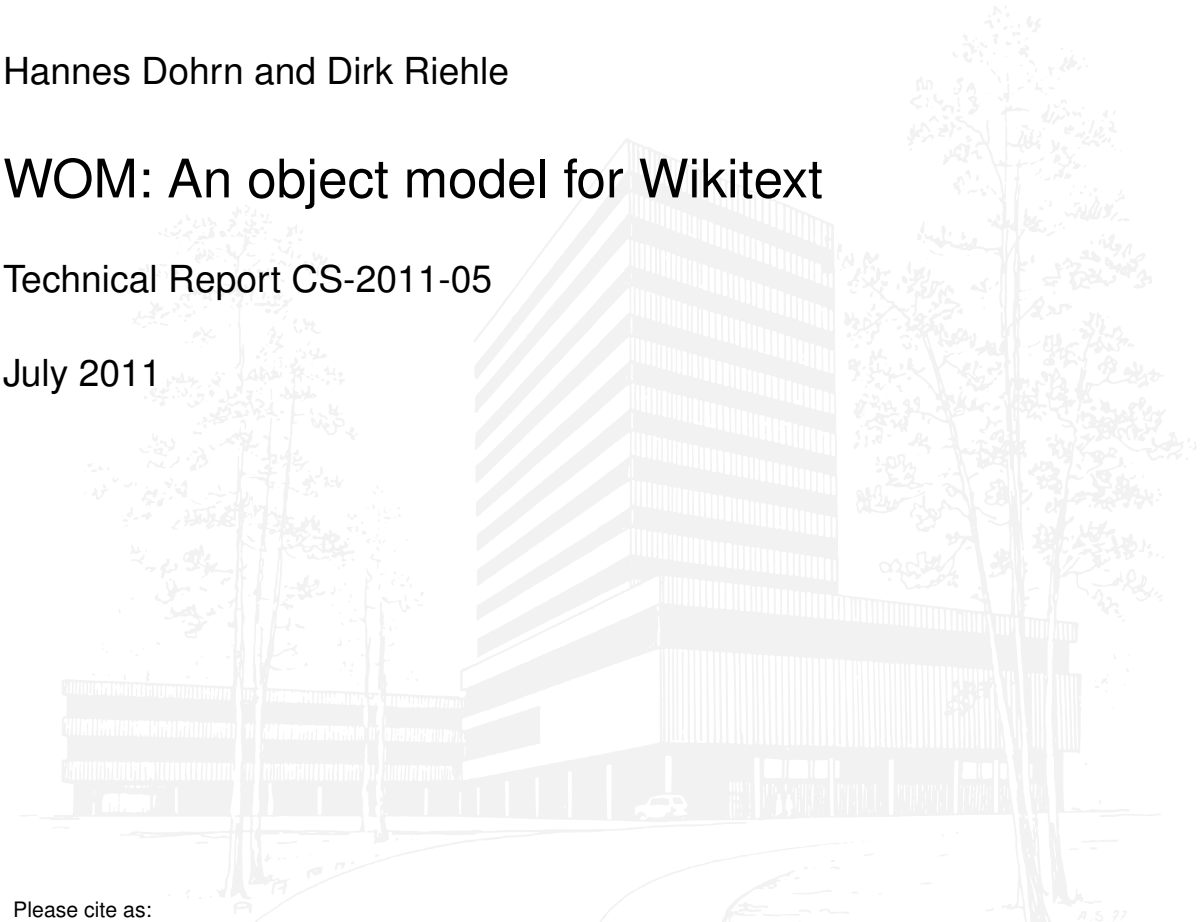
Technical Reports / ISSN 2191-5008

Hannes Dohrn and Dirk Riehle

WOM: An object model for Wikitext

Technical Report CS-2011-05

July 2011



Please cite as:
Hannes Dohrn and Dirk Riehle, “WOM: An object model for Wikitext,” University of Erlangen, Dept. of Computer Science, Technical Reports, CS-2011-05, July 2011.

Friedrich-Alexander-Universität Erlangen-Nürnberg

Friedrich-Alexander-Universität Erlangen-Nürnberg
Department Informatik

Martensstr. 3 · 91058 Erlangen · Germany

www.informatik.uni-erlangen.de

WOM: An object model for Wikitext

Hannes Dohrn and Dirk Riehle
hannes.dohrn@cs.fau.de, dirk@riehle.org
Professorship for Open Source Software
Dept. of Computer Science, University of Erlangen, Germany

Abstract

Wikipedia is a rich encyclopedia that is not only of great use to its contributors and readers but also to researchers and providers of third party software around Wikipedia. However, Wikipedia's content is only available as Wikitext, the markup language in which articles on Wikipedia are written, and whoever needs to access the content of an article has to implement their own parser or has to use one of the available parser solutions. Unfortunately, those parsers which convert Wikitext into a high-level representation like an abstract syntax tree (AST) define their own format for storing and providing access to this data structure. Further, the semantics of Wikitext are only defined implicitly in the MediaWiki software itself. This situation makes it difficult to reason about the semantic content of an article or exchange and modify articles in a standardized and machine-accessible way. To remedy this situation we propose a markup language, called XWML, in which articles can be stored and an object model, called WOM, that defines how the contents of an article can be read and modified.

Index Terms

AST, DOM, HTML, Object oriented programming, Sweble, Wiki, Wikipedia, WOM, XHTML, XML, XWML

Table of Contents

1. Introduction

MediaWiki's Wikitext is one of the most widely used markup languages on the Internet today. It is used to describe wiki articles in wikis that are driven by the MediaWiki engine. A big portion of Wikitext can be found in Wikipedia for which the MediaWiki engine was initially developed. The English Wikipedia alone has over 2,5 million articles as of June 2011. And there are many more sites on the Internet that use the MediaWiki software as wiki engine.

This makes Wikitext one of the most important computer languages used in today's world. However, until recently understanding of Wikitext was poor. Only the MediaWiki software was able to transform Wikitext into HTML. The MediaWiki software itself is practically the defining authority of Wikitext. Unfortunately, even MediaWiki does not produce a higher-level representation of Wikitext like an abstract syntax tree (AST), but only performs a conversion to HTML.

Furthermore, the semantics of an article are only defined implicitly by the MediaWiki software, informal documentation and XHTML. This stems from the fact that MediaWiki converts Wikitext to the XHTML dialect of the HTML family of languages. And since many elements of Wikitext are mapped directly to XHTML, a big portion of an article's semantics is defined by the HTML standard.

The lack of a precise specification for Wikipedia and other MediaWiki content has stalled evolution. Today without precise content specification everything has to go in lock-step with the MediaWiki software. We previously argued for the significance of precisely defined standards in our work on WikiCreole [1]. The same applies here: A precise specification of how to write Wikipedia and MediaWiki content is crucial for re-starting technology evolution and innovation for the MediaWiki ecosystem. Hence this report that by way of a precise content specification language and format decouples the various components and allows them to start developing independently, which in turn will let them take up speed.

The lack of a high-level and machine-accessible representation of an article written in Wikitext has lead to the development of numerous alternative parsers [2] by the large community of Wikipedia and MediaWiki. But only recently parsers were developed that not only fully support the rich syntax of Wikitext but also generate an abstract representation of the content of an article [3], [4], [5], [6].

However, the abstract syntax trees produced by this new generation of parsers are not standardized and focus on the syntax of a MediaWiki article. Many applications only require a semantic representation of an article's content and can completely neglect it's syntactic structure. Also, many researchers and providers of third party software would benefit from a standardized representation of a wiki article and a standardized interface to such a data structure to parse and modify an article.

To remedy this situation this report provides the description of a data interchange format called *eXtensible Wikitext Markup Language* (XWML 1.0) to store and

exchange wiki articles as well as a set of Java interfaces that define a standardized way to work with articles. In this report we define the Wikitext Object Model (WOM) 1.0 as the combination of the XWM language and a programming interface specification. The interface specification is provided as a set of Java interfaces for pragmatic reasons and might be rewritten in the future using a full-blown Interface Definition Language like the OMG's IDL [7]. We also hope to add interface specifications in other programming languages. Common to all interface specifications should be that they offer the same operations and navigation facilities as the Java interfaces presented in this report.

This technical report is based on our work on the Sweble parser [6], its notion of an AST and its round-trip support. However, any parser that can generate an AST can be used in conjunction with the data structures and data description language presented in this technical report.

The remainder of this report is structured as follows. In section 2 we give a deeper insight into Wikitext. This is followed by a rationale which explains the major design decisions that are presented in this report. In section 3 we give a textual specification of the eXtensible Wikitext Markup Language, its semantics and the Java interfaces. In section 4 we give a brief overview of the most important differences between XWML and XHTML. The appendix provides formal definitions of XWML 1.0 and the Java interfaces that together define the WOM.

2. Wikitext and Wikitext Processing

The following sections give a brief overview of the Wikitext language, its processing by MediaWiki and finally the rendering and display by a browser. It concludes with the rationale that drives the design of the WOM.

2.1. Wikitext Tutorial

MediaWiki's Wikitext is a markup language that is thought as an easier interface to the language of the web, HTML. Using special formatting characters like `''''` (three apostrophes), parts of the text in an article are marked and rendered differently.

The following table contains some examples of Wikitext markup:

Text set in italic or bold font	`''Italic text''` `'''Bold text'''`
Internal link (pointing to a page inside the same wiki)	`[[target page|Link text]]`
External link	`[http://example.com Link text]`
A horizontal line (<hr />)	`----`

Table with four cells on two rows	`{| class="wikitable"`
	`| Cell 1.1 || Cell 1.2`
	`|-`
	`| Cell 2.1 || Cell 2.2`
	`|}`

An itemization list	`* Item 1` `* Item 2`

Preformatted text (mind the space at the beginning of each line)	`␣This text is rendered using`
	`␣a fixed font and spaces are`
	`␣preserved.`

2.2. Stages of the Pipeline

To generate HTML from Wikitext the original document has to pass a non-trivial pipeline of processing steps. We group these steps into three stages known as *preprocessing*, *expansion* and *conversion*. The entire pipeline is illustrated in figure 1.

The first stage is called **preprocessing** and recognizes *transclusion* statements, *template parameters*, *parser functions*, *parser variables* and *tag extensions* and generates a simple abstract syntax tree (AST). This stage is dedicated only to the recognition of these elements and does not perform any conversions from Wikitext to HTML.

The second stage is called **expansion**. In this stage the elements that were recognized in the preprocessing stage are resolved by replacing the original statement

with the text from a template or the output produced by a parser function etc.

The third and last stage is called **conversion** and converts the Wikitext generated by preprocessing and expansion into *XHTML 1.0 Transitional* . The transformation from Wikitext to HTML is done in numerous smaller transformation steps that each transform the page as a whole. Since the transformation process can yield invalid XHTML due to invalid nesting of elements or missing opening or closing tags an HTML tidying step is performed after conversion.

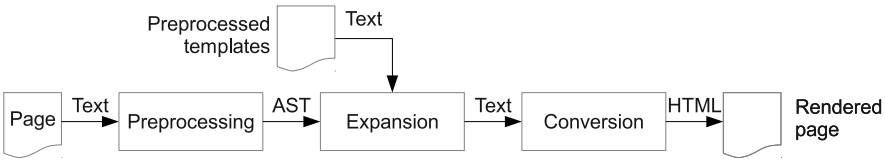


Fig. 1. The MediaWiki pipeline.

2.3. Parsing Wikitext

While the original MediaWiki software converts from Wikitext directly to HTML, recent parsers are able to generate a pure AST representation of a page. As a consequence not only does the preprocessing stage yield an AST but also will the conversion phase yield an AST containing, for example, a bold node where the original Wikitext contained bold HTML markup.

In the following discussion about pipeline reconfiguration we assume that an AST generating parser is used.

2.4. Reconfiguration of the Pipeline

The pipeline as described above is the default pipeline as is implemented in the MediaWiki software. This standard configuration is used to transform Wikitext into HTML which is then rendered by a browser for display. However, many more uses of the rich content of wiki pages emerged and they all require a machine-processable representation of the semantic content of the Wikitext:

- data analysis,
- WYSIWYG editing,
- storage,
- etc.

These use cases all have one thing in common: They want to skip the expansion process. When editing an article in a WYSIWYG editor one does not want to see expanded templates. Instead one wants to edit the original transclusion statement.

Also a transclusion statement usually contains semantic information in structured form. This structure is lost (or at least obscured) when the transclusion is performed and the statement is replaced by the template's content.

A simple solution to obtain an un-expanded representation of a page is to skip expansion. This approach will work for many pages that can be found in Wikipedia and other MediaWiki based wikis, however, not all possible pages support this approach. Skipping expansion can cause various problems. The following paragraphs until the next section will present some cases that cause such problems.

Pages used as templates are not required to be *syntactically closed* by the MediaWiki software. If, for example, a template opens a bold formatting tag but doesn't provide a matching closing tag, the bold formatting will "leak" out of the template and will also affect the Wikitext that follows the transclusion statement, as is illustrated in figure 2. When skipping expansion or performing expansion after conversion, the Wikitext following the transclusion statement will lack any formatting that would have been leaked from the template.

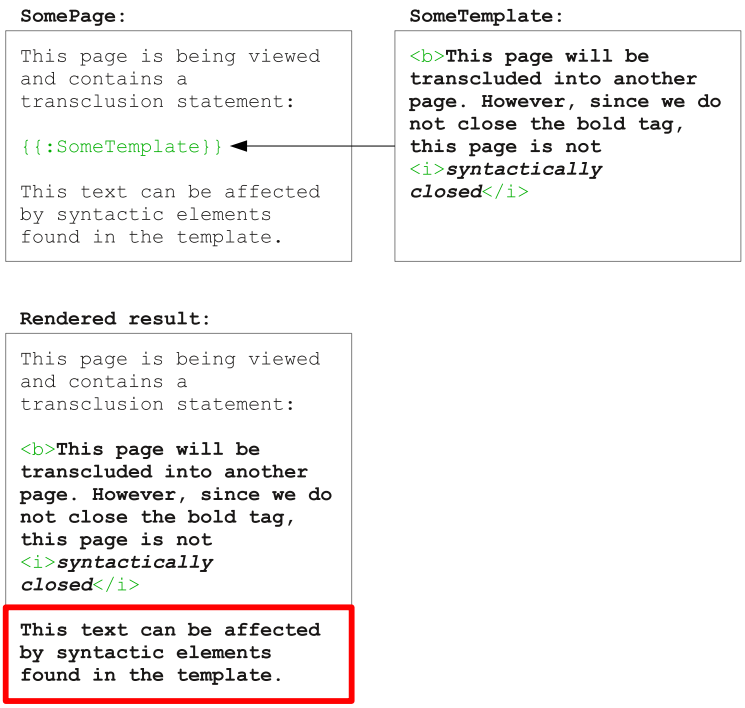


Fig. 2. Template leaking formatting information. The text after the transclusion (red box) will be rendered in bold because a bold tag was not closed in the transcluded template.

While an effect as shown in figure 2 is unintentional most of the time, this behavior of MediaWiki is often used intentionally as illustrated in figure 3.

Figure 4 illustrates the reconfigured pipeline in which expansion is left out. In this scenario every page and also every template will first be preprocessed and parsed

TablePage:

```
A table with one row
containing two cells and
another row containing a
common footer:

{|
 | Cell 1
 | Cell 2
{{TableFooter}}
```

TableFooter:

```
|-
| colspan="2" | Footer text
|}
```

Fig. 3. A template is used as common table footer. If a certain class of table occurs more than once and always has the same footer, it can be beneficial to move the footer to a template.

into an abstract syntax tree (in figure 4, the AST is further converted into XWML). However, an AST is a data structure with a rigid layout that does not allow for unclosed tags and similar syntactic anomalies.

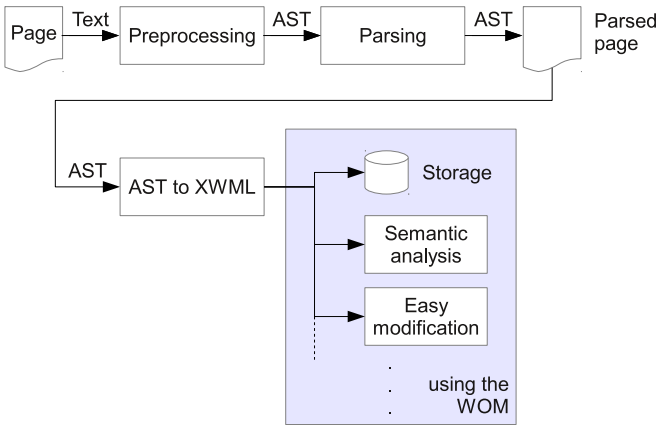


Fig. 4. Reconfigured pipeline without expansion. After parsing the AST is converted into XWML, thus facilitating further processing of the semantic content of a page based on the *Wikitext Object Model*.

Figure 5 shows the backend of the reconfigured pipeline in which the late expansion happens and the page is eventually rendered into another representation for display. When using this reconfigured pipeline with the examples from figures 2 and 3 the rendered results would differ.

In this set-up, both the page and the templates it transcludes are already converted into an AST or XWML and therefore into a syntactically closed form. As long as a bold formatting element is understood as an element with a scope, the scope has to be closed somewhere. In case of the syntactically unclosed page "SomeTemplate", the bold element will be closed at the end of the template page at the latest. As a consequence templates can no longer leak formatting into the transcluding page.

The case of the second example in figure 3 is slightly different. Here the template page contains syntactic elements (the table row marker "|-", the table end marker

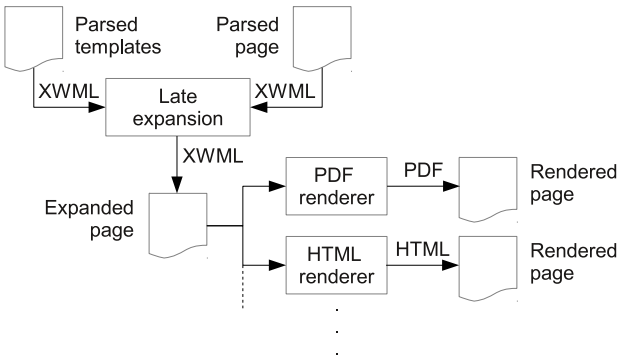


Fig. 5. Late expansion of a page. When using the reconfigured pipeline, expansion happens after parsing and is entirely based on the AST or XWML representation of a page and the templates.

"|}", etc.) that only make sense in the context of a table. But in the template page itself a table definition was never started. And in MediaWiki there is no rule that requires one to transclude such a template only into a context in which a table definition was already started. As a result, the parser will treat the special table markers as plain text. After transclusion into the page that is supposed to use the template as table footer, the respective table won't be closed by the template. Instead the content of the template will be put as plain text into the currently open table cell and the table itself will only be closed at the end of the page.

Despite these limitations described above, the proposed reconfiguration of the pipeline will be the basis of the object model presented in this report for it has strong advantages:

A document after preprocessing and parsing presents the semantically richest form possible: Before preprocessing and parsing the document is available in Wikitext which is inaccessible to machine processing. If one would perform expansion before parsing, template parameters which could have been easily extracted from the transclusion statement are now obscured in the content of the transcluded template or even irrecoverable.

Only an unexpanded article can be easily edited by authors: If templates would show up in their expanded form in an editor, authors would get lost in the transcluded content.

Storing unexpanded but parsed pages saves processing time: If a template is altered all pages that transclude this template will have to be re-parsed completely, including all other templates they depend on. If, however, one stores semantically closed ASTs, altering a template will not require the re-parsing of other pages.

2.5. Rationale

As explained, Wikitext is a front-end to HTML. Since the MediaWiki software generates its output as XHTML 1.0 Transitional, we decided to also base XWML on the design of the XHTML specification. The details of XWML and how it was based on XHTML are layed out in section 3.

We assume that XWML is generated from an AST after the conversion stage but with expansion skipped. As a consequence we expect that only syntactically closed templates will be used in the Wikitext.

There is no syntactically wrong or otherwise illegal Wikitext. This is because the original MediaWiki parser accepts all Wikitext and will generate HTML output for any input. However, since the specification presented in this report defines a strict content model there is Wikitext that cannot be represented in XWML. **This is intentional.** Software that converts from Wikitext to XWML has the liberty to choose how to deal with Wikitext that cannot be represented in XWML (by, for example, ignoring illegal parts of the original Wikitext). In this report we consider Wikitext that cannot be represented in XWML as **syntactically invalid Wikitext**.

2.6. State of this document

This technical report is our first attempt at providing a feature-complete, formal specification of Wikitext. While we believe that our report is already feature-complete in the sense that it covers everything one can express with Wikitext, we are aware of the fact that our specification of semantics and restrictions is not yet complete. For example we do not yet specify how exactly a string of type Username has to be formatted. Instead we refer the reader to the MediaWiki software that implicitly defines restrictions on user names. The same holds for images and other Wikitext specific tags. How an image has to be rendered in order to conform with MediaWiki is not specified in detail yet.

We hope that our initial work will encourage the community around Wikipedia and MediaWiki to join in and help us to eventually provide a complete and formal specification of Wikitext and its features.

3. Specification

This section defines the Wikitext Object Model. The WOM consists of a content model similar to that of HTML, a markup language, XWML, that describes the content of a page written in Wikitext and a set of Java interfaces that define how to access and modify an article inside a program. The content model of WOM is specified mostly in terms of XML Schema. The elements are also mostly specified in terms of XML Schema.

The specification starts with the introduction of the simple data types of WOM in section 3.1. Section 3.2 will given an overview of the content model of WOM, followed by a listing of the elements of XWML that are taken from XHTML 1.0 Transitional in section 3.6. This is followed by a detailed description of the new elements introduced by XWML. Finally, the Java interfaces are explained in section 3.8.

3.1. Simple Types and Enumerations

This section introduces the simple data types defined by WOM. These data types appear explicitly in XWML but are also used by the Java interfaces. However, the Java interfaces don't define a counterpart for every type introduced here but map some of those data types to one of Java's simple data types.

Only simple data types not already defined in XHTML 1.0 Transitional will be explicitly defined in this section. For XHTML data types please refer to [8]. If in the remainder of this report a simple data type is referenced that is already defined in XHTML 1.0 Transitional, the data type will be labeled with the namespace *xhtml*, as for example in "*xhtml:Pixels*".

The following simple data types are defined and used only by the XHTML 1.0 Transitional portion of WOM:

- xhtml:Character
- xhtml:Color
- xhtml:LanguageCode
- xhtml:Length
- xhtml:LIStyle
- xhtml:Number
- xhtml:OLStyle
- xhtml:Script
- xhtml:StyleSheet
- xhtml:ULStyle

The following simple data types are defined by XHTML 1.0 Transitional but are also used by WOM-only elements:

- xhtml:Datetime
- xhtml:Pixels
- xhtml:Text
- xhtml:URI

The simple data types are derived from the built-in types of the XML Schema language [9]. In this section as well as in the remainder of this report XML Schema data types are referenced using the namespace *xs*, as for example in "*xs:string*".

3.1.1. HeadingLevel

Designates the level of a heading from 1 (most important) to 6 (least important).

Base type:	*xs:nonNegativeInteger*
Restriction:	Restricted to the interval $[1, 6]$

3.1.2. MagicWord

The name of a MediaWiki magic word.

Base type:	*xs:string*
Restriction:	Restricted to valid names for magic word as implicitly defined by the MediaWiki software.

3.1.3. Namespace

The name of a MediaWiki namespace.

Base type:	*xs:string*
Restriction:	Restricted to valid MediaWiki namespace names implicitly defined by the MediaWiki software.

3.1.4. PageTitle

A MediaWiki page title possibly including a namespace and a path (of subpages).

Base type:	*xs:string*
Restriction:	Restricted to valid MediaWiki page titles implicitly defined by the MediaWiki software.

3.1.5. ImageFormat

Specifies how to display an image.[1]

[1] See http://www.mediawiki.org/wiki/Help:Images#Format (June 2011)

Base type: *xs:token*

Restriction: Restricted to the following values:

Value	Description
unrestrained	Render as inline image. The image can be reduced and enlarged to any size.
frameless	Render as inline image. Respect user preferences for image width.
thumbnail	Render as floating image. The image size can be reduced but can not be enlarged beyond the original image size.
frame	Render as floating image. The user preferences will be respected, size specification will be ignored.

3.1.6. ImageHAlign

Specifies how an image is aligned horizontally.[2]

Base type: *xs:token*

Restriction: Restricted to the following values:

Value	Description
default	The image will be rendered inline where the image tag occurs.
none	The image will be rendered as block element aligned to the left side of the page.
left	The image will be rendered as floating block element aligned to the left side of the page.
center	The image will be rendered as block element centered on the page.
right	The image will be rendered as floating block element aligned to the left side of the page.

3.1.7. ImageVAlign

Specifies how the image is aligned vertically.[3]

[2] See http://www.mediawiki.org/wiki/Help:Images#Horizontal_alignment (June 2011)

[3] See http://www.mediawiki.org/wiki/Help:Images#Vertical_alignment (June 2011)

Base type: *xs:token*

Restriction: Restricted to the following values:

Value	Description
baseline	Align the bottom of the image with the baseline of the text.
sub	Align the bottom of the image with the baseline for subscript text.
super	Align the bottom of the image with the baseline for superscript text.
top	Align the top of the image with the top of the tallest element on the line.
text-top	Align the top of the image with the top of the current font.
middle	Place the image in the middle of the current line.
bottom	Align the bottom of the image with the bottom of the lowest element on the line.
text-bottom	Align the bottom of the image with the bottom of the current font.

3.1.8. SignatureFormat

Specifies how to render a signature.[4]

Base type: *xs:token*

Restriction: Restricted to the following values:

Value	Description
user	Render only the username (as link to the user's page), e.g.: `Username`
timestamp	Render only the timestamp, e.g.: `12:34, 1 February 2008 (UTC)`
user-timestamp	Render username (as link to the user's page) and timestamp, e.g.: `Username 12:34, 1 February 2008 (UTC)`

3.1.9. Username

A MediaWiki username.

Base type: *xs:string*

Restriction: Restricted to valid MediaWiki usernames implicitly defined by the MediaWiki software.

3.1.10. Version

A version is given as numbers separated by dots. Furthermore, a version can have a suffix that is separated from the version number by a dash.

[4] See http://www.mediawiki.org/wiki/Help:Signatures (June 2011)

Base type:	*xs:string*
Restriction:	Restricted to the pattern `[0-9]+(\.[0-9]+)*(-[A-Za-z][A-Za-z0-9_]*)?`

3.2. Content model

An important concept in the specification of WOM is the content model as it can also be found in HTML. The content model describes which elements can occur where in the content and is detailed in the *Child elements* part of each element's description. The structure of the children is given as a regular expression with the following syntax and operators:

$e*$	Kleene-Closure: Zero or more occurences of e.
e_{opt}	Optional: Zero or one occurence of e.
$e_1\ e_2 \ldots e_n$	Sequence: First e_1, then e_2, ..., finally e_n.
(e)	Grouping
$\{e_1, e_2, \ldots, e_n\}$	Set: One of e_1, e_2, ..., e_n.
$[s]$	Set: One element out of the set s.
$s_1 \setminus s_2$	Set difference: All elements in s_1, but not in s_2.
<any>	Any element.

The regular expression in the *Child elements* field can be prefixed by the MIXED flag. The presence of this flag indicates that the child elements of that element can be mixed with plain text (the equivalent of the *mixed* attribute on *xs:complexType* elements in XML schema)

Transclusions, tag extensions and parser functions can appear nearly everywhere in Wikitext. And they can even expand to Wikitext that is syntactically illegal when it comes to conversion from Wikitext to XWML. Consider the following piece of Wikitext:

```
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
  {{SomeTemplate}}
</ul>
```

A unordered list contains two explicit list items and then a transclusion statement. According to the original XHTML 1.0 Transitional specification unordered lists can only contain `<li>` elements. The extended content model of XWML also allows preprocessor statements to occur in such lists. However, in order to guarantee that expansion again yields a valid XWML document such transclusion statements are required to resolve to content that is valid in the respective place in the document.

In the case of the above-mentioned list whose content model allows *([Preprocessor elements]|li)** as child elements (see section 3.6), a transclusion statement must

resolve to *li**.

Therefore, the rule for all preprocessor statements is that they must only resolve to those elements of XWML that are valid in the location where the transclusion statement occurs.

3.3. Attribute groups

WOM does not define own attribute groups, however, it relies on attribute groups defined in XHTML 1.0 Transitional. Below is a list of attribute groups that are not only needed by XHTML 1.0 Transitional but are also referenced by some of the non-XHTML elements of WOM. Please refer to [9] for a detailed description of these attributes.

3.3.1. Core attributes

- id
- class
- style
- title

3.3.2. I18n attributes

- lang
- dir

3.3.3. Event attributes

- onclick
- ondblclick
- onmousedown
- onmouseup
- onmouseover
- onmousemove
- onmouseout
- onmousepress
- onkeydown
- onkeyup

3.3.4. Universal attributes

A group containing the attributes that can be specified for almost all XHTML elements. This group corresponds to *xhtml:attrs*.

- [Core attributes]
- [I18n attributes]
- [Event attributes]

3.4. Element Groups

Element groups help in the specification of the content model of WOM. Certain elements only accept certain other elements as their children. To simplify the specification of a set of accepted children elements are assigned to element groups.

The element groups defined in this section are an extension of the element groups found in XHTML 1.0 Transitional. Compared to XHTML the [Preprocessor elements] group was added and elements have been added or removed from the other element groups, as shown below.

3.4.1. Preprocessor elements

- comment
- magicword
- param
- tagext
- transclusion

3.4.2. Inline fontstyle elements

- b
- big
- font
- i
- s
- small
- strike
- tt
- u

3.4.3. Inline phrase elements

- abbr
- cite
- code
- dfn
- em
- kbd
- samp
- strong
- sub
- sup
- var

3.4.4. Inline link elements

- extlink
- intlink
- url

3.4.5. Inline miscellaneous elements

- br
- element
- image
- nowiki
- signature
- span

3.4.6. Edit elements

- del
- ins

3.4.7. Inline elements

- [Preprocessor elements]
- [Edit elements]
- [Inline fontstyle elements]
- [Inline phrase elements]
- [Inline link elements]
- [Inline miscellaneous elements]

3.4.8. Block list elements

- dl
- ol
- ul

3.4.9. Block preformatted elements

- pre
- semipre

3.4.10. Block miscellaneous elements

- blockquote
- center
- div
- element
- hr
- p
- table

3.4.11. Block elements

- [Preprocessor elements]
- [Edit elements]
- [Block list elements]
- [Block preformatted elements]
- [Block miscellaneous elements]

3.4.12. Flow elements

- [Inline elements]
- [Block elements]

3.4.13. Miscellaneous elements

- arg (← *transclusion*)
- attr (← *tagext*)
- body (← *page*, *section*)
- caption (← *table*)
- category (← *page*)
- dd (← *dl*)
- default (← *param*)
- dt (← *dl*)
- elembody (← *element*)
- heading (← *section*)

- imgcaption (← *image*)
- li (← *ol*, *ul*)
- name (← *arg*, *param*, *transclusion*)
- page
- redirect (← *page*)
- section (← *body*)
- tagextbody (← *tagext*)
- tbody (← *table*)
- td (← *tr*)
- th (← *tr*)
- title (← *extlink*, *extlink*)
- tr (← *tbody*)
- value (← *arg*)

3.5. Element Content Groups

Element content groups help to give an overview of what elements accept which kind of child elements.

3.5.1. Inline content elements

This group subsumes all elements that accept only inline elements as their child elements.

- abbr
- b
- big
- caption
- cite
- code
- dfn
- div
- dt
- em
- font
- heading
- i
- kbd
- p
- s
- samp
- semipre*
- small
- span
- strike
- strong
- sub
- sup
- title*
- tt
- u
- var

Legend:

* These elements don't accept all inline elements. See the respective element specifications for details.

3.5.2. Block content elements

This group subsumes all elements that accept only block elements as their child elements.

- blockquote
- body
- center
- dd
- imgcaption
- li
- td
- th

3.5.3. Flow content elements

This group subsumes all elements that accept only inline or block elements as their child elements.

- del
- div
- ins

3.5.4. Preprocessor content elements

This group subsumes all elements that accept only elements generated in the preprocessor stage as child elements.

- default
- name
- value

3.6. XHTML Element Reference

Wikitext allows the use of a subset of HTML in a Wikitext page. When rendering such a page as HTML these elements are written directly to the XHTML 1.0 Transitional output and therefore behave exactly as specified in the XHTML 1.0 Transitional specification. Instead of repeating the specification of all allowed HTML elements, we refer the reader to the original specification [8], [10] and point out the differences.

Since WOM specifies its own content model, what changes for the HTML elements as specified by XHTML 1.0 Transitional is the context in which an HTML element can occur and the elements that an HTML element can contain as content.

The contexts in which certain HTML elements can occur in WOM is defined by element groups in section 3.4. These element groups are used in the XML Schema specification in appendix A as well as in this section to define the content of an element.

The differences between the original XHTML elements and their WOM equivalents are:

1) WOM uses a different definition of the [Inline elements], [Block elements] and [Flow elements] content types as opposed to the original *xhtml:Inline*, *xhtml:Block* and *xhtml:Flow* types.
2) To allow unified access to the textual content of a page, WOM requires text to always appear inside p elements. As a consequence, WOM requires the [Block elements] content model where XHTML 1.0 Transitional allows the *xhtml:Flow* content model.
3) Preprocessor elements are unique to Wikitext and not known to XHTML. While the [Inline elements], [Block elements] and [Flow elements] groups

already incorporate preprocessor elements, they have to be added to those XHTML elements that accept neither of these three types of content.

4) While Wikitext does not support the <tbody> element we still support it and disallow <tr> elements inside a <table> element. We do this to prevent the mixture of table row elements and the body element that can be found in HTML. However, until Wikitext supports the <tbody> element it will be treated like a pure container element with no attributes.

The following elements in WOM are taken from XHTML. The definitions found in XHTML 1.0 Transitional apply, however, the content of these elements has to be redefined as is done in the following enumeration:

3.6.1. abbr

Child elements: [Inline elements]*

3.6.2. b

Child elements: [Inline elements]*

3.6.3. big

Child elements: [Inline elements]*

3.6.4. blockquote

Child elements: [Block elements]*

3.6.5. br

Child elements: –

3.6.6. caption

Child elements: [Inline elements]*

3.6.7. center

Child elements: [Block elements]*

3.6.8. cite

Child elements: [Inline elements]*

3.6.9. code

Child elements: [Inline elements]*

3.6.10. dd

Child elements: [Block elements]*

3.6.11. del

Child elements: [Flow elements]*

3.6.12. dfn

Child elements: [Inline elements]*

3.6.13. div

Child elements: [Flow elements]*

3.6.14. dl

Child elements: ([Preprocessor elements]|dt|dd)*

3.6.15. dt

Child elements: [Inline elements]*

3.6.16. em

Child elements: [Inline elements]*

3.6.17. font

Child elements: [Inline elements]*

3.6.18. hr

Child elements: –

3.6.19. i

Child elements: [Inline elements]*

3.6.20. ins

Child elements: [Flow elements]*

3.6.21. kbd

Child elements: [Inline elements]*

3.6.22. li

Child elements: [Block elements]*

3.6.23. ol

Child elements: ([Preprocessor elements]|li)*

3.6.24. p

Child elements: [Inline elements]*

3.6.25. s

Child elements: [Inline elements]*

3.6.26. samp

Child elements: [Inline elements]*

3.6.27. small

Child elements: [Inline elements]*

3.6.28. span

Child elements: [Inline elements]*

3.6.29. strike

Child elements: [Inline elements]*

3.6.30. strong

Child elements: [Inline elements]*

3.6.31. sub

Child elements: [Inline elements]*

3.6.32. sup

Child elements: [Inline elements]*

3.6.33. table

Child elements: ([Preprocessor elements]|caption)$_{opt}$([Preprocessor elements]|tbody)$_{opt}$

3.6.34. tbody

Child elements: ([Preprocessor elements]|tr)*

3.6.35. td

Child elements: [Block elements]*

3.6.36. th

Child elements: [Block elements]*

3.6.37. tr

Child elements: ([Preprocessor elements]|th|td)*

3.6.38. tt

Child elements: [Inline elements]*

3.6.39. u

Child elements: [Inline elements]*

3.6.40. ul

Child elements: ([Preprocessor elements]|li)*

3.6.41. var

Child elements: [Inline elements]*

3.7. Element Reference

In this section all elements that are introduced in WOM and are not already part of XHTML 1.0 Transitional will be defined. For elements that are already defined in XHTML (e.g.: `<table>`) please refer to [8].

Most of the elements introduced by WOM don't possess the [Universal attributes] that are common to most XHTML elements. Those elements that do accept [Universal attributes] usually do so because Wikitext sometimes offers two ways to specify a certain element: via native Wikitext syntax or as XHTML element.

As in XHTML, boolean properties are specified using empty attributes in XWML. These attributes are either present (*true*) or absent (*false*). When present the value of these attributes has to be set to their name, e.g.: `upright="upright"`.

3.7.1. arg

The arg element specifies an argument to a template page. arg elements are specified as part of a transclusion statement.

Parent elements:	transclusion
Child elements:	name_{opt} value

3.7.2. attr

Describes an attribute that is passed to a tag extension or an arbitrary XML element. Both name and value of the attribute are given as attributes.

Parent elements: element, tagext
Child elements: –

Attribute	Type	Required	Description
name	*xs:Name*	yes	The name of the attribute.
value	*xs:string*	yes	The value of the attribute.

3.7.3. body

General node to hold block level content in *page* and *section* nodes.

Parent elements: page, section
Child elements: [Block elements]*

3.7.4. category

Adds the page to a category.

Parent elements: page
Child elements: –

Attribute	Type	Required	Description
category	PageTitle	yes	The category to which this page should be added.

3.7.5. comment

Holds an XML-like comment found in Wikitext.

Parent elements: [Flow content elements], [Preprocessor content elements]
Child elements: *xhtml:Text*

3.7.6. default

Holds the default value of a template parameter. If no argument is specified in the transclusion statement the default value will be used as parameter value.

Parent elements: param
Child elements: MIXED [Preprocessor elements]*

3.7.7. element

Represents an arbitrary XML element.

Parent elements: [Flow content elements]
Child elements: attr* elembody$_{opt}$

Attribute	Type	Required	Description
name	*xs:Name*	yes	The name of the XML element.

3.7.8. elembody

The content of an arbitrary XML tag specified by element.

Parent elements: element
Child elements: MIXED <any>*

3.7.9. extlink

Describes a bracketed external link.

Parent elements: [Inline content elements]
Child elements: title$_{opt}$

Attribute	Type	Required	Description
target	*xhtml:URI*	yes	The target URL to which this link points.

3.7.10. heading

Holds the heading of a section. If a heading was specified using HTML syntax (<h1> – <h6>) this element holds the attributes specified on the respective HTML element while the level of the heading (indicated by the number after the h) is stored at the section element.

Parent elements: section
Child elements: MIXED [Inline elements]*

Attribute	Type	Required	Description
align	*xhtml:TextAlign*	no	Text alignment for the heading
[Universal attributes]			

3.7.11. image

Describes a Wikitext image.

Parent elements: [Inline content elements]
Child elements: imgcaption$_{opt}$

Attribute	Type	Required	Description
source	PageTitle	yes	The page title of the image to render.
format	ImageFormat	no	Specifies how the image should be rendered. **Default:** `"unrestrained"`.
border	`"border"` (empty attribute)	no	Specifies if the image should be rendered with a border.
halign	ImageHAlign	no	The horizontal alignment of the image. **Default:** `"right"` when using the *thumbnail* or *frame* format, otherwise `"default"`.
valign	ImageVAlign	no	The vertical alignment of the image. Only applies to inline images that are not floating. **Default:** baseline.
width	*xhtml:Pixels*	no	The width of the image in pixels.
height	*xhtml:Pixels*	no	The height of the image in pixels.
upright	`"upright"` (empty attribute)	no	Resize the image according to user preferences.
urllink	*xhtml:URI*	no	If given the image will link to the given URL instead of linking to the wiki page of the image. This attribute and "pagelink" are mutually exclusive.
pagelink	PageTitle	no	If given the image will link to the given wiki page instead of linking to the wiki page of the image. This attribute and "urllink" are mutually exclusive.
alt	*xhtml:Text*	no	An alternative text for the image. If not specified the image name will be used as alternative text for the image.

The child element imgcaption is only allowed for framed images (see ImageFormat).

3.7.12. imgcaption

Describes the caption of an image.

Parent elements: image
Child elements: MIXED [Inline elements]*

3.7.13. intlink

Describes an internal Wikitext link.

Parent elements: [Inline content elements]
Child elements: title$_{opt}$

Attribute	Type	Required	Description
target	PageTitle	yes	The page title of the image to render.

3.7.14. magicword

Describes a magic word.

Parent elements: [Flow content elements]
Child elements: –

Attribute	Type	Required	Description
name	MagicWord	yes	The name of the magic word.

3.7.15. name

name elements specify the name of arg, param and transclusion elements. After expansion the content of a name element must evaluate to a string that forms a valid name. What constitutes a valid name depends on the element in which the name element occurs.

Parent elements: arg, param, transclusion
Child elements: MIXED [Preprocessor elements]*

3.7.16. nowiki

Contains text that must not be interpreted as Wikitext.

Parent elements: [Inline content elements]
Child elements: *xhtml:Text*

3.7.17. page

The root element of a page.

Parent elements: –
Child elements: redirect_{opt} category* body

Attribute	Type	Required	Description
version	Version	yes	The XWML version. Fixed to "1.0".
namespace	Namespace	no	The namespace in which the page resides.
path	PageTitle	no	A path of subpages that leads to the page.
title	PageTitle	yes	The title of the page.

3.7.18. param

A template parameter that will be replaced by an argument passed to the respective transclusion statement.

Parent elements: [Flow content elements], [Preprocessor content elements]
Child elements: name default$_{opt}$

3.7.19. pre

Describes preformatted content that must not be parsed and will be displayed with a fixed font.

Parent elements: [Block content elements]
Child elements: *xhtml:Text*

Attribute	Type	Required	Description
[Universal attributes]			

3.7.20. redirect

Indicates that this page is a redirection page and declares the target page of the redirection.

Parent elements: page
Child elements: –

Attribute	Type	Required	Description
target	PageTitle	yes	The target page of the redirection.

3.7.21. section

A section consisting of heading and body.

Parent elements: body
Child elements: heading body

Attribute	Type	Required	Description
level	HeadingLevel	yes	The level of the heading from 1 (most important) to 6 (least important).

3.7.22. semipre

Describes preformatted, parsed content that will be displayed with a fixed font. The semipre element must not contain the elements image, big, small, sub, sup or font.

Parent elements: [Block content elements]
Child elements: MIXED ([Inline elements]\{image, big, small, sub, sup, font})*

3.7.23. signature

Describes a signature.

Parent elements: [Inline content elements]
Child elements: –

Attribute	Type	Required	Description
format	SignatureFormat	yes	How the signature should be formatted.
author	Username	yes	The name of the author making the signature.
timestamp	*xhtml:Datetime*	yes	The time of the signature.

3.7.24. tagext

An invocation of a tag extension.

Parent elements: [Flow content elements]
Child elements: attr* tagextbody$_{opt}$

Attribute	Type	Required	Description
name	*xs:Name*	yes	The name of the tag extension.

3.7.25. tagextbody

The textual content inside the invocation of a tag extension.

Parent elements: tagext
Child elements: *xhtml:Text*

3.7.26. title

Holds the title of an external or internal link. This element must not be empty or contain only whitespace. Elements from [Inline link elements] must not appear as children of a title element.

Parent elements: extlink, intlink
Child elements: MIXED ([Inline elements]\[Inline link elements])*

3.7.27. transclusion

Transcludes another page (called *template*) into the page in which the transclusion statement appears. The template is preprocessed and expanded before the transclusion statement is replaced by the textual content of the template.

Parent elements: [Flow content elements], [Preprocessor content elements]
Child elements: name arg*

3.7.28. url

Describes a plain URL.

Parent elements: [Inline content elements]
Child elements: –

Attribute	Type	Required	Description
target	*xhtml:URI*	yes	The target URL to which this link points.

3.7.29. value

value elements specify the value of an arg element.

Parent elements: arg
Child elements: MIXED [Preprocessor elements]*

3.8. Java Interface reference

The following sections present the Java interfaces that complete the Wikitext Object Model. The actual interfaces can be found in appendix B. The following goals drove the design of the interfaces:

- Develop a general and familiar interface.
- Hide the subtleties of Wikitext behind a simple but powerful interface.
- Don't loose the original Wikitext representation. Once a wiki page was modified using the WOM interfaces, it should be possible to write out the result in Wikitext and **not** loose the user's formatting in areas where no modification took place.

The following sections will give an overview of how we achieve these goals.

3.8.1. The WOM tree

Wikitext as a language is closely related to HTML. And as HTML offers its *Document Object Model* (DOM) as a way to represent and work with an HTML document inside a program, we provide the *Wikitext Object Model* (WOM) to represent and work with a Wikitext page.

As in XWML, the individual elements of the Wikitext Object Model form a tree, the *WOM tree*. In XWML the elements are given as XML elements in an XML document. In case of the WOM tree, the elements are represented by Java objects that are connected to a tree-like data structure. Although the XML elements of XWML and the Java interfaces of the WOM are closely related there is no

one-to-one mapping between the two representations. XWML purely serves to represent the content of a Wikitext page for storage or exchange. The Java interfaces additionally provide a way to easily access and modify the document stored in the WOM tree inside a program. To meet this requirement, the Java interfaces are not an exact counterpart of the XML elements of XWML.

3.8.2. The WomNode interface

Every node in the WOM tree inherits from the WomNode interface which provides low-level access to the WOM tree. As illustrated in figure 6 there are three navigation axes in a WOM tree:

Vertical:
: When on a node x one can go up to its parent node $parent(x)$ or go down to its first child $firstChild(x)$ or last child $lastChild(x)$,

Horizontal:
: When on a node x one can go left $prevSibling(x)$ or right $nextSibling(x)$ to the sibling nodes that are children of the same parent node $parent(x)$,

Attributes:
: When on a node x one can query a set of attributes $attrs(x)$ associated with that node.

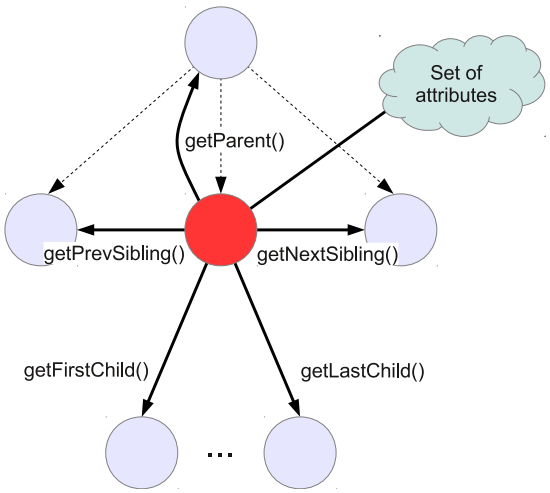


Fig. 6. Navigation in the WOM tree, starting on the red node.

This low-level interface allows arbitrary modifications of the WOM tree that includes illegal modifications that are not allowed, for example, by the content model specified in section 3. Examples for illegal modifications that are supported by the low-level interface are:

- Set an attribute to an unsupported value.

- Add an attribute that is not supported by the node.
- Assign child nodes to a node that does not support child nodes.

This behavior guarantees downwards compatibility with today's Wikitext documents which are not required to validate against XHTML or XWML.

3.8.3. Attributes

As in HTML, WOM attributes are name/value pairs. And like in HTML, the names of WOM attributes must form legal XML names and are case-insensitive.

In a WOM tree, attributes are also nodes of type WomNode. They have a parent node (the node they are attached to) and siblings but no children. While one can iterate through the attributes $A = attrs(x)$ of node x by navigating with $prevSibling(a), a \in A$ and $nextSibling(a)$, the order in which the attributes are retrieved is not guaranteed.

Also there can only be one attribute with a certain name for a given node x, that is $\forall_{n_a,n_b \in attrs(x) | n_a \neq n_b} : lc(name(n_a)) \neq lc(name(n_b))$, where $lc(s)$ returns the lower-case version of the string s.

3.8.4. The high-level interfaces

The Java representation also offers high-level interfaces to the different kinds of nodes. Examples for such interfaces are:

- WomList
- WomTransclusion
- WomHorizontalRule

The WomList node is an abstraction for list nodes like WomOrderedList or WomUnorderedList. The other two interfaces provide access to concrete elements of Wikitext.

These high-level interfaces offer better access to the attributes and content of specific nodes or kinds of nodes. There are, however, differences between the low-level interface and the high-level interfaces:

- Using the high-level interfaces one can only perform *legal* modifications.
- The high-level interface offers a view of only the *valid* parts of a Wikitext page.

This shall be illustrated using the abstract WomList interface as an example that is implemented by the WomOrderedList and WomUnorderedList interfaces. Consider the following Wikitext snippet:

```
1  This is a Wikitext bullet list:
2  * Item 1
3  * Item 2
4
5  This is a similar list, however, realized using HTML tags:
6  <ul>
7  <li>Item 1</li>
8  A piece of text that really should not be
9  here according to the XHTML content model.
10 <li>Item 2</li>
11 </ul>
```

Lines 1 and 2 contain a Wikitext-style unordered list (bullet list) while lines 6–11 contain a HTML-style unordered list. First thing to notice is that, although both lists use different representations in Wikitext (native Wikitext vs. HTML), they will both be represented by an instance of the WomUnorderedList interface in the WOM tree. As will be discussed in section 3.8.5 the underlying, distinctive representation is not lost but hidden away as long as one only uses the WOM tree to traverse the document.

Further, we can see that in the HTML-style list textual content is part of the surrounding `<ul>` element which is not allowed by the XHTML content model. However, Wikitext does not prohibit this and we therefore have to accept it as valid Wikitext. In the native Wikitext version of the list this is not possible. One could violate the XHTML content model by inserting illegal elements into the individual items (equivalent to inserting illegal elements into the `<li>` elements) but one cannot insert illegal elements in between the items of a native Wikitext bullet list.

In order to offer high-level access to the lists shown in the above code snippet, the list interface WomList (which is extended by WomOrderedList and WomUnorderedList) offers an index based view of the list. The interface offers the following (simplified) method signatures for content access and modification:

- `int getItemNum()`
- `Collection<WomListItem> getItems()`
- `WomListItem getItem(int index)`
- `WomListItem replaceItem(int index, WomListItem item)`
- `WomListItem removeItem(int index)`
- `void appendItem(WomListItem item)`
- `void insertItem(int beforeIndex, WomListItem item)`

Using the above methods one can access and modify individual items of the list using an index to point at individual items. The two lists in the above Wikitext snippet assign the same indices to their items (when accessed through the WomList interface). That is to say that in both cases *Item 1* is addressed by the index 0 and *Item 2* is addressed by the index 1. Although the second HTML-style list contains text content in between the two items. The reason that *Item 2* is still addressed with index 1 in the HTML-style list is that the high-level interfaces offer a view of the valid parts of the underlying document. As a consequence, the high-level

interface for a list ignores the textual content which is not supposed to be there. Yet, using the low-level interface that is implemented by all high-level interfaces one can still access and modify these illegal elements.

On the other hand, one also cannot perform illegal operations through a high-level interface. Either because the high-level interface doesn't offer an opportunity (given the methods above one only can add proper list items to the list and nothing else) or because an exception is thrown if one tries to perform a modification that is not allowed by the content model.

Further, the WomList interface extends the WomUniversalAttributes interface which offers access to the universal attributes common to most HTML elements. And WomList also provides methods to modify those HTML attributes which are specific to HTML lists:

Excerpt of the attribute accessor methods in WomList:

- `String getItemType()`
- `String setItemType(String type)`

  ...

Excerpt of the attribute accessor methods in WomUniversalAttributes:

- `String getId()`
- `String setId(String id)`
- `String getStyle()`
- `String setStyle(String style)`

  ...

Again, the same rules apply as for the content of the list: Illegal modifications of attributes are not possible through a high-level interface and illegal attributes are ignored by the high-level interface. In the case of attributes this means that one can only set attributes that are allowed by the content model and one can set these attributes only to values allowed by the content model. To be more precise:

- Setting an attribute to an illegal value raises an exception.
- Retrieving an attribute that is set to an illegal value (i.e. through the low-level interfaces) will retrieve the default value for the attribute or pretend that the attribute is not given for the element if the XHTML standard does not assign a default value to that particular attribute.

Access and modification of illegal attributes is possible through the low-level interface.

Finally, the WomList interface extends the WomNode interface and thus also offers low-level access to the list and its content.

Through this separation into low-level and high-level interfaces legacy Wikitext is preserved and one can still work with the WOM tree on a higher level of abstraction that offers a consistent view of the content.

3.8.5. The data structure underlying the WOM tree

This technical report provides a specification for the interfaces of the WOM tree but no specific implementation for the individual interfaces. There are different ways to implement the actual data structure behind the interfaces. Here we want to define two different data structures that can form the backbone of the WOM tree:

- an XWML-like data structure and
- an *Abstract Syntax Tree* (AST) backed data structure.

XWML-like data structure: The XWML-like data structure is a straightforward implementation of the capabilities also found in the actual XWM language. It is most useful when working with an article that is only available in XWML or if one only wants to perform semantic transformations on the content and has no need to write the resulting document back to Wikitext.

The XWML-like data structure encodes the full semantic content of a Wikitext page and can be rendered into different formats (like HTML or PDF) without loosing semantic information. However, it cannot be rendered back into the original Wikitext, since syntactic formatting information cannot be stored in a data structure similar to XWML and is therefore also not encoded in the XWML-like data structure.

On the technical level, the XWML-like data structure stores the attributes, sibling and child information in each implementation of an interface. One could also go as far as to say that individual nodes only provide fields for those attributes and children allowed by the XWML content model to get a lean and resource efficient implementation of the WOM. However, such a lean representation can only represent invalid Wikitext in a limited way.

AST backed data structure: The data structure backed by an *Abstract Syntax Tree* (AST) is more complex and powerful than the XWML like data structure. It assumes that the WOM tree is based on the parse tree (i.e. AST) of a Wikitext document and switching between these two representations is possible even after applying modifications to the WOM tree representation. As a result, syntactic formatting information of the original Wikitext is preserved in those areas of the document where no modifications took place.

Consider the following Wikitext:

```
1 [[File:Example.jpg|50xpx|Sunflower]]
```

Using the Sweble Wikitext parser [6] one can produce an AST of the above Wikitext that is illustrated in figure 7. The white circles are AST nodes, the yellow boxes present the properties of the individual nodes that contain the semantic information

extracted from the Wikitext and the gray boxes contain the original Wikitext representation of the respective node.

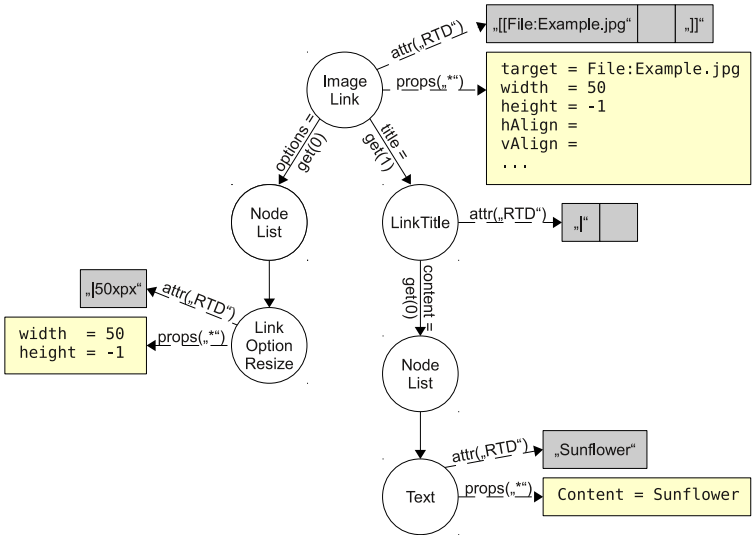


Fig. 7. AST node of an internal link as produced by the Sweble parser [5]

The AST backed WOM implementation does not have to duplicate the properties already attached to the AST nodes. Instead the individual nodes of the WOM tree point to their respective counterpart AST node. This isn't always a one-to-one mapping. A single WOM text node can represent multiple AST text nodes, for example. And some WOM nodes like the WomTableColumn node do not have any representation in the AST. The LinkOption nodes (like the LinkOptionResize node in figure 7) don't have a representation in the WOM tree. They have to be kept in the AST to remember the order in which the options were given in the original Wikitext.

When modifying the WOM tree, the changes are written back to the linked AST representation immediately. However, performing the modification through the WOM interface simplifies the task significantly. In the above example, if one wants to change the width attribute of the image, one calls the method

```
public Integer setWidth(Integer width)
```

with a non-`null` value on the WomImage object. If one wants to remove the width attribute, one calls the same method with a `null` value. The WOM tree implementation then takes care of altering or removing the respective AST nodes.

The one major advantage of the AST backed implementation of the WOM tree is the preservation of the original Wikitext. By switching from WOM tree to AST one can reconstruct the original Wikitext document by traversing the AST and printing out the values in the Round-Trip Data (RTD) attributes (see [6] for details). If

modifications were applied to the WOM tree, only those AST nodes affected by the changes will change their RTD attributes.

If, for example, one would change the *width* attribute of the image to 70 pixels, the width property in the *ImageLink* node would be adapted and the width property as well as the RTD attribute in the *LinkOptionResize* node would be adapted. At this point, the RTD attribute of the LinkOptionResize node will change to "|70xpx" or "|70px", which is an alternative syntax for "|70xpx". Which representation is chosen depends on the WOM tree implementation. It is, however, guaranteed that the formatting of the remaining document will be preserved.

3.8.6. Normalization

Normalization is performed to simplify and standardize the WOM tree. The following rules apply:

Text nodes:
: Depending on the parse tree (AST) that backs the WOM tree, text can appear as multiple successive text nodes instead of one text node. When the WOM tree is in its normalized form successive text nodes in the AST are always represented as a single text node in the WOM tree.

XML entities:
: XML entities are not represented as XML entities in the WOM tree. Instead they are resolved and appear as simple text. As a consequence, if a text node contains, for example, an ampersand (`&`), one cannot deduce from the WOM tree whether this character was specified as XML entity or as plain ampersand character.

Newlines:
: All Unicode [11] newline character sequences (`U+000A`, `U+000B`, `U+000C`, `U+000D`, `U+000D U+000A`, `U+0085`, `U+2028`, `U+2029`) are represented by the character `U+000A`. This is to say that when querying a text node that represents Wikitext containing a `U+000D` character, the text node will return a string in which `U+000D` is replaced by `U+000A`. However, the text node will not change the underlying representation unless it is altered. If the text of the text node is modified, the text node may also internally replace `U+000D` by `U+000A`.

3.8.7. Java interface overview

The following list gives an overview of the interfaces and types in the Wikitext Object Model. An interface A that is indented below another interface B means that A extends B. The following font styles are used to distinguish the different types of types and interfaces:

Simple data types and enumerations
Abstract interface that does not represent a concrete element
An interface exposing an attribute group
An interface representing a concrete element
`An interface that offers a different view of the data`

The interfaces and data types:

- WomBulletStyle
- WomClear
- WomColor
- *WomCoreAttributes*
  - WomBreak
  - *WomUniversalAttributes*
- *WomEventAttributes*
  - *WomUniversalAttributes*
- WomHorizAlign
- *WomI18nAttributes*
  - *WomUniversalAttributes*
- WomI18nDir
- WomImageFormat
- WomImageHAlign
- WomImageVAlign
- **WomLink**
  - WomCategory
  - WomExtLink
  - WomImage
  - WomIntLink
  - WomRedirect
  - WomUrl
- **WomNode**
  - WomArg
  - WomAttr
  - WomAttribute
  - **WomBlockElement**
    - WomBlockquote
    - WomCenter
    - WomDefinitionList
    - WomDel
    - WomDiv
    - WomHorizontalRule
    - WomIns
    - **WomList**
      - WomOrderedList
      - WomUnorderedList
    - WomParagraph
    - WomPre
    - WomSection
    - WomSemiPre
    - WomTable
  - WomBody
  - WomDefault
  - **WomDefinitionListItem**
    - WomDefinitionListDef
    - WomDefinitionListTerm
  - WomElementBody
  - WomHeading
  - WomImageCaption
  - **WomInlineElement**
    - WomAbbr
    - WomBig
    - WomBold
    - WomBreak
    - WomCite
    - WomCode
    - WomDel
    - WomDfn
    - WomElement
    - WomEmphasize
    - WomExtLink
    - WomImage
    - WomIns
    - WomIntLink
    - WomItalics
    - WomKbd
    - WomNowiki
    - WomSamp
    - WomSignature
    - WomSmall
    - WomSpan
    - WomStrike
    - WomStrong
    - WomSub
    - WomSup

    - WomTeletype
    - WomUnderline
    - WomUrl
    - WomVar
  - WomListItem
  - WomName
  - WomPage
  - **WomProcessingInstruction**
    - WomComment
    - WomCategory
    - WomMagicWord
    - WomParam
    - WomRedirect
    - WomTagExtension
    - WomTransclusion
  - WomTableCaption
  - **WomTableCellBase**
    - WomTableCell
    - WomTableHeaderCell
  - WomTableCellScope
  - **WomTablePartition**
    - WomTableBody
  - WomTableRow
  - WomTagExtBody
  - WomText
  - WomTitle
  - WomValue
- WomNodeType
- WomSignatureFormat
- WomTableCaptionAlign
- `WomTableColumn`
- WomTableFrame
- WomTableRules
- WomTableVAlign
- WomUnit
- *WomUniversalAttributes*
  - WomAbbr
  - WomBig
  - WomBlockquote
  - WomBold
  - WomCenter
  - WomCite
  - WomCode
  - WomDefinitionList
  - WomDefinitionListDef
  - WomDefinitionListTerm
  - WomDel
  - WomDfn
  - WomDiv
  - WomEmphasize
  - WomFont
  - WomHeading
  - WomHorizontalRule
  - WomIns
  - WomItalics
  - WomKbd
  - WomList
  - WomListItem
  - WomParagraph
  - WomPre
  - WomSamp
  - WomSmall
  - WomSpan
  - WomStrike
  - WomStrong
  - WomSub
  - WomSup
  - WomTable
  - WomTableBody
  - WomTableCaption
  - WomTableCell
  - WomTableHeaderCell
  - WomTableRow
  - WomTeletype
  - WomUnderline
  - WomVar
- WomValueWithUnit

4. Comparison between XHTML and XWML

XWML 1.0 elements not found in XHTML 1.0 Transitional:

- arg
- attr
- body
- default
- category
- element
- elembody
- extlink
- heading
- image
- imgcaption
- intlink
- magicword
- name
- nowiki
- page
- param
- pre
- redirect
- section
- semipre
- signature
- tagext
- tagextbody
- title
- transclusion
- url
- value

XHTML 1.0 Transitional elements not available in XWML 1.0:

- Misc elements:
  - html
  - head
  - body*
- All header elements:
  - base
  - link
  - meta
  - object
  - script
  - style
  - title
- Body elements:
  - a
  - acronym†
  - address†
  - applet
  - area
  - basefont
  - bdo
  - dir
  - form
  - h1 - h6
  - iframe
  - img
  - isindex
  - map
  - menu
  - noframes
  - noscript
  - object
  - param*
  - pre*
  - q†
  - script
- Table elements:
  - col†
  - colgroup†
  - tbody†
  - tfoot†
  - thead†
- All form elements§:
  - button
  - fieldset
  - input
  - label
  - legend
  - optgroup
  - option
  - select
  - textarea

Legend:

†	These elements are not supported by a vanilla MediaWiki installation, however, they might be supported in the future.
§	These elements are not supported by a vanialla MediaWiki and probably should not be supported as HTML elements in the future. However, supporting these elements through a MediaWiki specific abstraction might be interesting.
*	There are elements with the same name in XWML 1.0 but they have different meaning and/or function.

The following elements are supported by XHTML 1.0 Transitional and by XWML 1.0 but support should probably be faded out in future versions of XWML to become compliant with XHTML 1.0 Strict:

- center
- font
- s
- strike
- u

Elements found in XHTML 1.0 Frameset that are also not part of XWML 1.0:

- frame
- frameset

5. References

[1] M. Junghans, D. Riehle, R. Gurram, M. Kaiser, M. Lopes, and U. Yalcinalp, "A grammar for standardized wiki markup," in *Proceedings of the 4th International Symposium on Wikis*. New York, NY, USA: ACM, 2008, pp. 21:1–21:8.

[2] Various authors, "Alternative parsers for Mediawiki," http://www.mediawiki.org/wiki/Alternative_parsers, May 2011.

[3] AboutUs.org, "kiwi - Yet Another Peg WikiText Parser," http://github.com/aboutus/kiwi/, May 2011.

[4] dbpedia.org, "DBpedia," http://dbpedia.org/, May 2011.

[5] H. Dohrn and D. Riehle, "Design and implementation of the sweble wikitext parser: Unlocking the structure within wikipedia," to appear in the *Proceedings of the 7th International Symposium on Wikis and Open Collaboration*.

[6] sweble.org, "Sweble - Sweetly Enabling the Web," http://sweble.org/, May 2011.

[7] Object Management Group, "OMG IDL Syntax and Semantics," http://www.omg.org/cgi-bin/doc?formal/02-06-39, July 2011.

[8] The World Wide Web Consortium, "XHTML 1.0 Transitional," http://www.w3.org/2002/08/xhtml/xhtml1-transitional.xsd, May 2011.

[9] ——, "XML Schema Part 0: Primer Second Edition," http://www.w3.org/TR/xmlschema-0/, October 2004.

[10] ——, "HTML 4.01 Specification," http://www.w3.org/TR/1999/REC-html401-19991224, December 1999.

[11] The Unicode Consortium, *The Unicode Standard, Version 5.0*. Addison-Wesley, 2007.

A. XWML 1.0 Schema

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xs:schema xmlns="http://sweble.org/xwml-1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema
       "
3    targetNamespace="http://sweble.org/xwml-1.0" elementFormDefault="qualified">
4
5    <xs:import namespace="http://www.w3.org/XML/1998/namespace"
6      schemaLocation="http://www.w3.org/2001/xml.xsd" />
7
8    <!-- ==[ XHTML Data Types ]========================================== -->
9
10   <xs:simpleType name="Character">
11     <xs:annotation>
12       <xs:documentation>
13         see
14         <a href="http://www.w3.org/2002/08/xhtml/xhtml1-transitional.xsd">XHTML 1.0
               Transitional</a>
15       </xs:documentation>
16     </xs:annotation>
17     <xs:restriction base="xs:string">
18       <xs:length value="1" fixed="true" />
19     </xs:restriction>
20   </xs:simpleType>
21
22   <xs:simpleType name="Color">
23     <xs:annotation>
24       <xs:documentation>
25         see
26         <a href="http://www.w3.org/2002/08/xhtml/xhtml1-transitional.xsd">XHTML 1.0
               Transitional</a>
27       </xs:documentation>
28     </xs:annotation>
29     <xs:restriction base="xs:string">
30       <xs:pattern value="[A-Za-z]+|#[0-9A-Fa-f]{3}|#[0-9A-Fa-f]{6}" />
31     </xs:restriction>
32   </xs:simpleType>
33
34   <xs:simpleType name="LanguageCode">
35     <xs:annotation>
36       <xs:documentation>
37         see
38         <a href="http://www.w3.org/2002/08/xhtml/xhtml1-transitional.xsd">XHTML 1.0
               Transitional</a>
39       </xs:documentation>
40     </xs:annotation>
41     <xs:restriction base="xs:language" />
42   </xs:simpleType>
43
44   <xs:simpleType name="Length">
45     <xs:annotation>
46       <xs:documentation>
47         see
48         <a href="http://www.w3.org/2002/08/xhtml/xhtml1-transitional.xsd">XHTML 1.0
               Transitional</a>
49       </xs:documentation>
50     </xs:annotation>
51     <xs:restriction base="xs:string">
52       <xs:pattern value="[-+]?(\d+|\d+(\.\d+)?%)" />
53     </xs:restriction>
54   </xs:simpleType>
55
56   <xs:simpleType name="LiStyle">
57     <xs:annotation>
58       <xs:documentation>
59         see
60         <a href="http://www.w3.org/2002/08/xhtml/xhtml1-transitional.xsd">XHTML 1.0
               Transitional</a>
61       </xs:documentation>
62     </xs:annotation>
63     <xs:restriction base="xs:string" />
64   </xs:simpleType>
65
66   <xs:simpleType name="Number">
67     <xs:annotation>
```

```
68        <xs:documentation>
69          see
70          <a href="http://www.w3.org/2002/08/xhtml/xhtml1-transitional.xsd">XHTML 1.0
                Transitional</a>
71        </xs:documentation>
72      </xs:annotation>
73      <xs:restriction base="xs:nonNegativeInteger">
74        <xs:pattern value="[0-9]+" />
75      </xs:restriction>
76    </xs:simpleType>
77
78    <xs:simpleType name="OlStyle">
79      <xs:annotation>
80        <xs:documentation>
81          see
82          <a href="http://www.w3.org/2002/08/xhtml/xhtml1-transitional.xsd">XHTML 1.0
                Transitional</a>
83        </xs:documentation>
84      </xs:annotation>
85      <xs:restriction base="xs:string" />
86    </xs:simpleType>
87
88    <xs:simpleType name="Script">
89      <xs:annotation>
90        <xs:documentation>
91          see
92          <a href="http://www.w3.org/2002/08/xhtml/xhtml1-transitional.xsd">XHTML 1.0
                Transitional</a>
93        </xs:documentation>
94      </xs:annotation>
95      <xs:restriction base="xs:string" />
96    </xs:simpleType>
97
98    <xs:simpleType name="StyleSheet">
99      <xs:annotation>
100       <xs:documentation>
101         see
102         <a href="http://www.w3.org/2002/08/xhtml/xhtml1-transitional.xsd">XHTML 1.0
                Transitional</a>
103       </xs:documentation>
104     </xs:annotation>
105     <xs:restriction base="xs:string" />
106   </xs:simpleType>
107
108   <!-- ==[ XHTML Data Types also used by XWML ]============================== -->
109
110   <xs:simpleType name="Datetime">
111     <xs:annotation>
112       <xs:documentation>
113         see
114         <a href="http://www.w3.org/2002/08/xhtml/xhtml1-transitional.xsd">XHTML 1.0
                Transitional</a>
115       </xs:documentation>
116     </xs:annotation>
117     <xs:restriction base="xs:dateTime" />
118   </xs:simpleType>
119
120   <xs:simpleType name="Pixels">
121     <xs:annotation>
122       <xs:documentation>
123         see
124         <a href="http://www.w3.org/2002/08/xhtml/xhtml1-transitional.xsd">XHTML 1.0
                Transitional</a>
125       </xs:documentation>
126     </xs:annotation>
127     <xs:restriction base="xs:nonNegativeInteger" />
128   </xs:simpleType>
129
130   <xs:simpleType name="Text">
131     <xs:annotation>
132       <xs:documentation>
133         see
134         <a href="http://www.w3.org/2002/08/xhtml/xhtml1-transitional.xsd">XHTML 1.0
                Transitional</a>
135       </xs:documentation>
136     </xs:annotation>
```

```
137     <xs:restriction base="xs:string" />
138   </xs:simpleType>
139
140   <xs:simpleType name="URI">
141     <xs:annotation>
142       <xs:documentation>
143         see
144         <a href="http://www.w3.org/2002/08/xhtml/xhtml1-transitional.xsd">XHTML 1.0
                Transitional</a>
145       </xs:documentation>
146     </xs:annotation>
147     <xs:restriction base="xs:anyURI" />
148   </xs:simpleType>
149
150   <!-- ==[ XWML Data Types ]============================================== -->
151
152   <xs:simpleType name="HeadingLevel">
153     <xs:annotation>
154       <xs:documentation>
155         Designates the level of a heading from 1 (most
156         important) to 6 (least
157         important).
158       </xs:documentation>
159     </xs:annotation>
160     <xs:restriction base="xs:nonNegativeInteger">
161       <xs:minInclusive value="1" />
162       <xs:maxInclusive value="6" />
163     </xs:restriction>
164   </xs:simpleType>
165
166   <xs:simpleType name="MagicWord">
167     <xs:annotation>
168       <xs:documentation>The name of a MediaWiki magic word.</xs:documentation>
169     </xs:annotation>
170     <xs:restriction base="xs:string" />
171   </xs:simpleType>
172
173   <xs:simpleType name="Namespace">
174     <xs:annotation>
175       <xs:documentation>The name of a MediaWiki namespace.</xs:documentation>
176     </xs:annotation>
177     <xs:restriction base="xs:string" />
178   </xs:simpleType>
179
180   <xs:simpleType name="PageTitle">
181     <xs:annotation>
182       <xs:documentation>
183         A MediaWiki page title, possibly including namespace and
184         a path (of
185         subpages).
186       </xs:documentation>
187     </xs:annotation>
188     <xs:restriction base="xs:string" />
189   </xs:simpleType>
190
191   <xs:simpleType name="Username">
192     <xs:annotation>
193       <xs:documentation>A MediaWiki username.</xs:documentation>
194     </xs:annotation>
195     <xs:restriction base="xs:string" />
196   </xs:simpleType>
197
198   <xs:simpleType name="Version">
199     <xs:annotation>
200       <xs:documentation>
201         A version is given as numbers separated by dots.
202         Furthermore, a version
203         can have a suffix that is separated from the version number by a dash.
204       </xs:documentation>
205     </xs:annotation>
206     <xs:restriction base="xs:string">
207       <xs:pattern value="[0-9]+(\.[0-9]+)*(-[A-Za-z][A-Za-z0-9_]*)?" />
208     </xs:restriction>
209   </xs:simpleType>
210
211   <!-- ==[ XHTML Enumerations ]========================================== -->
```

```
212
213   <xs:simpleType name="Enum.blockAlign">
214     <xs:restriction base="xs:token">
215       <xs:enumeration value="center" />
216       <xs:enumeration value="left" />
217       <xs:enumeration value="right" />
218     </xs:restriction>
219   </xs:simpleType>
220
221   <xs:simpleType name="Enum.captionAlign">
222     <xs:restriction base="xs:token">
223       <xs:enumeration value="bottom" />
224       <xs:enumeration value="left" />
225       <xs:enumeration value="right" />
226       <xs:enumeration value="top" />
227     </xs:restriction>
228   </xs:simpleType>
229
230   <xs:simpleType name="Enum.cellHAlign">
231     <xs:restriction base="xs:token">
232       <xs:enumeration value="center" />
233       <xs:enumeration value="char" />
234       <xs:enumeration value="justify" />
235       <xs:enumeration value="left" />
236       <xs:enumeration value="right" />
237     </xs:restriction>
238   </xs:simpleType>
239
240   <xs:simpleType name="Enum.cellVAlign">
241     <xs:restriction base="xs:token">
242       <xs:enumeration value="baseline" />
243       <xs:enumeration value="bottom" />
244       <xs:enumeration value="middle" />
245       <xs:enumeration value="top" />
246     </xs:restriction>
247   </xs:simpleType>
248
249   <xs:simpleType name="Enum.clear">
250     <xs:restriction base="xs:token">
251       <xs:enumeration value="all" />
252       <xs:enumeration value="left" />
253       <xs:enumeration value="none" />
254       <xs:enumeration value="right" />
255     </xs:restriction>
256   </xs:simpleType>
257
258   <xs:simpleType name="Enum.compact">
259     <xs:restriction base="xs:token">
260       <xs:enumeration value="compact" />
261     </xs:restriction>
262   </xs:simpleType>
263
264   <xs:simpleType name="Enum.dir">
265     <xs:restriction base="xs:token">
266       <xs:enumeration value="ltr" />
267       <xs:enumeration value="rtl" />
268     </xs:restriction>
269   </xs:simpleType>
270
271   <xs:simpleType name="Enum.noshade">
272     <xs:restriction base="xs:token">
273       <xs:enumeration value="noshade" />
274     </xs:restriction>
275   </xs:simpleType>
276
277   <xs:simpleType name="Enum.nowrap">
278     <xs:restriction base="xs:token">
279       <xs:enumeration value="nowrap" />
280     </xs:restriction>
281   </xs:simpleType>
282
283   <xs:simpleType name="Enum.scope">
284     <xs:restriction base="xs:token">
285       <xs:enumeration value="row" />
286       <xs:enumeration value="col" />
287       <xs:enumeration value="rowgroup" />
```

```
      <xs:enumeration value="colgroup" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="Enum.tableFrame">
    <xs:restriction base="xs:token">
      <xs:enumeration value="above" />
      <xs:enumeration value="below" />
      <xs:enumeration value="border" />
      <xs:enumeration value="box" />
      <xs:enumeration value="hsides" />
      <xs:enumeration value="lhs" />
      <xs:enumeration value="rhs" />
      <xs:enumeration value="void" />
      <xs:enumeration value="vsides" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="Enum.tableRules">
    <xs:restriction base="xs:token">
      <xs:enumeration value="all" />
      <xs:enumeration value="cols" />
      <xs:enumeration value="groups" />
      <xs:enumeration value="none" />
      <xs:enumeration value="rows" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="Enum.ulStyle">
    <xs:restriction base="xs:token">
      <xs:enumeration value="circle" />
      <xs:enumeration value="disc" />
      <xs:enumeration value="square" />
    </xs:restriction>
  </xs:simpleType>

  <!-- ==[ XWML Enumerations ]============================================== -->

  <xs:simpleType name="Enum.imageFormat">
    <xs:annotation>
      <xs:documentation>
        Specifies how the image will be rendered/placed.
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:token">
      <xs:enumeration value="unrestrained" />
      <xs:enumeration value="frameless" />
      <xs:enumeration value="thumbnail" />
      <xs:enumeration value="frame" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="Enum.imageHAlign">
    <xs:annotation>
      <xs:documentation>
        Specifies how an image is aligned horizontically.
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:token">
      <xs:enumeration value="default" />
      <xs:enumeration value="none" />
      <xs:enumeration value="left" />
      <xs:enumeration value="center" />
      <xs:enumeration value="right" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="Enum.imageVAlign">
    <xs:annotation>
      <xs:documentation>
        Specifies how an image is aligned vertically.
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:token">
      <xs:enumeration value="baseline" />
      <xs:enumeration value="sub" />
```

```
      <xs:enumeration value="super" />
      <xs:enumeration value="top" />
      <xs:enumeration value="text-top" />
      <xs:enumeration value="middle" />
      <xs:enumeration value="bottom" />
      <xs:enumeration value="text-bottom" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="Enum.signatureFormat">
    <xs:annotation>
      <xs:documentation>
        Specifies how a signature should be rendered.
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:token">
      <xs:enumeration value="user" />
      <xs:enumeration value="timestamp" />
      <xs:enumeration value="user-timestamp" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="Enum.border">
    <xs:restriction base="xs:token">
      <xs:enumeration value="border" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="Enum.upright">
    <xs:restriction base="xs:token">
      <xs:enumeration value="upright" />
    </xs:restriction>
  </xs:simpleType>

  <!-- ================================================================== -->

  <xs:attributeGroup name="attributes.core">
    <xs:annotation>
      <xs:documentation>
      </xs:documentation>
    </xs:annotation>
    <xs:attribute name="id" type="xs:ID" />
    <xs:attribute name="class" type="xs:NMTOKENS" />
    <xs:attribute name="style" type="StyleSheet" />
    <xs:attribute name="title" type="Text" />
  </xs:attributeGroup>

  <xs:attributeGroup name="attributes.i18n">
    <xs:annotation>
      <xs:documentation>
      </xs:documentation>
    </xs:annotation>
    <xs:attribute name="lang" type="LanguageCode" />
    <xs:attribute ref="xml:lang" />
    <xs:attribute name="dir" type="Enum.dir" />
  </xs:attributeGroup>

  <xs:attributeGroup name="attributes.events">
    <xs:annotation>
      <xs:documentation>
      </xs:documentation>
    </xs:annotation>
    <xs:attribute name="onclick" type="Script" />
    <xs:attribute name="ondblclick" type="Script" />
    <xs:attribute name="onmousedown" type="Script" />
    <xs:attribute name="onmouseup" type="Script" />
    <xs:attribute name="onmouseover" type="Script" />
    <xs:attribute name="onmousemove" type="Script" />
    <xs:attribute name="onmouseout" type="Script" />
    <xs:attribute name="onkeypress" type="Script" />
    <xs:attribute name="onkeydown" type="Script" />
    <xs:attribute name="onkeyup" type="Script" />
  </xs:attributeGroup>

  <xs:attributeGroup name="attributes.universal">
    <xs:attributeGroup ref="attributes.core" />
```

```
    <xs:attributeGroup ref="attributes.i18n" />
    <xs:attributeGroup ref="attributes.events" />
  </xs:attributeGroup>

  <xs:attributeGroup name="attributes.textAlign">
    <xs:attribute name="align">
      <xs:simpleType>
        <xs:restriction base="xs:token">
          <xs:enumeration value="left" />
          <xs:enumeration value="center" />
          <xs:enumeration value="right" />
          <xs:enumeration value="justify" />
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:attributeGroup>

  <xs:attributeGroup name="attributes.cellHAlign">
    <xs:attribute name="align" type="Enum.cellHAlign" />
    <xs:attribute name="char" type="Character" />
    <xs:attribute name="charoff" type="Length" />
  </xs:attributeGroup>

  <xs:attributeGroup name="attributes.cellVAlign">
    <xs:attribute name="valign" type="Enum.cellVAlign" />
  </xs:attributeGroup>

  <!-- ================================================================== -->

  <xs:group name="elements.preprocessor">
    <xs:choice>
      <xs:element ref="comment" />
      <xs:element ref="magicword" />
      <xs:element ref="param" />
      <xs:element ref="tagext" />
      <xs:element ref="transclusion" />
    </xs:choice>
  </xs:group>

  <xs:group name="elements.inline.fontstyle.pre">
    <xs:choice>
      <xs:element ref="b" />
      <xs:element ref="i" />
      <xs:element ref="s" />
      <xs:element ref="strike" />
      <xs:element ref="tt" />
      <xs:element ref="u" />
    </xs:choice>
  </xs:group>

  <xs:group name="elements.inline.fontstyle.extra">
    <xs:choice>
      <xs:element ref="big" />
      <xs:element ref="font" />
      <xs:element ref="small" />
    </xs:choice>
  </xs:group>

  <xs:group name="elements.inline.fontstyle">
    <xs:choice>
      <xs:group ref="elements.inline.fontstyle.pre" />
      <xs:group ref="elements.inline.fontstyle.extra" />
    </xs:choice>
  </xs:group>

  <xs:group name="elements.inline.phrase.pre">
    <xs:choice>
      <xs:element ref="abbr" />
      <xs:element ref="cite" />
      <xs:element ref="code" />
      <xs:element ref="dfn" />
      <xs:element ref="em" />
      <xs:element ref="kbd" />
      <xs:element ref="samp" />
      <xs:element ref="strong" />
      <xs:element ref="var" />
```

```
    </xs:choice>
  </xs:group>

  <xs:group name="elements.inline.phrase.extra">
    <xs:choice>
      <xs:element ref="sub" />
      <xs:element ref="sup" />
    </xs:choice>
  </xs:group>

  <xs:group name="elements.inline.phrase">
    <xs:choice>
      <xs:group ref="elements.inline.phrase.pre" />
      <xs:group ref="elements.inline.phrase.extra" />
    </xs:choice>
  </xs:group>

  <xs:group name="elements.inline.link">
    <xs:choice>
      <xs:element ref="extlink" />
      <xs:element ref="intlink" />
      <xs:element ref="url" />
    </xs:choice>
  </xs:group>

  <xs:group name="elements.inline.misc.pre">
    <xs:choice>
      <xs:element ref="br" />
      <xs:element ref="nowiki" />
      <xs:element ref="signature" />
      <xs:element ref="span" />
    </xs:choice>
  </xs:group>

  <xs:group name="elements.inline.misc.extra">
    <xs:choice>
      <xs:element ref="image" />
    </xs:choice>
  </xs:group>

  <xs:group name="elements.inline.misc">
    <xs:choice>
      <xs:group ref="elements.inline.misc.pre" />
      <xs:group ref="elements.inline.misc.extra" />
    </xs:choice>
  </xs:group>

  <xs:group name="elements.edit">
    <xs:choice>
      <xs:element ref="del" />
      <xs:element ref="ins" />
    </xs:choice>
  </xs:group>

  <xs:group name="elements.inline">
    <xs:choice>
      <xs:group ref="elements.inline.fontstyle" />
      <xs:group ref="elements.inline.phrase" />
      <xs:group ref="elements.inline.link" />
      <xs:group ref="elements.inline.misc" />
    </xs:choice>
  </xs:group>

  <xs:group name="elements.block.list">
    <xs:choice>
      <xs:element ref="dl" />
      <xs:element ref="ol" />
      <xs:element ref="ul" />
    </xs:choice>
  </xs:group>

  <xs:group name="elements.block.preformatted">
    <xs:choice>
      <xs:element ref="pre" />
      <xs:element ref="semipre" />
    </xs:choice>
```

```
592   </xs:group>
593
594   <xs:group name="elements.block.body">
595     <xs:choice>
596       <xs:element ref="section" />
597     </xs:choice>
598   </xs:group>
599
600   <xs:group name="elements.block.misc">
601     <xs:choice>
602       <xs:element ref="blockquote" />
603       <xs:element ref="center" />
604       <xs:element ref="div" />
605       <xs:element ref="hr" />
606       <xs:element ref="p" />
607       <xs:element ref="table" />
608     </xs:choice>
609   </xs:group>
610
611   <xs:group name="elements.block">
612     <xs:choice>
613       <xs:group ref="elements.block.list" />
614       <xs:group ref="elements.block.preformatted" />
615       <xs:group ref="elements.block.misc" />
616     </xs:choice>
617   </xs:group>
618
619   <xs:group name="elements.inlblk">
620     <xs:choice>
621       <xs:group ref="elements.preprocessor" />
622       <xs:group ref="elements.edit" />
623       <xs:element ref="element" />
624     </xs:choice>
625   </xs:group>
626
627   <!-- ================================================================== -->
628
629   <xs:complexType name="Preprocessor" mixed="true">
630     <xs:choice minOccurs="0" maxOccurs="unbounded">
631       <xs:group ref="elements.preprocessor" />
632     </xs:choice>
633   </xs:complexType>
634
635   <xs:complexType name="Inline" mixed="true">
636     <xs:choice minOccurs="0" maxOccurs="unbounded">
637       <xs:group ref="elements.inline" />
638       <xs:group ref="elements.inlblk" />
639     </xs:choice>
640   </xs:complexType>
641
642   <xs:complexType name="Inline.pre" mixed="true">
643     <xs:choice>
644       <xs:group ref="elements.inline.fontstyle.pre" />
645       <xs:group ref="elements.inline.phrase.pre" />
646       <xs:group ref="elements.inline.link" />
647       <xs:group ref="elements.inline.misc.pre" />
648       <xs:group ref="elements.inlblk" />
649     </xs:choice>
650   </xs:complexType>
651
652   <xs:complexType name="Inline.title" mixed="true">
653     <xs:choice>
654       <xs:group ref="elements.inline.fontstyle" />
655       <xs:group ref="elements.inline.phrase" />
656       <xs:group ref="elements.inline.misc" />
657       <xs:group ref="elements.inlblk" />
658     </xs:choice>
659   </xs:complexType>
660
661   <xs:complexType name="Block">
662     <xs:choice minOccurs="0" maxOccurs="unbounded">
663       <xs:group ref="elements.block" />
664       <xs:group ref="elements.inlblk" />
665     </xs:choice>
666   </xs:complexType>
667
```

```
  <xs:complexType name="Block.body">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:group ref="elements.block" />
      <xs:group ref="elements.block.body" />
      <xs:group ref="elements.inlblk" />
    </xs:choice>
  </xs:complexType>

  <xs:complexType name="Flow" mixed="true">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:group ref="elements.inline" />
      <xs:group ref="elements.block" />
      <xs:group ref="elements.inlblk" />
    </xs:choice>
  </xs:complexType>

  <!-- ============================================================== -->

  <xs:element name="arg">
    <xs:annotation>
      <xs:documentation>
        The arg element specifies an argument to a template page. arg elements
        are specified as part of a transclusion statement.
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="name" minOccurs="0" />
        <xs:element ref="value" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="attr">
    <xs:annotation>
      <xs:documentation>
        Describes an attribute that is passed to a tag extension. Both name and
        value of the attribute are given as attributes.
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:attribute name="name" use="required" type="xs:Name" />
      <xs:attribute name="value" use="required" type="xs:string" />
    </xs:complexType>
  </xs:element>

  <xs:element name="body">
    <xs:annotation>
      <xs:documentation>
        General node to hold block level content in page and section nodes.
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="Block.body" />
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="category">
    <xs:annotation>
      <xs:documentation>
        Puts this page into the mentioned category.
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:attribute name="category" use="required" type="PageTitle" />
    </xs:complexType>
  </xs:element>

  <xs:element name="comment">
    <xs:annotation>
      <xs:documentation>
        XML-like comment found in Wikitext.
      </xs:documentation>
    </xs:annotation>
```

```
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="Text" />
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="default">
    <xs:annotation>
      <xs:documentation>
        Holds the default value of a template parameter. If no argument is
        specified in the transclusion statement the default value will be used
        as parameter value.
      </xs:documentation>
    </xs:annotation>
    <xs:complexType mixed="true">
      <xs:complexContent>
        <xs:extension base="Preprocessor" />
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="element">
    <xs:annotation>
      <xs:documentation>
        Represents and arbitrary XML element.
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="attr" minOccurs="0" maxOccurs="unbounded" />
        <xs:element ref="elembody" minOccurs="0" />
      </xs:sequence>
      <xs:attribute name="name" use="required" type="xs:Name" />
    </xs:complexType>
  </xs:element>

  <xs:element name="elembody">
    <xs:annotation>
      <xs:documentation>
        The content of an arbitrary XML tag specified by "element".
      </xs:documentation>
    </xs:annotation>
    <xs:complexType mixed="true">
      <xs:sequence>
        <xs:any minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="extlink">
    <xs:annotation>
      <xs:documentation>
        Describes a bracketed external Wikitext link.
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="title" minOccurs="0" />
      </xs:sequence>
      <xs:attribute name="target" use="required" type="URI" />
    </xs:complexType>
  </xs:element>

  <xs:element name="heading">
    <xs:annotation>
      <xs:documentation>
        Holds the heading of a section.
      </xs:documentation>
    </xs:annotation>
    <xs:complexType mixed="true">
      <xs:complexContent>
        <xs:extension base="Inline">
          <xs:attributeGroup ref="attributes.textAlign" />
          <xs:attributeGroup ref="attributes.universal" />
        </xs:extension>
```

```
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="image">
    <xs:annotation>
      <xs:documentation>
        Describes a Wikitext image.
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="imgcaption" minOccurs="0" />
      </xs:sequence>
      <xs:attribute name="source" use="required" type="PageTitle" />
      <xs:attribute name="format" type="Enum.imageFormat" default="unrestrained" />
      <xs:attribute name="border" type="Enum.border" />
      <xs:attribute name="halign" type="Enum.imageHAlign" default="none" />
      <xs:attribute name="valign" type="Enum.imageVAlign" default="baseline" />
      <xs:attribute name="width" type="Pixels" />
      <xs:attribute name="height" type="Pixels" />
      <xs:attribute name="upright" type="Enum.upright" />
      <xs:attribute name="urllink" type="URI" />
      <xs:attribute name="pagelink" type="PageTitle" />
      <xs:attribute name="alt" type="Text" />
    </xs:complexType>
  </xs:element>

  <xs:element name="imgcaption">
    <xs:annotation>
      <xs:documentation>
        Describes the caption of an image.
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="Inline" />
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="intlink">
    <xs:annotation>
      <xs:documentation>
        Describes a internal Wikitext link.
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="title" minOccurs="0" />
      </xs:sequence>
      <xs:attribute name="target" use="required" type="PageTitle" />
    </xs:complexType>
  </xs:element>

  <xs:element name="magicword">
    <xs:annotation>
      <xs:documentation>
        Descibes a magic word.
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:attribute name="name" use="required" type="MagicWord" />
    </xs:complexType>
  </xs:element>

  <xs:element name="name">
    <xs:annotation>
      <xs:documentation>
        Name elements specify the name of arg, param and transclusion
        elements. After expansion the content of a name element must evaluate
        to a string that forms a valid name. What constitutes a valid name
        depends on the element in which the name element occurs.
      </xs:documentation>
    </xs:annotation>
    <xs:complexType mixed="true">
```

```
      <xs:complexContent>
        <xs:extension base="Preprocessor" />
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="nowiki">
    <xs:annotation>
      <xs:documentation>
        Contains text that must not be interpreted.
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="Text" />
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="page">
    <xs:annotation>
      <xs:documentation>
        The root element of a page.
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="redirect" minOccurs="0" />
        <xs:element ref="category" minOccurs="0" maxOccurs="unbounded" />
        <xs:element ref="body" />
      </xs:sequence>
      <xs:attribute name="version" use="required" type="Version" />
      <xs:attribute name="namespace" type="Namespace" />
      <xs:attribute name="path" type="PageTitle" />
      <xs:attribute name="title" use="required" type="PageTitle" />
    </xs:complexType>
  </xs:element>

  <xs:element name="param">
    <xs:annotation>
      <xs:documentation>
        A template parameter that will be replaced by an argument passed to the
        respective transclusion statement.
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="name" />
        <xs:element ref="default" minOccurs="0" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="pre">
    <xs:annotation>
      <xs:documentation>
        Describes preformatted content that must not be parsed and will be
        displayed with a fixed font.
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="Text">
          <xs:attributeGroup ref="attributes.universal" />
          <xs:attribute name="width" type="Number" />
          <xs:attribute ref="xml:space" fixed="preserve" />
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="redirect">
    <xs:annotation>
      <xs:documentation>
        Indicates that this page is a redirection page and declares the target
        page of the redirection.
```

```
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:attribute name="target" use="required" type="PageTitle" />
    </xs:complexType>
  </xs:element>

  <xs:element name="section">
    <xs:annotation>
      <xs:documentation>
        A section consiting of heading and body.
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="heading" />
        <xs:element ref="body" />
      </xs:sequence>
      <xs:attribute name="level" use="required" type="HeadingLevel" />
    </xs:complexType>
  </xs:element>

  <xs:element name="semipre">
    <xs:annotation>
      <xs:documentation>
        Describes preformatted, parsed content that will be displayed with a
        fixed font. The semipre element must not contain the elements image,
        big, small, sub, sup or font.
      </xs:documentation>
    </xs:annotation>
    <xs:complexType mixed="true">
      <xs:complexContent>
        <xs:extension base="Inline.pre" />
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="signature">
    <xs:annotation>
      <xs:documentation>
        Describes a signature.
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:attribute name="format" use="required" type="Enum.signatureFormat" />
      <xs:attribute name="author" use="required" type="Username" />
      <xs:attribute name="timestamp" use="required" type="Datetime" />
    </xs:complexType>
  </xs:element>

  <xs:element name="tagext">
    <xs:annotation>
      <xs:documentation>
        An invocation of a tag extension.
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="attr" minOccurs="0" maxOccurs="unbounded" />
        <xs:element ref="tagextbody" minOccurs="0" />
      </xs:sequence>
      <xs:attribute name="name" use="required" type="xs:Name" />
    </xs:complexType>
  </xs:element>

  <xs:element name="tagextbody">
    <xs:annotation>
      <xs:documentation>
        The textual content of a tag extension.
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="Text" />
      </xs:simpleContent>
    </xs:complexType>
```

```
1048  </xs:element>
1049
1050  <xs:element name="title">
1051    <xs:annotation>
1052      <xs:documentation>
1053        Holds the title of an external or internal link. This element must not
1054        be empty or contain only whitespace. Elements from the link group must
1055        not appear as children of a title element.
1056      </xs:documentation>
1057    </xs:annotation>
1058    <xs:complexType mixed="true">
1059      <xs:complexContent>
1060        <xs:extension base="Inline.title" />
1061      </xs:complexContent>
1062    </xs:complexType>
1063  </xs:element>
1064
1065  <xs:element name="transclusion">
1066    <xs:annotation>
1067      <xs:documentation>
1068        Transcludes another page (called template) into the page in which the
1069        transclusion statement appears. The template is preprocessed and
1070        expanded before the transclusion statement is replaced by the expanded
1071        textual content of the template.
1072      </xs:documentation>
1073    </xs:annotation>
1074    <xs:complexType>
1075      <xs:sequence>
1076        <xs:element ref="name" />
1077        <xs:element ref="arg" minOccurs="0" maxOccurs="unbounded" />
1078      </xs:sequence>
1079    </xs:complexType>
1080  </xs:element>
1081
1082  <xs:element name="url">
1083    <xs:annotation>
1084      <xs:documentation>
1085        Describes a plain URL.
1086      </xs:documentation>
1087    </xs:annotation>
1088    <xs:complexType>
1089      <xs:attribute name="target" use="required" type="URI" />
1090    </xs:complexType>
1091  </xs:element>
1092
1093  <xs:element name="value">
1094    <xs:annotation>
1095      <xs:documentation>
1096        value elements specify the value of an arg element.
1097      </xs:documentation>
1098    </xs:annotation>
1099    <xs:complexType mixed="true">
1100      <xs:complexContent>
1101        <xs:extension base="Preprocessor" />
1102      </xs:complexContent>
1103    </xs:complexType>
1104  </xs:element>
1105
1106  <!-- ================================================================== -->
1107
1108  <xs:element name="b">
1109    <xs:complexType mixed="true">
1110      <xs:complexContent>
1111        <xs:extension base="Inline">
1112          <xs:attributeGroup ref="attributes.universal" />
1113        </xs:extension>
1114      </xs:complexContent>
1115    </xs:complexType>
1116  </xs:element>
1117
1118  <xs:element name="i">
1119    <xs:complexType mixed="true">
1120      <xs:complexContent>
1121        <xs:extension base="Inline">
1122          <xs:attributeGroup ref="attributes.universal" />
1123        </xs:extension>
```

```
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="s">
    <xs:complexType mixed="true">
      <xs:complexContent>
        <xs:extension base="Inline">
          <xs:attributeGroup ref="attributes.universal" />
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="strike">
    <xs:complexType mixed="true">
      <xs:complexContent>
        <xs:extension base="Inline">
          <xs:attributeGroup ref="attributes.universal" />
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="tt">
    <xs:complexType mixed="true">
      <xs:complexContent>
        <xs:extension base="Inline">
          <xs:attributeGroup ref="attributes.universal" />
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="u">
    <xs:complexType mixed="true">
      <xs:complexContent>
        <xs:extension base="Inline">
          <xs:attributeGroup ref="attributes.universal" />
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="big">
    <xs:complexType mixed="true">
      <xs:complexContent>
        <xs:extension base="Inline">
          <xs:attributeGroup ref="attributes.universal" />
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="font">
    <xs:complexType mixed="true">
      <xs:complexContent>
        <xs:extension base="Inline">
          <xs:attributeGroup ref="attributes.core" />
          <xs:attributeGroup ref="attributes.i18n" />
          <xs:attribute name="color" type="Color" />
          <xs:attribute name="face" />
          <xs:attribute name="size" />
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="small">
    <xs:complexType mixed="true">
      <xs:complexContent>
        <xs:extension base="Inline">
          <xs:attributeGroup ref="attributes.universal" />
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
```

```
1200    </xs:element>
1201
1202    <xs:element name="abbr">
1203      <xs:complexType mixed="true">
1204        <xs:complexContent>
1205          <xs:extension base="Inline">
1206            <xs:attributeGroup ref="attributes.universal" />
1207          </xs:extension>
1208        </xs:complexContent>
1209      </xs:complexType>
1210    </xs:element>
1211
1212    <xs:element name="cite">
1213      <xs:complexType mixed="true">
1214        <xs:complexContent>
1215          <xs:extension base="Inline">
1216            <xs:attributeGroup ref="attributes.universal" />
1217          </xs:extension>
1218        </xs:complexContent>
1219      </xs:complexType>
1220    </xs:element>
1221
1222    <xs:element name="code">
1223      <xs:complexType mixed="true">
1224        <xs:complexContent>
1225          <xs:extension base="Inline">
1226            <xs:attributeGroup ref="attributes.universal" />
1227          </xs:extension>
1228        </xs:complexContent>
1229      </xs:complexType>
1230    </xs:element>
1231
1232    <xs:element name="dfn">
1233      <xs:complexType mixed="true">
1234        <xs:complexContent>
1235          <xs:extension base="Inline">
1236            <xs:attributeGroup ref="attributes.universal" />
1237          </xs:extension>
1238        </xs:complexContent>
1239      </xs:complexType>
1240    </xs:element>
1241
1242    <xs:element name="em">
1243      <xs:complexType mixed="true">
1244        <xs:complexContent>
1245          <xs:extension base="Inline">
1246            <xs:attributeGroup ref="attributes.universal" />
1247          </xs:extension>
1248        </xs:complexContent>
1249      </xs:complexType>
1250    </xs:element>
1251
1252    <xs:element name="kbd">
1253      <xs:complexType mixed="true">
1254        <xs:complexContent>
1255          <xs:extension base="Inline">
1256            <xs:attributeGroup ref="attributes.universal" />
1257          </xs:extension>
1258        </xs:complexContent>
1259      </xs:complexType>
1260    </xs:element>
1261
1262    <xs:element name="samp">
1263      <xs:complexType mixed="true">
1264        <xs:complexContent>
1265          <xs:extension base="Inline">
1266            <xs:attributeGroup ref="attributes.universal" />
1267          </xs:extension>
1268        </xs:complexContent>
1269      </xs:complexType>
1270    </xs:element>
1271
1272    <xs:element name="strong">
1273      <xs:complexType mixed="true">
1274        <xs:complexContent>
1275          <xs:extension base="Inline">
```

```
          <xs:attributeGroup ref="attributes.universal" />
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="var">
    <xs:complexType mixed="true">
      <xs:complexContent>
        <xs:extension base="Inline">
          <xs:attributeGroup ref="attributes.universal" />
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="sub">
    <xs:complexType mixed="true">
      <xs:complexContent>
        <xs:extension base="Inline">
          <xs:attributeGroup ref="attributes.universal" />
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="sup">
    <xs:complexType mixed="true">
      <xs:complexContent>
        <xs:extension base="Inline">
          <xs:attributeGroup ref="attributes.universal" />
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="br">
    <xs:complexType>
      <xs:attributeGroup ref="attributes.core" />
      <xs:attribute name="clear" type="Enum.clear" default="none" />
    </xs:complexType>
  </xs:element>

  <xs:element name="span">
    <xs:complexType mixed="true">
      <xs:complexContent>
        <xs:extension base="Inline">
          <xs:attributeGroup ref="attributes.universal" />
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="del">
    <xs:complexType mixed="true">
      <xs:complexContent>
        <xs:extension base="Flow">
          <xs:attributeGroup ref="attributes.universal" />
          <xs:attribute name="cite" type="URI" />
          <xs:attribute name="datetime" type="Datetime" />
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="ins">
    <xs:complexType mixed="true">
      <xs:complexContent>
        <xs:extension base="Flow">
          <xs:attributeGroup ref="attributes.universal" />
          <xs:attribute name="cite" type="URI" />
          <xs:attribute name="datetime" type="Datetime" />
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
```

```
1352
1353    <xs:element name="dl">
1354      <xs:complexType>
1355        <xs:choice maxOccurs="unbounded">
1356          <xs:group ref="elements.preprocessor" />
1357          <xs:element ref="dt" />
1358          <xs:element ref="dd" />
1359        </xs:choice>
1360        <xs:attributeGroup ref="attributes.universal" />
1361        <xs:attribute name="compact">
1362          <xs:simpleType>
1363            <xs:restriction base="xs:token">
1364              <xs:enumeration value="compact" />
1365            </xs:restriction>
1366          </xs:simpleType>
1367        </xs:attribute>
1368      </xs:complexType>
1369    </xs:element>
1370
1371    <xs:element name="dt">
1372      <xs:complexType mixed="true">
1373        <xs:complexContent>
1374          <xs:extension base="Inline">
1375            <xs:attributeGroup ref="attributes.universal" />
1376          </xs:extension>
1377        </xs:complexContent>
1378      </xs:complexType>
1379    </xs:element>
1380
1381    <xs:element name="dd">
1382      <xs:complexType>
1383        <xs:complexContent>
1384          <xs:extension base="Block">
1385            <xs:attributeGroup ref="attributes.universal" />
1386          </xs:extension>
1387        </xs:complexContent>
1388      </xs:complexType>
1389    </xs:element>
1390
1391    <xs:element name="ol">
1392      <xs:complexType>
1393        <xs:sequence>
1394          <xs:group ref="elements.preprocessor" />
1395          <xs:element ref="li" maxOccurs="unbounded" />
1396        </xs:sequence>
1397        <xs:attributeGroup ref="attributes.universal" />
1398        <xs:attribute name="compact" type="Enum.compact" />
1399        <xs:attribute name="start" type="Number" />
1400        <xs:attribute name="type" type="OlStyle" />
1401      </xs:complexType>
1402    </xs:element>
1403
1404    <xs:element name="ul">
1405      <xs:complexType>
1406        <xs:sequence>
1407          <xs:group ref="elements.preprocessor" />
1408          <xs:element ref="li" maxOccurs="unbounded" />
1409        </xs:sequence>
1410        <xs:attributeGroup ref="attributes.universal" />
1411        <xs:attribute name="compact" type="Enum.compact" />
1412        <xs:attribute name="type" type="Enum.ulStyle" />
1413      </xs:complexType>
1414    </xs:element>
1415
1416    <xs:element name="li">
1417      <xs:complexType>
1418        <xs:complexContent>
1419          <xs:extension base="Block">
1420            <xs:attributeGroup ref="attributes.universal" />
1421            <xs:attribute name="type" type="LiStyle" />
1422            <xs:attribute name="value" type="Number" />
1423          </xs:extension>
1424        </xs:complexContent>
1425      </xs:complexType>
1426    </xs:element>
1427
```

```
  <xs:element name="blockquote">
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="Block">
          <xs:attributeGroup ref="attributes.universal" />
          <xs:attribute name="cite" type="URI" />
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="center">
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="Block">
          <xs:attributeGroup ref="attributes.universal" />
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="div">
    <xs:complexType mixed="true">
      <xs:complexContent>
        <xs:extension base="Flow">
          <xs:attributeGroup ref="attributes.universal" />
          <xs:attributeGroup ref="attributes.textAlign" />
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="hr">
    <xs:complexType>
      <xs:attributeGroup ref="attributes.universal" />
      <xs:attribute name="align" type="Enum.blockAlign" />
      <xs:attribute name="noshade" type="Enum.noshade" />
      <xs:attribute name="size" type="Pixels" />
      <xs:attribute name="width" type="Length" />
    </xs:complexType>
  </xs:element>

  <xs:element name="p">
    <xs:complexType mixed="true">
      <xs:complexContent>
        <xs:extension base="Inline">
          <xs:attributeGroup ref="attributes.universal" />
          <xs:attributeGroup ref="attributes.textAlign" />
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="table">
    <xs:complexType>
      <xs:sequence>
        <xs:choice>
          <xs:group ref="elements.preprocessor" />
          <xs:element ref="caption" />
        </xs:choice>
        <xs:choice>
          <xs:group ref="elements.preprocessor" />
          <xs:element ref="tbody" />
        </xs:choice>
      </xs:sequence>
      <xs:attributeGroup ref="attributes.universal" />
      <xs:attribute name="summary" type="Text" />
      <xs:attribute name="width" type="Length" />
      <xs:attribute name="border" type="Pixels" />
      <xs:attribute name="frame" type="Enum.tableFrame" />
      <xs:attribute name="rules" type="Enum.tableRules" />
      <xs:attribute name="cellspacing" type="Length" />
      <xs:attribute name="cellpadding" type="Length" />
      <xs:attribute name="align" type="Enum.blockAlign" />
      <xs:attribute name="bgcolor" type="Color" />
    </xs:complexType>
```

```
</xs:element>

<xs:element name="caption">
  <xs:complexType mixed="true">
    <xs:complexContent>
      <xs:extension base="Inline">
        <xs:attributeGroup ref="attributes.universal" />
        <xs:attribute name="align" type="Enum.captionAlign" />
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>

<xs:element name="tbody">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="tr" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="tr">
  <xs:complexType>
    <xs:choice maxOccurs="unbounded">
      <xs:group ref="elements.preprocessor" />
      <xs:element ref="th" />
      <xs:element ref="td" />
    </xs:choice>
    <xs:attributeGroup ref="attributes.universal" />
    <xs:attributeGroup ref="attributes.cellHAlign" />
    <xs:attributeGroup ref="attributes.cellVAlign" />
    <xs:attribute name="bgcolor" type="Color" />
  </xs:complexType>
</xs:element>

<xs:element name="th">
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="Block">
        <xs:attributeGroup ref="attributes.universal" />
        <xs:attributeGroup ref="attributes.cellHAlign" />
        <xs:attributeGroup ref="attributes.cellVAlign" />
        <xs:attribute name="abbr" type="Text" />
        <xs:attribute name="axis" />
        <xs:attribute name="bgcolor" type="Color" />
        <xs:attribute name="colspan" default="1" type="Number" />
        <xs:attribute name="headers" type="xs:IDREFS" />
        <xs:attribute name="height" type="Length" />
        <xs:attribute name="nowrap" type="Enum.nowrap" />
        <xs:attribute name="rowspan" default="1" type="Number" />
        <xs:attribute name="scope" type="Enum.scope" />
        <xs:attribute name="width" type="Length" />
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>

<xs:element name="td">
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="Block">
        <xs:attributeGroup ref="attributes.universal" />
        <xs:attributeGroup ref="attributes.cellHAlign" />
        <xs:attributeGroup ref="attributes.cellVAlign" />
        <xs:attribute name="abbr" type="Text" />
        <xs:attribute name="axis" />
        <xs:attribute name="bgcolor" type="Color" />
        <xs:attribute name="colspan" default="1" type="Number" />
        <xs:attribute name="headers" type="xs:IDREFS" />
        <xs:attribute name="height" type="Length" />
        <xs:attribute name="nowrap" type="Enum.nowrap" />
        <xs:attribute name="rowspan" default="1" type="Number" />
        <xs:attribute name="scope" type="Enum.scope" />
        <xs:attribute name="width" type="Length" />
      </xs:extension>
    </xs:complexContent>
```

```
1580       </xs:complexType>
1581     </xs:element>
1582
1583 </xs:schema>
```

B. XWML 1.0 Java Interfaces

WomAbbr.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * Denotes an abbreviation.
5   *
6   * Corresponds to the XHTML 1.0 Transitional element "abbr".
7   *
8   * <b>Child elements:</b> Mixed, [Inline elements]*
9   */
10 public interface WomAbbr
11         extends
12             WomInlineElement,
13             WomUniversalAttributes
14 {
15 }
```

WomArg.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * An argument to a Wikitext transclusion statement.
5   *
6   * Corresponds to the WXML 1.0 element "arg".
7   *
8   * <b>Child elements:</b> name? value
9   */
10 public interface WomArg
11         extends
12             WomNode
13 {
14     /**
15      * Get the name of the argument.
16      *
17      * Operates on the first &lt;name> element found among this node's children.
18      *
19      * @return The name of the argument or <code>null</code> if the argument
20      *         does not have a name.
21      */
22     public WomName getName();
23
24     /**
25      * Set the name of the argument.
26      *
27      * Operates on the first &lt;name> element found among this node's children.
28      * If no name node is found, the name will be added as the first child.
29      *
30      * @param name
31      *            The new name of the argument or <code>null</code> to turn the
32      *            argument into an anonymous argument.
33      * @return The old name of the argument.
34      */
35     public WomName setName(WomName name);
36
37     /**
38      * Get the value of the argument.
39      *
40      * Operates on the first &lt;value> element found among this node's
41      * children.
42      *
43      * @return The name of the value.
44      */
45     public WomValue getArgValue();
46
47     /**
48      * Set the value of the argument.
```

```
     *
     * Operates on the first &lt;value> element found among this node's
     * children. If no value node is found, the value will be added as the first
     * child.
     *
     * @param value
     *            The new value of the argument.
     * @return The old value of the argument.
     * @throws NullPointerException
     *             Thrown when <code>null</code> is passed as value.
     */
    public WomValue setArgValue(WomValue value) throws NullPointerException;
}
```

WomAttr.java

```
package org.sweble.wikitext.engine.wom;

/**
 * An attribute of a tag extension.
 *
 * Corresponds to the WXML 1.0 element "attr".
 *
 * <b>Child elements:</b> -
 */
public interface WomAttr
        extends
            WomNode
{
    /**
     * Get the name of the attribute. Attribute names are case-insensitive.
     *
     * Corresponds to the XWML 1.0 attribute "name".
     *
     * @return The name of the attribute.
     */
    public String getName();

    /**
     * Set the name of the attribute. Attribute names are case-insensitive.
     *
     * Corresponds to the XWML 1.0 attribute "name".
     *
     * @param name
     *            The new name of the attribute.
     * @return The old name of the attribute.
     * @throws IllegalArgumentException
     *             If an attribute with the given name already exists or the
     *             given name was empty or not a valid XML name.
     * @throws NullPointerException
     *             Thrown if the <code>null</code> was specified as name.
     */
    public String setName(String name) throws IllegalArgumentException,
         NullPointerException;

    /**
     * Retrieve the value of the attribute.
     *
     * Corresponds to the XWML 1.0 attribute "value".
     *
     * @return The value of the attribute.
     */
    public String getAttrValue();

    /**
     * Retrieve the value of the attribute.
     *
     * Corresponds to the XWML 1.0 attribute "value".
     *
     * @param value
     *            The new value of the attriute.
     * @return The old value of the attribute.
     * @throws NullPointerException
```

```
57      *             Thrown when <code>null</code> is passed as value.
58      */
59     public String setAttrValue(String value) throws NullPointerException;
60 }
```

WomAttribute.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * An attribute node.
5   *
6   * Objects of this class represent attributes that can be attached to other
7   * nodes that support attributes. An attribute node can only have other
8   * attribute nodes as siblings. An attribute node cannot have children or
9   * attributes of its own.
10  *
11  * <b>Child elements:</b> -
12  */
13 public interface WomAttribute
14         extends
15             WomNode
16 {
17     /**
18      * Retrieve the name of the attribute. Attribute names are case-insensitive.
19      *
20      * @return The name of the attribute.
21      */
22     public String getName();
23
24     /**
25      * Set the name of the attribute. Attribute names are case-insensitive.
26      *
27      * @param name
28      *             The new name of the attribute.
29      * @return The old name of the attribute.
30      * @throws IllegalArgumentException
31      *              If an attribute with the given name already exists or the
32      *              given name was empty or not a valid XML name.
33      * @throws NullPointerException
34      *              Thrown if the <code>null</code> was specified as name.
35      */
36     public String setName(String name) throws IllegalArgumentException,
           NullPointerException;
37
38     /**
39      * Return the value of the attribute.
40      *
41      * @return The value of the attribute.
42      */
43     @Override
44     public String getValue();
45
46     /**
47      * Set the value of the attribute.
48      *
49      * @param value
50      *             The new value of the attribute.
51      * @return The old value of the attribute.
52      */
53     public String setValue(String value);
54 }
```

WomBig.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * Denotes text that should be displayed in a larger font.
```

```
 *
 * Corresponds to the XHTML 1.0 Transitional element "big".
 *
 * <b>Child elements:</b> Mixed, [Inline elements]*
 */
public interface WomBig
        extends
            WomInlineElement,
            WomUniversalAttributes
{
}
```

WomBlockElement.java

```
package org.sweble.wikitext.engine.wom;

/**
 * An element that behaves like an XHTML 1.0 Transitional block element.
 *
 * This is an interface that groups elements that can behave like block
 * elements.
 */
public interface WomBlockElement
        extends
            WomNode
{
}
```

WomBlockquote.java

```
package org.sweble.wikitext.engine.wom;

/**
 * Denotes a long quotation.
 *
 * Corresponds to the XHTML 1.0 Transitional element "blockquote".
 *
 * <b>Child elements:</b> [Block elements]*
 */
public interface WomBlockquote
        extends
            WomBlockElement,
            WomUniversalAttributes
{
    /**
     * Get source of the quotation.
     *
     * Corresponds to the XHTML 1.0 Transitional attribute "cite".
     *
     * @return The source of the citation or <code>null</code> if the attribute
     *         is not specified.
     */
    public String getCite();

    /**
     * Set the source of the quotation.
     *
     * Corresponds to the XHTML 1.0 Transitional attribute "cite".
     *
     * @param source
     *            The source of the citation or <code>null</code> to remove the
     *            attribute.
     * @return The source of the citation.
     */
    public String setCite(String source);
}
```

WomBody.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * The body of a page or section.
5   *
6   * Corresponds to the WXML 1.0 element "body".
7   *
8   * <b>Child elements:</b> [Block elements]*
9   */
10 public interface WomBody
11         extends
12             WomNode
13 {
14 }
```

WomBold.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * Denotes text that should be rendered as bold text.
5   *
6   * Corresponds to the XHTML 1.0 Transitional element "b".
7   *
8   * <b>Child elements:</b> Mixed, [Inline elements]*
9   */
10 public interface WomBold
11         extends
12             WomInlineElement,
13             WomUniversalAttributes
14 {
15 }
```

WomBreak.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * Denotes a single line break.
5   *
6   * Corresponds to the XHTML 1.0 Transitional element "br".
7   *
8   * <b>Child elements:</b> -
9   */
10 public interface WomBreak
11         extends
12             WomInlineElement,
13             WomCoreAttributes
14 {
15     /**
16      * Get the sides on which floating elements are not allowed.
17      *
18      * Corresponds to the XHTML 1.0 Transitional attribute "clear".
19      *
20      * @return The sides on which floating elements are not allowed.
21      */
22     public WomClear getClear();
23
24     /**
25      * Set the sides on which floating elements are not allowed.
26      *
27      * Corresponds to the XHTML 1.0 Transitional attribute "clear".
28      *
29      * @param clear
30      *             The new sides on which floating elements are not allowed or
31      *             <code>null</code> if the attribute is not specified.
32      * @return The old sides on which floating elements are not allowed.
```

```
33      */
34     public WomClear setClear(WomClear clear);
35 }
```

WomBulletStyle.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * The type of bullet points used by unordered lists.
5   */
6  public enum WomBulletStyle
7  {
8      CIRCLE,
9      DISC,
10     SQUARE,
11 }
```

WomCategory.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * A category statement.
5   *
6   * Corresponds to the XWML 1.0 element "category".
7   *
8   * <b>Child elements:</b> -
9   */
10 public interface WomCategory
11         extends
12             WomProcessingInstruction,
13             WomLink
14 {
15     /**
16      * Return the category to which the page will be assigned.
17      *
18      * Corresponds to the XWML 1.0 attribute "category".
19      *
20      * @return The category.
21      */
22     public String getCategory();
23
24     /**
25      * Set the category to which the page will be assigned.
26      *
27      * Corresponds to the XWML 1.0 attribute "category".
28      *
29      * @param category
30      *             The new category.
31      * @return The old category.
32      */
33     public String setCategory(String category);
34
35     // ==[ Link interface ]=================================================
36
37     /**
38      * Returns an empty title since category statements do not provide a title.
39      *
40      * @return An empty title.
41      */
42     @Override
43     public WomTitle getLinkTitle();
44
45     /**
46      * Return the name of the category this statement is pointing to.
47      *
48      * @return The category.
49      */
```

```
50      @Override
51      public String getLinkTarget();
52  }
```

WomCenter.java

```
1   package org.sweble.wikitext.engine.wom;
2
3   /**
4    * Denotes a block that will be centered.
5    *
6    * Corresponds to the XHTML 1.0 Transitional element "center".
7    *
8    * <b>Child elements:</b> [Block elements]*
9    */
10  public interface WomCenter
11          extends
12              WomBlockElement,
13              WomUniversalAttributes
14  {
15  }
```

WomCite.java

```
1   package org.sweble.wikitext.engine.wom;
2
3   /**
4    * Denotes text as citation.
5    *
6    * Corresponds to the XHTML 1.0 Transitional element "cite".
7    *
8    * <b>Child elements:</b> Mixed, [Inline elements]*
9    */
10  public interface WomCite
11          extends
12              WomInlineElement,
13              WomUniversalAttributes
14  {
15  }
```

WomClear.java

```
1   package org.sweble.wikitext.engine.wom;
2
3   /**
4    * The sides of an element on which floating elements are not allowed.
5    */
6   public enum WomClear
7   {
8       NONE,
9       LEFT,
10      RIGHT,
11      BOTH
12  }
```

WomCode.java

```
1   package org.sweble.wikitext.engine.wom;
2
3   /**
4    * Denotes text as being source code.
```

```
 *
 * Corresponds to the XHTML 1.0 Transitional element "code".
 *
 * <b>Child elements:</b> Mixed, [Inline elements]*
 */
public interface WomCode
        extends
            WomInlineElement,
            WomUniversalAttributes
{
}
```

WomColor.java

```
package org.sweble.wikitext.engine.wom;

/**
 * A color value.
 */
public interface WomColor
{
    /**
     * Set the RGB values of the color.
     *
     * Values that are out of range will be clipped into the interval [0, 255].
     *
     * @param r
     *             A value for the red component between 0 and 255.
     * @param g
     *             A value for the green component between 0 and 255.
     * @param b
     *             A value for the blue component between 0 and 255.
     */
    public void setRGB(int r, int g, int b);

    /**
     * Get the red color component.
     *
     * @return A value between 0 and 255.
     */
    public int getRed();

    /**
     * Get the green color component.
     *
     * @return A value between 0 and 255.
     */
    public int getGreen();

    /**
     * Get the blue color component.
     *
     * @return A value between 0 and 255.
     */
    public int getBlue();
}
```

WomComment.java

```
package org.sweble.wikitext.engine.wom;

/**
 * Denotes an XML-style comment in the Wikitext.
 *
 * Corresponds to the WXML 1.0 element "comment".
 *
 * <b>Child elements:</b> Text
 */
public interface WomComment
```

```
        extends
            WomProcessingInstruction
{
    /**
     * Return the text of the comment.
     *
     * @return The text of the comment.
     */
    @Override
    public String getValue();

    /**
     * Set the text of the comment.
     *
     * @param text
     *            The new text of the comment.
     * @return The old text of the comment.
     * @throws NullPointerException
     *             Thrown if <code>null</code> is passed as text.
     * @throws IllegalArgumentException
     *             Thrown if the given text violates the syntax of XML comments
     *             (e.g.: " -" would violate the XML comment syntax since it
     *             would become "&lt;-- ---&gt;").
     */
    public String setValue(String text) throws IllegalArgumentException,
        NullPointerException;
}
```

WomCoreAttributes.java

```
package org.sweble.wikitext.engine.wom;

/**
 * The XHTML 1.0 Transitional core attributes.
 */
public interface WomCoreAttributes
{
    /**
     * Get the unique id of the element.
     *
     * Corresponds to the XHTML 1.0 Transitional attribute "id".
     *
     * @return The unique id of the element or <code>null</code> if the
     *         attribute is not specified.
     */
    public String getId();

    /**
     * Set the unique id of the element.
     *
     * Corresponds to the XHTML 1.0 Transitional attribute "id".
     *
     * @param id
     *            The new id of the element or <code>null</code> to remove the
     *            attribute.
     * @return The old unique id of the element.
     */
    public String setId(String id) throws IllegalArgumentException;

    /**
     * Get the stylesheet classes assigned to this element.
     *
     * Corresponds to the XHTML 1.0 Transitional attribute "class".
     *
     * @return A string containing the names of the stylesheet classes,
     *         separated by space. <code>null</code> if the attribute is not
     *         specified.
     */
    public String getClasses();

    /**
     * Set the stylesheet classes assigned to this element.
     *
```

```
44      * Corresponds to the XHTML 1.0 Transitional attribute "class".
45      *
46      * @param classes
47      *            A string containing the new classes or <code>null</code> to
48      *            remove the attribute.
49      * @return A string containing the old classes.
50      */
51     public String setClasses(String classes);
52
53     /**
54      * Get CSS styles directly associated with this element.
55      *
56      * Corresponds to the XHTML 1.0 Transitional attribute "style".
57      *
58      * @return The CSS styles directly associated with this element or
59      *         <code>null</code> if the attribute is not specified.
60      */
61     public String getStyle();
62
63     /**
64      * Directly associate CSS styles with this element.
65      *
66      * Corresponds to the XHTML 1.0 Transitional attribute "style".
67      *
68      * @param style
69      *            The new CSS styles to associate with this element or
70      *            <code>null</code> to remove the attribute.
71      * @return The old CSS styles that were associated with this element.
72      */
73     public String setStyle(String style);
74
75     /**
76      * Get the title of the element.
77      *
78      * Corresponds to the XHTML 1.0 Transitional attribute "title".
79      *
80      * @return The title of the element or <code>null</code> if the attribute is
81      *         not specified.
82      */
83     public String getTitlte();
84
85     /**
86      * Get the title of the element.
87      *
88      * Corresponds to the XHTML 1.0 Transitional attribute "title".
89      *
90      * @param title
91      *            The new title of this element or <code>null</code> to remove
92      *            the attribute.
93      * @return The old title of this element.
94      */
95     public String setTitlte(String title);
96 }
```

WomDefault.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * The default value of a template parameter.
5   *
6   * Corresponds to the WXML 1.0 element "default".
7   *
8   * <b>Child elements:</b> Mixed, [Preprocessor elements]*
9   */
10 public interface WomDefault
11         extends
12             WomNode
13 {
14 }
```

WomDefinitionList.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  import java.util.Collection;
4
5  /**
6   * Denotes a definition list.
7   *
8   * Definition lists can be accessed in two different ways. First by addressing
9   * individual terms and treating the definitions following each term as
10  * belonging to the respective term. Second by addressing terms and definitions
11  * through a common index.
12  *
13  * <b>Term-oriented interface:</b><br />
14  * Terms are accessed via an integer index where <code>0</code> denotes the
15  * first term. Only terms are counted using this index. All other child elements
16  * are ignored.However, they can be iterated using the methods provided by the
17  * WomNode interface.
18  *
19  * <b>Item-oriented interface:</b><br />
20  * Terms and definitions can also be accessed via an integer where
21  * <code>0</code> denotes the first term <b>or</b> definition. <b>Only terms and
22  * definitions are counted.</b> All other child elements are skipped in the
23  * enumeration and are not accessible through this interface. However, they can
24  * be iterated using the methods provided by the WomNode interface.
25  *
26  * Corresponds to the XHTML 1.0 Transitional element "dl".
27  *
28  * <b>Child elements:</b> ([Preprocessor elements]|dd|dt)*
29  */
30 public interface WomDefinitionList
31         extends
32             WomBlockElement,
33             WomUniversalAttributes
34 {
35     // ==[ Term-oriented interface ]================================================
36
37     /**
38      * Get the number of terms in this list.
39      *
40      * @return The number of terms in this list.
41      */
42     public int getTermNum();
43
44     /**
45      * Get a collection containing all terms.
46      *
47      * @return A collection containing all terms.
48      */
49     public Collection<WomDefinitionListTerm> getTerms();
50
51     /**
52      * Get a certain term from the list.
53      *
54      * @param index
55      *             The zero-based index of the term to retrieve. Only terms are
56      *             counted by this index!
57      * @return The term with the given index.
58      * @throws IndexOutOfBoundsException
59      *              Thrown if the given index is out of range.
60      */
61     public WomDefinitionListTerm getTerm(int index) throws IndexOutOfBoundsException;
62
63     /**
64      * Replace a certain term and all its definitions by another term and its
65      * definitions.
66      *
67      * @param index
68      *             The zero-based index of the term to replace. Only terms are
69      *             counted by this index!
70      * @param item
71      *             The replacement term.
72      * @throws IndexOutOfBoundsException
73      *              Thrown if the given index is out of range.
74      */
```

```
75      public void replaceTerm(int index, WomDefinitionListTerm term) throws
            IndexOutOfBoundsException;
76
77      /**
78       * Replace a certain term and all its definitions by another term and its
79       * definitions.
80       *
81       * @param search
82       *            The term to replace.
83       * @param item
84       *            The replacement term.
85       * @throws IllegalArgumentException
86       *             Thrown if the given term <code>term</code> is not a term of
87       *             this list.
88       */
89      public void replaceTerm(WomDefinitionListTerm search, WomDefinitionListTerm replace)
            throws IllegalArgumentException;
90
91      /**
92       * Remove a term and all its definitions from the list.
93       *
94       * @param index
95       *            The zero-based index of the term to remove. Only terms are
96       *            counted by this index!
97       * @throws IndexOutOfBoundsException
98       *             Thrown if the given index is out of range.
99       */
100     public void removeTerm(int index) throws IndexOutOfBoundsException;
101
102     /**
103      * Remove a term and all its definitions from the list.
104      *
105      * @param term
106      *            The term to remove.
107      * @throws IllegalArgumentException
108      *             Thrown if the given term <code>term</code> is not a term of
109      *             this list.
110      */
111     public void removeTerm(WomDefinitionListTerm term) throws IllegalArgumentException;
112
113     /**
114      * Append a term and its definitions to the list.
115      *
116      * @param term
117      *            The term to append.
118      */
119     public void appendTerm(WomDefinitionListTerm term);
120
121     /**
122      * Insert a term and its definitions at the given index into the list.
123      *
124      * @param beforeIndex
125      *            The index of the term in front of which the new term and its
126      *            definitions is to be inserted. This index only counts terms!
127      * @param term
128      *            The term to insert. The term will have the given index
129      *            <code>beforeIndex</code> after insertion.
130      * @throws IndexOutOfBoundsException
131      *             Thrown if <code>0 <= beforeIndex <= getItemNum()</code> does
132      *             not hold.
133      */
134     public void insertItem(int beforeIndex, WomDefinitionListTerm term) throws
            IndexOutOfBoundsException;
135
136     /**
137      * Insert a term and its definitions at the given index into the list.
138      *
139      * @param before
140      *            The term in front of which the new should be inserted. This
141      *            index only counts terms!
142      * @param term
143      *            The term to insert.
144      * @throws IllegalArgumentException
145      *             Thrown if the given term <code>term</code> is not a term of
146      *             this list.
147      */
```

```
148     public void insertItem(WomDefinitionListTerm before, WomDefinitionListTerm term)
            throws IllegalArgumentException;
149
150     // ==[ Item-oriented interface ]=============================================
151
152     /**
153      * Get the number of terms and definitions in this list.
154      *
155      * @return The number of terms and definitions in this list.
156      */
157     public int getItemNum();
158
159     /**
160      * Get a collection containing all terms and definitions.
161      *
162      * @return A collection with all items of the list.
163      */
164     public Collection<WomDefinitionListItem> getItems();
165
166     /**
167      * Get a certain term or definition from the list.
168      *
169      * @param index
170      *            The zero-based index of the item to retrieve.
171      * @return The item with the given index.
172      * @throws IndexOutOfBoundsException
173      *             If the given index is out of range.
174      */
175     public WomDefinitionListItem getItem(int index) throws IndexOutOfBoundsException;
176
177     /**
178      * Replace a certain term or definition in the list.
179      *
180      * If the replacement item is a term with definitions attached to it,
181      * definitions <b>will not</b> be replaced. Instead the definitions attached
182      * to the replacement term will be discarded!
183      *
184      * @param index
185      *            The zero-based index of the item to replace.
186      * @param item
187      *            The replacement item.
188      * @return The old item with the given index.
189      * @throws IndexOutOfBoundsException
190      *             If the given index is out of range.
191      */
192     public WomDefinitionListItem replaceItem(int index, WomDefinitionListItem item)
            throws IndexOutOfBoundsException;
193
194     /**
195      * Remove a term or definition from the list.
196      *
197      * If the item that should be removed is a term, <b>only the term</b> will
198      * be removed. Its definitions are left untouched and become the definitions
199      * of the preceding term (if there is a preceding term).
200      *
201      * @param index
202      *            The zero-based index of the item to remove.
203      * @return The removed item.
204      * @throws IndexOutOfBoundsException
205      *             If the given index is out of range.
206      */
207     public WomDefinitionListItem removeItem(int index) throws IndexOutOfBoundsException;
208
209     /**
210      * Append term or definition to the list.
211      *
212      * If the item that should be appended is a term with definitions attached,
213      * the definitions <b>will not</b> be inserted into the list. Instead the
214      * term's definitions will be discarded.
215      *
216      * @param item
217      *            The item to append.
218      */
219     public void appendItem(WomDefinitionListItem item);
220
221     /**
```

```
222       * Insert a term or definition at the given index into the list.
223       *
224       * If the item that should be appended is a term with definitions attached,
225       * the definitions <b>will not</b> be inserted into the list. Instead the
226       * term's definitions will be discarded.
227       *
228       * @param beforeIndex
229       *            The index of the item in front of which the new item is to be
230       *            inserted.
231       * @param item
232       *            The item to insert. The item will have the given index
233       *            <code>beforeIndex</code> after insertion.
234       * @throws IndexOutOfBoundsException
235       *             Thrown if <code>0 <= beforeIndex <= getItemNum()</code> does
236       *             not hold.
237       */
238      public void insertItem(int beforeIndex, WomDefinitionListItem item) throws
             IndexOutOfBoundsException;
239
240      // ==[ The XHTML Attributes ]==========================================
241
242      /**
243       * Tells whether the list should be displayed as compact list.
244       *
245       * Corresponds to the XHTML 1.0 Transitional attribute "compact".
246       *
247       * @return Whether the "compact" flag is given or not.
248       */
249      public boolean isCompact();
250
251      /**
252       * Set whether the list should be displayed as compact list.
253       *
254       * Corresponds to the XHTML 1.0 Transitional attribute "compact".
255       *
256       * @param compact
257       *            Set (<code>true</code>) or remove (<code>false</code>) the
258       *            "compact" flag.
259       * @return The old state.
260       */
261      public boolean setCompact(boolean compact);
262  }
```

WomDefinitionListDef.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * Denotes a definition list definition.
5   *
6   * Corresponds to the XHTML 1.0 Transitional element "dd".
7   *
8   * <b>Child elements:</b> [Block elements]*
9   */
10 public interface WomDefinitionListDef
11         extends
12             WomDefinitionListItem,
13             WomUniversalAttributes
14 {
15 }
```

WomDefinitionListItem.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * Interface to the items &lt;dt> and &lt;dd> of a definition list.
5   */
6  public interface WomDefinitionListItem
```

```
        extends
            WomNode
{
}
```

WomDefinitionListTerm.java

```
package org.sweble.wikitext.engine.wom;

import java.util.Collection;

/**
 * Denotes a definition list term.
 *
 * While in HTML a definition list is a loose collection of terms and
 * definitions, the WOM interfaces also offer a more abstract view on a
 * definition list in which one or more definitions belong to a term.
 *
 * Therefore, if one retrieves a term from a definition list, the definitions
 * that follow the term can also be addressed and altered through the term
 * interface (this interface).
 *
 * However, the definitions that are considered part of a term are still
 * <b>not</b> children of that term! They are only attached to the term
 * "virtually". Altering definitions through the term interface will actually
 * alter the definition list to which the term belongs. If a term does not yet
 * belong to a definition list, definitions that get attached to the term will
 * be temporarily stored in that term (but not as its children) and will be
 * attached to the definition list once the term is attached to a definition
 * list.
 *
 * Individual definitions are addressed using a zero-based integer index where
 * <code>0</code> denotes the first definition that follows this term in the
 * definition list. If this term is followed by another term in the definition
 * list and there are not definitions between the two terms, than this term does
 * not have any definitions.
 *
 * Corresponds to the XHTML 1.0 Transitional element "dt".
 *
 * <b>Child elements:</b> Mixed, [Inline elements]*
 */
public interface WomDefinitionListTerm
        extends
            WomDefinitionListItem,
            WomUniversalAttributes
{
    /**
     * Get the number of definitions of this term.
     *
     * @return The number of definitions of this term.
     */
    public int getDefNum();

    /**
     * Get a collection containing all definitions of this term.
     *
     * @return A collection containing all definitions of this term.
     */
    public Collection<WomDefinitionListDef> getDefs();

    /**
     * Get a certain definition of this term.
     *
     * @param index
     *            The zero-based index of the definition to retrieve.
     * @return The definition with the given index.
     * @throws IndexOutOfBoundsException
     *             If the given index is out of range.
     */
    public WomDefinitionListDef getDef(int index) throws IndexOutOfBoundsException;

    /**
     * Replace a definition of this term.
```

```
67          *
68          * @param index
69          *            The zero-based index of the definition to replace.
70          * @param def
71          *            The replacement definition.
72          * @return The replaced definition.
73          * @throws IndexOutOfBoundsException
74          *             If the given index is out of range.
75          */
76         public WomDefinitionListDef replaceDef(int index, WomDefinitionListDef def) throws
                IndexOutOfBoundsException;
77
78         /**
79          * Remove a definition if this term.
80          *
81          * @param index
82          *            The zero-based index of the definition to remove.
83          * @return The removed definition.
84          * @throws IndexOutOfBoundsException
85          *             If the given index is out of range.
86          */
87         public WomDefinitionListDef removeDef(int index) throws IndexOutOfBoundsException;
88
89         /**
90          * Append a definition to the term.
91          *
92          * @param def
93          *            The item to append.
94          */
95         public void appendDef(WomDefinitionListDef def);
96
97         /**
98          * Insert a definition at the given index into the list of definitions of
99          * this term.
100         *
101         * @param beforeIndex
102         *            The index of the definition in front of which the new
103         *            definition is to be inserted.
104         * @param def
105         *            The definition to insert. The definition will have the given
106         *            index <code>beforeIndex</code> after insertion.
107         * @throws IndexOutOfBoundsException
108         *             Thrown if <code>0 <= beforeIndex <= getDefNum()</code> does
109         *             not hold.
110         */
111        public void insertItem(int beforeIndex, WomDefinitionListDef def) throws
               IndexOutOfBoundsException;
112 }
```

WomDel.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  import java.util.Date;
4
5  /**
6   * Denotes text or a block that has been removed.
7   *
8   * Corresponds to the XHTML 1.0 Transitional element "del".
9   *
10  * <b>Child elements:</b> Mixed, [Flow elements]*
11  */
12 public interface WomDel
13         extends
14             WomInlineElement,
15             WomBlockElement,
16             WomUniversalAttributes
17 {
18     /**
19      * Get the url of a document that specifies the reasons for the change.
20      *
21      * Corresponds to the XHTML 1.0 Transitional attribute "cite".
22      *
```

```
23      * @return The url or <code>null</code> if the attribute is not specified.
24      */
25     public String getCite();
26
27     /**
28      * Set the url of a document that specifies the reasons for the change.
29      *
30      * Corresponds to the XHTML 1.0 Transitional attribute "cite".
31      *
32      * @param url
33      *            The new url or <code>null</code> to remove the attribute.
34      * @return The The old url.
35      */
36     public String getCite(String url);
37
38     /**
39      * Get the timestamp when the text or block was deleted.
40      *
41      * Corresponds to the XHTML 1.0 Transitional attribute "cite".
42      *
43      * @return The date and time of the deletion or <code>null</code> if the
44      *         attribute is not specified.
45      */
46     public Date getDatetime();
47
48     /**
49      * Set the timestamp when the text or block was deleted.
50      *
51      * Corresponds to the XHTML 1.0 Transitional attribute "cite".
52      *
53      * @param timestamp
54      *            The new timestamp or <code>null</code> to remove the
55      *            attribute.
56      * @return The old timestamp.
57      */
58     public Date getDatetime(Date timestamp);
59 }
```

WomDfn.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * Denotes a definition term.
5   *
6   * Corresponds to the XHTML 1.0 Transitional element "dfn".
7   *
8   * <b>Child elements:</b> Mixed, [Inline elements]*
9   */
10 public interface WomDfn
11         extends
12             WomInlineElement,
13             WomUniversalAttributes
14 {
15 }
```

WomDiv.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * Denotes a general block.
5   *
6   * Corresponds to the XHTML 1.0 Transitional element "div".
7   *
8   * <b>Child elements:</b> Mixed, [Flow elements]*
9   */
10 public interface WomDiv
11         extends
```

```
            WomBlockElement,
            WomUniversalAttributes
{
    /**
     * Get the alignment of the content inside the tag.
     *
     * Corresponds to the XHTML 1.0 Transitional attribute "align".
     *
     * @return The alignment or <code>null</code> if the attribute is not
     *         specified.
     */
    public WomHorizAlign getAlign();

    /**
     * Set the alignment of the content inside the tag.
     *
     * Corresponds to the XHTML 1.0 Transitional attribute "align".
     *
     * @param align
     *            The new alignment or <code>null</code> to remove the
     *            attribute. Only the values LEFT, RIGHT, CENTER and JUSTIFY are
     *            allowed.
     * @return The old alignment.
     * @throws IllegalArgumentException
     *             Thrown if an illegal value is specified as alignment.
     */
    public WomHorizAlign setAlign(WomHorizAlign align) throws IllegalArgumentException;
}
```

WomElement.java

```
package org.sweble.wikitext.engine.wom;

import java.util.Collection;

/**
 * An arbitrary XML element not part of the XHTML 1.0 Transitional
 * specification.
 *
 * Corresponds to the WXML 1.0 element "element".
 *
 * <b>Child elements:</b> attr* elembody?
 */
public interface WomElement
        extends
            WomInlineElement
{
    /**
     * Get the name of the element.
     *
     * Corresponds to the XWML 1.0 attribute "name".
     *
     * @return The name of the element.
     */
    public String getName();

    /**
     * Set the name of the element.
     *
     * Corresponds to the XWML 1.0 attribute "name".
     *
     * @param name
     *            The new name of the element.
     * @return The old name of the element.
     * @throws IllegalArgumentException
     *             Thrown if an empty name is passed or the name is not a valid
     *             XML name.
     * @throws NullPointerException
     *             Thrown if <code>null</code> is passed as name.
     */
    public String setName(String name) throws IllegalArgumentException,
         NullPointerException;

```

```
42      /**
43       * Return a collection containing the XML attributes of the element.
44       *
45       * @return A collection containing the XML attributes of the element.
46       */
47      public Collection<WomAttr> getElemAttributes();
48
49      /**
50       * Return the value of an attribute. If no attribute with the given name
51       * exists <code>null</code> is returned.
52       *
53       * @return The attribute with the given name or <code>null</code>.
54       */
55      public WomAttr getElemAttribute(String name);
56
57      /**
58       * Remove an attribute.
59       *
60       * @return The removed attribute node or <code>null</code> if no such
61       *         attribute exists.
62       */
63      public WomAttr removeElemAttribute(String name);
64
65      /**
66       * Sets an attribute node. If the attribute already exists, it will be
67       * replaced by the given attribute. Otherwise, a new attribute will be
68       * created.
69       *
70       * @param value
71       *            Passing <code>null</code> as value will remove the attribute.
72       * @return The old attribute or <code>null</code> if the attribute did not
73       *         exist.
74       */
75      public WomAttr setElemAttribute(String name, String value);
76
77      /**
78       * Sets an attribute node. If the attribute already exists, it will be
79       * replaced by the given attribute.
80       *
81       * @return The old attribute or <code>null</code> if the attribute did not
82       *         exist.
83       */
84      public WomAttr setElemAttribute(WomAttr attr);
85
86      /**
87       * Get the body of the element.
88       *
89       * @return The body of the element or <code>null</code> if the element only
90       *         consists of an empty tag. An empty element that consists of a
91       *         start tag and an end tag returns an empty body.
92       */
93      public WomElementBody getBody();
94
95      /**
96       * Set the body of the element.
97       *
98       * @param body
99       *            The new body of the element or <code>null</code> to turn the
100      *            element into an empty tag.
101      * @return The old body of the element.
102      */
103     public WomElementBody setBody(WomElementBody body);
104 }
```

WomElementBody.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * The body of a page or section.
5   *
6   * Corresponds to the WXML 1.0 element "elembody".
7   *
```

```
 * <b>Child elements:</b> Mixed, &lt;any>*
 */
public interface WomElementBody
        extends
            WomNode
{
}
```

WomEmphasize.java

```
package org.sweble.wikitext.engine.wom;

/**
 * Denotes text that should be rendered as emphasized text.
 *
 * Corresponds to the XHTML 1.0 Transitional element "em".
 *
 * <b>Child elements:</b> Mixed, [Inline elements]*
 */
public interface WomEmphasize
        extends
            WomInlineElement,
            WomUniversalAttributes
{
}
```

WomEventAttributes.java

```
package org.sweble.wikitext.engine.wom;

/**
 * The XHTML 1.0 Transitional event attributes.
 */
public interface WomEventAttributes
{
    /**
     * Return the "onclick" event handler call.
     *
     * Corresponds to the XHTML 1.0 Transitional attribute "onclick".
     *
     * @return The "onclick" event handler call or <code>null</code> if the
     *         attribute is not specified.
     */
    public String getOnclick();

    /**
     * Set the "onclick" event handler call.
     *
     * Corresponds to the XHTML 1.0 Transitional attribute "onclick".
     *
     * @param handler
     *            The new handler or <code>null</code> to remove the attribute.
     * @return The old handler.
     */
    public String setOnclick(String handler);

    /**
     * Return the "ondblclick" event handler call.
     *
     * Corresponds to the XHTML 1.0 Transitional attribute "ondblclick".
     *
     * @return The "ondblclick" event handler call or <code>null</code> if the
     *         attribute is not specified.
     */
    public String getOndblclick();

    /**
     * Set the "ondblclick" event handler call.
     *
```

```
     * Corresponds to the XHTML 1.0 Transitional attribute "ondblclick".
     *
     * @param handler
     *            The new handler or <code>null</code> to remove the attribute.
     * @return The old handler.
     */
    public String setOndblclick(String handler);

    /**
     * Return the "onmousedown" event handler call.
     *
     * Corresponds to the XHTML 1.0 Transitional attribute "onmousedown".
     *
     * @return The "onmousedown" event handler call or <code>null</code> if the
     *         attribute is not specified.
     */
    public String getOnmousedown();

    /**
     * Set the "onmousedown" event handler call.
     *
     * Corresponds to the XHTML 1.0 Transitional attribute "onmousedown".
     *
     * @param handler
     *            The new handler or <code>null</code> to remove the attribute.
     * @return The old handler.
     */
    public String setOnmousedown(String handler);

    /**
     * Return the "onmouseup" event handler call.
     *
     * Corresponds to the XHTML 1.0 Transitional attribute "onmouseup".
     *
     * @return The "onmouseup" event handler call or <code>null</code> if the
     *         attribute is not specified.
     */
    public String getOnmouseup();

    /**
     * Set the "onmouseup" event handler call.
     *
     * Corresponds to the XHTML 1.0 Transitional attribute "onmouseup".
     *
     * @param handler
     *            The new handler or <code>null</code> to remove the attribute.
     * @return The old handler.
     */
    public String setOnmouseup(String handler);

    /**
     * Return the "onmouseover" event handler call.
     *
     * Corresponds to the XHTML 1.0 Transitional attribute "onmouseover".
     *
     * @return The "onmouseover" event handler call or <code>null</code> if the
     *         attribute is not specified.
     */
    public String getOnmouseover();

    /**
     * Set the "onmouseover" event handler call.
     *
     * Corresponds to the XHTML 1.0 Transitional attribute "onmouseover".
     *
     * @param handler
     *            The new handler or <code>null</code> to remove the attribute.
     * @return The old handler.
     */
    public String setOnmouseover(String handler);

    /**
     * Return the "onmousemove" event handler call.
     *
     * Corresponds to the XHTML 1.0 Transitional attribute "onmousemove".
     *
```

```
118       * @return The "onmousemove" event handler call or <code>null</code> if the
119       *         attribute is not specified.
120       */
121      public String getOnmousemove();
122
123      /**
124       * Set the "onmousemove" event handler call.
125       *
126       * Corresponds to the XHTML 1.0 Transitional attribute "onmousemove".
127       *
128       * @param handler
129       *            The new handler or <code>null</code> to remove the attribute.
130       * @return The old handler.
131       */
132      public String setOnmousemove(String handler);
133
134      /**
135       * Return the "onmouseout" event handler call.
136       *
137       * Corresponds to the XHTML 1.0 Transitional attribute "onmouseout".
138       *
139       * @return The "onmouseout" event handler call or <code>null</code> if the
140       *         attribute is not specified.
141       */
142      public String getOnmouseout();
143
144      /**
145       * Set the "onmouseout" event handler call.
146       *
147       * Corresponds to the XHTML 1.0 Transitional attribute "onmouseout".
148       *
149       * @param handler
150       *            The new handler or <code>null</code> to remove the attribute.
151       * @return The old handler.
152       */
153      public String setOnmouseout(String handler);
154
155      /**
156       * Return the "onkeypress" event handler call.
157       *
158       * Corresponds to the XHTML 1.0 Transitional attribute "onkeypressw".
159       *
160       * @return The "onkeypress" event handler call or <code>null</code> if the
161       *         attribute is not specified.
162       */
163      public String getOnkeypress();
164
165      /**
166       * Set the "onkeypress" event handler call.
167       *
168       * Corresponds to the XHTML 1.0 Transitional attribute "onkeypressw".
169       *
170       * @param handler
171       *            The new handler or <code>null</code> to remove the attribute.
172       * @return The old handler.
173       */
174      public String setOnkeypress(String handler);
175
176      /**
177       * Return the "onkeydown" event handler call.
178       *
179       * Corresponds to the XHTML 1.0 Transitional attribute "onkeydown".
180       *
181       * @return The "onkeydown" event handler call or <code>null</code> if the
182       *         attribute is not specified.
183       */
184      public String getOnkeydown();
185
186      /**
187       * Set the "onkeydown" event handler call.
188       *
189       * Corresponds to the XHTML 1.0 Transitional attribute "onkeydown".
190       *
191       * @param handler
192       *            The new handler or <code>null</code> to remove the attribute.
193       * @return The old handler.
```

```
194       */
195      public String setOnkeydown(String handler);
196
197      /**
198       * Return the "onkeyup" event handler call.
199       *
200       * Corresponds to the XHTML 1.0 Transitional attribute "onkeyup".
201       *
202       * @return The "onkeyup" event handler call or <code>null</code> if the
203       *         attribute is not specified.
204       */
205      public String getOnkeyup();
206
207      /**
208       * Set the "onkeyup" event handler call.
209       *
210       * Corresponds to the XHTML 1.0 Transitional attribute "onkeyup".
211       *
212       * @param handler
213       *            The new handler or <code>null</code> to remove the attribute.
214       * @return The old handler.
215       */
216      public String setOnkeyup(String handler);
217  }
```

WomExtLink.java

```
1   package org.sweble.wikitext.engine.wom;
2
3   import java.net.URL;
4
5   /**
6    * Denotes a Wikitext bracketed external link.
7    *
8    * Corresponds to the WXML 1.0 element "extlink".
9    *
10   * <b>Child elements:</b> title?
11   */
12  public interface WomExtLink
13          extends
14              WomInlineElement,
15              WomLink
16  {
17      /**
18       * Get the title of the external link.
19       *
20       * @return The title of the external link or <code>null</code> if the link
21       *         does not specify a title.
22       */
23      public WomTitle getTitle();
24
25      /**
26       * Set the title of the external link.
27       *
28       * @param title
29       *            The new title of the external link or <code>null</code> to
30       *            remove the title.
31       * @return The old link title node.
32       */
33      public WomTitle setTitle(WomTitle title);
34
35      /**
36       * Retrieve the target of this link.
37       *
38       * Corresponds to the XWML 1.0 attribute "target".
39       *
40       * @return The target of this link.
41       */
42      public URL getTarget();
43
44      /**
45       * Set a new target for this external link.
46       *
```

```
     * Corresponds to the XWML 1.0 attribute "target".
     *
     * @param target
     *            The new target of the external link.
     * @return The old target of the external link.
     * @throws NullPointerException
     *             Thrown if <code>null</code>is passed as URL.
     */
    public URL setTarget(URL target) throws NullPointerException;

    // ==[ Link interface ]==================================================

    /**
     * Returns the title of the external link. If the external link does not
     * specify a title, an empty title is returned.
     *
     * @return The title of the external link.
     */
    @Override
    public WomTitle getLinkTitle();

    /**
     * Retrieve the target of this link.
     *
     * @return The target of this link.
     */
    @Override
    public URL getLinkTarget();
}
```

WomFont.java

```
package org.sweble.wikitext.engine.wom;

/**
 * Specifies the font color, font face and size of the text content.
 *
 * Corresponds to the XHTML 1.0 Transitional element "font".
 *
 * <b>Child elements:</b> Mixed, [Inline elements]*
 */
public interface WomFont
        extends
            WomInlineElement,
            WomCoreAttributes,
            WomI18nAttributes
{
    /**
     * Get the color of the text content.
     *
     * Corresponds to the XHTML 1.0 Transitional attribute "color".
     *
     * @return The text color or <code>null</code> if the attribute is not
     *         specified.
     */
    public WomColor getColor();

    /**
     * Set the color of the text content.
     *
     * Corresponds to the XHTML 1.0 Transitional attribute "color".
     *
     * @param color
     *            The new color of the text content or <code>null</code> to
     *            remove the attribute.
     * @return The old color of the text content.
     */
    public WomColor setColor(WomColor color);

    /**
     * Get the name of the font face of the text content.
     *
     * Corresponds to the XHTML 1.0 Transitional attribute "face".
```

```
42      *
43      * @return The name of the font face or <code>null</code> if the attribute
44      *         is not specified.
45      */
46     public String getFace();
47
48     /**
49      * Set the name of the font face.
50      *
51      * Corresponds to the XHTML 1.0 Transitional attribute "face".
52      *
53      * @param face
54      *            The name of the new font face.
55      * @return The name of the old font face.
56      */
57     public String setFace(String face);
58
59     /**
60      * Get the size of the text content.
61      *
62      * Corresponds to the XHTML 1.0 Transitional attribute "size".
63      *
64      * @return The size of the text content. A value between 1 and 7.
65      */
66     public int getSize();
67
68     /**
69      * Set the text size.
70      *
71      * Corresponds to the XHTML 1.0 Transitional attribute "size".
72      *
73      * @param size
74      *            The new text size. A value between 1 and 7.
75      * @return The old text size.
76      */
77     public int setSize(int size);
78 }
```

WomHeading.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * Denotes the heading of a section.
5   *
6   * The level of the heading is stored in the section node which is always the
7   * parent node of a heading. This interface is only concerned with the content
8   * of the heading and the content's formatting.
9   *
10  * Corresponds to the WXML 1.0 element "heading". Also partly corresponds to the
11  * XHTML 1.0 Transitional elements "h1" - "h6".
12  *
13  * <b>Child elements:</b> Mixed, [Inline elements]*
14  */
15 public interface WomHeading
16         extends
17             WomNode,
18             WomUniversalAttributes
19 {
20     /**
21      * Get the alignment of the content inside the tag.
22      *
23      * Corresponds to the XHTML 1.0 Transitional attribute "align".
24      *
25      * @return The alignment or <code>null</code> if the attribute is not
26      *         specified.
27      */
28     public WomHorizAlign getAlign();
29
30     /**
31      * Set the alignment of the content inside the tag.
32      *
33      * Corresponds to the XHTML 1.0 Transitional attribute "align".
```

```
34      *
35      * @param align
36      *            The new alignment or <code>null</code> to remove the
37      *            attribute. Only the values LEFT, RIGHT, CENTER and JUSTIFY are
38      *            allowed.
39      * @return The old alignment.
40      * @throws IllegalArgumentException
41      *             Thrown if an illegal value is specified as alignment.
42      */
43     public WomHorizAlign setAlign(WomHorizAlign align) throws IllegalArgumentException;
44 }
```

WomHorizAlign.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * Alignment attribute on DIV, H1-H6, HR, P, TABLE, TD, TH and TR elements.
5   */
6  public enum WomHorizAlign
7  {
8      LEFT,
9      RIGHT,
10     CENTER,
11
12     /**
13      * Not applicable to HR and TABLE.
14      */
15     JUSTIFY,
16
17     /**
18      * Only applicable to TD, TH and TR.
19      */
20     CHAR
21 }
```

WomHorizontalRule.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * Denotes a horizontal rule.
5   *
6   * Corresponds to the XHTML 1.0 Transitional element "hr".
7   *
8   * <b>Child elements:</b> -
9   */
10 public interface WomHorizontalRule
11         extends
12             WomBlockElement,
13             WomUniversalAttributes
14 {
15     /**
16      * Get the alignment of the horizontal rule.
17      *
18      * Corresponds to the XHTML 1.0 Transitional attribute "align".
19      *
20      * @return The alignment of the horizontal rule or <code>null</code> if the
21      *          attribute is not specified.
22      */
23     public WomHorizAlign getAlign();
24
25     /**
26      * Set the alignment of the horizontal rule.
27      *
28      * Corresponds to the XHTML 1.0 Transitional attribute "align".
29      *
30      * @param align
31      *            The alignment. Only the values <code>left</code>,
```

```
32          *            <code>center</code> and <code>right</code> are allowed.
33          * @return The old alignment of the horizontal rule.
34          */
35         public WomHorizAlign setAlign(WomHorizAlign align);
36
37         /**
38          * Get whether the horizontal rule is display with a 3-D effect (shade) or
39          * without (no-shade).
40          *
41          * Corresponds to the XHTML 1.0 Transitional attribute "noshade".
42          *
43          * @return <code>True</true> for no 3-D effect, <code>false</code> for a 3-D
44          *         effect.
45          */
46         public boolean isNoshade();
47
48         /**
49          * Set whether the horizontal rule is display with a 3-D effect (shade) or
50          * without (no-shade).
51          *
52          * Corresponds to the XHTML 1.0 Transitional attribute "noshade".
53          *
54          * @param noshade
55          *            The new setting.
56          * @return The old setting.
57          */
58         public boolean setNoshade(boolean noshade);
59
60         /**
61          * Get the thickness of the horizontal rule in pixels.
62          *
63          * Corresponds to the XHTML 1.0 Transitional attribute "size".
64          *
65          * @return The thickness in pixels or <code>null</code> if the attribute is
66          *         not specified.
67          */
68         public Integer getSize();
69
70         /**
71          * Set the thickness of the horizontal rule in pixels.
72          *
73          * Corresponds to the XHTML 1.0 Transitional attribute "size".
74          *
75          * @param size
76          *            The new thickness in pixels or <code>null</code> to remove the
77          *            attribute.
78          * @return The old thickness in pixels.
79          */
80         public Integer setSize(Integer size);
81
82         /**
83          * Get the width of the horizontal rule.
84          *
85          * Corresponds to the XHTML 1.0 Transitional attribute "width".
86          *
87          * @return The width in pixels or percent or <code>null</code> if the
88          *         attribute is not specified.
89          */
90         public WomValueWithUnit getWidth();
91
92         /**
93          * Set the width of the horizontal rule.
94          *
95          * Corresponds to the XHTML 1.0 Transitional attribute "width".
96          *
97          * @param size
98          *            The new width in pixels or percent or <code>null</code> to
99          *            remove the attribute.
100         * @return The old width in pixels or percent.
101         */
102        public WomValueWithUnit setWidth(WomValueWithUnit width);
103    }
```

WomI18nAttributes.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * XHTML 1.0 Transitional Internationalization attributes.
5   */
6  public interface WomI18nAttributes
7  {
8      /**
9       * Get the text direction of the element.
10      *
11      * Corresponds to the XHTML 1.0 Transitional attribute "dir".
12      *
13      * @return The text direction of the element's content or <code>null</code>
14      *         if the attribute is not specified.
15      */
16     public WomI18nDir getDir();
17
18     /**
19      * Get the text direction of the element.
20      *
21      * Corresponds to the XHTML 1.0 Transitional attribute "dir".
22      *
23      * @param dir
24      *            The new text direction or <code>null</code> to remove the
25      *            attribute.
26      * @return The old text direction.
27      */
28     public WomI18nDir setDir(WomI18nDir dir);
29
30     /**
31      * Get the language code for content inside this element.
32      *
33      * Corresponds to the XHTML 1.0 Transitional attribute "lang".
34      *
35      * @return The language code of the element's content or <code>null</code>
36      *         if the attribute is not specified.
37      */
38     public String getLang();
39
40     /**
41      * Get the language code for content inside this element.
42      *
43      * Corresponds to the XHTML 1.0 Transitional attribute "lang".
44      *
45      * @param lang
46      *            The new language code or <code>null</code> to remove the
47      *            attribute.
48      * @return The old language code.
49      */
50     public String setLang(String lang);
51
52     /**
53      * Get the language code for content inside this element.
54      *
55      * Corresponds to the XHTML 1.0 Transitional attribute "xml:lang".
56      *
57      * @return The language code of the element's content or <code>null</code>
58      *         if the attribute is not specified.
59      */
60     public String getXmlLang();
61
62     /**
63      * Get the language code for content inside this element.
64      *
65      * Corresponds to the XHTML 1.0 Transitional attribute "xml:lang".
66      *
67      * @param lang
68      *            The new language code or <code>null</code> to remove the
69      *            attribute.
70      * @return The old language code.
71      */
72     public String setXmlLang(String lang);
73 }
```

WomI18nDir.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * Text direction inside an element.
5   */
6  public enum WomI18nDir
7  {
8      LTR,
9      RTL
10 }
```

WomImage.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  import java.net.URL;
4
5  /**
6   * Denotes a Wikitext image.
7   *
8   * Corresponds to the WXML 1.0 element "image".
9   *
10  * <b>Child elements:</b> imgcaption?
11  */
12 public interface WomImage
13         extends
14             WomInlineElement,
15             WomLink
16 {
17     /**
18      * Get the source of the image.
19      *
20      * Corresponds to the XWML 1.0 attribute "source".
21      *
22      * @return The page title of the image.
23      */
24     public String getSource();
25
26     /**
27      * Set the source of the image.
28      *
29      * Corresponds to the XWML 1.0 attribute "source".
30      *
31      * @param source
32      *             The page title of the new image source.
33      * @return The page title of the old image source.
34      * @throws IllegalArgumentException
35      *             Thrown if an empty source is passed or the source is not a
36      *             valid page title.
37      * @throws NullPointerException
38      *             Thrown if <code>null</code> is passed as source.
39      */
40     public String setSource(String source) throws IllegalArgumentException,
          NullPointerException;
41
42     /**
43      * Get the image rendering format.
44      *
45      * @return The image rendering format.
46      */
47     public WomImageFormat getFormat();
48
49     /**
50      * Set the image rendering format.
51      *
52      * @param format
53      *             The new image rendering format or <code>null</code> to remove
54      *             the attribute. Passing <code>unrestrained</code> also removes
55      *             the attribute as this is the default.
56      * @return The old image rendering format.
57      * @throws NullPointerException
```

```
58        *             Thrown if <code>null</code> is given as format.
59        */
60      public WomImageFormat setFormat(WomImageFormat format) throws NullPointerException;
61
62      /**
63       * Get whether the image will be rendered with a grey border.
64       *
65       * Corresponds to the XWML 1.0 attribute "border".
66       *
67       * @return <code>True</code> if the image is rendered with a grey border,
68       *         <code>false</code> otherwise.
69       */
70      public boolean isBorder();
71
72      /**
73       * Set whether the image should be rendered with a grey border
74       *
75       * Corresponds to the XWML 1.0 attribute "border".
76       *
77       * @param border
78       *            <code>True</code> if the image should be rendered with a grey
79       *            border, <code>false</code> otherwise.
80       * @return The old setting.
81       */
82      public boolean setBorder(boolean border);
83
84      /**
85       * Get the horizontal alignment of the image.
86       *
87       * Corresponds to the XWML 1.0 attribute "halign".
88       *
89       * @return The horizontal alignment or <code>null</code> if the attribute is
90       *         not specified.
91       */
92      public WomImageHAlign getHAlign();
93
94      /**
95       * Set the horizontal alignment of the image.
96       *
97       * Corresponds to the XWML 1.0 attribute "halign".
98       *
99       * @param halign
100      *            The new setting or <code>null</code> to remove the attribute.
101      * @return The old setting.
102      */
103     public WomImageHAlign setHAlign(WomImageHAlign halign);
104
105     /**
106      * Get the vertical alignment of the image. Only applies to inline,
107      * non-floating images.
108      *
109      * Corresponds to the XWML 1.0 attribute "valign".
110      *
111      * @return The vertical alignment of the image or <code>null</code> if the
112      *         attribute is not specified.
113      */
114     public WomImageVAlign getVAlign();
115
116     /**
117      * Set the vertical alignment of the image. Only applies to inline,
118      * non-floating images.
119      *
120      * Corresponds to the XWML 1.0 attribute "valign".
121      *
122      * @param halign
123      *            The new setting or <code>null</code> to remove the attribute.
124      * @return The old setting.
125      */
126     public WomImageVAlign setVAlign(WomImageVAlign valign);
127
128     /**
129      * Get the width to which the image should be scaled before rendering.
130      *
131      * Corresponds to the XWML 1.0 attribute "width".
132      *
133      * @return The width in pixels or <code>null</code> if the attribute is not
```

```
134         *         specified.
135         */
136        public Integer getWidth();
137
138        /**
139         * Set the width to which the image should be scaled before rendering.
140         *
141         * Corresponds to the XWML 1.0 attribute "width".
142         *
143         * @param width
144         *            The new width in pixels or <code>null</code> to remove the
145         *            attribute.
146         * @return The old width in pixels.
147         */
148        public Integer setWidth(Integer width);
149
150        /**
151         * Get the height to which the image should be scaled before rendering.
152         *
153         * Corresponds to the XWML 1.0 attribute "height".
154         *
155         * @return The height in pixels or <code>null</code> if the attribute is not
156         *         specified.
157         */
158        public Integer getHeight();
159
160        /**
161         * Set the height to which the image should be scaled before rendering.
162         *
163         * Corresponds to the XWML 1.0 attribute "height".
164         *
165         * @param height
166         *            The new height in pixels or <code>null</code> to remove the
167         *            attribute.
168         * @return The old height in pixels.
169         */
170        public Integer setHeight(Integer height);
171
172        /**
173         * Whether the image will be resized according to user preferences.
174         *
175         * Corresponds to the XWML 1.0 attribute "upright".
176         *
177         * @return <code>True</code> if the image will be resized according to user
178         *         preferences, <code>false</code> otherwise.
179         */
180        public boolean isUpright();
181
182        /**
183         * Set whether the image should will be resized according to user
184         * preferences.
185         *
186         * Corresponds to the XWML 1.0 attribute "upright".
187         *
188         * @param upright
189         *            <code>True</code> if the image should be resized according to
190         *            user preferences, <code>false</code> otherwise.
191         *
192         * @return The old setting.
193         */
194        public boolean setUpright(boolean upright);
195
196        /**
197         * Get the optional URL to which the image will link when clicked. The
198         * optional URL and optional page link are mutually exclusive.
199         *
200         * Corresponds to the XWML 1.0 attribute "urllink".
201         *
202         * @return The URL to link to or <code>null</code> if the attribute is not
203         *         specified.
204         */
205        public URL getUrlLink();
206
207        /**
208         * Set the optional URL to which the image will link when clicked. The
209         * optional URL and optional page link are mutually exclusive. If the
```

```
210       * attribute "pagelink" is specified and this method is called to set a
211       * "urllink", the "pagelink" attribute will be removed.
212       *
213       * Corresponds to the XWML 1.0 attribute "urllink".
214       *
215       * @param url
216       *            The new URL to which the image should link or
217       *            <code>null</code> to remove the attribute.
218       * @return The old URL.
219       */
220      public URL setUrlLink(URL url);
221
222      /**
223       * Get the optional page to which the image will link when clicked. The
224       * optional page link and optional URL are mutually exclusive.
225       *
226       * Corresponds to the XWML 1.0 attribute "pagelink".
227       *
228       * @return The page to link to or <code>null</code> if the attribute is not
229       *         specified.
230       */
231      public String getPageLink();
232
233      /**
234       * Set the optional page to which the image will link when clicked. The
235       * optional page and link optional URL are mutually exclusive. If the
236       * attribute "urllink" is specified and this method is called to set a
237       * "pagelink", the "urllink" attribute will be removed.
238       *
239       * Corresponds to the XWML 1.0 attribute "pagelink".
240       *
241       * @param page
242       *            The new page to which the image should link.
243       * @return The old page.
244       */
245      public String setPageLink(String page);
246
247      /**
248       * Get the alternative text of the image.
249       *
250       * Corresponds to the XWML 1.0 attribute "alt".
251       *
252       * @return The alternative text or <code>null</code> if the attribute is not
253       *         specified.
254       */
255      public String getAlt();
256
257      /**
258       * Set the alternative text of the image.
259       *
260       * Corresponds to the XWML 1.0 attribute "alt".
261       *
262       * @param alt
263       *            The new alternative text of the image or <code>null</code> to
264       *            remove the attribute.
265       * @return The old alternative text.
266       */
267      public String setAlt(String alt);
268
269      // ==[ Link interface ]======================================================
270
271      /**
272       * Returns the alternative text of the image. If no alternative text is
273       * given an empty title will be returned.
274       *
275       * @return The title of this image.
276       */
277      @Override
278      public WomTitle getLinkTitle();
279
280      /**
281       * Return the target this image links to. This is the page of the image
282       * itself or another page or url if the <code>pagelink</code> or
283       * <code>urllink</code> attributes are set.
284       *
285       * @return The target this image links to.
```

```
286       */
287      @Override
288      public Object getLinkTarget();
289  }
```

WomImageCaption.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * The caption of a framed image.
5   *
6   * Corresponds to the WXML 1.0 element "imgcaption".
7   *
8   * <b>Child elements:</b> Mixed, [Inline elements]*
9   */
10 public interface WomImageCaption
11         extends
12             WomNode
13 {
14 }
```

WomImageFormat.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * The different formats to render and place Wikitext images.
5   */
6  public enum WomImageFormat
7  {
8      UNRESTRAINED,
9      FRAMELESS,
10     THUMBNAIL,
11     FRAME
12 }
```

WomImageHAlign.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * Horizontal alignment of a Wikitext image.
5   */
6  public enum WomImageHAlign
7  {
8      DEFAULT,
9      NONE,
10     LEFT,
11     CENTER,
12     RIGHT,
13 }
```

WomImageVAlign.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * Vertical alignment of an inline, non-floating Wikitext image.
5   */
6  public enum WomImageVAlign
```

```
{
    BASELINE,
    SUB,
    SUPER,
    TOP,
    TEXT_TOP,
    MIDDLE,
    BOTTOM,
    TEXT_BOTTOM,
}
```

WomInlineElement.java

```
package org.sweble.wikitext.engine.wom;

/**
 * An element that behaves like an XHTML 1.0 Transitional inline element.
 *
 * This is an interface that groups elements that can behave like inline
 * elements.
 */
public interface WomInlineElement
        extends
            WomNode
{
}
```

WomIns.java

```
package org.sweble.wikitext.engine.wom;

import java.util.Date;

/**
 * Denotes text or a block that has been added.
 *
 * Corresponds to the XHTML 1.0 Transitional element "ins".
 *
 * <b>Child elements:</b> Mixed, [Flow elements]*
 */
public interface WomIns
        extends
            WomInlineElement,
            WomBlockElement,
            WomUniversalAttributes
{
    /**
     * Get the url of a document that specifies the reasons for the change.
     *
     * Corresponds to the XHTML 1.0 Transitional attribute "cite".
     *
     * @return The url or <code>null</code> if the attribute is not specified.
     */
    public String getCite();

    /**
     * Set the url of a document that specifies the reasons for the change.
     *
     * Corresponds to the XHTML 1.0 Transitional attribute "cite".
     *
     * @param url
     *            The new url or <code>null</code> to remove the attribute.
     * @return The The old url.
     */
    public String getCite(String url);

    /**
     * Get the timestamp when the text or block was deleted.
     *
```

```
     * Corresponds to the XHTML 1.0 Transitional attribute "cite".
     *
     * @return The date and time of the deletion or <code>null</code> if the
     *         attribute is not specified.
     */
    public Date getDatetime();

    /**
     * Set the timestamp when the text or block was deleted.
     *
     * Corresponds to the XHTML 1.0 Transitional attribute "cite".
     *
     * @param timestamp
     *            The new timestamp or <code>null</code> to remove the
     *            attribute.
     * @return The old timestamp.
     */
    public Date getDatetime(Date timestamp);
}
```

WomIntLink.java

```
package org.sweble.wikitext.engine.wom;

/**
 * Denotes a Wikitext internal link.
 *
 * Corresponds to the WXML 1.0 element "intlink".
 *
 * <b>Child elements:</b> title?
 */
public interface WomIntLink
        extends
            WomInlineElement,
            WomLink
{
    /**
     * Get the title of the internal link.
     *
     * @return The title of the internal link or <code>null</code> if the link
     *         does not specify a title.
     */
    public WomTitle getTitle();

    /**
     * Set the title of the internal link.
     *
     * @param title
     *            The new title of the internal link or <code>null</code> to
     *            remove the title.
     * @return The old link title.
     */
    public WomTitle setTitle(WomTitle title);

    /**
     * Get the target of the internal link.
     *
     * Corresponds to the XWML 1.0 attribute "target".
     *
     * @return The target of the internal link.
     */
    public String getTarget();

    /**
     * Set the target of this internal link.
     *
     * Corresponds to the XWML 1.0 attribute "target".
     *
     * @param target
     *            The new target of the internal link.
     * @return The old target of the internal link.
     * @throws IllegalArgumentException
     *             Thrown if the given target is empty or not a valid page
```

```
52          *             title.
53          * @throws NullPointerException
54          *             Thrown if <code>null</code> is passed as target.
55          */
56         public String setTarget(String target) throws IllegalArgumentException,
                NullPointerException;
57
58         // ==[ Link interface ]=================================================
59
60         /**
61          * Returns the title of the internal link. If the internal link does not
62          * specify a title, the target (which specifies a page title) is returned as
63          * title.
64          *
65          * @return The title of the internal link.
66          */
67         @Override
68         public WomTitle getLinkTitle();
69
70         /**
71          * Retrieve the target of this link.
72          *
73          * @return The target of this link.
74          */
75         @Override
76         public String getLinkTarget();
77 }
```

WomItalics.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * Denotes text that should be rendered as italic text.
5   *
6   * Corresponds to the XHTML 1.0 Transitional element "i".
7   *
8   * <b>Child elements:</b> Mixed, [Inline elements]*
9   */
10 public interface WomItalics
11         extends
12             WomInlineElement,
13             WomUniversalAttributes
14 {
15 }
```

WomKbd.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * Denotes a key on the keyboard.
5   *
6   * Corresponds to the XHTML 1.0 Transitional element "kbd".
7   *
8   * <b>Child elements:</b> Mixed, [Inline elements]*
9   */
10 public interface WomKbd
11         extends
12             WomInlineElement,
13             WomUniversalAttributes
14 {
15 }
```

WomLink.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * This interface groups elements of Wikitext that link to different pages/urls.
5   */
6  public interface WomLink
7  {
8      /**
9       * Returns the title of this link.
10      *
11      * @return The title node or <code>null</code> if the link does not support
12      *         a title or does not specify a title.
13      */
14     public WomTitle getLinkTitle();
15
16     /**
17      * Retrieve the target of this link.
18      *
19      * @return The target of this link.
20      */
21     public Object getLinkTarget();
22 }
```

WomList.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  import java.util.Collection;
4
5  /**
6   * Interface to lists like <code>&lt;ul></code> or <code>&lt;ol></code>.
7   *
8   * List items are accessed via an integer index where <code>0</code> denotes the
9   * first list item. <b>Only valid list items are counted.</b> If a list is given
10  * in HTML that contains invalid content (e.g.: text or elements other than
11  * <code>&lt;li></code>), these elements are skipped in the enumeration and are
12  * not accessible through this interface. However, they can be iterated using
13  * the methods provided by the WomNode interface.
14  *
15  * Corresponds to the XHTML 1.0 Transitional element "ul" or "ol".
16  *
17  * <b>Child elements:</b> ([Preprocessor elements]|li)*
18  */
19 public interface WomList
20         extends
21             WomBlockElement,
22             WomUniversalAttributes
23 {
24     /**
25      * Get the number of items in the list.
26      *
27      * @return The number of items in the list.
28      */
29     public int getItemNum();
30
31     /**
32      * Get a collection containing all items.
33      *
34      * @return A collection with all items of the list.
35      */
36     public Collection<WomListItem> getItems();
37
38     /**
39      * Get a certain item from the list.
40      *
41      * @param index
42      *            The zero-based index of the item to retrieve.
43      * @return The item with the given index.
44      * @throws IndexOutOfBoundsException
45      *             If the given index is out of range.
46      */
```

```
47      public WomListItem getItem(int index) throws IndexOutOfBoundsException;
48
49      /**
50       * Replace a certain item in the list.
51       *
52       * @param index
53       *            The zero-based index of the item to replace.
54       * @param item
55       *            The replacement item.
56       * @return The old item with the given index.
57       * @throws IndexOutOfBoundsException
58       *             If the given index is out of range.
59       */
60      public WomListItem replaceItem(int index, WomListItem item) throws
             IndexOutOfBoundsException;
61
62      /**
63       * Remove an item from the list.
64       *
65       * @param index
66       *            The zero-based index of the item to remove.
67       *
68       * @return The removed item.
69       * @throws IndexOutOfBoundsException
70       *             If the given index is out of range.
71       */
72      public WomListItem removeItem(int index) throws IndexOutOfBoundsException;
73
74      /**
75       * Append an item to the list.
76       *
77       * @param item
78       *            The item to append.
79       */
80      public void appendItem(WomListItem item);
81
82      /**
83       * Insert an item at the given index into the list.
84       *
85       * @param beforeIndex
86       *            The index of the item in front of which the new item is to be
87       *            inserted.
88       * @param item
89       *            The item to insert. The item will have the given index
90       *            <code>beforeIndex</code> after insertion.
91       * @throws IndexOutOfBoundsException
92       *             Thrown if <code>0 <= beforeIndex <= getItemNum()</code> does
93       *             not hold.
94       */
95      public void insertItem(int beforeIndex, WomListItem item) throws
             IndexOutOfBoundsException;
96
97      // ==[ The XHTML Attributes ]==========================================
98
99      /**
100      * Tells whether the list should be displayed as compact list.
101      *
102      * Corresponds to the XHTML 1.0 Transitional attribute "compact".
103      *
104      * @return Whether the "compact" flag is given or not.
105      */
106     public boolean isCompact();
107
108     /**
109      * Set whether the list should be displayed as compact list.
110      *
111      * Corresponds to the XHTML 1.0 Transitional attribute "compact".
112      *
113      * @param compact
114      *            Set (<code>true</code>) or remove (<code>false</code>) the
115      *            "compact" flag.
116      * @return The old state.
117      */
118     public boolean setCompact(boolean compact);
119
120     /**
```

```
121         * Get the number of the first item in the list.
122         *
123         * Corresponds to the XHTML 1.0 Transitional attribute "start".
124         *
125         * @return The number of the first item or <code>null</code> if the
126         *         attribute is not specified.
127         */
128        public Integer getStart();
129
130        /**
131         * Set the number of the first list item.
132         *
133         * Corresponds to the XHTML 1.0 Transitional attribute "start".
134         *
135         * @param start
136         *            The new number of the first list item or <code>null</code> to
137         *            remove the attribute.
138         * @return The old number of the first list item.
139         */
140        public Integer setStart(Integer start);
141    }
```

WomListItem.java

```
1   package org.sweble.wikitext.engine.wom;
2
3   /**
4    * Denotes a list item.
5    *
6    * Corresponds to the XHTML 1.0 Transitional element "li".
7    *
8    * <b>Child elements:</b> [Block elements]*
9    */
10  public interface WomListItem
11          extends
12              WomNode,
13              WomUniversalAttributes
14  {
15      /**
16       * Get the type of bullet point the list item uses.
17       *
18       * Corresponds to the XHTML 1.0 Transitional attribute "type".
19       *
20       * @return The type of bullet point or <code>null</code> if the attribute is
21       *         not specified.
22       */
23      public String getItemType();
24
25      /**
26       * Get the type of bullet point the list item uses.
27       *
28       * Corresponds to the XHTML 1.0 Transitional attribute "type".
29       *
30       * @param type
31       *            The new type of bullet point or <code>null</code> to remove
32       *            the attribute.
33       * @return The old type of bullet point.
34       */
35      public String setItemType(String type);
36
37      /**
38       * Get the number of the list item.
39       *
40       * Corresponds to the XHTML 1.0 Transitional attribute "value".
41       *
42       * @return The number of the list item or <code>null</code> if the attribute
43       *         is not specified.
44       */
45      public Integer getItemValue();
46
47      /**
48       * Set the number of the list item.
49       *
```

```
     * Corresponds to the XHTML 1.0 Transitional attribute "value".
     *
     * @param number
     *            The new number of the list item or <code>null</code> to remove
     *            the attribute.
     * @return The old number of the list item.
     */
    public Integer getItemValue(Integer number);
}
```

WomMagicWord.java

```
package org.sweble.wikitext.engine.wom;

/**
 * Denotes a magic word.
 *
 * Corresponds to the WXML 1.0 element "magicword".
 *
 * <b>Child elements:</b> -
 */
public interface WomMagicWord
        extends
            WomProcessingInstruction
{
    /**
     * Get the name of the magic word.
     *
     * Corresponds to the XWML 1.0 attribute "name".
     *
     * @return The name of the magic word.
     */
    public String getName();

    /**
     * Set the name of the magic word.
     *
     * Corresponds to the XWML 1.0 attribute "name".
     *
     * @param name
     *            The new name of the magic word.
     * @return The old name of the magic word.
     * @throws IllegalArgumentException
     *             Thrown if name is not a valid name for a magic word or if
     *             <code>null</code> is passed as name.
     * @throws NullPointerException
     *             Thrown if <code>null</code> is passed as name.
     */
    public String setName(String name) throws IllegalArgumentException,
         NullPointerException;
}
```

WomName.java

```
package org.sweble.wikitext.engine.wom;

/**
 * The name of a transclusion, transclusion argument or template parameter.
 *
 * Corresponds to the WXML 1.0 element "name".
 *
 * <b>Child elements:</b> Mixed, [Preprocessor elements]*
 */
public interface WomName
        extends
            WomNode
{
}
```

WomNode.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  import java.io.Serializable;
4  import java.util.Collection;
5
6  /**
7   * The parent interface of every node in the Wikitext Object Model.
8   */
9  public interface WomNode
10         extends
11             Cloneable,
12             Serializable
13 {
14     // ==[ Reflection ]=========================================================
15
16     /**
17      * Returns the name of a node.
18      */
19     public String getNodeName();
20
21     /**
22      * Returns the type of a node.
23      */
24     public WomNodeType getNodeType();
25
26     // ==[ Textual content ]====================================================
27
28     /**
29      * Return the text content of a node. Returns <code>null</node> for other
30      * types of nodes. An empty text node will return the empty string and not
31      * <code>null</code>!
32      */
33     public String getText();
34
35     /**
36      * Return the value of a text node and other value carrying nodes. Returns
37      * <code>null</node> for other types of nodes. Attributes with an empty
38      * value or empty text nodes will return the empty string and not
39      * <code>null</code>!
40      */
41     public String getValue();
42
43     // ==[ Text manipulation ]==================================================
44
45     /**
46      * Append text to the text of this node.
47      *
48      * @throws UnsupportedOperationException
49      *             If this node is not a text node.
50      */
51     public void appendText(String text) throws UnsupportedOperationException;
52
53     /**
54      * Delete a range of the text of this node.
55      *
56      * @param from
57      *            The first character that will be deleted.
58      * @param length
59      *            The number of characters that will be deleted.
60      *
61      * @return The deleted text.
62      *
63      * @throws UnsupportedOperationException
64      *             If this node is not a text node.
65      * @throws IndexOutOfBoundsException
66      *             If the given range is invalid.
67      */
68     public String deleteText(int from, int length) throws UnsupportedOperationException,
           IndexOutOfBoundsException;
69
70     /**
71      * Insert text at a specified position.
72      *
73      * @throws UnsupportedOperationException
```

```
74        *             If this node is not a text node.
75        * @throws IndexOutOfBoundsException
76        *             If the given range is invalid.
77        */
78       public void insertText(int at, String text) throws UnsupportedOperationException,
              IndexOutOfBoundsException;
79
80       /**
81        * Replaces the text of this node with another text.
82        *
83        * @return The replaced text.
84        *
85        * @throws UnsupportedOperationException
86        *             If this node is not a text node.
87        */
88       public String replaceText(String text) throws UnsupportedOperationException;
89
90       /**
91        * Replaces a specified range of the text of this node with another text.
92        *
93        * @param from
94        *            The first character that will be replaced.
95        * @param length
96        *            The number of characters that will be replaced.
97        * @param text
98        *            The new text that will replace the given range of characters.
99        *
100       * @return The replaced text.
101       *
102       * @throws UnsupportedOperationException
103       *             If this node is not a text node.
104       * @throws IndexOutOfBoundsException
105       *             If the given range is invalid.
106       */
107      public String replaceText(int from, int length, String text) throws
             UnsupportedOperationException, IndexOutOfBoundsException;
108
109      // ==[ Attributes ]=========================================================
110
111      /**
112       * Returns whether this node supports attributes.
113       */
114      public boolean supportsAttributes();
115
116      /**
117       * Return a collection containing the XML attributes of a node. Nodes that
118       * don't support attributes will return an empty collection.
119       */
120      public Collection<WomAttribute> getAttributes();
121
122      /**
123       * Return the value of an attribute node. If no attribute with the given
124       * name exists <code>null</code> is returned. Nodes that don't support
125       * attributes will return <code>null</code>.
126       */
127      public String getAttribute(String name);
128
129      /**
130       * Return an attribute. If no attribute with the given name exists
131       * <code>null</code> is returned. Nodes that don't support attributes will
132       * return <code>null</code>.
133       */
134      public WomAttribute getAttributeNode(String name);
135
136      // ==[ Attribute modification ]=============================================
137
138      /**
139       * Remove an attribute.
140       *
141       * @return The removed attribute node or <code>null</code> if no such
142       *         attribute exists.
143       *
144       * @throws UnsupportedOperationException
145       *             If the node does not support attributes.
146       */
```

```
147     public WomAttribute removeAttribute(String name) throws
            UnsupportedOperationException;
148
149     /**
150      * Remove an attribute.
151      *
152      * @throws UnsupportedOperationException
153      *             If the node does not support attributes.
154      * @throws IllegalArgumentException
155      *             If the given node is not an attribute of this node.
156      */
157     public void removeAttributeNode(WomAttribute attr) throws
            UnsupportedOperationException, IllegalArgumentException;
158
159     /**
160      * Sets an attribute node. If the attribute already exists, it will be
161      * replaced by the given attribute. Otherwise, a new attribute will be
162      * created.
163      *
164      * @param value
165      *            Passing <code>null</code> as value will remove the attribute.
166      *
167      * @return The old attribute or <code>null</code> if the attribute did not
168      *         exist.
169      *
170      * @throws UnsupportedOperationException
171      *             If the node does not support attributes.
172      */
173     public WomAttribute setAttribute(String name, String value) throws
            UnsupportedOperationException;
174
175     /**
176      * Sets an attribute node. If the attribute already exists, it will be
177      * replaced by the given attribute.
178      *
179      * @return The old attribute or <code>null</code> if the attribute did not
180      *         exist.
181      *
182      * @throws UnsupportedOperationException
183      *             If the node does not support attributes.
184      */
185     public WomAttribute setAttributeNode(WomAttribute attr) throws
            UnsupportedOperationException;
186
187     // ==[ Navigation ]=========================================================
188
189     /**
190      * Return the parent node of this node.
191      */
192     public WomNode getParent();
193
194     /**
195      * Return whether this node has any children.
196      */
197     public boolean hasChildNodes();
198
199     /**
200      * Returns a collection containing the children of this node.
201      */
202     public Collection<WomNode> childNodes();
203
204     /**
205      * Returns the first child of this node or <code>null</code> if this node
206      * has no children.
207      */
208     public WomNode getFirstChild();
209
210     /**
211      * Returns the last child of this node or <code>null</code> if this node has
212      * no children.
213      */
214     public WomNode getLastChild();
215
216     /**
217      * Return the next node on the same level as this node. Returns
218      * <code>null</code> if there is no next sibling.
```

```
219         */
220     public WomNode getNextSibling();
221 
222     /**
223      * Return the previous node on the same level as this node. Returns
224      * <code>null</code> if there is no previous sibling.
225      */
226     public WomNode getPrevSibling();
227 
228     // ==[ Tree modification ]==========================================
229 
230     /**
231      * Adds a node to the end of the list of children of this node.
232      *
233      * @throws UnsupportedOperationException
234      *             If the node does not support children.
235      */
236     public void appendChild(WomNode child) throws UnsupportedOperationException;
237 
238     /**
239      * Insert a node into the list of children of this node. The node will be
240      * inserted before another given child node.
241      *
242      * @throws UnsupportedOperationException
243      *             If the node does not support children.
244      * @throws IllegalArgumentException
245      *             If the <code>before</code> node is not a child of this node.
246      */
247     public void insertBefore(WomNode before, WomNode child) throws
            UnsupportedOperationException, IllegalArgumentException;
248 
249     /**
250      * Remove the given child node from the list of children.
251      *
252      * @throws UnsupportedOperationException
253      *             If the node does not support children.
254      * @throws IllegalArgumentException
255      *             If the <code>child</code> node is not a child of this node.
256      */
257     public void removeChild(WomNode child) throws UnsupportedOperationException,
            IllegalArgumentException;
258 
259     /**
260      * Replace a given child node with another node.
261      *
262      * @return Returns <code>true</code> if the child was replaced. If the given
263      *         node is not a child of this node, <code>false</code> is returned.
264      *
265      * @throws UnsupportedOperationException
266      *             If the node does not support children.
267      * @throws IllegalArgumentException
268      *             If the <code>search</code> node is not a child of this node.
269      */
270     public void replaceChild(WomNode search, WomNode replace) throws
            UnsupportedOperationException, IllegalArgumentException;
271 
272     // ==[ Cloning ]===================================================
273 
274     /**
275      * Create a copy of this node. The created node will not be part of the WOM
276      * tree which the original node belonged to (or any other tree, until it is
277      * added to some other tree).
278      */
279     public WomNode cloneNode();
280 }
```

WomNodeType.java

```
1 package org.sweble.wikitext.engine.wom;
2 
3 /**
4  * A rough categorization of the different kinds of node types found in a WOM
5  * tree.
```

```
 */
public enum WomNodeType
{
    DOCUMENT,
    ELEMENT,
    ATTRIBUTE,
    TEXT,
    COMMENT,
}
```

WomNowiki.java

```
package org.sweble.wikitext.engine.wom;

/**
 * Wraps text that must not be interpreted.
 *
 * Corresponds to the WXML 1.0 element "nowiki".
 *
 * <b>Child elements:</b> Text
 */
public interface WomNowiki
        extends
            WomInlineElement
{
    /**
     * Return the text inside the nowiki element.
     *
     * @return The text inside the nowiki element.
     */
    @Override
    public String getValue();

    /**
     * Set the text inside the nowiki element.
     *
     * @param text
     *             The new text.
     * @return The old text.
     * @throws NullPointerException
     *              Thrown if <code>null</code> is passed as text.
     * @throws IllegalArgumentException
     *              Thrown if the given text contains "&lt;/nowiki>".
     */
    public String setValue(String text) throws IllegalArgumentException,
         NullPointerException;
}
```

WomOrderedList.java

```
package org.sweble.wikitext.engine.wom;

/**
 * Denotes an ordered list.
 *
 * Corresponds to the XHTML 1.0 Transitional element "ol".
 *
 * See WomList for details.
 */
public interface WomOrderedList
        extends
            WomList
{
    /**
     * Get the type of bullet point the list items use.
     *
     * Corresponds to the XHTML 1.0 Transitional attribute "type".
     *
     * @return The type of bullet point or <code>null</code> if the attribute is
```

```
20          *         not specified.
21          */
22        public String getItemType();
23
24        /**
25          * Set the type of bullet point the list items should use.
26          *
27          * Corresponds to the XHTML 1.0 Transitional attribute "type".
28          *
29          * @param type
30          *            The new type of bullet point or <code>null</code> to remove
31          *            the attribute.
32          * @return The old type of bullet point.
33          */
34        public String setItemType(String type);
35    }
```

WomPage.java

```
1   package org.sweble.wikitext.engine.wom;
2
3   /**
4    * The root node of every page.
5    *
6    * Corresponds to the WXML 1.0 element "page".
7    *
8    * <b>Child elements:</b> redirect? body
9    */
10  public interface WomPage
11          extends
12              WomNode
13  {
14      /**
15       * Returns the full name of an page including namespace and path.
16       *
17       * @return The namespace, path and page name concatenated.
18       */
19      public String getName();
20
21      /**
22       * Returns the version of the XWML object model.
23       *
24       * @return The version of the XWML object model.
25       */
26      public String getVersion();
27
28      /**
29       * Returns the name of the page without namespace and without path.
30       *
31       * Corresponds to the XWML 1.0 attribute "title".
32       *
33       * @return The name of the page without namespace and without path.
34       */
35      public String getTitle();
36
37      /**
38       * Set the name of the page without namespace and without path.
39       *
40       * Corresponds to the XWML 1.0 attribute "title".
41       *
42       * @param title
43       *            The new title of the page.
44       * @return The old title of the page.
45       * @throws IllegalArgumentException
46       *             Thrown if the given title is empty or not a valid MediaWiki
47       *             page title.
48       * @throws NullPointerException
49       *             Thrown if the given title is <code>null</code>.
50       */
51      public String setTitle(String title) throws IllegalArgumentException,
            NullPointerException;
52
53      /**
```

```
54        * Returns the canonical namespace name.
55        *
56        * Corresponds to the XWML 1.0 attribute "namespace".
57        *
58        * @return The canonoical namespace name or <code>null</code> if the
59        *         attribute is not specified.
60        */
61       public String getNamespace();
62
63       /**
64        * Set the canonical namespace name.
65        *
66        * Corresponds to the XWML 1.0 attribute "namespace".
67        *
68        * @param namespace
69        *            The new namespace name or <code>null</code> to remove the
70        *            attribute.
71        * @return The old namespace name.
72        */
73       public String setNamespace(String namespace);
74
75       /**
76        * Returns the path of pages that lead to this subpage.
77        *
78        * Corresponds to the XWML 1.0 attribute "path".
79        *
80        * @return The path of pages that lead to this subpage or <code>null</code>
81        *         if this attribute is not given and this page, therefore, is not a
82        *         subpage.
83        */
84       public String getPath();
85
86       /**
87        * Set the path of pages that lead to this subpage.
88        *
89        * @param path
90        *            The new path or <code>null</code> to remove the attribute.
91        * @return The old path.
92        */
93       public String setPath(String path);
94
95       /**
96        * Tell whether this page is a redirecting page.
97        *
98        * @return <code>True</code> if this page redirects to another page,
99        *         <code>false</code> otherwise.
100       */
101      public boolean isRedirect();
102
103      /**
104       * Get the redirection statement.
105       *
106       * Operates on the first &lt;redirect> element found among this node's
107       * children.
108       *
109       * @return The redirection statement or <code>null</code> if this page does
110       *         not redirect.
111       */
112      public WomRedirect getRedirect();
113
114      /**
115       * Set a redirection.
116       *
117       * Operates on the first &lt;redirect> element found among this node's
118       * children. If no redirect node is found, the redirect will be added as the
119       * first child.
120       *
121       * @param redirect
122       *            The new redirection to set or <code>null</code> to remove a
123       *            redirection.
124       * @return The old redirection.
125       */
126      public WomRedirect setRedirect(WomRedirect redirect);
127
128      /**
129       * Get the page body.
```

```
130       *
131       * Operates on the first &lt;body> element found among this node's children.
132       *
133       * @return The body.
134       */
135      public WomBody getBody();
136
137      /**
138       * Set the page body.
139       *
140       * Operates on the first &lt;body> element found among this node's children.
141       *
142       * @param body
143       *             The new body.
144       * @return The old body.
145       * @throws NullPointerException
146       *              Thrown if <code>null</code> is given as body.
147       */
148      public WomBody setBody(WomBody body) throws NullPointerException;
149  }
```

WomParagraph.java

```
1   package org.sweble.wikitext.engine.wom;
2
3   /**
4    * Denotes a paragraph.
5    *
6    * Corresponds to the XHTML 1.0 Transitional element "p".
7    *
8    * <b>Child elements:</b> Mixed, [Inline elements]*
9    */
10  public interface WomParagraph
11          extends
12              WomBlockElement,
13              WomUniversalAttributes
14  {
15      /**
16       * Get the alignment of the content inside the tag.
17       *
18       * Corresponds to the XHTML 1.0 Transitional attribute "align".
19       *
20       * @return The alignment.
21       */
22      public WomHorizAlign getAlign();
23
24      /**
25       * Set the alignment of the content inside the tag.
26       *
27       * Corresponds to the XHTML 1.0 Transitional attribute "align".
28       *
29       * @param align
30       *             The new alignment.
31       * @return The old alignment.
32       */
33      public WomHorizAlign setAlign(WomHorizAlign align);
34  }
```

WomParam.java

```
1   package org.sweble.wikitext.engine.wom;
2
3   /**
4    * A template substitution parameter.
5    *
6    * Corresponds to the XWML 1.0 element "param".
7    *
8    * <b>Child elements:</b> name default?
9    */
```

```
10 public interface WomParam
11         extends
12             WomProcessingInstruction
13 {
14     /**
15      * Get the name of the parameter.
16      *
17      * Operates on the first &lt;name> element found among this node's children.
18      *
19      * @return The name of the parameter.
20      */
21     public WomName getName();
22
23     /**
24      * Set the name of the parameter.
25      *
26      * Operates on the first &lt;name> element found among this node's children.
27      *
28      * @param name
29      *             The new name of the parameter.
30      * @return The old name of the parameter.
31      * @throws NullPointerException
32      *              Thrown if <code>null</code> is given as name.
33      */
34     public WomName setName(WomName name) throws NullPointerException;
35
36     /**
37      * Get the default value of the parameter.
38      *
39      * Operates on the first &lt;default> element found among this node's
40      * children.
41      *
42      * @return The default value of the parameter or <code>null</code> if no
43      *         default value is specified.
44      */
45     public WomValue getDefault();
46
47     /**
48      * Set the default value of the parameter.
49      *
50      * Operates on the first &lt;default> element found among this node's
51      * children.
52      *
53      * @param value
54      *             The new default value of the parameter or <code>null</code> to
55      *             remove the default value.
56      * @return The old default value of the parameter.
57      */
58     public WomValue setDefault(WomValue value);
59 }
```

WomPre.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * Denotes a block of preformatted text.
5   *
6   * Corresponds to the XWML 1.0 element "pre".
7   *
8   * <b>Child elements:</b> Text
9   */
10 public interface WomPre
11         extends
12             WomBlockElement,
13             WomUniversalAttributes
14 {
15     /**
16      * Return the text inside the pre element.
17      *
18      * @return The text inside the pre element.
19      */
20     @Override
```

```
21      public String getValue();
22
23      /**
24       * Set the text inside the pre element.
25       *
26       * @param text
27       *            The new text.
28       * @return The old text.
29       * @throws NullPointerException
30       *             Thrown if <code>null</code> is passed as text.
31       * @throws IllegalArgumentException
32       *             Thrown if the given text contains "&lt;/pre>".
33       */
34      public String setValue(String text) throws IllegalArgumentException,
            NullPointerException;
35
36      /**
37       * Get the number of characters per line.
38       *
39       * Corresponds to the XHTML 1.0 Transitional attribute "width".
40       *
41       * @return The number of characters per line or <code>null</code> if the
42       *         attribute is not specified.
43       */
44      public Integer getWidth();
45
46      /**
47       * Set the number of characters per line.
48       *
49       * Corresponds to the XHTML 1.0 Transitional attribute "width".
50       *
51       * @param width
52       *            The new number of characters per line or <code>null</code> to
53       *            remove the attribute.
54       * @return The old number of characters per line.
55       */
56      public Integer setWidth(Integer width);
57  }
```

WomProcessingInstruction.java

```
1   package org.sweble.wikitext.engine.wom;
2
3   /**
4    * All Wikitext processing instructions like, for example, magic words inherit
5    * from this interface.
6    */
7   public interface WomProcessingInstruction
8           extends
9               WomNode
10  {
11  }
```

WomRedirect.java

```
1   package org.sweble.wikitext.engine.wom;
2
3   /**
4    * A redirection statement.
5    *
6    * Corresponds to the XWML 1.0 element "redirect".
7    *
8    * <b>Child elements:</b> -
9    */
10  public interface WomRedirect
11          extends
12              WomProcessingInstruction,
13              WomLink
14  {
```

```
15      /**
16       * Return the target page of the redirection.
17       *
18       * Corresponds to the XWML 1.0 attribute "target".
19       *
20       * @return The target page to redirect to.
21       */
22      public String getTarget();
23
24      /**
25       * Set the target page of the redirection.
26       *
27       * Corresponds to the XWML 1.0 attribute "target".
28       *
29       * @param page
30       *            The new target of the redirection.
31       * @return The old target of the redirection.
32       */
33      public String setTarget(String page);
34
35      // ==[ Link interface ]=====================================================
36
37      /**
38       * Returns an empty title since redirection statements do not provide a
39       * title.
40       *
41       * @return An empty title.
42       */
43      @Override
44      public WomTitle getLinkTitle();
45
46      /**
47       * Return the target page of the redirection.
48       *
49       * @return The target page to redirect to.
50       */
51      @Override
52      public String getLinkTarget();
53  }
```

WomSamp.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * Denotes a text as example.
5   *
6   * Corresponds to the XHTML 1.0 Transitional element "samp".
7   *
8   * <b>Child elements:</b> Mixed, [Inline elements]*
9   */
10 public interface WomSamp
11         extends
12             WomInlineElement,
13             WomUniversalAttributes
14 {
15 }
```

WomSection.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * A section in Wikitext.
5   *
6   * Corresponds to the XWML 1.0 element "section".
7   *
8   * <b>Child elements:</b> heading body
9   */
```

```
10 public interface WomSection
11         extends
12             WomBlockElement
13 {
14     /**
15      * Get the level of the section. Ranges from 1 (most important) to 6 (least
16      * important).
17      *
18      * Corresponds to the XWML 1.0 attribute "level".
19      *
20      * @return The level of the section.
21      */
22     public int getLevel();
23
24     /**
25      * Set the level of the section. Ranges from 1 (most important) to 6 (least
26      * important). A section with level <code>x</code> cannot be contained in a
27      * section with level <code>y</code> or any of its children if
28      * <code>x <= y</code>.
29      *
30      * Corresponds to the XWML 1.0 attribute "level".
31      *
32      * @param level
33      *            The new level of the section.
34      * @return The old level of the section.
35      * @throws IllegalArgumentException
36      *             Thrown if this section is contained in a section with level
37      *             <code>y</code> or any of its children and
38      *             <code>level <= y</code> or if the given level does not lie in
39      *             the range [1,6].
40      */
41     public int setLevel(int level) throws IllegalArgumentException;
42
43     /**
44      * Return the heading of this section.
45      *
46      * Operates on the first &lt;heading> element found among this node's
47      * children.
48      *
49      * @return This heading of this section.
50      */
51     public WomHeading getHeading();
52
53     /**
54      * Set the heading of this section.
55      *
56      * Operates on the first &lt;heading> element found among this node's
57      * children.
58      *
59      * @param heading
60      *            The new heading.
61      * @return The old heading.
62      * @throws NullPointerException
63      *             Thrown if <code>null</code> is given as heading.
64      */
65     public WomHeading setHeading(WomHeading heading) throws NullPointerException;
66
67     /**
68      * Return the body of this section.
69      *
70      * Operates on the first &lt;body> element found among this node's children.
71      *
72      * @return The body of this section.
73      */
74     public WomBody getBody();
75
76     /**
77      * Set the body of this section.
78      *
79      * Operates on the first &lt;body> element found among this node's children.
80      *
81      * @param body
82      *            The new body.
83      * @return The old body.
84      * @throws NullPointerException
85      *             Thrown if <code>null</code> is given as body.
```

```
86      */
87     public WomBody setBody(WomBody body) throws NullPointerException;
88 }
```

WomSemiPre.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * Denotes a preformatted block of text that can contain other formatting
5   * markup.
6   *
7   * Corresponds to the XWML 1.0 element "semipre".
8   *
9   * <b>Child elements:</b> Mixed, ([Inline elements] \ {image, big, small, sub,
10  * sup, font})*
11  */
12 public interface WomSemiPre
13         extends
14             WomBlockElement
15 {
16 }
```

WomSignature.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  import java.util.Date;
4
5  /**
6   * A Wikitext signature.
7   *
8   * Corresponds to the XWML 1.0 element "signature".
9   *
10  * <b>Child elements:</b> -
11  */
12 public interface WomSignature
13         extends
14             WomInlineElement
15 {
16     /**
17      * Get the signature format that describes how the signature should be
18      * rendered.
19      *
20      * Corresponds to the XWML 1.0 attribute "format".
21      *
22      * @return The signature format.
23      */
24     public WomSignatureFormat getSignatureFormat();
25
26     /**
27      * Set the signature format that describes how the signature should be
28      * rendered.
29      *
30      * Corresponds to the XWML 1.0 attribute "format".
31      *
32      * @param format
33      *             The new signature format.
34      * @return The old signature format.
35      * @throws NullPointerException
36      *              Thrown if <code>null</code> is given as format.
37      */
38     public WomSignatureFormat setSignatureFormat(WomSignatureFormat format) throws
            NullPointerException;
39
40     /**
41      * Get the name of the author.
42      *
43      * Corresponds to the XWML 1.0 attribute "author".
```

```
44      *
45      * @return The author name.
46      */
47     public String getAuthor();
48
49     /**
50      * Set the author name.
51      *
52      * Corresponds to the XWML 1.0 attribute "author".
53      *
54      * @param author
55      *            The new name of the author.
56      * @return The old author name.
57      * @throws IllegalArgumentException
58      *             Thrown if the given author name is not a valid MediaWiki user
59      *             name.
60      * @throws NullPointerException
61      *             Thrown if <code>null</code> is given as format.
62      */
63     public String setAuthor(String author) throws IllegalArgumentException,
            NullPointerException;
64
65     /**
66      * Get the date and time of the signature.
67      *
68      * Corresponds to the XWML 1.0 attribute "timestamp".
69      *
70      * @return The date and time of the signature.
71      */
72     public Date getTimestamp();
73
74     /**
75      * Set the date and time of the signature.
76      *
77      * Corresponds to the XWML 1.0 attribute "timestamp".
78      *
79      * @param timestamp
80      *            The new date and time of the signature.
81      * @return The old date and time of the signature.
82      * @throws NullPointerException
83      *             Thrown if <code>null</code> is given as format.
84      */
85     public Date setTimestamp(Date timestamp) throws NullPointerException;
86 }
```

WomSignatureFormat.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * Defines how a signature will be rendered.
5   */
6  public enum WomSignatureFormat
7  {
8      USERNAME,
9      TIMESTAMP,
10     USERNAME_AND_TIMESTAMP
11 }
```

WomSmall.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * Denotes text that should be displayed in a smaller font.
5   *
6   * Corresponds to the XHTML 1.0 Transitional element "small".
7   *
8   * <b>Child elements:</b> Mixed, [Inline elements]*
```

```
 */
public interface WomSmall
        extends
            WomInlineElement,
            WomUniversalAttributes
{
}
```

WomSpan.java

```
package org.sweble.wikitext.engine.wom;

/**
 * Denotes a general inline span.
 *
 * Corresponds to the XHTML 1.0 Transitional element "span".
 *
 * <b>Child elements:</b> Mixed, [Inline elements]*
 */
public interface WomSpan
        extends
            WomInlineElement,
            WomUniversalAttributes
{
}
```

WomStrike.java

```
package org.sweble.wikitext.engine.wom;

/**
 * Denotes text that should be displayed as strikethrough text.
 *
 * Corresponds to the XHTML 1.0 Transitional element "strike" and "s".
 *
 * <b>Child elements:</b> Mixed, [Inline elements]*
 */
public interface WomStrike
        extends
            WomInlineElement,
            WomUniversalAttributes
{
}
```

WomStrong.java

```
package org.sweble.wikitext.engine.wom;

/**
 * Denotes text that should be rendered as strong (highlighted) text.
 *
 * Corresponds to the XHTML 1.0 Transitional element "strong".
 *
 * <b>Child elements:</b> Mixed, [Inline elements]*
 */
public interface WomStrong
        extends
            WomInlineElement,
            WomUniversalAttributes
{
}
```

WomSub.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * Denotes text that should be rendered as subscript text.
5   *
6   * Corresponds to the XHTML 1.0 Transitional element "sub".
7   *
8   * <b>Child elements:</b> Mixed, [Inline elements]*
9   */
10 public interface WomSub
11         extends
12             WomInlineElement,
13             WomUniversalAttributes
14 {
15 }
```

WomSup.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * Denotes text that should be rendered as superscript text.
5   *
6   * Corresponds to the XHTML 1.0 Transitional element "sup".
7   *
8   * <b>Child elements:</b> Mixed, [Inline elements]*
9   */
10 public interface WomSup
11         extends
12             WomInlineElement,
13             WomUniversalAttributes
14 {
15 }
```

WomTable.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * Denotes a table.
5   *
6   * Corresponds to the XHTML 1.0 Transitional element "table".
7   *
8   * <b>Child elements:</b> ([Preprocessor elements]|caption)? ([Preprocessor
9   * elements]|tbody)?
10  */
11 public interface WomTable
12         extends
13             WomBlockElement,
14             WomUniversalAttributes
15 {
16     /**
17      * Get the caption of the table.
18      *
19      * Operates on the first &lt;caption> element found among the table's
20      * children.
21      *
22      * @return The caption of the table.
23      */
24     public WomTableCaption getCaption();
25
26     /**
27      * Set the caption of the table.
28      *
29      * Operates on the first &lt;caption> element found among the table's
30      * children. If no caption is found, the caption will be added as the first
31      * child of the table.
```

```
32        *
33        * @param caption
34        *            The new caption of the table.
35        * @return The old caption of the table.
36        */
37       public WomTableCaption setCaption(WomTableCaption caption);
38
39       /**
40        * Get the body of the table.
41        *
42        * Operates on the first &lt;tbody> element found among the table's
43        * children.
44        *
45        * @return The body of the table.
46        */
47       public WomTableBody getBody();
48
49       /**
50        * Set the body of the table.
51        *
52        * Operates on the first &lt;tbody> element found among the table's
53        * children. If no body is found, the body will be added as the first child
54        * of the table.
55        *
56        * @param body
57        *            The new body of the table.
58        * @return The old body of the table.
59        */
60       public WomTableBody setBody(WomTableBody body);
61
62       // ==[ The XHTML Attributes ]==========================================
63
64       /**
65        * Get the alignment of the table.
66        *
67        * Corresponds to the XHTML 1.0 Transitional attribute "align".
68        *
69        * @return The alignment of the table or <code>null</code> if the attribute
70        *         is not specified.
71        */
72       public WomHorizAlign getAlign();
73
74       /**
75        * Set the alignment of the table.
76        *
77        * Corresponds to the XHTML 1.0 Transitional attribute "align".
78        *
79        * @param align
80        *            The alignment or <code>null</code> to remove the attribute.
81        *            Only the values <code>left</code>, <code>center</code> and
82        *            <code>right</code> are allowed.
83        * @return The old alignment of the table.
84        */
85       public WomHorizAlign setAlign(WomHorizAlign align);
86
87       /**
88        * Get the thickness of the table border.
89        *
90        * Corresponds to the XHTML 1.0 Transitional attribute "border".
91        *
92        * @return The thickness of the table border in pixels or <code>null</code>
93        *         if the attribute is not given.
94        */
95       public Integer getBorder();
96
97       /**
98        * The the thickness of the table border.
99        *
100       * Corresponds to the XHTML 1.0 Transitional attribute "border".
101       *
102       * @param thickness
103       *            the new thickness of the table border in pixels or
104       *            <code>null</code> to remove the attribute.
105       * @return The old thickness in pixels.
106       */
107      public int setBorder(int thickness);
```

```
108
109     /**
110      * Get the background color of the table.
111      *
112      * Corresponds to the XHTML 1.0 Transitional attribute "bgcolor".
113      *
114      * @return The background color of the table or <code>null</code> if the
115      *         attribute is not specified.
116      */
117     public WomColor getBgColor();
118
119     /**
120      * Set the background color of the table.
121      *
122      * Corresponds to the XHTML 1.0 Transitional attribute "bgcolor".
123      *
124      * @param color
125      *            The new background color of the table or <code>null</code> to
126      *            remove the attribute.
127      * @return The old background color of the table.
128      */
129     public WomColor setBgColor(WomColor color);
130
131     /**
132      * Get the spacing between cell wall and cell content.
133      *
134      * Corresponds to the XHTML 1.0 Transitional attribute "cellpadding".
135      *
136      * @return The spacing between cell wall and content in pixels or
137      *         <code>null</code> if the attribute is not specified.
138      */
139     public int getCellPadding();
140
141     /**
142      * Set the spacing between cell wall and cell content.
143      *
144      * Corresponds to the XHTML 1.0 Transitional attribute "cellpadding".
145      *
146      * @param padding
147      *            The new spacing between cell wall and content in pixels or
148      *            <code>null</code> to remove the attribute.
149      * @return The old spacing between cell wall and content in pixels.
150      */
151     public int setCellPadding(int padding);
152
153     /**
154      * Get the space between cells.
155      *
156      * Corresponds to the XHTML 1.0 Transitional attribute "cellspacing".
157      *
158      * @return The space between cells in pixels or <code>null</code> if the
159      *         attribute is not specified.
160      */
161     public int getCellSpacing();
162
163     /**
164      * Set the space between cells.
165      *
166      * Corresponds to the XHTML 1.0 Transitional attribute "cellspacing".
167      *
168      * @param spacing
169      *            The new space between cells in pixels or <code>null</code> to
170      *            remove the attribute.
171      * @return The old space between cells in pixels.
172      */
173     public int setCellSpacing(int spacing);
174
175     /**
176      * Get the outer border parts that will be rendered.
177      *
178      * Corresponds to the XHTML 1.0 Transitional attribute "frame".
179      *
180      * @return The outer border parts that will be rendered or <code>null</code>
181      *         if the attribute is not specified.
182      */
183     public WomTableFrame getFrame();
```

```
184
185     /**
186      * Set the outer border parts to be rendered.
187      *
188      * Corresponds to the XHTML 1.0 Transitional attribute "frame".
189      *
190      * @param frame
191      *            The new outer border parts to render or <code>null</code> to
192      *            remove the attribute.
193      * @return The old setting.
194      */
195     public WomTableFrame setFrame(WomTableFrame frame);
196
197     /**
198      * Get the inner border parts that will be rendered.
199      *
200      * Corresponds to the XHTML 1.0 Transitional attribute "rules".
201      *
202      * @return The inner border parts that will be rendered or <code>null</code>
203      *         if the attribute is not specified.
204      */
205     public WomTableRules getRules();
206
207     /**
208      * Set the inner border parts to be rendered.
209      *
210      * Corresponds to the XHTML 1.0 Transitional attribute "rules".
211      *
212      * @param rules
213      *            The new inner border parts to render or <code>null</code> to
214      *            remove the attribute.
215      * @return The old setting.
216      */
217     public WomTableRules setRules(WomTableRules rules);
218
219     /**
220      * Get a textual summary of the table's content.
221      *
222      * Corresponds to the XHTML 1.0 Transitional attribute "summary".
223      *
224      * @return A summary or <code>null</code> if the attribute is not specified.
225      */
226     public String getSummary();
227
228     /**
229      * Set a textual summary of the table's content.
230      *
231      * Corresponds to the XHTML 1.0 Transitional attribute "summary".
232      *
233      * @param summary
234      *            The new summary or <code>null</code> to remove the attribute.
235      * @return The old summary.
236      */
237     public String setSummary(String summary);
238
239     /**
240      * Get the table width.
241      *
242      * Corresponds to the XHTML 1.0 Transitional attribute "width".
243      *
244      * @return The table width in pixels or percent or <code>null</code> if the
245      *         attribute is not specified.
246      */
247     public WomValueWithUnit getWidth();
248
249     /**
250      * Set the table width.
251      *
252      * Corresponds to the XHTML 1.0 Transitional attribute "width".
253      *
254      * @param width
255      *            The new table width in pixels or percent or <code>null</code>
256      *            to remove the attribute.
257      * @return The old setting.
258      */
259     public WomValueWithUnit setWidth(WomValueWithUnit width);
```

```
}
```

WomTableBody.java

```
package org.sweble.wikitext.engine.wom;

/**
 * Denotes the body section of a table.
 *
 * Corresponds to the XHTML 1.0 Transitional element "tbody".
 *
 * See WomTablePartition for details.
 */
public interface WomTableBody
        extends
            WomTablePartition,
            WomUniversalAttributes
{
}
```

WomTableCaption.java

```
package org.sweble.wikitext.engine.wom;

/**
 * Denotes a table caption.
 *
 * Corresponds to the XHTML 1.0 Transitional element "caption".
 *
 * <b>Child elements:</b> Mixed, [Inline elements]*
 */
public interface WomTableCaption
        extends
            WomNode,
            WomUniversalAttributes
{
    /**
     * Get the alignment of the caption.
     *
     * Corresponds to the XHTML 1.0 Transitional attribute "align".
     *
     * @return The alignment of the caption or <code>null</code> if the
     *         attribute is not specified.
     */
    public WomTableCaptionAlign getAlign();

    /**
     * Set the alignment of the caption.
     *
     * Corresponds to the XHTML 1.0 Transitional attribute "align".
     *
     * @param align
     *            The new alignment of the caption or <code>null</code> to
     *            remove the attribute.
     * @return The old alignment of the caption.
     */
    public WomTableCaptionAlign setAlign(WomTableCaptionAlign align);
}
```

WomTableCaptionAlign.java

```
package org.sweble.wikitext.engine.wom;

/**
 * Defines how a table caption is aligned.
```

```
 */
public enum WomTableCaptionAlign
{
    LEFT,
    RIGHT,
    TOP,
    BOTTOM
}
```

WomTableCell.java

```
package org.sweble.wikitext.engine.wom;

/**
 * Denotes a table cell.
 *
 * Corresponds to the XHTML 1.0 Transitional element "td".
 *
 * See WomTableCellBase for details.
 */
public interface WomTableCell
        extends
            WomTableCellBase,
            WomUniversalAttributes
{
}
```

WomTableCellBase.java

```
package org.sweble.wikitext.engine.wom;

/**
 * Interface exposing attributes common to table cells and table header cells.
 *
 * Corresponds to the XHTML 1.0 Transitional element "th" or "td".
 *
 * <b>Child elements:</b> [Block elements]*
 */
public interface WomTableCellBase
        extends
            WomNode
{
    /**
     * Get the zero-based index of the row in which this cell is located.
     *
     * @return The zero-based index of the row in which this cell is located.
     */
    public int getRow();

    /**
     * Get the zero-based index of the row in which this cell is located.
     *
     * @return The zero-based index of the row in which this cell is located.
     */
    public int getCol();

    // ==[ The XHTML Attributes ]==========================================

    /**
     * Get an abbreviation of the cell's content.
     *
     * Corresponds to the XHTML 1.0 Transitional attribute "abbr".
     *
     * @return The abbreviation or <code>null</code> if the attribute is not
     *         specified.
     */
    public String getAbbr();

    /**
```

```
41      * Set an abbreviation of the cell's content.
42      *
43      * Corresponds to the XHTML 1.0 Transitional attribute "abbr".
44      *
45      * @param abbr
46      *            The new abbreviation or <code>null</code> to remove the
47      *            attribute.
48      * @return The old abbreviation.
49      */
50     public String setAbbr(String abbr);
51
52     /**
53      * Get categories assigned to this cell.
54      *
55      * Corresponds to the XHTML 1.0 Transitional attribute "axis".
56      *
57      * @return The categories or <code>null</code> if the attribute is not
58      *         specified.
59      */
60     public String getAxis();
61
62     /**
63      * Assign categories to this cell.
64      *
65      * Corresponds to the XHTML 1.0 Transitional attribute "axis".
66      *
67      * @param axis
68      *            The new categories or <code>null</code> to remove the
69      *            attribute.
70      * @return The old categories.
71      */
72     public String setAxis(String axis);
73
74     /**
75      * Get the scope of this cell.
76      *
77      * Corresponds to the XHTML 1.0 Transitional attribute "scope".
78      *
79      * @return The scope of this cell or <code>null</code> if the attribute is
80      *         not specified.
81      */
82     public WomTableCellScope getScope();
83
84     /**
85      * Set the scope of this cell.
86      *
87      * Corresponds to the XHTML 1.0 Transitional attribute "scope".
88      *
89      * @param scope
90      *            The new scope or <code>null</code> to remove the attribute.
91      * @return The old scope.
92      */
93     public WomTableCellScope setScope(WomTableCellScope scope);
94
95     /**
96      * Get the horizontal alignment of the cell's content.
97      *
98      * Corresponds to the XHTML 1.0 Transitional attribute "align".
99      *
100     * @return The horizontal alignment or <code>null</code> if the attribute is
101     *         not specified.
102     */
103    public WomHorizAlign getAlign();
104
105    /**
106     * Set the horizontal alignment of the cell's content.
107     *
108     * Corresponds to the XHTML 1.0 Transitional attribute "align".
109     *
110     * @param align
111     *            The new horizontal alignment or <code>null</code> to remove
112     *            the attribute.
113     * @return The old horizontal alignment.
114     */
115    public WomHorizAlign setAlign(WomHorizAlign align);
116
```

```
117     /**
118      * Get the vertical alignment of the cell's content.
119      *
120      * Corresponds to the XHTML 1.0 Transitional attribute "valign".
121      *
122      * @return The vertical alignment or <code>null</code> if the attribute is
123      *         not specified.
124      */
125     public WomTableVAlign getVAlign();
126 
127     /**
128      * Set the vertical alignment of the cell's content.
129      *
130      * Corresponds to the XHTML 1.0 Transitional attribute "valign".
131      *
132      * @param valign
133      *            The new vertical alignment or <code>null</code> to remove the
134      *            attribute.
135      * @return The old vertical alignment.
136      */
137     public WomTableVAlign setTableVAlign(WomHorizAlign valign);
138 
139     /**
140      * Get background color of the cell.
141      *
142      * Corresponds to the XHTML 1.0 Transitional attribute "bgcolor".
143      *
144      * @return The background color or <code>null</code> if the attribute is not
145      *         specified.
146      */
147     public WomColor getBgColor();
148 
149     /**
150      * Set the background color of the cell.
151      *
152      * Corresponds to the XHTML 1.0 Transitional attribute "bgcolor".
153      *
154      * @param color
155      *            The new background color or <code>null</code> to remove the
156      *            attribute.
157      * @return The old background color.
158      */
159     public WomColor setBgColor(WomColor color);
160 
161     /**
162      * Get the cell's alignment character.
163      *
164      * Corresponds to the XHTML 1.0 Transitional attribute "char".
165      *
166      * @return The alignment character or <code>null</code> if the attribute is
167      *         not specified.
168      */
169     public Character getChar();
170 
171     /**
172      * Set the cell's alignment character.
173      *
174      * Corresponds to the XHTML 1.0 Transitional attribute "char".
175      *
176      * @param ch
177      *            The new alignment character or <code>null</code> to remove the
178      *            attribute.
179      * @return The old alignment character.
180      */
181     public Character setChar(Character ch);
182 
183     /**
184      * Get the position of the alignment character.
185      *
186      * Corresponds to the XHTML 1.0 Transitional attribute "charoff".
187      *
188      * @return The position of the alignment character or <code>null</code> if
189      *         the attribute is not specified.
190      */
191     public Integer getCharoff();
192 
```

```
193     /**
194      * Set the position of the alignment character.
195      *
196      * Corresponds to the XHTML 1.0 Transitional attribute "charoff".
197      *
198      * @param charoff
199      *            The new position or <code>null</code> to remove the attribute.
200      * @return The old position.
201      */
202     public Integer setCharoff(Integer charoff);
203
204     /**
205      * Get number of columns this cell spans.
206      *
207      * Corresponds to the XHTML 1.0 Transitional attribute "colspan".
208      *
209      * @return The number of columns this cell spans or <code>null</code> if the
210      *         attribute is not specified.
211      */
212     public Integer getColspan();
213
214     /**
215      * Set the number of columns this cell spans.
216      *
217      * If the cell covers other cells after the change the covered cells will be
218      * removed. If the cell covers less cells after the change new, empty cells
219      * will be created for the uncovered cells.
220      *
221      * Corresponds to the XHTML 1.0 Transitional attribute "colspan".
222      *
223      * @param span
224      *            The new number of columns or <code>null</code> to remove the
225      *            attribute.
226      * @return The old number of columns.
227      * @throws IllegalArgumentException
228      *             Thrown if the cell is spanning beyond the table's dimensions.
229      */
230     public Integer setColspan(Integer span) throws IllegalArgumentException;
231
232     /**
233      * Get the number of rows this cell spans.
234      *
235      * Corresponds to the XHTML 1.0 Transitional attribute "rowspan".
236      *
237      * @return The number of rows this cell spans or <code>null</code> if the
238      *         attribute is not specified.
239      */
240     public Integer getRowspan();
241
242     /**
243      * Set the number of rows this cell spans.
244      *
245      * If the cell covers other cells after the change the covered cells will be
246      * removed. If the cell covers less cells after the change new, empty cells
247      * will be created for the uncovered cells.
248      *
249      * Corresponds to the XHTML 1.0 Transitional attribute "rowspan".
250      *
251      * @param span
252      *            The new number of rows or <code>null</code> to remove the
253      *            attribute.
254      * @return The old number of rows.
255      * @throws IllegalArgumentException
256      *             Thrown if the cell is spanning beyond the table's dimensions.
257      */
258     public Integer setRowspan(Integer span) throws IllegalArgumentException;
259
260     /**
261      * Tell whether content inside a cell should not wrap.
262      *
263      * Corresponds to the XHTML 1.0 Transitional attribute "nowrap".
264      *
265      * @return <code>True</code> if the cell's content should not wrap,
266      *         <code>false</code> otherwise.
267      */
268     public boolean isNowrap();
```

```
269
270     /**
271      * Set whether the content inside a cell should not wrap.
272      *
273      * Corresponds to the XHTML 1.0 Transitional attribute "nowrap".
274      *
275      * @param nowrap
276      *            The new setting.
277      * @return The old setting.
278      */
279     public boolean setNowrap(boolean nowrap);
280
281     /**
282      * Get the width of the cell.
283      *
284      * Corresponds to the XHTML 1.0 Transitional attribute "width".
285      *
286      * @return The width of the cell in pixels or percent or <code>null</code>
287      *         if the attribute is not specified.
288      */
289     public WomValueWithUnit getWidth();
290
291     /**
292      * Set the cell's width.
293      *
294      * Corresponds to the XHTML 1.0 Transitional attribute "width".
295      *
296      * @param width
297      *            The new width of the cell or <code>null</code> to remove the
298      *            attribute.
299      * @return The old width of the cell.
300      */
301     public WomValueWithUnit setWidth(WomValueWithUnit width);
302
303     /**
304      * Get the height of the cell.
305      *
306      * Corresponds to the XHTML 1.0 Transitional attribute "height".
307      *
308      * @return The height of the cell in pixels or percent or <code>null</code>
309      *         if the attribute is not specified.
310      */
311     public WomValueWithUnit getHeight();
312
313     /**
314      * Set the height of the cell.
315      *
316      * Corresponds to the XHTML 1.0 Transitional attribute "height".
317      *
318      * @param height
319      *            The new height of the cell or <code>null</code> to remove the
320      *            attribute.
321      * @return The old height of the cell.
322      */
323     public WomValueWithUnit setHeight(WomValueWithUnit height);
324 }
```

WomTableCellScope.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * Defines an association between a header cell and a data cell.
5   */
6  public enum WomTableCellScope
7  {
8      COL,
9      COLGROUP,
10     ROW,
11     ROWGROUP
12 }
```

WomTableColumn.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * A table column.
5   *
6   * This is an auxiliary interface that has no representation in XWML but is
7   * provides easier handling of table column.
8   *
9   * <b>Child elements:</b> -
10  */
11 public interface WomTableColumn
12 {
13     /**
14      * Get the zero-based index of this column.
15      *
16      * @return The zero-based index of this column.
17      */
18     public int getColIndex();
19
20     /**
21      * Get the number of cells (including rowspan calculations) in this column.
22      *
23      * @return The number of cells.
24      */
25     public int getNumRows();
26
27     /**
28      * Get a cell from this column.
29      *
30      * @param row
31      *            The row in which the cell is located. If the addressed cell
32      *            doesn't exist in itself but is part of a spanning cell then
33      *            the spanning cell will be returned instead.
34      * @return The requested cell.
35      * @throws IndexOutOfBoundsException
36      *             Thrown if <code>row &lt; 0</code> or
37      *             <code>row >= getNumCols()</code>.
38      */
39     public WomTableCellBase getCell(int row) throws IndexOutOfBoundsException;
40
41     /**
42      * Replace a cell with another cell.
43      *
44      * If a spanning cell is replaced with a cell that spans fewer cells, the
45      * remaining cells not covered by the replacement cell are filled with new,
46      * empty cells. If a cell is replaced by a cell that spans more cells, the
47      * cells covered by the replacement cell will be removed.
48      *
49      * @param row
50      *            The zero-based index of the cell to replace.
51      * @param replace
52      *            The replacement cell.
53      * @throws IndexOutOfBoundsException
54      *             Thrown if <code>before &lt; 0</code> or
55      *             <code>row >= getNumRows()</code>.
56      * @throws IllegalArgumentException
57      *             Thrown if the replacement cell is spanning beyond the table's
58      *             dimensions.
59      */
60     public void replaceCell(int row, WomTableCellBase replace) throws
           IndexOutOfBoundsException, IllegalArgumentException;
61
62     /**
63      * Replace a cell with another cell.
64      *
65      * If a spanning cell is replaced with a cell that spans fewer cells, the
66      * remaining cells not covered by the replacement cell are filled with new,
67      * empty cells. If a cell is replaced by a cell that spans more cells, the
68      * cells covered by the replacement cell will be removed.
69      *
70      * @param search
71      *            The cell to replace.
72      * @param replace
73      *            The replacement cell.
```

```
74      * @throws IllegalArgumentException
75      *             Thrown if the given cell <code>search</code> is not a cell of
76      *             this column.
77      * @throws IllegalArgumentException
78      *             Thrown if the replacement cell is spanning beyond the table's
79      *             dimensions.
80      */
81     public void replaceCell(WomTableCellBase search, WomTableCellBase replace) throws
            IllegalArgumentException;
82 }
```

WomTableFrame.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * Specifies which parts of the outside table border should be visible.
5   */
6  public enum WomTableFrame
7  {
8      VOID,
9      ABOVE,
10     BELOW,
11     HSIDES,
12     LHS,
13     RHS,
14     VSIDES,
15     BOX,
16     BORDER
17 }
```

WomTableHeaderCell.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * Denotes a table header cell.
5   *
6   * Corresponds to the XHTML 1.0 Transitional element "th".
7   *
8   * See WomTableCellBase for details.
9   */
10 public interface WomTableHeaderCell
11         extends
12             WomTableCellBase,
13             WomUniversalAttributes
14 {
15 }
```

WomTablePartition.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * The interface to access the cells, rows and columns of table head, body or
5   * foot.
6   *
7   * Rows and cells are accessed via integer indices. <b>Only valid items are
8   * counted.</b> If a table partition is given in HTML that contains invalid
9   * content (e.g.: text or elements other than <code>&lt;tr></code>), these
10  * elements are skipped in the enumeration and are not accessible through this
11  * interface. However, they can be iterated using the methods provided by the
12  * WomNode interface.
13  *
14  * Corresponds to the XHTML 1.0 Transitional element "thead", "tbody" or
```

```
 * "tfoot".
 *
 * <b>Child elements:</b> ([Preprocessor elements]|tr)*
 */
public interface WomTablePartition
        extends
            WomNode
{
    /**
     * Get the number of columns.
     *
     * @return The number of columns.
     */
    public int getNumCols();

    /**
     * Get the number of rows.
     *
     * @return The number of rows.
     */
    public int getNumRows();

    /**
     * Get the i'th row.
     *
     * @param row
     *            The zero-based index of the row to retrieve.
     * @return The i'th row.
     * @throws IndexOutOfBoundsException
     *             Thrown if <code>row &lt; 0</code> or
     *             <code>row >= getNumRows()</code>.
     */
    public WomTableRow getRow(int row) throws IndexOutOfBoundsException;

    /**
     * Get the i'th column.
     *
     * @param col
     *            The zero-based index of the column to retrieve.
     * @return The i'th column.
     * @throws IndexOutOfBoundsException
     *             Thrown if <code>row &lt; 0</code> or
     *             <code>col >= getNumCols()</code>.
     */
    public WomTableColumn getCol(int col) throws IndexOutOfBoundsException;

    /**
     * Retrieve the specified cell.
     *
     * If the specified cell does not exist but is part of a rowspan/colspan
     * cell, then the respective rowspan/colspan cell will be returned.
     *
     * @param row
     *            The zero-based index of the row in which the cell is found.
     * @param col
     *            The zero-based index of the column in which the cell is found.
     * @return The specified cell.
     * @throws IndexOutOfBoundsException
     *             If the specified cell does not exist.
     */
    public WomTableCellBase getCell(int row, int col) throws IndexOutOfBoundsException;

    // ==[ Row modification ]===============================================

    /**
     * Append a new row to the end of the table.
     *
     * @param row
     *            The row to append.
     */
    public void appendRow(WomTableRow row);

    /**
     * Insert a row in front of another specified row.
     *
     * @param before
```

```
91        *            The index of the row in front of which the new row is to be
92        *            inserted.
93        * @param row
94        *            The row to insert.
95        * @throws IndexOutOfBoundsException
96        *             Thrown if <code>before &lt; 0</code> or
97        *             <code>before > getNumRows()</code>.
98        */
99       public void insertRow(int before, WomTableRow row) throws IndexOutOfBoundsException;
100
101       /**
102        * Insert a row in front of another specified row.
103        *
104        * @param before
105        *            The row in front of which the new row is to be inserted.
106        * @param row
107        *            The row to insert.
108        * @throws IllegalArgumentException
109        *             Thrown if <code>before</code> is not a row of this table.
110        */
111       public void insertRow(WomTableRow before, WomTableRow row) throws
               IllegalArgumentException;
112
113       /**
114        * Replace a row with another row.
115        *
116        * @param row
117        *            The index of the row to replace.
118        * @param replace
119        *            The replacement row.
120        * @throws IndexOutOfBoundsException
121        *             Thrown if <code>row &lt; 0</code> or
122        *             <code>row > getNumRows()</code>.
123        */
124       public void replaceRow(int row, WomTableRow replace) throws
               IndexOutOfBoundsException;
125
126       /**
127        * Replace a row with another row.
128        *
129        * @param search
130        *            The row to replace.
131        * @param replace
132        *            The replacement row.
133        * @throws IllegalArgumentException
134        *             Thrown if <code>search</code> is not a row of this table.
135        */
136       public void replaceRow(WomTableRow search, WomTableRow replace) throws
               IllegalArgumentException;
137
138       /**
139        * Remove a row from this table.
140        *
141        * @param row
142        *            The index of the row to remove.
143        * @throws IndexOutOfBoundsException
144        *             Thrown if <code>row &lt; 0</code> or
145        *             <code>row > getNumRows()</code>.
146        */
147       public void removeRow(int row) throws IndexOutOfBoundsException;
148
149       /**
150        * Remove a row from this table.
151        *
152        * @param row
153        *            The row to remove.
154        * @throws IllegalArgumentException
155        *             Thrown if <code>row</code> is not a row of this table.
156        */
157       public void removeRow(WomTableRow row) throws IllegalArgumentException;
158
159       // ==[ Column modification ]=========================================
160
161       /**
162        * Append a new column to the end of the table.
163        *
```

```
164         * @param col
165         *            The column to append.
166         */
167        public void appendCol(WomTableColumn col);
168
169        /**
170         * Insert a column in front of another specified column.
171         *
172         * @param before
173         *            The index of the column in front of which the new column is to
174         *            be inserted.
175         * @param col
176         *            The column to insert.
177         * @throws IndexOutOfBoundsException
178         *             Thrown if <code>before &lt; 0</code> or
179         *             <code>before > getNumCols()</code>.
180         */
181        public void insertCol(int before, WomTableColumn col) throws
               IndexOutOfBoundsException;
182
183        /**
184         * Insert a column in front of another specified column.
185         *
186         * @param before
187         *            The column in front of which the new column is to be inserted.
188         * @param col
189         *            The column to insert.
190         * @throws IllegalArgumentException
191         *             Thrown if <code>before</code> is not a column of this table.
192         */
193        public void insertCol(WomTableColumn before, WomTableColumn col) throws
               IllegalArgumentException;
194
195        /**
196         * Replace a column with another column.
197         *
198         * @param col
199         *            The index of the column to replace.
200         * @param replace
201         *            The replacement column.
202         * @throws IndexOutOfBoundsException
203         *             Thrown if <code>col &lt; 0</code> or
204         *             <code>col > getNumCols()</code>.
205         */
206        public void replaceCol(int col, WomTableColumn replace) throws
               IndexOutOfBoundsException;
207
208        /**
209         * Replace a column with another column.
210         *
211         * @param search
212         *            The column to replace.
213         * @param replace
214         *            The replacement column.
215         * @throws IllegalArgumentException
216         *             Thrown if <code>search</code> is not a column of this table.
217         */
218        public void replaceCol(WomTableColumn search, WomTableColumn replace) throws
               IllegalArgumentException;
219
220        /**
221         * Remove a column from this table.
222         *
223         * @param col
224         *            The index of the column to remove.
225         * @throws IndexOutOfBoundsException
226         *             Thrown if <code>col &lt; 0</code> or
227         *             <code>col > getNumCols()</code>.
228         */
229        public void removeCol(int col) throws IndexOutOfBoundsException;
230
231        /**
232         * Remove a column from this table.
233         *
234         * @param col
235         *            The column to remove.
```

```
236         * @throws IllegalArgumentException
237         *             Thrown if <code>col</code> is not a column of this table.
238         */
239        public void removeCol(WomTableColumn col) throws IllegalArgumentException;
240
241        // ==[ The XHTML Attributes ]==========================================
242
243        /**
244         * Get the horizontal alignment of the row's content.
245         *
246         * Corresponds to the XHTML 1.0 Transitional attribute "align".
247         *
248         * @return The horizontal alignment or <code>null</code> if the attribute is
249         *         not specified.
250         */
251        public WomHorizAlign getAlign();
252
253        /**
254         * Set the horizontal alignment of the row's content.
255         *
256         * Corresponds to the XHTML 1.0 Transitional attribute "align".
257         *
258         * @param align
259         *            The new horizontal alignment or <code>null</code> to remove
260         *            the attribute.
261         * @return The old horizontal alignment.
262         */
263        public WomHorizAlign setAlign(WomHorizAlign align);
264
265        /**
266         * Get the vertical alignment of the row's content.
267         *
268         * Corresponds to the XHTML 1.0 Transitional attribute "valign".
269         *
270         * @return The vertical alignment or <code>null</code> if the attribute is
271         *         not specified.
272         */
273        public WomTableVAlign getVAlign();
274
275        /**
276         * Set the vertical alignment of the row's content.
277         *
278         * Corresponds to the XHTML 1.0 Transitional attribute "valign".
279         *
280         * @param valign
281         *            The new vertical alignment or <code>null</code> to remove the
282         *            attribute.
283         * @return The old vertical alignment.
284         */
285        public WomTableVAlign setTableVAlign(WomHorizAlign valign);
286
287        /**
288         * Get the row's alignment character.
289         *
290         * Corresponds to the XHTML 1.0 Transitional attribute "char".
291         *
292         * @return The alignment character or <code>null</code> if the attribute is
293         *         not specified.
294         */
295        public Character getChar();
296
297        /**
298         * Set the row's alignment character.
299         *
300         * Corresponds to the XHTML 1.0 Transitional attribute "char".
301         *
302         * @param ch
303         *            The new alignment character or <code>null</code> to remove the
304         *            attribute.
305         * @return The old alignment character.
306         */
307        public Character setChar(Character ch);
308
309        /**
310         * Get the position of the alignment character.
311         *
```

```
312         * Corresponds to the XHTML 1.0 Transitional attribute "charoff".
313         *
314         * @return The position of the alignment character or <code>null</code> if
315         *         the attribute is not specified.
316         */
317        public Integer getCharoff();
318
319        /**
320         * Set the position of the alignment character.
321         *
322         * Corresponds to the XHTML 1.0 Transitional attribute "charoff".
323         *
324         * @param charoff
325         *            The new position or <code>null</code> to remove the attribute.
326         * @return The old position.
327         */
328        public Integer setCharoff(Integer charoff);
329    }
```

WomTableRow.java

```
1   package org.sweble.wikitext.engine.wom;
2
3   /**
4    * A table row.
5    *
6    * Cells are accessed via integer indices. <b>Only valid items are counted.</b>
7    * If a table row is given in HTML that contains invalid content (e.g.: text or
8    * elements other than <code>&lt;th></code> or <code>&lt;td></code>), these
9    * elements are skipped in the enumeration and are not accessible through this
10   * interface. However, they can be iterated using the methods provided by the
11   * WomNode interface.
12   *
13   * Corresponds to the XHTML 1.0 Transitional element "tr".
14   *
15   * <b>Child elements:</b> ([Preprocessor elements]|th|td)*
16   */
17  public interface WomTableRow
18          extends
19              WomNode,
20              WomUniversalAttributes
21  {
22      /**
23       * Get the zero-based index of this row.
24       *
25       * @return The zero-based index of this row.
26       */
27      public int getRowIndex();
28
29      /**
30       * Get the number of cells (including colspan calculations) in this row.
31       *
32       * @return The number of cells.
33       */
34      public int getNumCols();
35
36      /**
37       * Get a cell from this row.
38       *
39       * @param col
40       *             The column in which the cell is located. If the addressed cell
41       *             doesn't exist in itself but is part of a spanning cell then
42       *             the spanning cell will be returned instead.
43       * @return The requested cell.
44       * @throws IndexOutOfBoundsException
45       *              Thrown if <code>col &lt; 0</code> or
46       *              <code>col >= getNumCols()</code>.
47       */
48      public WomTableCellBase getCell(int col) throws IndexOutOfBoundsException;
49
50      /**
51       * Replace a cell with another cell.
52       *
```

```
53        * If a spanning cell is replaced with a cell that spans fewer cells, the
54        * remaining cells not covered by the replacement cell are filled with new,
55        * empty cells. If a cell is replaced by a cell that spans more cells, the
56        * cells covered by the replacement cell will be removed.
57        *
58        * @param col
59        *            The zero-based index of the cell to replace.
60        * @param replace
61        *            The replacement cell.
62        * @throws IndexOutOfBoundsException
63        *             Thrown if <code>col &lt; 0</code> or
64        *             <code>before >= getNumCols()</code>.
65        * @throws IllegalArgumentException
66        *             Thrown if the replacement cell is spanning beyond the table's
67        *             dimensions.
68        */
69       public void replaceCell(int col, WomTableCellBase replace) throws
              IndexOutOfBoundsException, IllegalArgumentException;
70
71       /**
72        * Replace a cell with another cell.
73        *
74        * If a spanning cell is replaced with a cell that spans fewer cells, the
75        * remaining cells not covered by the replacement cell are filled with new,
76        * empty cells. If a cell is replaced by a cell that spans more cells, the
77        * cells covered by the replacement cell will be removed.
78        *
79        * @param search
80        *            The cell to replace.
81        * @param replace
82        *            The replacement cell.
83        * @throws IllegalArgumentException
84        *             Thrown if the given cell <code>search</code> is not a cell of
85        *             this row.
86        * @throws IllegalArgumentException
87        *             Thrown if the replacement cell is spanning beyond the table's
88        *             dimensions.
89        */
90       public void replaceCell(WomTableCellBase search, WomTableCellBase replace) throws
              IllegalArgumentException;
91
92       // ==[ The XHTML Attributes ]=============================================
93
94       /**
95        * Get the horizontal alignment of the row's content.
96        *
97        * Corresponds to the XHTML 1.0 Transitional attribute "align".
98        *
99        * @return The horizontal alignment or <code>null</code> if the attribute is
100       *         not specified.
101       */
102      public WomHorizAlign getAlign();
103
104      /**
105       * Set the horizontal alignment of the row's content.
106       *
107       * Corresponds to the XHTML 1.0 Transitional attribute "align".
108       *
109       * @param align
110       *            The new horizontal alignment.
111       * @return The old horizontal alignment.
112       */
113      public WomHorizAlign setAlign(WomHorizAlign align);
114
115      /**
116       * Get the vertical alignment of the row's content.
117       *
118       * Corresponds to the XHTML 1.0 Transitional attribute "valign".
119       *
120       * @return The vertical alignment or <code>null</code> if the attribute is
121       *         not specified.
122       */
123      public WomTableVAlign getVAlign();
124
125      /**
126       * Set the vertical alignment of the row's content.
```

```
127         *
128         * Corresponds to the XHTML 1.0 Transitional attribute "valign".
129         *
130         * @param valign
131         *            The new vertical alignment.
132         * @return The old vertical alignment.
133         */
134        public WomTableVAlign setTableVAlign(WomHorizAlign valign);
135
136        /**
137         * Get background color of the row.
138         *
139         * Corresponds to the XHTML 1.0 Transitional attribute "bgcolor".
140         *
141         * @return The background color or <code>null</code> if the attribute is not
142         *         specified.
143         */
144        public WomColor getBgColor();
145
146        /**
147         * Set the background color of the row.
148         *
149         * Corresponds to the XHTML 1.0 Transitional attribute "bgcolor".
150         *
151         * @param color
152         *            The new background color.
153         * @return The old background color.
154         */
155        public WomColor setBgColor(WomColor color);
156
157        /**
158         * Get the row's alignment character.
159         *
160         * Corresponds to the XHTML 1.0 Transitional attribute "char".
161         *
162         * @return The alignment character or <code>null</code> if the attribute is
163         *         not specified.
164         */
165        public Character getChar();
166
167        /**
168         * Set the row's alignment character.
169         *
170         * Corresponds to the XHTML 1.0 Transitional attribute "char".
171         *
172         * @param ch
173         *            The new alignment character.
174         * @return The old alignment character.
175         */
176        public Character setChar(Character ch);
177
178        /**
179         * Get the position of the alignment character.
180         *
181         * Corresponds to the XHTML 1.0 Transitional attribute "charoff".
182         *
183         * @return The position of the alignment character or <code>null</code> if
184         *         the attribute is not specified.
185         */
186        public Integer getCharoff();
187
188        /**
189         * Set the position of the alignment character.
190         *
191         * Corresponds to the XHTML 1.0 Transitional attribute "charoff".
192         *
193         * @param charoff
194         *            The new position.
195         * @return The old position.
196         */
197        public Integer setCharoff(Integer charoff);
198    }
```

WomTableRules.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * Specifies which parts of the inside table borders should be visible.
5   */
6  public enum WomTableRules
7  {
8      NONE,
9      GROUPS,
10     ROWS,
11     COLS,
12     ALL
13 }
```

WomTableVAlign.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * Defines vertical alignment of content inside a table cell.
5   */
6  public enum WomTableVAlign
7  {
8      TOP,
9      MIDDLE,
10     BOTTOM,
11     BASELINE,
12 }
```

WomTagExtBody.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * Wraps the content of a tag extension.
5   *
6   * Corresponds to the WXML 1.0 element "tagextbody".
7   *
8   * <b>Child elements:</b> Text
9   */
10 public interface WomTagExtBody
11         extends
12             WomNode
13 {
14     /**
15      * Return the text inside the tagextbody element.
16      *
17      * @return The text inside the tagextbody element.
18      */
19     @Override
20     public String getValue();
21
22     /**
23      * Set the text inside the tagextbody element.
24      *
25      * @param text
26      *             The new text.
27      * @return The old text.
28      * @throws NullPointerException
29      *             Thrown if <code>null</code> is passed as text.
30      * @throws IllegalArgumentException
31      *             Thrown if the given text contains "&lt;/tagextbody>".
32      */
33     public String setValue(String text) throws IllegalArgumentException,
           NullPointerException;
34 }
```

WomTagExtension.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  import java.util.Collection;
4
5  /**
6   * A Wikitext call to a tag extension.
7   *
8   * Corresponds to the XWML 1.0 element "signature".
9   *
10  * <b>Child elements:</b> attr* tagextbody?
11  */
12 public interface WomTagExtension
13         extends
14             WomProcessingInstruction
15 {
16     /**
17      * Get the name of the tag extension.
18      *
19      * Corresponds to the XWML 1.0 attribute "name".
20      *
21      * @return The name of the tag extension.
22      */
23     public String getName();
24
25     /**
26      * Set the name of the tag extension.
27      *
28      * Corresponds to the XWML 1.0 attribute "name".
29      *
30      * @param name
31      *             The new name of the tag extension.
32      * @return The old name of the tag extension.
33      * @throws IllegalArgumentException
34      *             If the given name is empty or not a valid XML name.
35      * @throws NullPointerException
36      *             Thrown if <code>null</code> is given as name.
37      */
38     public String setName(String name) throws IllegalArgumentException,
           NullPointerException;
39
40     /**
41      * Returns the attributes attached to the tag extension.
42      *
43      * @return The attributes attached to the tag extension.
44      */
45     public Collection<WomAttr> getTagAttributes();
46
47     /**
48      * Retrieve a tag attribute.
49      *
50      * @param name
51      *             The name of the attribute to retrieve.
52      * @return The attribute or <code>null</code> if no attribute with the given
53      *         name exists.
54      */
55     public WomAttr getTagAttribute(String name);
56
57     /**
58      * Add or replace an attribute of the tag extension call.
59      *
60      * @param attribute
61      *             The attribute to add or replace.
62      * @return The old attribute with the same name or <code>null</code> if
63      *         there was no attribute with the same.
64      */
65     public WomAttr setTagAttribute(WomAttr attribute);
66
67     /**
68      * Remove a attribute from the tag extension call.
69      *
70      * @param name
71      *             The name of the attribute to remove.
72      * @return The removed attribute or <code>null</code> if no attribute with
73      *         the given name exists.
```

```
74      */
75     public WomAttr removeTagAttribute(String name);
76
77     /**
78      * Get the body of the element.
79      *
80      * @return The body of the element or <code>null</code> if the element only
81      *         consists of an empty tag. An empty element that consists of a
82      *         start tag and an end tag returns an empty body.
83      */
84     public WomTagExtBody getBody();
85
86     /**
87      * Set the body of the element.
88      *
89      * @param body
90      *             The new body of the element or <code>null</code> to turn the
91      *             element into an empty tag.
92      * @return The old body of the element.
93      */
94     public WomTagExtBody setBody(WomTagExtBody body);
95 }
```

WomTeletype.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * Denotes text that should be rendered as teletype text.
5   *
6   * Corresponds to the XHTML 1.0 Transitional element "tt".
7   *
8   * <b>Child elements:</b> Mixed, [Inline elements]*
9   */
10 public interface WomTeletype
11         extends
12             WomInlineElement,
13             WomUniversalAttributes
14 {
15 }
```

WomText.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * A node containing plain text.
5   *
6   * Corresponds to the XHTML 1.0 Transitional type "Text".
7   *
8   * <b>Child elements:</b> -
9   */
10 public interface WomText
11         extends
12             WomNode
13 {
14     /**
15      * Return the text content of this node.
16      *
17      * @return The text stored in this node.
18      */
19     @Override
20     public String getText();
21
22     /**
23      * Return the text content of this node.
24      *
25      * @return The text stored in this node.
26      */
```

```
27      @Override
28      public String getValue();
29
30      @Override
31      public void appendText(String text) throws UnsupportedOperationException;
32
33      @Override
34      public String deleteText(int from, int length) throws UnsupportedOperationException,
              IndexOutOfBoundsException;
35
36      @Override
37      public void insertText(int at, String text) throws UnsupportedOperationException,
             IndexOutOfBoundsException;
38
39      @Override
40      public String replaceText(String text) throws UnsupportedOperationException;
41
42      @Override
43      public String replaceText(int from, int length, String text) throws
             UnsupportedOperationException, IndexOutOfBoundsException;
44  }
```

WomTitle.java

```
1   package org.sweble.wikitext.engine.wom;
2
3   /**
4    * The title of an internal or external link.
5    *
6    * Corresponds to the WXML 1.0 element "title".
7    *
8    * <b>Child elements:</b> ([Inline elements] \ {extlink, intlink, image, url})*
9    */
10  public interface WomTitle
11          extends
12              WomNode
13  {
14  }
```

WomTransclusion.java

```
1   package org.sweble.wikitext.engine.wom;
2
3   import java.util.Collection;
4
5   /**
6    * A Wikitext transclusion statement.
7    *
8    * Also this node is called transclusion statement it can also represent a
9    * parser function invocation or a parser variable substitution. Which of the
10   * above applies depends on the name the transclusion statement specifies and
11   * what that name represents in the context of the wiki in which the statement
12   * is evaluated.
13   *
14   * Corresponds to the WXML 1.0 element "transclusion".
15   *
16   * <b>Child elements:</b> name arg*
17   */
18  public interface WomTransclusion
19          extends
20              WomProcessingInstruction
21  {
22      /**
23       * Get the name of the (template) page to transclude.
24       *
25       * Operates on the first &lt;name> element found among this node's children.
26       *
27       * @return The name of template page to transclude.
28       */
```

```
29      public WomName getName();
30
31      /**
32       * Set the namem of the (template) page to transclude.
33       *
34       * Operates on the first &lt;name> element found among this node's children.
35       *
36       * @param page
37       *             The name of the template page to transclude.
38       * @return The old page.
39       * @throws NullPointerException
40       *              Thrown if <code>null</code> is given as name.
41       */
42      public WomName setName(WomName name) throws NullPointerException;
43
44      /**
45       * Returns the arguments of the transclusion statement.
46       *
47       * @return The arguments of the transclusion statement.
48       */
49      public Collection<WomArg> getArguments();
50
51      /**
52       * Retrieve a transclusion argument.
53       *
54       * @param name
55       *             The name of the argument to retrieve.
56       * @return The argument or <code>null</code> if no argument with the given
57       *         name exists.
58       */
59      public WomArg getArgument(String name);
60
61      /**
62       * Add or replace an argument to the transclusion statement.
63       *
64       * @param argument
65       *             The argument to add.
66       * @return The old argument with the same name or <code>null</code> if there
67       *         was no argument with the same.
68       */
69      public WomArg setArgument(WomArg argument);
70
71      /**
72       * Remove an argument from the transclusion statement.
73       *
74       * @param name
75       *             The name of the argument to remove.
76       * @return The removed argument or <code>null</code> if no argument with the
77       *         given name exists.
78       */
79      public WomArg removeArgument(String name);
80  }
```

WomUnderline.java

```
1   package org.sweble.wikitext.engine.wom;
2
3   /**
4    * Denotes text that should be rendered as underlined text.
5    *
6    * Corresponds to the XHTML 1.0 Transitional element "u".
7    *
8    * <b>Child elements:</b> Mixed, [Inline elements]*
9    */
10  public interface WomUnderline
11          extends
12              WomInlineElement,
13              WomUniversalAttributes
14  {
15  }
```

WomUnit.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * A HTML unit of measurement.
5   */
6  public enum WomUnit
7  {
8      PIXELS,
9      PERCENT,
10 }
```

WomUniversalAttributes.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * Interface inherited by elements that support the XHTML 1.0 Transitional
5   * universal attributes.
6   */
7  public interface WomUniversalAttributes
8          extends
9              WomCoreAttributes,
10             WomI18nAttributes,
11             WomEventAttributes
12 {
13 }
```

WomUnorderedList.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * Denotes an unordered list.
5   *
6   * Corresponds to the XHTML 1.0 Transitional element "ul".
7   *
8   * See WomList for details.
9   */
10 public interface WomUnorderedList
11         extends
12             WomList
13 {
14     /**
15      * Get the type of bullet point the list items use.
16      *
17      * Corresponds to the XHTML 1.0 Transitional attribute "type".
18      *
19      * @return The type of bullet point or <code>null</code> if the attribute is
20      *         not specified.
21      */
22     public WomBulletStyle getItemType();
23
24     /**
25      * Set the type of bullet point the list items should use.
26      *
27      * Corresponds to the XHTML 1.0 Transitional attribute "type".
28      *
29      * @param type
30      *             The new type of bullet point or <code>null</code> to remove
31      *             the attribute.
32      * @return The old type of bullet point.
33      */
34     public WomBulletStyle setItemType(WomBulletStyle type);
35 }
```

WomUrl.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  import java.net.URL;
4
5  /**
6   * A Wikitext plain url.
7   *
8   * Corresponds to the WXML 1.0 element "url".
9   *
10  * <b>Child elements:</b> -
11  */
12 public interface WomUrl
13         extends
14             WomInlineElement,
15             WomLink
16 {
17     /**
18      * Get the target for this external link.
19      *
20      * @return The target of the external link.
21      */
22     public URL getTarget();
23
24     /**
25      * Set the target for this external link.
26      *
27      * @param target
28      *            The new target of the external link.
29      * @return The old target of the external link.
30      * @throws NullPointerException
31      *             Thrown if <code>null</code> is given as target.
32      */
33     public URL setTarget(URL target) throws NullPointerException;
34
35     // ==[ Link interface ]=================================================
36
37     /**
38      * Returns the URL as title of this link.
39      *
40      * @return The URL as title of this link.
41      */
42     @Override
43     public WomTitle getLinkTitle();
44
45     /**
46      * Retrieve the target of this link.
47      *
48      * @return The target of this link.
49      */
50     @Override
51     public URL getLinkTarget();
52 }
```

WomValue.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * The value of a transclusion argument.
5   *
6   * Corresponds to the WXML 1.0 element "value".
7   *
8   * <b>Child elements:</b> Mixed, [Preprocessor elements]*
9   */
10 public interface WomValue
11         extends
12             WomNode
13 {
14 }
```

WomValueWithUnit.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * An HTML value associated with its unit of measurement.
5   */
6  public interface WomValueWithUnit
7  {
8      /**
9       * Get the unit of measurement of this value.
10      *
11      * @return The unit.
12      */
13     public WomUnit getUnit();
14
15     /**
16      * Get the actual value.
17      *
18      * @return The value.
19      */
20     public float getValue();
21
22     /**
23      * Get the actual value rounded to an integer.
24      *
25      * @return The value as integer.
26      */
27     public int getIntValue();
28
29     /**
30      * Set a float value with its unit.
31      *
32      * @param value
33      *            The value to set.
34      * @param unit
35      *            The unit to set.
36      */
37     public void set(float value, WomUnit unit);
38
39     /**
40      * Set an integer value with its unit.
41      *
42      * @param value
43      *            The value to set.
44      * @param unit
45      *            The unit to set.
46      */
47     public void set(int value, WomUnit unit);
48 }
```

WomVar.java

```
1  package org.sweble.wikitext.engine.wom;
2
3  /**
4   * Denotes a text as variable.
5   *
6   * Corresponds to the XHTML 1.0 Transitional element "var".
7   *
8   * <b>Child elements:</b> Mixed, [Inline elements]*
9   */
10 public interface WomVar
11         extends
12             WomInlineElement,
13             WomUniversalAttributes
14 {
15 }
```