Wiky: A Bidirectional Markup Converter

Wiky is a clientside Wiki markup to HTML converter written in javascript. As it is bidirectional, it can convert Wiki markup to HTML and later convert that generated HTML text back to Wiki markup. Optionally *Wiky* will create math formulas from a simple notation similar to LaTeX.

```
=== Sum of Integers ===
It is well known, that
[http://en.wikipedia.org/wiki/Gauss,Gauss]
at the age *of 10* found the _formula_
$\sum_{i=1}^n i = {n (n+1)}/2$
```

Sum of Integers

It is well known, that Gauss at the age **of 10** found the *formula*

$$\sum_{i=1}^{n} i = \frac{n\,(n+1)}{2}$$

Wiky is **neither** an editor **nor** a widget, though it might be used by these. It is just a converter.

The primary object of *Wiky*'s developement was its use in a lightweight eLearning environment. It had to meet the requirements:

- Clientside only.
- Creating validating XHTML markup, which is …
- … cross browser with no need of plugins or addons.
- Minimal CSS dependency.
- Suitable for *inplace editing* with Ajax components and weblogs.
- Easily extensible and modifyable Syntax.

You can play with it in the WikyBox and download the most current version. I have tested *Wiky* for several month now with a couple of students using

- Firefox 1.5
- Internet Explorer 6
- Opera 8.5

You may use it freely under the Creative Commons GNU LGPL License.

Wiky Syntax

Wikitext is written in a lightweight markup language with specific syntax rules. The choice of a wiki syntax is nontrivial, as there is no common wiki markup standard. Even if there are some attempts of standardization, no final agreement can be seen.

You can comfortably compare syntax differences with WikiMatrix.

Wiky's syntax is heavily influenced by markdown and MediaWiki.

Wiky Markup	HTML
Block Formatting	

= H1 = ... ====== H6 ======	<h1> H1 </h1 ... <h6> H6 </h6>
paragraphs separated by an empty line	paragraphs separated by an empty line
line break by two\\ slashes followed by a space or newline	line break by two slashes followed by a space or newline
["quotation"]	<blockquote> quote </blockquote>
[%code block%]	<pre>code block</pre>
Inline Formatting	
strong	strong
emphasized	emphasized
^superscript^	^{superscript}
~subscript~	_{subscript}
%code%	<code>code</code>
?ABBR(abbreviation)?	<abbr title="abbreviation">ABBR</abbr>
Links & Images	
http://u.ri	http://u.ri
[http://u.ri, title]	title
uri/image.png	
[img:uri/image.png,title]	title
[http://u.ri, title]	
Lists	
; term: definition	<dl><dt>term</dt><dd>definition</dd></dl>
* unordered * list	 <li class="u">unordered <li class="u">list
1. ordered 1. list	 <li class="1">ordered <li class="1">list
Table	
[|one|two| ||three|four|]	<table><tr><td>one</td><td>two</td></tr> <tr><td>three</td><td>four</td></tr></table>

Usage and Implementation

Wiky consists of some ordered sets of rules based on regular expressions. It doesn't rely on external CSS, so its HTML output can be used smoothly with RSS or ATOM feeds. The use of internal style attributes is possible and useful, but always an option.

Wiky is lightweight (~20 kB) and implemented as a single static object with two public methods

```
Wiky.toHtml(wikistr);
Wiky.toWiki(htmlstr);
```

for converting some text from Wiki markup to HTML and vice versa. A minimal *Wiky* demo application looks like so.

```
<html>
  <head>
    <script type="text/javascript" src="wiky.js"></script>
  </head>
  <body>
    <textarea id="input"></textarea>
    <button onclick="alert(Wiky.toHtml(document.getElementById('input').value));">
       toHtml
    </button>
  </body>
</html>
```

Extensibility and Modifications

The simple architecture of *Wiky* makes adding extensions easy. There are several rule sections, implemented as *arrays*.

```
var Wiky = {
  rules: { // wiki to html
    pre: [],
    post: [],
    nonwikiblocks: [],
    wikiblocks: [],
    nonwikiinlines: [],
    wikiinlines: [],
    escapes: [],
    shortcuts: [],
    code: [],
    lang: [],
  },
  inverse: { // html to wiki
  /* ... */
  }
};
```

Each rule consists of a *regular expression* and a corresponding string or function template in case of a match.

```
{ rex:/.../g, tmplt:"..." },
```

So adding one ore more rules, you simply add those to the corresponding rule sections. If we would want to map the shortcut '(R)' to the character '®', we could write (preferrable in a separate file)

```
Wiky.rules.shortcuts = Wiky.rules.shortcuts.concat(
  { rex:/\(R\)/g, tmplt:"&#174;" });
Wiky.inverse.shortcuts = Wiky.inverse.shortcuts.concat(
  { rex:/&#174;/g, tmplt:"(R)" });
```

When adding rules, you have to take care, if they interfere with existing rules. *Wiky* comes with two extensions:

- *wiky.lang.js* = syntax highlighting for *XML* and *Javascript*.
- *wiky.math.js* = LateX like math formulas.

The mechanism to extend or modify these rules is transparent and easy as in the example above.

Math Formulas

The primary use case of creating and editing learning content also includes the requirement of handling math formulas. A LaTeX style markup to describe math formulas is simple, intuitive and well proven for years.

So *Wiky* uses a markup similar to LaTeX as input format. It creates HTML+CSS for math rendering in order to be cross browser without the need of plugins *(So note: 'wiky.math.js' does rely on CSS — in contrast to 'wiky.js')*.

The technique used here is heavily inspired by George Chavchanidze's great project site at Maiden. *Wiky* implements only a subset of *Maiden*'s functionality and directly converts to HTML instead of XML.

In case you prefer to directly generate MathML, I would like to recommend Peter Jipsen's cool LaTeX to MathML translator AsciiMathML.

I might write more about *Wiky*'s math capabilities in a future article. You can try it out in WikyBox and will also find a short syntax reference there.

If you want to use the math extension of *Wiky*, you need to include the files:

- '*wiky.js*'
- '*wiky.math.js*'
- '*wiky.math.css*'