dllup markup language
y2015m03d12

Daniel L. Lu

March 12, 2015

1 Introduction

dllup is a lightweight markup language designed for creating simple websites. It is a way to write an easily readable text file which the dllup parser turns into a static HTML page, or a LaTeX document. This document will cover the use of dllup to produce HTML webpages.

The name "dllup" is a portmanteau of my username and "markup". It was inspired by jemdoc and markdown. It supports LaTeX equations, tables, figure captioning, syntax highlighting of source code, and automatic section numbering with table of contents. However, it lacks many other features as it is deliberately simple.

1.1 Philosophy

- The raw text file must be fast to type up.
- The output must be semantically correct HTML5 and be parseable by text-only browsers like Lynx and accessibility tools. Users can apply whatever CSS styles as they wish.
- In descending order of importance: it should be *simple*, *consistent*, *correct*. Compare with the "worse is better" approach.

1.2 Comparison to other languages

Why make yet another markup language when there are already so many available? The answer is:

1. Just for fun.

2. I want several features not available in current markup languages, and I don't need several features which are available in current markup languages.

Thus, this project is for my own personal use. That notwithstanding, here are comparisons to jemdoc and Markdown.

1.2.1 Comparison to Markdown

There are several differences:

- **dllup is much stricter**. Markdown is designed to support a large variety of ways to format things in a text file. For example, an `<h1>` can be formatted in over three ways (prefix, both prefix and suffix, and "underline" of equal signs). In dllup, everything is done one way only. This is because I wrote my language with the intention of having only one user (myself), and I tend to prefer doing things consistently.
- **dllup supports math equations**. Many attempts exist to bolt on MathJax support to Markdown (see: MathJax in use). I have made the decision instead to render math as SVG on the serverside. There are pros and cons: the good thing is that this avoids the lengthy render time of MathJax, allows repeated equations to be cached, and works on browsers without Javascript. The bad thing is that it takes a bit more bandwidth to send a large SVG file (a few kB per equation) and there is no subpixel hinting.
- **dllup supports captioned figures**. Markdown only supports inline images, although it is possible to give it a kind of caption. dllup uses semantically correct HTML5 tags `<figure>` and `<figcaption>` instead.
- **dllup does not implement several features**, such as integration with publishing platforms (Markdown supports Movable Type), and things like automatic links, and so on.

There are also several similarities: the syntax for headers, links, lists, numbered lists, emphasis, and strong text are also valid Markdown. So, if you take a simple dllup document and put it into a markdown parser, there is a good chance it will be parsed correctly. The converse is not true in general.

1.2.2 Comparison to Jemdoc

There are several differences:

- **dllup equations are SVG**. The equations in jemdoc were originally rasterized PNG images which wastes bandwidth, and looks horrible on retina screens, in print, or when zoomed in. But it should be noted there is a version of jemdoc which uses MathJax instead, with the same pros and cons as described in the previous section.
- **dllup supports captioned figures**. While jemdoc supports image blocks, it does not directly support the semantically correct <figure> caption since it targets XHTML 1.1 instead of HTML5.
- **dllup does not implement several features**. This includes other character sets, and so on. There are also no bibliographic reference packages for dllup.
- **dllup is not as portable**, as it requires external software to handle SVG equations.

There is a key similarity: the goal of maintaining personal webpages, especially in an academic setting, is identical.

2 Syntax

2.1 Article header

If the document contains a line consisting of three equal signs, then everything before it will be considered the **header**. The first line of the header is the document title, and subsequent paragraphs (separated by an empty line) will show up as paragraphs. These paragraphs can be used for useful meta-data (such as the date of creation, tags, and so forth), but should not be used for actual content.

Everything after the three equal signs will be body text, explained in the following sections.

```
dllup markup language

===

# Introduction

**dllup** is a lightweight markup language...
```

2.2 Code

2.2.1 Raw HTML

Separate raw HTML code from the rest of the document by lines consisting of *only* three question marks (no whitespace).

```
Blah blah.

???
<script>
window.alert('hi');
</script>
???

Blah blah blah.
```

2.2.2 Code blocks

A code block is separated from the rest of the document by lines consisting of *only* three or four tildes (no whitespace). Three tildes means the content will get syntax highlighted (automatic guess for syntax, unless the first line specifies the language by the word "lang" followed by the language name), and four tildes means the content wil not get syntax highlighted.

```
Blah blah.

~~~
lang c++
#include <iostream>
int main() {
    std::cout << "Hello world!" << endl;
}
~~~

Blah blah blah.
```

For example,

```c++
#include <iostream>
int main() {
    std::cout << "Hello world!" << endl;
}
```

2.3 Block elements

All block elements are separated by at least one empty line. For example there must be an empty line between a header and the paragraph that follows it. This behaviour is similar to LaTeX and allows you to use hard wraps (i.e. line breaks) to enforce a maximum line length.

2.3.1 Section headers

Headers are specified by preceding octothorpes, up to six. These will show up in the Table of Contents, which appears in the article header. Using more than three levels of section headers is not recommended.

```
# This is a first level header

## This is a second level header

### This is a third level header
that is very, very long

###### This is a sixth level header
```

2.3.2 Blockquotes

Blockquotes are preceded by a greater than symbol.

```
> The quick brown fox jumps over the lazy dog and
also jumps over the lazy cat.
```

This gets rendered as:

> The quick brown fox jumps over the lazy dog and also jumps over the lazy cat.

2.3.3 Images

Images are captioned figures by default and consist of the word "pic", the URL, the alt tag, a colon with spaces on either side, and the caption. All fields are required. You can put multiple images in the same block, one on each line.

```
pic http://i.imgur.com/WpEUM8S.jpg Ghost, a novelty chess set : _Ghost_ is
     a novelty chess set I designed which looks cool but is totally
    impractical for playing. The pieces are very flat and can be stacked
    together for compact storage.
```

This gets rendered as:

2.3.4 Display math equations

Display math equations are preceded by a dollar sign.

```
$ x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}
```

This gets rendered as:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{1}$$

Figure 1: *Ghost* is a novelty chess set I designed which looks cool but is totally impractical for playing. The pieces are very flat and can be stacked together for compact storage.

2.3.5 Lists

Lists are preceded by an asterisk or a number followed by a dot.

```
* Item one.
* Item two.
Multi-line list item.
** Nested.
*** More nested.
** Nested.
* Item 3.

1. Item one
99. Item ninety-nine? Nah it's actually item two.
```

This gets rendered as:

- Item one.
- Item two. Multi-line list item.
  - Nested.
    - More nested.
  - Nested.
- Item 3.

1. Item one
2. Item ninety-nine? Nah it's actually item two.

Nesting is supported only for unordered lists, but not for ordered ones. The first number of a numbered list must be 1, although for the remaining list items any number works.

2.3.6 Tables

Tables consist of rows where columns are separated by the vertical pipe.

```
| Puppy colour | Awesome? | Purpleness |
| ------------ | -------- | ---------- |
| Red          | $\times$ | 0.1        |
| Orange       | $\times$ | 0          |
| Yellow       | $\times$ | 0          |
| Green        | $\times$ | 0          |
| Blue         | $\times$ | 0.5        |
| Purple       | $\checkmark$ | 1      |
Summary of the awesomeness of puppy colours.
```

This gets rendered as:

Puppy colour	Awesome?	Purpleness
Red	$\times$	0.1
Orange	$\times$	0
Yellow	$\times$	0
Green	$\times$	0
Blue	$\times$	0.5
Purple	✓	1

Table 1: Summary of the awesomeness of puppy colours.

The first line is always interpreted as a header. A row consisting of only |, −, and whitespace is ignored. The vertical pipes don't need to line up and extra whitespace is discarded. The last line is always interpreted as a table caption.

2.3.7 Big button

A big button is useful in case you want to link to an important page, or a link to download something, or to perform a critical action.

```
:: big button http://www.dllu.net/
```

This gets rendered as:

big button

2.3.8 Paragraph

A block with none of the keywords/symbols in front counts as just a paragraph.

2.4 Inline elements

Inline elements, or *span* elements, apply to any kind of normal text such as paragraph text, header text, caption text, list content, and so on. But they do not apply to math equations or source code. Here are the types of elements processed in order of precedence.

2.4.1 Monospace code snippet

A `monospace code snippet` comes between backticks.

```
A `monospace code snippet` comes between backticks.
```

2.4.2 Inline math

Inline mathematics $y = mx + b$ comes between dollar signs.

```
Inline mathematics $y = mx + b$ comes between dollar signs.
```

2.4.3 Links

Links follow the Markdown syntax.

```
Links follow [the Markdown syntax](http://daringfireball.net/projects/
    markdown/basics).
```

2.4.4 Emphasis and strong

Text is *emphasized* or made **strong** by underscores and double asterisks.

```
Text is _emphasized_ or made **strong** by underscores and double
    asterisks.
```

2.4.5 References

Referring to or citing things is done using `(#asdf)`, for example (fig 1)(eq 1)(s 2.4.5)[1]. A reference list is similar to a regular unordered list, except each element starts with `* [#asdf]`, like this:

References

[1] Lu, Daniel. “dllup markup language”.

```
Referring to or citing things is done using ‘(#asdf)‘, for example (#fig1)
    (#eq1)(#s2.4.5)(#asdf). A reference can be created as ‘* [#asdf]‘,
    like this:

* [#asdf] Lu, Daniel. "dllup markup language".
```

2.5 Typographical character substitutions

dllup makes automatic substitutions:

- Real quotation marks that look like “66” and “99” instead of straight quotation marks.
- Real single quotation marks and apostrophes that look like ‘6’ and ‘9’.
- Dashes — em dashes for breaking sentences, and en dashes for ranges like A–Z.
- Ellipses...

2.5.1 Special character escaping

You can escape special characters using backslashes.

The dollar sign “$”, backtick “`”, for example, should be escaped as $ and ` respectively if you wish to use them in prose.

3 Implementation

dllup is implemented in Python 3. It is fewer than 250 lines of Python, most of which are one-line functions (compare with Markdown which is 1400 lines of Perl and jemdoc which is 1400 lines of Python).

The Python code aims to be concise and does not contain html information about the header and footer of the document.

For rendering math equations, dllup uses svgtex which uses MathJax through PhantomJS.

For syntax highlighting, dllup uses Pygments.

The source code of dllup is available on Bitbucket under the MIT license.

4 Installation

4.1 dllup to html

Using dllup is straightforward, but it is not as portable as jemdoc and not as polished. As such it will not check for dependencies and will simply crash if you do not set up your system exactly as described.

The parser is a single Python script that you can download here:

Download dllup

- Install PhantomJS
- Install Pygments
- Git clone svgtex to some directory

Ensure that you are using Python 3 (you can type `python -version` on the command line). Start running svgtex in the background before running dllup:

```
phantomjs main.js
```

Ensure that svgtex is set up to listen to port 16000.

4.1.1 Compiling a single file

To run, `dllup.py` does input and output on standard IO. By default it creates svg files for equations in a folder `site/texcache`.

```
mkdir site/texcache
./dllup.py < site/index.dllu > site/index.html
```

In Python you can also import dllup as a module:

```
#!/usr/bin/python

import dllup

s = """Example

===

#_Hi

The_quick_brown_fox_jumps_over_the_lazy_dog."""

print(dllup.parse(s))
```

4.2 dllup to LaTeX

Compiling dllup to LaTeX is easier since it doesn't require svgtex, phantomjs, or pygments. All you need is a working LaTeX installation and the following Python 3 script.

Download dlluptex

The output can be included in any LaTeX document, like so

```
\begin{document}
\input{index_dllu.tex}
\end{document}
```

I recommend my own LaTeX headers and template. For example, see this documentation page rendered in LaTeX (pdf) and the raw .tex output.

4.2.1 Compiling a single file

Usage is the same as for html, but easier.

```
./dlluptex.py < index.dllu > index_dllu.tex
pdflatex texheader.tex
pdflatex texheader.tex
```

4.3 Syntax highlighting

There is experimental syntax highlighting support for vim.

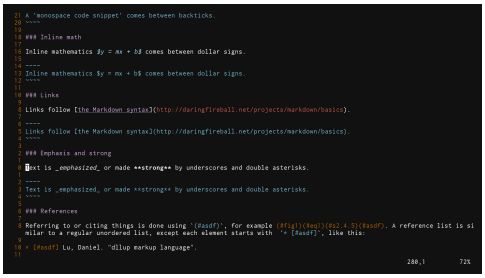

Figure 2: Syntax highlighting example in vim.

It is modified from the Markdown vim syntax file maintained by Tim Pope.

Download vim syntax

5 FAQ

Certain features don't work in my browser.

The output of dllup has been W3C validated and tested in the latest versions of Firefox, Google Chrome, and even Lynx. No attempt shall be made to support other browsers, although you are free to extend dllup as you please.

Why SVG math instead of MathJax or KaTeX?

MathJax is slow, causes page content to jump around as math is being rendered, and may freeze the browser on old hardware or slow browsers. KaTeX does not support critical math entities such as matrices. Saving math equations as SVG files also allows the SVG files to be cached in case a single equation shows up many times. Some users also prefer to disable Javascript for privacy reasons even when they are using a modern browser that supports SVG images.

Where can I find an example of a dllup file?

This documentation itself was written in dllup.

The parser crashed.

This can be due to a syntax error in the input. The dllup markup language is stricter than Markdown and has some syntax that is not obvious to people other than myself.