Alternatives to Word

Ben Klemens

30 May 2009

[Part six of six]

At this point, I hope I've demonstrated the efficiency gains in having a means of just writing content, a means of just formatting content, and a standard format linking the two.

In every case I can think of, the text writing part is in what you'd call a text editor. As above, there are many that you could choose from. Your OS provides a very basic one with zero learning curve and few features (Windows=Notepad, MacOS=TextPad, Unices=pico), but you can find others that are more comfortable to live in for large projects, like EMACS, vi, Ultraedit, Notepad++, and a whole lot more.

The rest of the story breaks down as to your preferred output and the closely-related question what standard you're going to lean on.

Plain text One option is to not use a formatting system at all. Just open up your preferred text editor and go to town.

Hemmingway:

- Brief.
- Did not use bullet points.
- Used un-formatted plain text.

But Hemmingway was fortunate enough to live before word processors. Today, an unadorned block of text is unacceptable, meaning that you will probably have to move your plain text to some sort of formatted system.

HTML The Web has a text-based standard that can be successfully read by dozens of web browsers on all types of computer. HTML documents from the birth of the web in the mid-80s can still be read today. Even Word can read HTML.

HTML stands for HyperText Markup Language, and although the HyperText part is probably not too relevant to the discussion here, the Markup Language part indicates that this is exactly the sort of semantic language discussed above. This is especially true with the advent of Cascading Style Sheets (CSS). CSS lets you define a class, and describe how that class is to be formatted on the screen. Then, you mark up your text with class delimiters: this is a header, this is a digression. That is, HTML with CSS is exactly the sort of semantic markup language that we're looking for.

Your colleagues will be able to read these documents with their web browser, and even edit them with software on their computer.

If you don't want to write the HTML markers yourself, there are a few systems that will turn easy-to-write plain text into proper HTML. Txt2tags[1], markdown[2], or textile[3] specify easy plain-text markers, like **boldface**, and then they'll filter that into the correct HTML.

LaTeX If you are in academia, use LaTeX. It was written for academic publishing, and universities are used to LaTeX users. It is designed around semantic markup of articles, books, and letters, and pegs them perfectly. This document is written in it, and as you can see, it looks beautiful. Any journal you want your papers to be seen in accept (and frequently prefer) LaTeX-formatted documents, and will provide you with a style sheet to apply to your document so that you can conform to their rules. Mathematics in Word looks amateurish, because only 0.02% of Word's buyers have equations in their papers; LaTeX's math typesetting makes you look smarter instantly.

It is not a strictly semantic markup, but a bit of a hybrid. I think it does a good job of combining the two, and if you want stricter semantics, then you are welcome to add `\def`s to the top of your documents to effect that.

One thing Word is good at, by the way, is deliberate inconsistency. If you want your first page in Helvetica, your second in Times, and your third page to be two-column format, this will be a pain in most semantically-oriented systems. But because Word's literal markup has no mechanism to impose consistency on the document, inconsistent formatting is much easier than in LaTeX. So there's my token compliment to Word.

If you are not in academia, then you have a stronger compatibility-with-Word problem, but consider using LaTeX anyway. Because there are reasonably effective (but imperfect) LaTeX-to-HTML translators, you can think of the language as a document-oriented HTML-producing language, and can then send HTML to your trapped-in-Word colleagues. This method will especially benefit those who want to use bibtex or makeindex to autogenerate the end matter in larger works.

Now, the above methods require work and learning, but I hope by now you agree that spending time learning something that you will use every day for years is worth the effort. But, I'm not going to tell you how to go about learning HTML and CSS markup or which text editor to use. You know how to ask your favorite search engine for "efficient text editor", "HTML tutorial" or what have you. Many of these open standards and tools are entirely free, so there is at least no financial cost to downloading the tools and playing around. Better than the search engines is to ask your favorite guru for help; many are happy to take time to help a friend work more efficiently.

Also, because standard formats are so open, there is probably somebody who has already fixed every problem you have, but it might be a separate tool. Some text editors include a spell checker, some expect you to choose an external full-time external

[1]`http://txt2tags.sourceforge.net/`

[2]`http://daringfireball.net/projects/markdown/`

[3]`http://www.textism.com/tools/textile/`

spell checker from the various available options. If you want to see the difference between your version of the document and the one your colleague edited, your editor may include a dedicated diff mode, or you may need a copy of the `diff` program.

Word Format You may have to use the Word document format at your workplace, though you can continue to use the structure above: use a plain text editor to write plain text, perhaps using format markers like those above, then, at the last minute, open the document in Word. That is, spend the bulk of your weeks of editing and revising working on content and worry about format and visual appeal only as a final step.

Because of Word's fundamentally first-person paradigm, you still need to change your format markers to real formatting yourself, but (1) Word's macro feature can help with this, and (2) you may still save time and effort, because the editing features of text editors can add that much more efficiency.

OpenOffice.org is a word processor initially from Sun Microsystems. [Due to trademark issues, they can't just call it OpenOffice.] Its key claim to fame is that it can read and write Microsoft's document formats very well, meaning that you can interoperate with your coworkers without their knowing that you aren't one of them.

Its stylist solves many of Word's style editor problems, so you may have better success with using it semantically. It has a built in bibliography database system. Maybe Mr. bdamm will get his wish for a basic vi keymap for efficient editing. Its own format is open, and you can save anything to PDF. So complaints about some details are alleviated, but it tries to imitate Word to the point of imitating Word's paradigmatic failings. The literal markup, intuitive-over-efficient, and one work = one view paradigms remain.

1 Conclusion

A great many people have spent a great deal of time thinking about how to best edit and format text, and most of them have come up with solutions that look very, very different from Word. Part of the reason for this is that the authors of Word were writing for Aunt Myrtle, while the author of LaTeX was writing a package for his own use; meaning that Word was built around ease of initial use, while LaTeX was built around efficiency. There is no metaphor that one could make between an HTML document with a cascading style sheet and a physical paper with text—but this is liberating and allows for new possibilities and an easier time with formatting.

Perhaps you are stuck with Word, and company policy dictates that you write and maintain long, complex business documents using the same tool Aunt Myrtle uses to write her thank-you notes. Hopefully this paper has given you some ideas for working more efficiently: use the style sheet, stick to plain text where possible, maybe get a copy of OpenOffice.org on the sly for saving to PDF. But hopefully you have the liberty to take the effort and time to learn some of the other paradigms. It will take you days or even weeks, your first documents will look amateurish, and over the next several years of your career you will thank yourself over and over again as you gracefully produce output with truly efficient tools.