A summary of the thesis *Difficulties maintaining separation of structure and presentation while using a browser based WYSIWYG-editor*

WYSIWYG and web content

Simon Rönnqvist

September 4, 2007

This is a summary of the Bachelor of Arts degree thesis:

Difficulties maintaining separation of structure and presentation while using a browser based WYSIWYG-editor

Copyright © 2007 Simon Rönnqvist
Some Rights Reserved.

This thesis summary is like the thesis itself licensed under a *Creative Commons Attribution-Noncommercial-Share Alike 3.0 License* available from `http://creativecommons.org/licenses/by-nc-sa/3.0/`.

Accordingly, you are free to:

- *to Share* – to copy, distribute and transmit the work
- *to Remix* – to adapt the work

Under the following conditions:

- *Attribution.* You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
- *Noncommercial.* You may not use this work for commercial purposes.
- *Share Alike.* If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.

Permissions beyond the scope of this license may be available upon request from the author. Contact information and material related to the thesis is found here:

`http://simon.fi/thesis`

According to recommendations by the World Wide Web Consortium (W3C) one should structure web content (using XHTML) according to semantic meaning and separately control how certain kinds of elements should be presented (using CSS). WYSIWYG-editors in CMS-environments tend to cause inconsistencies because they allow their users to within the content define the looks of each element that they are editing. A WYSIWYG which instead is intended for structuring content semantically is the Bitflux Editor. The thesis showed through a case study how a user of the Bitflux Editor was mislead by its WYSIWYG-nature to mark up content according to presentation instead of semantic meaning. (E.g. headings were used to achieve certain text styles.) Thus it was concluded that the WYSIWYG-concept is unsuitable for CMS-usage and for web content in general, since it causes inconsistency. The case study examined the troubadour Håkan Streng's web site, which represented a rather typical web site with mostly text content. Also a usability problem in WYSIWYG-editors intended for structural content was identified; they show presentation but expect their users to think according to structure. It was concluded that the solution to both this usability problem and to the problem with inconsistently marked up content could be having purely structural editors, that instead show the document structure while editing. One such concept is called WYSIWYM (What You See Is What You Mean), which could be tried as a replacement for WYSIWYG:s in CMS-contexts.

Contents

1 Introduction **5**

1.1 Basic concepts 5

1.2 Case background 6

1.3 Goal 6

2 Case study **7**

2.1 XHTML used in the case study 7

2.2 Results 8

2.3 Analysis 8

2.3.1 Overview 8

2.3.2 Unclear cases 8

2.3.3 Clear cases 9

2.3.4 Conclusion 11

3 Summary and discussion **12**

3.1 Evaluation 12

3.2 Recommendations 12

3.2.1 Abandon WYSIWYG for web content 12

3.2.2 Alternatives to WYSIWYG 14

1 Introduction

1.1 Basic concepts

Content management systems (CMS) are often equipped with so called WYSIWYG[1]-editors (from now on referred to as WYSIWYG). A WYSIWYG works in some ways like an ordinary word processor, striving to show the content editor what the end result would look like in real-time while editing.

A WYSIWYG often lets the content editor separately define the looks of each element. If this kind of functionality is used, it leads to poor separation of document structure[2] and presentation[3].

When defining presentation separately one defines how certain kinds of structural elements should be presented (in certain kinds of media if you wish). In practice poor separation of structure and presentation can lead to inconsistent looks of a site. It can also lead to problems when redesigning a site, since the presentational control isn't centralized. In some cases one would also want different styling for different presentational media, which is almost impossible if the presentation isn't defined separately.

The World Wide Web Consortium (W3C), lead by the inventor of the Web TIM BERNERS-LEE, develops web standards such as XHTML[4] (for structuring content) and CSS[5] (for defining presentation). Using web standards according to their recommendations is commonly referred to as standards based web design, or standards compliant web design. Keeping structure and presentation separate is a central part of standards based web design.

[1] What You See Is What You Get
[2] Structural elements are e.g. main headings, sub-headings and links.
[3] Presentation is how a document should be presented in a certain media, e.g. screen, print, screen readers or handheld devices.
[4] eXtensible Hyper-Text Markup Language
[5] Cascading Style Sheets

1.2 Case background

My production is the troubadour Håkan Streng's web site. The site is quite an ordinary kind of site, with mostly text content. The site utilizes a CMS equipped with a WYSIWYG for the content editor to use.

The production is an attempt to maintain separation of structure and presentation by using an in this respect strict WYSIWYG. One of the few web based WYSIWYG:s of today that validate the structure of the content when editing is the Bitflux Editor, and it is therefore used in the production. It is used within Flux CMS, the web based CMS for which it's primarily developed.

1.3 Goal

Due to the live validation the Bitflux Editor manages to maintain a valid XHTML 1.0 Strict[6] document structure in the production mentioned above. XHTML 1.0 Strict forbids use of obsolete[7] presentational markup[8]. The central argument of my thesis is that separation of structure and presentation cannot be maintained purely by technical means, while using a WYSIWYG such as the Bitflux Editor.

My analysis will consist of mapping out what kinds of failures to maintain a separation of structure and presentation occurred, in spite of the strict nature of the Bitflux Editor. I will later also discuss possible ways to avoid these problems. The target group for the thesis and this summary is people working with standards based web design for CMS-based sites, in other words basically anyone making a modern web site. Basic knowledge of XHTML is assumed, though some brief explanations are given on the XHTML-terminology needed to understand the thesis.

[6] XHTML 1.0 The Extensible HyperText Markup Language `http://www.w3.org/TR/xhtml1/`

[7] Some versions of XHTML's predecessor HTML was to some extent intended to specify presentational aspects of a web page. This kind of features have later been deprecated and are no longer allowed at all in XHTML 1.0 and 1.1 Strict.

[8] XHTML, the predecessor HTML and even it's predecessor SGML (used in print) are commonly referred to as markup.

2 Case study

2.1 XHTML used in the case study

Although basic knowledge of XHTML is assumed one can indeed follow the case study by being familiar with the concepts previously presented, along with just the few XHTML elements. Therefore those few XHTML elements are presented here, along with a very brief explanation of how XHTML (and in fact other XML) documents are structured:

All of the XHTML elements below except for *line break* and *image* wrap content. Wrapping means that they consist of one opening tag <*something*> and a closing tag <*/something*>, with the wrapped content in between the tags. The wrapped content can also contain other tags. XHTML-tags (also referred to as markup) can also contain attributes according to this format <*sometag someattribute="something"*>, as far as the thesis is concerned the attributes in the code samples can be ignored, since they're not discussed any further.

<p>	paragraph - wraps paragraphs
 	line break - used to enforce line breaks in certain places in a text
<h*n*>	heading - wraps headings, *n* is a number representing the heading level, e.g. <*h1*> for main headings, <*h2*> for sub-headings and so forth
	emphasize - wraps emphasized words or phrases
	strong emphasize - wraps strongly emphasized words or phrases
&	unordered lists - <*ul*> wraps the whole unordered list (normally presented as a bulleted list) and <*li*> wraps each list item
<a>	anchor - wraps linked words
<img />	image - refers to an image document that will be shown where the tag is placed

2.2 Results

All markup that from a semantic point of view wasn't appropriate was noted. This markup is then analyzed to find out to what extent this markup was made with the presentational result in mind, as shown by the Bitflux Editor.

Table 2.1: Number of instances with inappropriate markup

	Instances
Paragraphs	5
Line breaks	9
Headings	5
Emphasize	2
Strong emphasize	2
Total	23

2.3 Analysis

2.3.1 Overview

Some of the inappropriately used markup may have been added by mistake, while as most is very likely to have been added deliberately to affect the presentational outcome.

2.3.2 Unclear cases

Paragraphs and one heading

The empty paragraphs could easily have been added by mistake but are in most cases quite likely to have been made deliberately to make up space. Figure 2.1 on the following page on the other hand looks like it could have contained content, but eventually would have been left empty. First there is an empty heading followed by a paragraph, just as there would be if the empty heading would have been a properly used one. This makes it look like it could have been left like that by mistake.

Figure 2.1: An empty heading in combination with an empty paragraph, making empty space

2.3.3 Clear cases

Line breaks

The use of line breaks at the end of list items, headings and in one case a paragraph did not cause any visual result as opposed to when put in the beginning of a heading. However the Bitflux Editor requires its user to hold down the shift-button while pushing enter in order to achieve a line break, which makes it unlikely that any of the line breaks would have been added by mistake, even though not giving the supposedly expected visual result in most cases.

Headings

Figure 2.2 on the next page clearly shows that there is no reason to suspect that the inappropriately used headings (except for the one mentioned in 2.3.2) would have been made by mistake. Figure 2.2 also shows that they gave a clearly visible visual result.

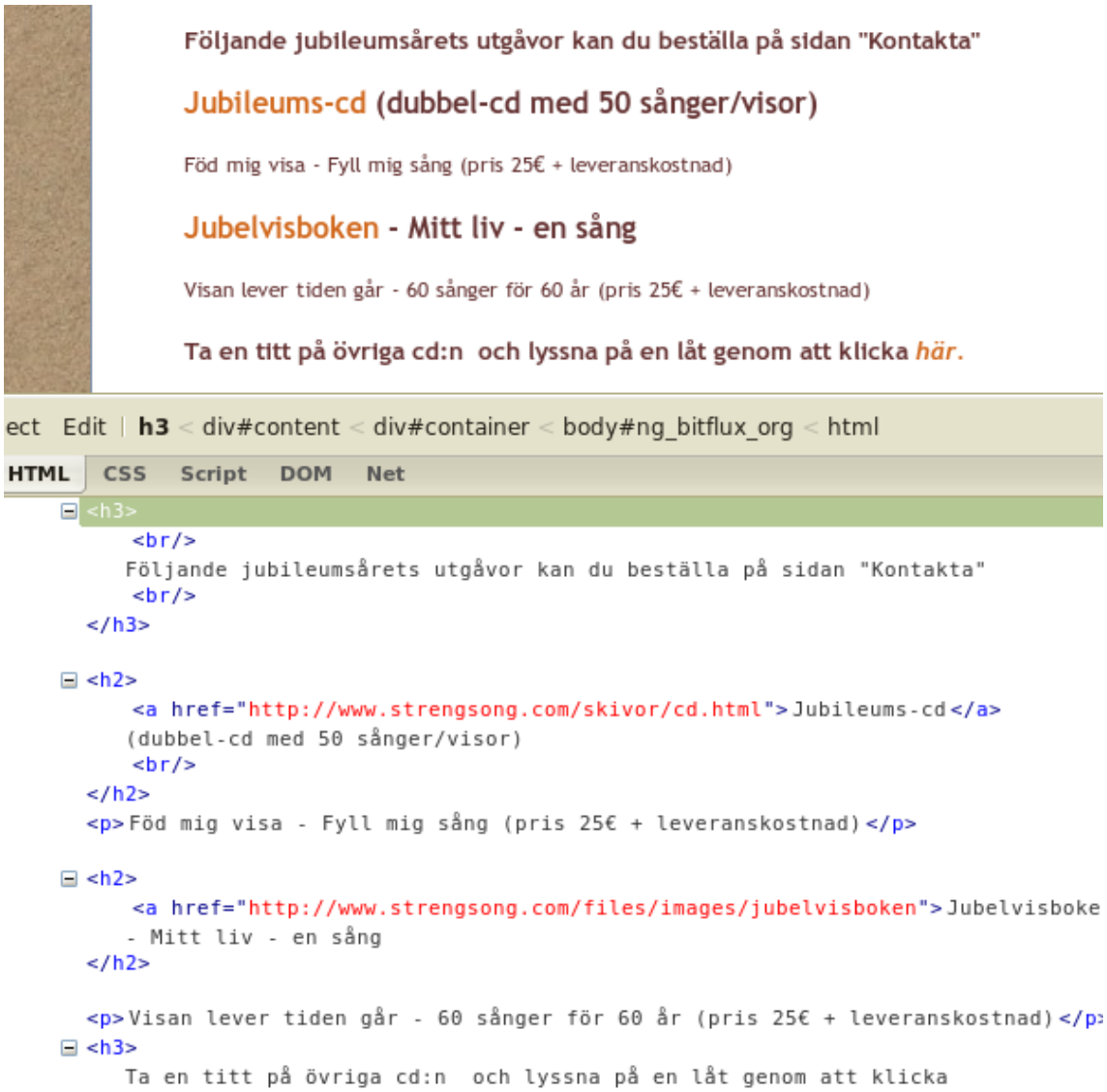

Figure 2.2: The first third level sub-heading is obviously inappropriately used since it's directly followed by a second level sub-heading instead of some content which it would represent. The page also ends with a third level heading, followed by no content at all. Also the two second level headings in the middle are inappropriately used because they actually contain text that is to it's characteristics more like ordinary page content, even though that may not be as obvious as with the previous examples.

Emphasize and strong emphasize

There is no reason to suspect that emphasize and strong emphasize would have been used for any other reason than a presentational. This also applies to the usage of strong emphasize in figure 2.3 on the following page, even though it does not happen to produce any visible result. The content editor is unlikely to have known that strong emphasize wouldn't have any visual effect within a heading, and the use of a combination of emphasize and strong emphasize doesn't semantically make sense either.

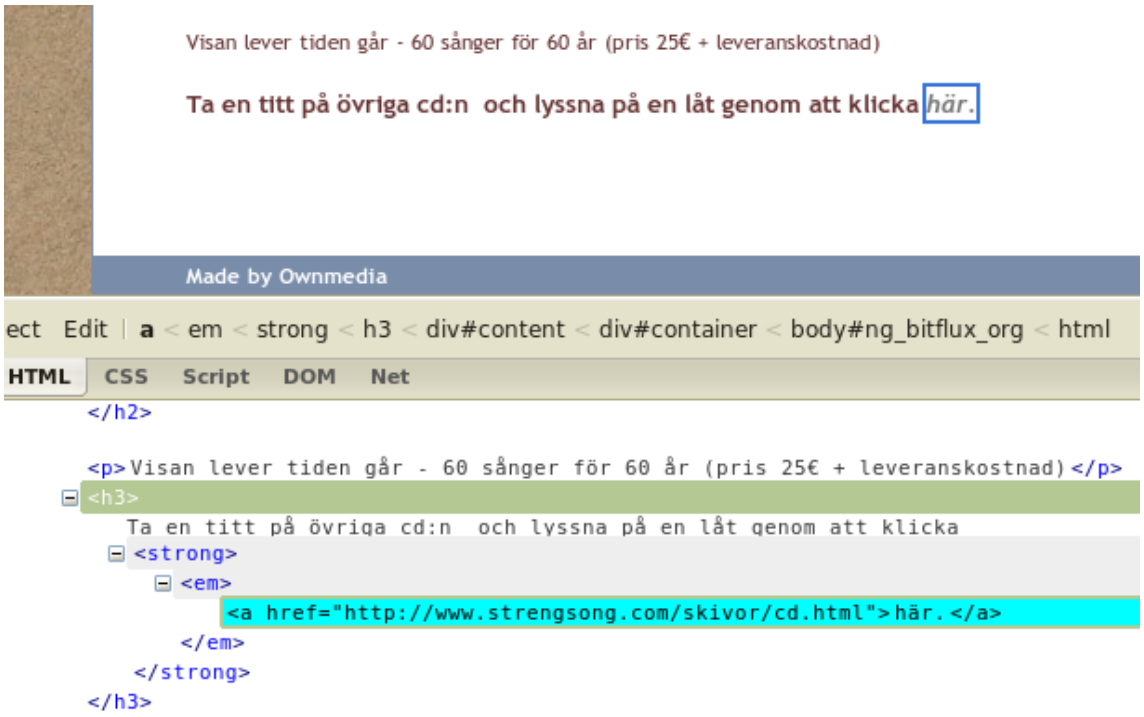

Figure 2.3: A linked word in a heading, made italic because of emphasize usage, while the use of strong emphasize has no visual effect since the fact that it's a part of a heading makes it bold already by default

2.3.4 Conclusion

All inappropriate usage of markup except for the usage of empty paragraphs and in one case an empty heading is more or less clearly deliberate, no matter whether it ended up producing a visible result or not. Exceptionally clear is the intent behind the misuse of headings to achieve certain text styling.

Even though the intent behind some of the cases remains a bit uncertain, it can be concluded that in some cases the content editor's ambition to control presentation is clearly behind the usage of inappropriate markup.

Apparently even a WYSIWYG intended for structural content can mislead its user into marking up the content according to presentation instead of semantic structure. The WYSIWYG-concept in itself becomes misleading in such a context. In the case study the content editor even received a briefing in semantic structuring. Even though these guidelines were followed to some extent, they were occasionally ignored when needed in order to achieve a certain presentational effect.

3 Summary and discussion

3.1 Evaluation

The central argument[1] that "separation of structure and presentation cannot be maintained purely by technical means, while using a WYSIWYG such as the Bitflux Editor" was proven. Even though the goal to prove that was attained, the method would have been more accurate if the content editor's intent with each action would have been monitored. In theory even any inappropriately used markup could have been added by mistake, however in practice this is very unlikely since there seems to be a clear pattern of seeking presentational effects behind most of it.

3.2 Recommendations

3.2.1 Abandon WYSIWYG for web content

Given that the WYSIWYG-concept initially was invented for print production, with one single presentational outcome, it's natural that it might not be fit for the Web. WYSIWYG:s sometimes do have their benefits due to superior cybernetics[2]. With (HTML/XHTML-based) web content this is however not the case, since the cybernetics remain false due to the multiple possible presentational outcomes. This means that the user might get tricked into thinking that he or she has ultimate control over how the content will look when presented. This ultimate control is impossible since the content will look slightly different in different ordinary browsers and very different in entirely different presentational media, such as mobile devices.

Non-CMS contexts

Some non-HTML/XHTML page elements or file formats such as Flash-movies, images and PDF-files are exceptional. These are generally not adapted to the different presentational contexts in any way apart from being re-sized, if they're viewable at all. They

[1] See 1.3 on page 6
[2] feedback and control

lack the flexibility of HTML/XHTML-based web content, but gain the possibility of authentic design cybernetics through a WYSIWYG. For example in an image manipulation program the WYSIWYG-concept makes sense, as opposed to when editing text content of HTML/XHTML-pages.

Stand-alone WYSIWYG-editors are often used to produce web content, there the WYSIWYG-concept has the same downsides as in a CMS-context. WYSIWYG-usage when designing templates for whole web sites, by web professionals knowledgeable of document structuring, is on the other hand out of the scope of the thesis. The WYSIWYG-concept might be useless or even harmful for them too, despite modern WYSIWYG:s such as Dreamweaver being designed with web standards in mind. The findings here can't however be directly applied on that kind of usage, since they concern content editing and not template design. A skilled web professional also knows that the WYSIWYG-view is just a rough preview, and that browser testing is inevitable.

Structural WYSIWYG:s

The case study showed that even a WYSIWYG that emphasizes structural content tends to sometimes have its user marking up the content according to presentation instead of structure. This tendency to trick its user into marking up content according presentation (by viewing something close to it) could be considered a usability flaw, since such a WYSIWYG is indeed meant for structuring content. In the case study more thorough instructions to the content editor could have helped, but that would still not have been a remedy for the usability flaw itself.

One could of course try to prevent some of these mistakes from happening by having an even stricter WYSIWYG, in the case of the Bitflux Editor by configuring its schema to be even stricter about what markup is allowed in which contexts. This would however not address the actual usability flaw either, only in some cases prevent the user from succeeding to apply markup according to presentation.

Conclusion

WYSIWYG:s in general give the notion of control over presentation to its user and should not be used when this notion is false, which it usually is when editing web content. By providing tools for manipulation of structure while still showing presentation, a structural WYSIWYG goes only half-way towards a solution. From a usability standpoint this kind behavior doesn't make any sense either.

The WYSIWYG-concept in itself seems to be the core problem, indirectly causing presentational markup and thereby inconsistently structured content. Therefore the widespread

usage of WYSIWYG-editors in CMS-environments is highly questionable, and should be reconsidered. In fact since most kinds of text content today (including non-web content) should be re-usable in different presentational contexts, that content would be better off managed through a structural editor as well.

3.2.2 Alternatives to WYSIWYG

The usability flaw of structural WYSIWYG:s could be corrected by making a WYSIWYG show structure instead of how the page supposedly would look, but then it would have been made into what is known as a WYSIWYM[3]. A WYSIWYM could also show elements more clearly and therefore avoid having the user mistakenly adding them, as opposed to a WYSIWYG which strives to show the eventual outcome, in the case of the Bitflux Editor leaving some of the changes made invisible to its user.

In some cases even editors more different from a WYSIWYG, such as one using wiki-style[4] markup or something similar like Textile[5] could be tried. Also some system built using traditional forms, having a different field for each element, could be tried out.

Using Textile instead of a WYSIWYG for the purpose of consistent markup seems to be a somewhat successful solution:

> I have set up a CMS for the intranet of a design-school here. I used Textile for html-formatting. About 20 professors and teachers with extremely different computer-skills are feeding content into this system. It turned out, that the code all those people produce in this CMS is highly consistent and mostly free of errors. None of these people had severe problems starting to use Textile, but of course they needed instructions.
>
> No WYSIWYG-editor works for people who don't know what they're doing. It would have taken the same or higher effort for those people to learn how to properly use a WYSIWYG-editor.
>
> Source `http://wiki.rubyonrails.org/rails/pages/Ruby+on+Rails+based+CMS`

A WYSIWYM or traditional forms however would probably be more self-explanatory and require less or no instructions to the user.

[3] A WYSIWYM (What You See Is What You Mean) shows what the different structural elements are, not how they'll eventually be presented. Recently a web based WYSIWYM called WYMeditor `http://www.wymeditor.org/` was released, and new versions are released frequently.

[4] A wiki-style editor lets the user edit plain text with a somewhat simplified markup compared to HTML or XHTML.

[5] `http://textism.com/tools/textile/`

In cases where the content editor should be allowed some control over the presentation one could look into the concept of Presentation Management Systems (PMS)[6], rather than allowing the usage of less strict WYSIWYG:s. In most cases however the content editor only needs control over content structure, since design issues usually should belong to XHTML and CSS savvy web designers.

In any case there is always a possibility that the content editor might look at the visual outcome by checking a page in a browser right after editing it, therefore being able to do further adjustments according to the visual outcome. No technical restrictions such as having only a non-WYSIWYG structural editor provided could prevent this from happening. Only having a workflow involving a separate moderator for content approval could prevent browser previewing. Using a structural editor such as a WYSIWYM in case studies otherwise similar to the one discussed here could clarify whether this would pose a problem in reality. Monitoring of the user's behavior would make sense in such a test case, to verify whether test and editing iterations take place.

In addition to technical solutions more thorough education of the content editor than in the case study of the thesis could be wise, not only education about semantics but also about how to write good web texts in general. Education about semantics would be especially important if browser checking would cause problems concerning the proper usage structural editors.

Relying on a WYSIWYG-editor to solve these kinds of problems is not, then, a sensible solution. It will often create more, and more serious, problems, than it solves.

[6] A tool used to control certain parts of the centralized presentation (CSS) `http://www.digital-web.com/articles/integrating_css_with_cms/`