Cascading Style Sheets: A Novel Approach Towards Productive Styling with Today's Standards

Matthias Keller
Steinbuch Centre for Computing (SCC)
Karlsruhe Institute of Technology (KIT)
D-76128 Karlsruhe, Germany
matthias.keller2@student.kit.edu

Martin Nussbaumer
Steinbuch Centre for Computing (SCC)
Karlsruhe Institute of Technology (KIT)
D-76128 Karlsruhe, Germany
martin.nussbaumer@kit.edu

ABSTRACT

In this paper we present an approach of generating Cascading Style Sheet documents automatically if the desired effect on the content elements is specified. While a Web user agent resolves the CSS rules and computes their effect, our approach handles the way back. We argue, that this can remarkably improve CSS productivity, since the process of CSS authoring always involves this direction implicitly. Our approach claims a new and innovative way to reuse chunks of markup together with its presentation. It furthermore bears potential for the optimization and reorganization of CSS documents. We describe criteria for CSS code quality we oriented on, including a quantitative indicator for the abstractness of a CSS presentation specification. An evaluation and recomputation of the CSS for 25.000 HTML documents shows that concerning these criteria the automatically generated code comes close to manually authored code.

Categories and Subject Descriptors

I.7.2 **[Document and Text Processing]:** Document Preparation – *Format and notation, Markup languages, Standards*

General Terms

Algorithms, Design, Languages

Keywords

HTML, CSS, presentation authoring, abstractness factor

1. INTRODUCTION

It is a success story: Cascading Style Sheets (CSS) as a language for specifying the presentation properties of structured documents has become a core technology in today's Web and changed the world of Web publishing in a positive way. The uncoupling of HTML from presentation information means a step towards a more abstract content document, allowing a more efficient content authoring and sharing. But we observed that the improvements on the content side made things more complicated on the presentation side [1]. Authoring CSS became a complex task and is sometimes referred to as "programming" [2]. The necessity of CSS debugging is well documented [3].

Style sheet languages allow separating content and presentation in a physical way only. In a logical way both are closely connected, since the presentation is useless without information associating elements of the content document with certain style rules. We found that there are two basic ways of linking content elements and style properties, *Content-to-presentation references* and *Presentation-to-content references.* When style attributes or class names, which do not express the content's semantics (cf. [4]), are used to assign CSS properties, this is a reference from a content element to a presentation rule. The mechanism CSS provides for implementing presentation-to-content references are the *CSS selectors*, which make up a substantial portion of CSS's complexity. Authoring and reusability of content clearly have priority over the authoring and reusability of presentational aspects. Thus freeing content from all presentation information and using presentation-to-content references is without doubt the silver bullet. But implementing presentation-to-content references with CSS references is more laborious: At least two extra steps are required after the presentation author has translated the desired visual (or aural) effect on a content document's rendering into property/value-pairs for certain content elements. The selector has to be designed and because of the complexity and strong interdependency of a presentation's selectors there is no way around testing the effect of changes on the rendered document. Eventually a debugging process is necessary.

WWW 2009, April 20-24, 2009, Madrid, Spain.
ACM 978-1-60558-487-4/09/04.

2. TOWARDS MORE PRODUCTIVITY

We believe that the way CSS is authored and managed today does not fully meet the designer's needs. We think that a formatting language should be applicable for designers without "programming" and "debugging".

Hypothesis: We assume that starting from structured content with assigned presentation properties, CSS code can be generated automatically that does not take second place to manually authored code in aspects of code quality.

The generation of CSS selectors means the computing of presentation-to-content references that implement a given assignment of presentation properties to content elements. Since the implementation of content-to-presentation references from the given assignment using the style attribute is straightforward, this would allow the transformation of content-to-presentation references into presentation-to-content references and back. The proposed approach could be used for several other applications than presentation authoring: reusing HTML content with presentation information, CSS optimization by recomputing existing Style Sheets, HTML optimization by reducing the presentation-induced structure and improved HTML/CSS export functionality of other than web authoring applications.

3. CODE QUALITY CHARACTERISTICS

From the fact that CSS requires "programming" and "debugging", which implies that it has to be authored manually, we conclude,

that current applications are not able to generate CSS code that meets the required quality. We propose a metric for CSS code quality that can explain the superiority of manually authored code.

Abstractness-factor: Presentation specifications with a high level of abstractness rely on structure elements with defined semantics beyond the document context and do not require additional presentation-motivated structure to be added in the content document. Thus a more abstract presentation specification improves the reusability of the content. A high abstraction level means also more general declarations, and this allows more tolerance towards content changes. From the characteristics of CSS selectors we derived a metric for measuring the degree of abstractness. The metric includes the ratio of tag selectors to class and id selectors used, as well as the amount descendant selectors restrict the scope of a declaration. A high abstractness-factor can be considered as an indicator for a high degree of content and presentation separation, if only presentation-to-content references are used. It also emphasizes the difference between manually authored presentation specifications and the generated HTML/CSS by the applications we tested, which do not use any abstract element selectors and therefore have an abstractness-factor of *0*. In contrast to this, the average abstractness-factor of manually authored code we found by evaluating 25.000 Websites (cf. sect. 4) was *0.41*. For the default style sheet shipped with Mozilla Firefox 2, reduced to all selectors our current parser supports, we computed an average abstractness-factor of *0.99*.

Other criteria: We believe that the proposed abstractness indicator is the most important criterion, but we also found that the code's *efficiency* has to be considered – what we did by comparing the number of declarations used in the CSS document. We also did account for readability issues in our solution by an adequate output formatting of the generated CSS code.

4. SOLUTION & EVALUATION

With a weighted summation of the abstractness indicator and efficiency criteria as evaluation function, the generation of CSS for a content document and a defined property assignment can be considered as an optimization problem. Applying any brute force algorithm would result in exponential complexity for the number of processed nodes. We do not believe that an exact approach which solves the problem in polynomial time can be found at all. Hence a heuristic algorithm was implemented using a number of subsequent transformation and optimization routines. In contrast to current applications the algorithm is able to handle descendant selectors in a similar way human authors do.

	Manually authored	Recomputed
Abstr. factor	0.410	0.412 (+0.4%)
Delcarations	141.4	148.8 (+5.2%)
Selector length	1.6	2.1 (+31.3%)

Table 1: Measured average values

To evaluate the solution a Web crawler was used to gather HTML documents from the Web and 25.000 documents were processed. All declarations without effect on the content document were removed, to make the manually authored CSS and the recomputed CSS comparable. Table 1 lists the measured average criteria for the original, manually authored code and the recomputed code. The abstractness factor could be even minimally increased by the recomputation. Regarding the measured code efficiency criteria, the manually authored code is still slightly superior in average.

Fig. 7 shows a scatter plot of the distribution of the abstractness factor for manual authored code vs. computed code. An interesting detail is the cluster at 0 on the x-axis. It represents CSS documents which only use class or id selectors to address content elements, like the ones exported from an application other than a Web authoring tool. The abstractness factor of these documents can be improved by the recomputation.

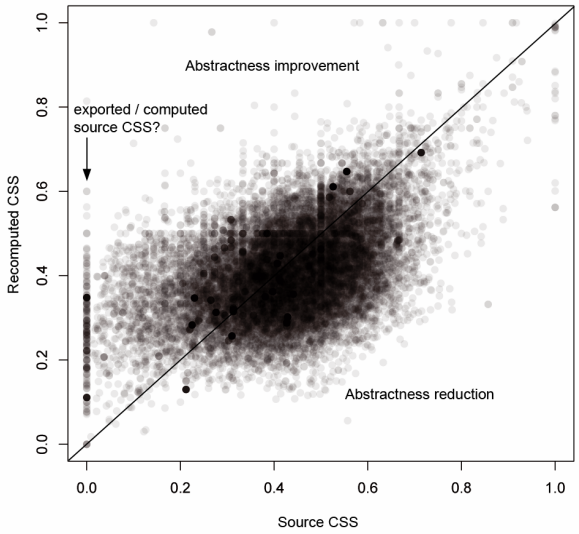

Figure 1: Abstractness factor of the source CSS plotted against the abstractness factor of the recomputed CSS

We draw two basic conclusions from the evaluated documents:

I. Regarding the examined criteria, our computer generated CSS code comes very close to the average manually generated one. But as assumed, the potential of the approach is not exhausted.

II. The abstractness factor as quantitative indicator and the intuitional understanding of how abstract or general a presentation specification is are correlated strongly.

While the algorithm's potential is far from being exhausted, it is sufficient to serve as foundation for a novel CSS editor, allowing a more productive authoring. Drag and drop functionality known from other than Web authoring applications can be implemented. And not only Web authoring applications will benefit from the proposed approach – an improved HTML/CSS export functionality of DTP or office applications enables a better cross media publishing of content *and* presentation.

For more information visit: http://research.tm.uni-karlsruhe.de/css

5. REFERENCE

[1] Weiss, A. The Web designer's dilemma: when standards and practice diverge. *netWorker* 10, 1 (Mar. 2006), 18-25.

[2] Reed, D. and Davies, J. The convergence of computer programming and graphic design. *J. Comput. Small Coll.* 21, 3 (Feb. 2006), 179-187.

[3] Quint, V. and Vatton, I. Editing with style. In *Proceedings of the 2007 ACM Symposium on Document Engineering* (Winnipeg, Manitoba, Canada, August 28 - 31, 2007), ACM, 151-160.

[4] Lie, H. W. and Saarela, J. Multipurpose Web publishing using HTML, XML, and CSS. *Commun. ACM 42*, 10 (Oct. 1999), 95-101