JPMTR 078 | 1512
DOI 10.14622/JPMTR-1512
UDC 7.012 : 004.91

Research paper
Received: 2015-12-22
Accepted: 2016-03-07

Formatting print layouts with CSS3

Christin Götz, Ulrich Nikolaus

Leipzig University of Applied Sciences (HTWK Leipzig),
Gustav-Freytag-Str. 42,
04277 Leipzig, Germany

E-mail: christin.goetz@pagina-tuebingen.de
ulrich.nikolaus@htwk-leipzig.de

Abstract

Cascading Style Sheets (CSS) have already been the *de facto* standard for the visual representation of digital content for some time now. However, advanced functions intended for the formatting of print layouts have been included only recently. With CSS level 3, which is still under development, several new features have been added to the standard, such as, for example, the definition of marginalia, footnotes, running heads or the support for advanced micro-typographic settings like OpenType features. In theory, these new features could be the key to a significant simplification of cross-media publishing, based only on the use of XML or (X)HTML and CSS3. In this paper, the current status of implementation of CSS3 features for the formatting of XML-based print layouts is discussed and its current support by rendering engines analyzed. The results suggest that CSS3 can be used at present for the formatting of simply structured content, but not for visually or semantically complex print layouts.

Keywords: XML, Cascading Style Sheets, cross-media publishing, electronic publishing, rendering engine

1. Introduction and background

The more the publication of e-books or other digital content formats becomes firmly established, the more important effective procedures for multi-channel publishing become for publisher competitiveness (Kleinfeld, 2013; Ott, 2014, p. 3; Quin, 2014, p. 253). Of course, various computer-based technologies for professional typesetting have been around since the 1980s – such as TeX by Donald E. Knuth (1984), LaTeX by Leslie Lamport (1994), Standard Generalized Markup Language (SGML) by Charles F. Goldfarb (1990) or various desktop publishing systems ranging from Aldus PageMaker and QuarkXpress to Adobe InDesign, to name but a few. However, they were mainly intended to enable high-resolution or desktop printers to translate a pre-defined design into printed form (cf. Crawford, 1994, p. 101).

The eXtensible Markup Language (XML), on the other hand, designed in 1998 with the purpose to become the "future lingua franca for the exchange of structured data" (Bosak, 1998, p. 120), has a much stronger focus on principles like media and platform independence. It has therefore become a key technology for cross-media publishing, where information contained in one single source can, in principle, be published without manual intervention on different output channels in order to become a printed book, a website, an e-book – or even an accessible document for handicapped users (cf. Nikolaus, 2010, p. 14).

However, in accordance with its media-independent approach, XML only describes the semantic structure of a document, not its layout. In order to specify a document's design, an additional stylesheet language is needed. Modern multi-channel publishing workflows automatically generate print layouts (along with various electronic products) by applying the eXtensible Stylesheet Language Formatting Objects (XSL-FO) to media independent (XML) data (Ott, 2014, p. 117; Quin, 2014, p. 253). However, XSL-FO has not become very popular within the publishing industry yet (Ott, 2014, p. 117; McKesson, 2012). Cascading Style Sheets (CSS), on the other hand, are extremely popular for the formatting of web content as well as e-books, although until now, they have barely been used in formatting of print layouts. While CSS has become a *de facto* standard for the styling of electronic documents, print layouts still have to be generated using different technology – this contradicts the basic concept to generate a coherent and truly homogeneous cross-media publishing workflow.

Nevertheless, an (albeit rather limited) *paged media* model has been added to CSS level 2 as early as 1998 – which

basically makes it possible to generate and format print layouts from HyperText Markup Language (HTML) or XML source documents using CSS as well (Ertel and Laborenz, 2014, p. 58; Harold and Means, 2004, p. 221; Lie and Day, 2005).

Since 1998 the CSS print layout functionality has been improved slowly but steadily by the World Wide Web Consortium (W3C). Thus, the current version CSS3 – with its modular design intended to simplify parallel development (Meyer, 2012) – comprises several CSS modules defining several new rules, functions, properties, values and selectors which can be used for the automatic generation of print layouts (Graham, 2014, p. 265; Quin, 2013, p. 261). Amongst others, CSS3 includes modules to:

- define and design multi-column layouts for elements in the *CSS Multi-column Layout Module* (W3C, 2011b),
- define color and opacity using different color models like RGB, HSL or CMYK in the *CSS Color Module Level 3* and *Level 4* (W3C, 2011a; W3C, 2014b), or
- make various micro-typographic settings – for instance in the *CSS Fonts Module Level 3* (W3C, 2013a) and the *CSS Text Module Level 3* (CSS3TEXT) (W3C, 2013d).

Apart from these more general CSS3 modules that can be used for other purposes as well (such as the formatting of digital content), CSS3 introduced several new modules that are of particular importance for formatting print layouts, such as:

- The *CSS Paged Media Module Level 3* (CSS3PAGE) describes a page model (or box model) for print layouts and defines rules, properties and special page selectors for the formatting of paged media (e.g. the definition of page size, orientation and margins) or the customization of headers and footers (size, styling and positioning). Furthermore, it allows for the positioning of content such as page counters in headers and footers (W3C, 2013c).
- The *CSS Generated Content for Paged Media Module* (CSS3GCPM) contains features to generate and to place content in special page areas automatically and to add, for instance, running heads, footnotes and cross-references to paged media (W3C, 2014d).
- The *CSS Fragmentation Module Level 3* (CSS3BREAK) includes properties to control pagination and defines fragmentation rules for page and text breaks which should be observed and applied when a static layout is generated (W3C, 2014c).

2. Methods

The objective of this paper was to explore how the new features of CSS3 can be used to render paged media layouts out of media independent data. In order to achieve this aim, a constructional approach was used: First, the structure, content and design of thirty textbooks and other non-fiction books (only Latin characters) with complex print layouts were analyzed to identify specific design elements that characterize paged media. Based on this analysis, a test document was developed to combine all these features into one single book. Next, a media independent XML file describing the content and structure of this test document and a CSS3 document containing the corresponding paged media formatting instructions were prepared. The content and layout files were validated (using both validation tools of the <*oXygen*/> XML editor and the W3C's online CSS Validation Service) to ensure the conformity of the XML content file with the semantic language DocBook V5 and the CSS layout document with the specification of CSS3.

From these two files, a paged media PDF document was generated automatically using two different rendering engines: *YesLogic Prince v9 rev5.0* and *Antenna House Formatter V6.2 (evaluation version)*, in further text designated as PR and AH, respectively. At the time of the tests, these two renderers were considered to be the two major commercial tools that already support features of the CSS3 Paged Media Module (cf. Kleinfeld, 2013), while the Paged Media support of other programs and projects was supposed to be insufficient (Fellenz and Fischer, 2013). Finally, the generated PDF documents were analyzed and the conformity of the resulting paged media layout to the initial layout specification was tested.

3. Results

In contrast to previous CSS versions, CSS3 includes a wide range of new rules, properties, values and selectors to format print layouts with complex structures. The CSS3 modules support various new macro- and micro-typographic settings for print layouts and provide a formatting functionality for paged media that is comparable to professional typesetting software applications, such as the creation of sophisticated running heads and footnotes, the addition of marginalia, the definition of baseline grids and floating images or the specification of advanced typographic settings for Open Type Fonts (OTF). Based on the nomenclature of classic layout processes, the new CSS3 features for print layouts can be categorized into six classes: *Page settings*, *Paragraph, Word and character formatting*, *Automatic content generation, Fragmentation options*, *Settings for printing and distribution*, and *Selectors*. For each of these categories, the following section will provide a brief overview

of the new CSS3 features for formatting print layouts. Next, the results of the rendering tests will be presented in tabular form, followed by illustrative examples.

3.1 Page settings

Compared to CSS2.1, CSS3 provides several new rules, properties and values to define paged media. Beside the description of page sizes, page margins (controlling print space) and page orientation (W3C, 2013c), it is also possible to create and control baseline grids and snaps (Lie, 2015; W3C, 2014e) or to define other layout elements, such as pagination, column titles, footnotes or margin columns, and to describe their position on the page and their styling (Lie, 2015; W3C, 2013c; W3C, 2014d). Furthermore, CSS3 supports the usage of sample pages (W3C, 2013c).

Thus, CSS3 offer a functionality comparable to DTP applications when it comes to the formatting of paged layouts. Being style sheets for markup languages, however, CSS files have to be parsed and rendered by corresponding user agents. The new features are, thus, only applicable if these renderers support them (see also Götz and Nikolaus, 2013). Unfortunately, this could not always be taken for granted. When the test document was rendered by the professional typesetting applications PR and AH, the CSS style definitions for page size, page margins (including different margin settings for left and right pages), page orientation and the positioning of column titles and page numbers, respectively, were interpreted according to the specification (Table 1).

However, the rendering of column titles was differing: While PR rendered column titles according to the CSS specification, AH produced some unexpected results

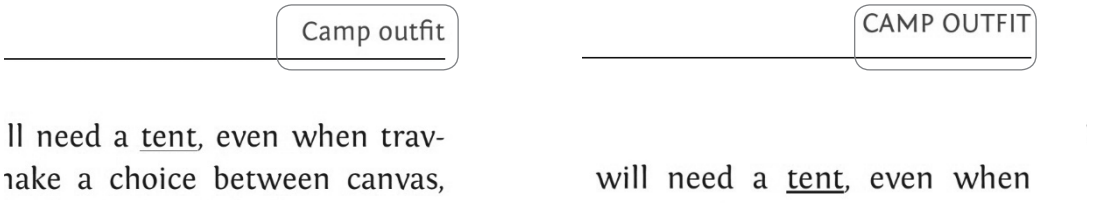

Figure 1: PR rendered column titles according to CSS specification (left) but AH (right) generated an additional offset and converted the column title to uppercase although this text transformation was only declared for the chapter titles and not for the column titles

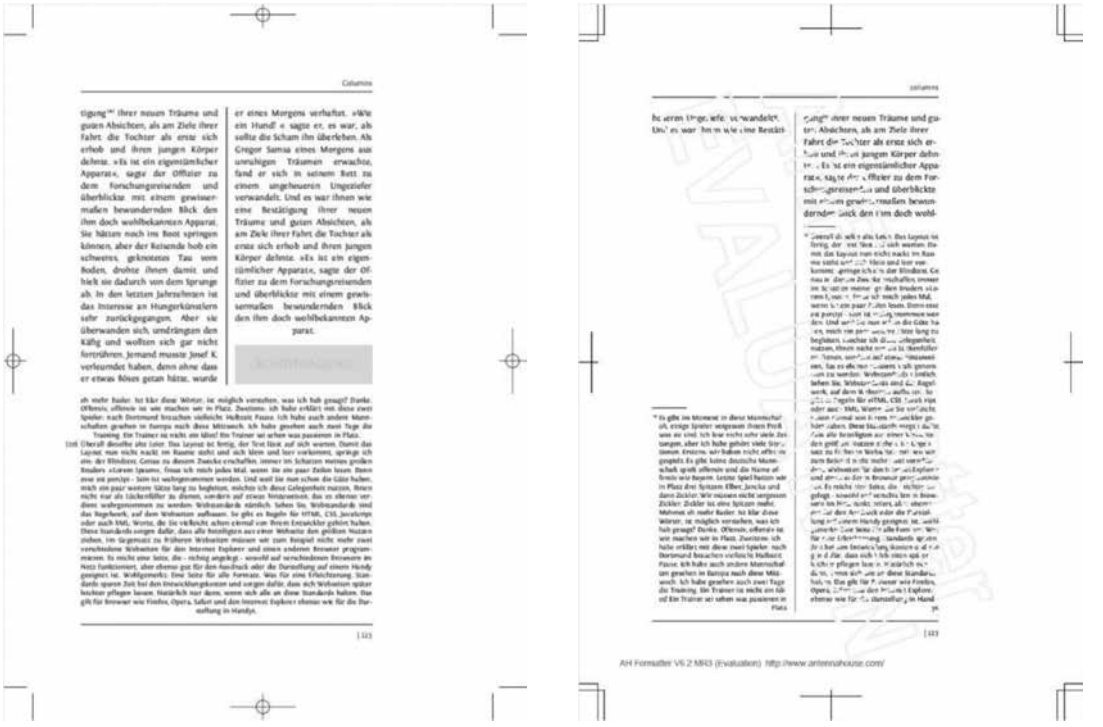

Figure 2: Rendering results of footnote styles by PR (left) and AH (right) for the CSS declaration that included a dividing line between main text and footnotes, a one column footnote area under the double-columned main text with a maximum height of 25 % of page size, and custom footnote markers and positioning

when the column title was generated automatically from the chapter titles using the property `string-set` and the function `string()`. In this case, a text transformation to uppercase that was only intended for the chapter titles themselves was unintentionally propagated to the column title as well (Figure 1).

Table 1: Excerpt of test results for page settings, for more detailed descriptions cf. Götz (2014)

Analyzed and tested new features of CSS3 for page setting	**PR**	**AH**
Page size [f]	●	●
Orientation	●	●
Print space, margins, double page layout	●	●
Column title [f]	●	◒
Page number [f]	●	●
Footnote (notes, counter, area) [f]	◒	◒
Margin column [d]	○	○
Page template [d] and page selector	◒	◒
Baseline grid [d] and register accuracy	○	○

Legend: [f] format, [d] definition, ● *full support,* ◒ *partial support,* ○ *no support*

The rendering of footnotes was also heterogeneous (Figure 2). The styling of both footnote element and footnote call were consistent (full CSS support), as was the styling of footnote counters (although user-defined counters were only rendered if they were defined using `strings` but not when the `@count-style` rule was used). Changes of the `display` property of footnote elements (from `block` to `inline`) were ignored by both renderers. The styling of footnote markers was fully supported by AH but only partially by PR (footnote markers were always positioned outside of the print area, regardless of the CSS property settings). Custom changes of the footnote area were totally ignored by PR, which only used predefined formatting settings, while AH supported a few style settings like `background`, `border-clip` and text spacing; though it ignored the CSS properties for `height` and `width` of footnote area as well).

The rendering of marginalia did not comply with the CSS specification either, although renderer-specific workarounds to define such layout elements (and to fill them with content) could be found. However, none of these solutions proved to be fully satisfactory, as an alignment neither to baseline grids nor to corresponding elements in the main text could be realized.

Additional rendering problems could be found while working with page templates and page selectors. Page templates offer the opportunity to pre-define common page layouts; page selectors can be used to define varying layouts for the first page of a document or chapter, for blank pages or all left or right pages, respectively (`:first`, `:blank`, `:left`, and `:right`). In general, both the page template functionality and the page selectors were supported by both the PR and AH renderers; however, the pseudo class `:nth()` intendet to define a different styling for every fifth or ninth page and so on, was not.

3.2 Paragraph, word and character formatting

The CSS3 specification provides new features for the formatting of textual content. In addition to CSS1 and 2.1 properties such as `font-family`, `font-size`, `font-weight` and `font-style`, it is now possible, for instance, to specify `font-stretch` (e. g. condensed or semi-expanded font faces) or to use OpenType features like standard or historical ligatures and small and petite capitals, respectively, through `font-variant` and `font-feature-settings` (W3C, 2013a). As to justification, CSS3TEXT offers, for example, `text-justify` adding new sophisticated justification algorithms and `hanging-punctuation` specifying if a punctuation mark at the start or end of a text line should be rendered outside or inside the line box (W3C, 2013d). Moreover, multi-column layouts can be specified (W3C, 2011b); also, leaders for tables of contents, registers and the like are defined and formatted using the `leaders()` function (W3C, 2014d). Customized counters for lists, etc., can be declared (with `@counter-style` and `::marker`) and additional settings for the positioning, floating and the column span of figures and boxes are available (`float` properties like `top-bottom`, `column-span`, `float-defer-column`, `float-defer-line`, `float-offset`). Text can now be wrapped around other content (W3C, 2013b; Lie, 2014) and boxes can be embroidered with rounded corners or shadows (W3C, 2014a).

Just like the renderers' support for page settings, the results for paragraph, word and character formatting were equally mixed (Table 2). Only the `leaders()` function was fully supported by both rendering engines. As to the font settings, only the CSS3 `@font-face` rule (used to embed fonts into paged media) was supported by both PR and AH, while `font-stretch` and `font-variant` were not supported by either one.

The justification test showed that currently only `text-align-last` (used to specify the alignment of the last line of text of a paragraph) is partially supported by PR. AH, on the other hand, fully supported the text justification properties `hanging-punctuation` and `text-align-last` – but only partially supported the new `text-align` values and ignored the properties `font-kerning` and `text-justify` (though a proprietary version of the latter is available: `(-ah-)text-justify-trim`).

As for multiple column page layouts, the column properties `columns`, `column-rule` and `column-fill` can already

be used. The `column-span` settings, however, were ignored by PR if used for content that was generated by pseudo elements, and AH ignored `column-span: all` if it was used in the footnote area (instead, the columns in the footnote area always matched the layout of the main text columns, see Figure 2).

Table 2: Excerpt of test results for paragraph, word and character formatting, for more detailed descriptions cf. Götz (2014)

Analyzed and tested new features of CSS3 for paragraph, word and character formatting	**PR**	**AH**
Font settings	◒	◒
Justification	◒	◒
Columns	◒	◒
Depiction and decoration (e.g. text indent, text orientation, text decoration, text transform)	◒	◒
Leaders for register	●	●
Individual counter for e. g. lists	◒	◒
Figures and boxes	◒	◒

Legend: ● *full support,* ◒ *partial support,*

The `writing-mode` feature (`top-to-bottom`, `right-to-left` and `left-to-right`) was supported by both renderers. PR additionally recognized the new value `hanging` of `text-indent`, while AH supported `text-transform` without limitation. Other functions were only partially implemented (AH: `text-orientation`, `text-decoration-style` and `text-shadow`) or unsupported (PR: `text-orientation`, `text-transform`, `text-shadow` and `text-decoration` properties; AH: `text-indent: hanging`, `text-decoration-underline-position` and `text-decoration-skip`; both: `text-indent: each-line`). Again, PR and AH sometimes offered proprietary alternatives (YesLogic, 2015a; Antenna House, 2014).

The CSS styles to define individual counters are equally unknown to the renderers, although the pseudo element `::marker` can be used to customize predefined counters for lists, footnotes and the like.

In regard to the formatting of figures, CSS3 offers many new features, but only a few of them can already be used. The property `float`, to begin with, is supposed to control the alignment of figures on the page (*vertically:* `float: top | bottom | top-bottom | bottom-top`; *horizontally:* `float: inside | outside`). The properties `top`, `bottom`, `inside` and `outside` are currently supported by both AH and PR, whereas `top-bottom` and `bottom-top` (intended to define preferred alignments that can, however, be overridden by the renderers, if necessary) were not. The alignment in multi-column layouts can be controlled using `column-span`. Here, only the values `none` and `all` were rendered correctly, while it was not possible to span an image across a specific number of columns using `integer` or `length` values. Both the new `float-defer-*` settings (which can be used to transfer a figure to another column or another page irrespective of its position in the original XML document) and the advanced new wrap configurations, `wrap-side` and `wrap-contrast` (Lie, 2014) or `clear-side` and `exclude-level` (W3C, 2013b), were not considered during the rendering process at all.

The performance of the renderers for CSS3 box model instructions – `border-image` to use bitmap images as border style patterns, `border-radius` to define rounded borders and `box-shadow` to attach shadows to an element (W3C, 2014a) was equally mixed. Border images were replaced by black solid borders by both PR and AH; `box-shadow` was only supported by AH, although the property was not interpreted according to CSS3 specification (box and content were converted to an image with low resolution and the RGB color mode was used, although CMYK colors had been defined throughout the document). The `border-radius-*` feature could be used according to specification, although PR did not accept the shorthand `border-radius` to define different radii for each corner; instead, the longhand terms such as `border-bottom-left-radius` had to be resorted to.

3.3 Automatically generated content

Automatically generated content, like paginations, column titles, footnotes, cross-references, tables of content or catchword indexes, is very important for effective cross-media publishing. Consequently, CSS3 offers various settings to generate paged media content automatically. Compared to previous CSS versions, CSS3 includes additional functionalities for:

- the automatic generation of paginations: `content: counter(page)` (W3C, 2013c),
- footnotes: the declaration `float: footnote` turns an element into a footnote; this conversion necessitates several background processes – the element has to be moved from the main text to the footnote area, a footnote marker has to be displayed before the element as well as a footnote call in the main text, and the footnote counter has to be incremented (W3C, 2014d; Lie, 2015),
- running headers and footers: the property `string-set` combined with `content: string()` is used to copy the textual content of an element into the header/footer or `position: running()` and `content: element()` to move an element including all of its substructures from the main text to the page margin boxes for headers and footers, respectively (W3C, 2014d),
- cross-references: `target-counter()` and `target-counters()` generate numbered cross-references and `target-text()` retrieves the text value of an element referred to using an URL (McKesson, 2012; Lie, 2015),

- bookmarks: CSS3 offers three properties to generate bookmarks: `bookmark-level` to define the hierarchy level of a bookmark in the bookmark structure, `bookmark-label` to declare the text which should be displayed to identify a bookmark entry and `bookmark-state` to specify the state of a bookmark hierarchy as open or closed – with open state bookmarks displaying the next level of bookmarks (W3C, 2014d; Lie, 2015).

Sophisticated as these new functions may be, there are still some features for automatic content creation left that the CSS3 specification does not include up to now: catchword indexes and tables of content. For the time being, this content has to be created semi-automatically using XSLT (XSL Transformations) and the CSS3 property for cross-references; for more details see Götz (2014).

Considering the features that are already included in the CSS3 specification, the test results show that only the automatic pagination and the definition of footnotes are currently supported satisfactorily by the renderers (Table 3); all other aforementioned functionalities for automatic content generation are only partially implemented. Column titles could be created using both methods mentioned above (`string-set: [<identifier> <content-list>];` and `content: string()` or `content: element()` and `position: running()`), but not all arguments intended to specify the mentioned content could be used by PR and AH (especially `first-except`, which allows for an empty header/footer on the page where a new column title is assigned, is currently unsupported).

As to the automatic generation of cross-references, the test shows that the `content` values `target-counter()` and `target-counters()` (to create dynamic paged cross-references that automatically adapt the pagination to page size or layout changes) can be used by both rendering engines. The property `target-text()`, to generate customized textual references like "see chapter *(name of chapter), p. (page number)*", however, is currently only supported by AH.

Table 3: Excerpt of test results for generated content, for more detailed descriptions cf. Götz (2014)

Analyzed and tested new features of CSS3 for automatic generated content	PR	AH
Table of content [d]	◐	◐
Index [d]	◐	◐
Column title [d]	◒	◒
Page number [d]	●	●
Footnote [d]	●	●
`Content()`-function	◒	○
Cross-reference	◒	◒
Bookmarks	◒	◒

Legend: [d] definition, ● full support, ◐ support only in combination with an XSLT, ◒ partial support, ○ no support

On the other hand, the properties `bookmark-level` and `bookmark-state` for the generation of bookmarks were already fully supported. Bookmark labels were tagged correctly by both PR and AH when the text labels were hard-coded via `string`. PR also supported the `content()` function that can be used to define bookmark labels dynamically, but only without any argument (`content()`) or if the property `text` was used (`content(text)`) – other arguments like `before`, `after` or `first-letter` were unsupported. AH offered no support at all of the `content()` function – neither for the labeling of bookmarks nor in any other case.

3.4 Fragmentation options

To control the fragmentation of pages, columns and regions, CSS3 provides the properties `break-before` and `break-after`. These are similar to the CSS2.1 properties `page-break-before` and `page-break-after`, but new values, like `page`, `column` (inserting mandatory page and column breaks, respectively) and `recto` (adding page breaks so that the following page is a recto page) are added (W3C, 2014c). For in-text breaks, CSS3TEXT contains e.g. the properties `line-break` (fragmentation options for punctuations), `word-break` (breaking rules for letters) and `hyphens` (control of word splitting) (W3C, 2013d). In the CSS3GCPM module, the W3C also adds the property `footnote-policy`. This is important for fragmentation if a footnote call is located near the bottom of the print space so that there is not enough space to render the complete footnote body on the same page. With `footnote-policy`, it can be specified whether a page break should be inserted at the start of the line or before the corresponding paragraph, so that both reference and body of the footnote are displayed together on the same page (W3C, 2014d).

Although only about 20 % of the CSS3 fragmentation settings were supported in the test (Table 4), the rendering engines use proprietary methods to define breaking rules. For page breaks, the new properties `break-before/-after` were only supported by AH, but both rendering engines accepted the old CSS2.1 `page-break-before/-after` properties (limited to the CSS2.1 value set `always`, `avoid`, `left` and `right`). CSS3 line breaks were ignored by both PR and AH. AH supported CSS3 word fragmentation settings and PR as well as AH used proprietary commands for hyphenation (`prince-hyphenate-lines`, `prince-hyphenate-before`, `prince-hyphenate-after` (YesLogic, 2015a), see Figure 3. CSS3 page break settings for footnotes were ignored, but again, PR added a proprietary command named `prince-footnote-policy`.

Table 4: Excerpt of test results for fragmentation options, for more detailed descriptions cf. Götz (2014)

Analysed and tested new features of CSS3 for fragmentation options	PR	AH
Page, column and area fragmentation	○(1)	●
Hyphenation	◒(2)	◒(2)
Line break	○	○
Word break	○	●
Fragmentation settings for footnotes	○(3)	○

Legend: ● *full support,* ◒ *partial support,* ○ *no support*

(1) *Instead of the CSS3* `break-before/-after` *commands, the CSS2.1 properties* `page-break-before/-after` *could be used to define page breaks*

(2) *Proprietary functions to influence hyphenation:* `prince-hyphenate-lines`, `prince-hyphenate-before`, `prince-hyphenate-after`

(3) *Workaround to define fragmentation for footnotes with* `prince-footnote-policy`

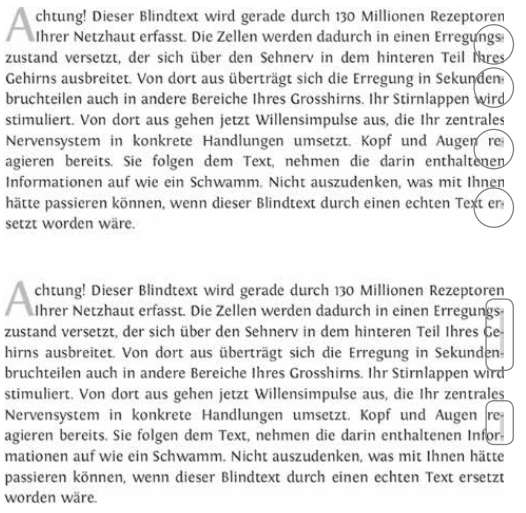

Figure 3: Two renderings showing the effect of a proprietary feature to influence hyphenation: due to the command `prince-hyphenate-lines: no-limit;` *in the lower half, any number of lines ending with a hyphenation could follow one other, whereas in the upper half, the maximum number of hyphenations is 1 (*`prince-hyphenate-lines:1;`*)*

3.5 Settings for printing and distribution

The new CSS3 features for printing and distribution include properties to switch between color spaces like RGB (W3C, 2011a) and CMYK (W3C, 2014b), to control image resolutions (W3C, 2012) and to define edge trim, corner marks and bleed (W3C, 2013c). These CSS3 functionalities were mostly supported in the test except for the definition of CMYK colors (Table 5). The rendering engine PR uses a proprietary command instead of the one specified by the W3C. Color management of images, is only supported by AH. Both renderers also use different visualizations for crop marks and bleed (Figure 4).

Table 5: Excerpt of test results for printing and distribution settings, for more detailed descriptions cf. Götz (2014)

Analyzed and tested new features of CSS3 for settings for printing and distribution	PR	AH
Bleed [d]	●	●
Marks [d]	●	●
Color settings and color management	○(1)	●
Image resolution	◒	◒

Legend: [d] definition, ● *full support,* ◒ *partial support,* ○ *no support*

(1) *PR uses the proprietary command* `cmyk()` *to define CMYK colors instead of the one specified by the W3C (*`device-cmyk()`*)*

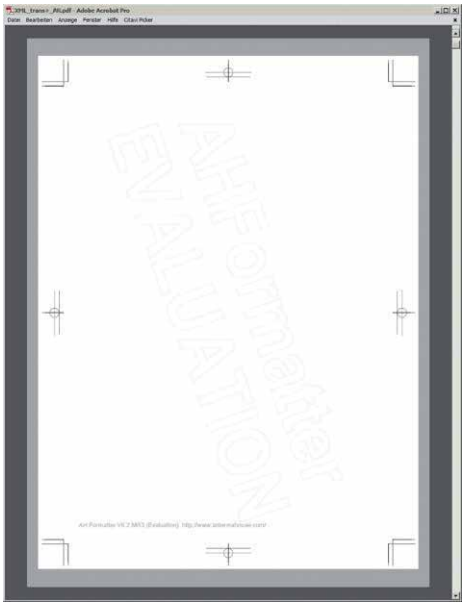

Figure 4: Rendering of bleed, marks and crops by PR (foreground) and AH (background); although the trimmed page size is identical, the untrimmed size differs, just as the visualization of marks and crops

3.6 Selectors

The CSS3 specification includes a wide range of new selectors to select and style elements separately. For instance, the new sibling selector `E ~ F` enables the selection of *all* following siblings of an element and not only the direct successor, which can be selected using `E + F`. The new pseudo class `:nth-child(n)` makes it possible to define styles e. g. for each odd table row with `tr:nth-child(odd) {…}`, and the pseudo element selector `::marker` to declare styles only for the marker, but not for the text, of a list with `li::marker {…}` (W3C, 2011c).

Fortunately, both rendering engines offer full support of all these new CSS3 selectors (Table 6), which facilitates the formatting of somewhat irregular but nonetheless recurring structures like, for instance, the odd-numbered rows of a table, which can now be addressed directly using `tr:nth-child(odd)` without recourse to inconvenient attribute definitions such as `<tr attr="odd">`.

This further improves the media-independent separation of content and layout between the XML file and the CSS document.

Table 6: Excerpt of test results for selectors, for more detailed descriptions cf. Götz (2014)

Analyzed and tested new selectors of CSS3	**PR**	**AH**
Attribute selectors	●	●
Pseudo class selectors	●	●
`::marker`, Pseudo element	●	●
`E ~ F`, Sibling selector	●	●

Legend: ● *full support,* ◐ *partial support,* ○ *no support*

4. Discussion

The aim of the present study was to test the current implementation status of CSS3 features that are needed for the formatting of XML-based print layouts in professional state-of-the-art rendering engines. In order to do this, a test document has been developed that contains the specific semantic elements of complex textbooks and other non-fiction books (only Latin characters). This book has been formatted using CSS3 and then rendered by the two state-of-the-art engines PR and AH.

The high potential of the new CSS3 features specification for a direct generation of print layouts from XML source documents becomes apparent if one considers the multitude of additional rules, functions, properties, values and selectors that have been described above. In principle, these new features could greatly enhance the efficiency of cross-media workflows. Even today, most digital publications such as websites, e-papers or e-books are using (X)HTML and CSS as source data formats to specify content and design (Quin, 2014). If print layouts could now be rendered from the same source documents as well, this would be another step towards a real single-source publishing, where a single media-independent file could be published on different channels just by exchanging the CSS. Double work and deviating processes for print layouts could thus be reduced.

This is not a mere future scenario, but is already becoming a reality: since 2013, for instance, the O'Reilly Media publishing house is developing HTMLBook, which is supposed to become an open, XHTML5-based standard for the authoring and production of both print and digital books (Kleinfeld, 2013). This initiative, however, is for the time being mainly focused on the semantic definition of a basic "book" structure (Kleinfeld, 2016) instead of the production of printed books. Whether this or other (X)HTML-based data formats are suited for the systematic editing, structuring and processing or typesetting data in a cross-media publishing workflow in conjunction with CSS3, needs further analysis. However, it has already been stated that CSS as a technology to format XML documents for print is starting to come at age (Quin, 2014), that even XSL-FO processor vendors are moving to also support CSS (Graham, 2014) and that CSS will supplant XSL-FO within the next few years in the world's publishing houses (Kelly, 2015).

As for the situation today, the test results presented in this paper show that only less than a half of the new CSS3 features for print production are currently supported by *YesLogic Prince* and the *Antenna House Formatter*. This means that the test results are basically dissatisfying: for the time being, it cannot reasonably be expected that flawless CSS3 renderings of complex XML documents for print layout can be achieved with the available technology. As it is often the case with formal languages, the implementation of new language features into the rendering engines is unable to keep pace with the language's specification process (in the context of an automatic generation of accessible publications, cf. Nikolaus (2010)).

Furthermore, the test reveals fundamental differences between the rendering results of both engines. Not only that both support only a limited number of the new CSS3 features, but also, these feature subsets are not identical. Some features *are* supported by both renderers, but the rendering results are nonetheless different and the same CSS instruction thus results in differing print layouts. Besides, the analyses show that both renderers add non-standard, proprietary functionality to the definitions of the W3C CSS specification. While this may be helpful in some cases where complex print layouts might not have been rendered otherwise, the proprietary functions may require additional efforts and application testing. Because renderer-specific approaches do not validate against the W3C's CSS specification, it cannot be checked automatically and is not interchangeable with other XML rendering solutions. While this might not be a problem as long as the print production takes place in a controlled, in-house software environment,

it might not be suitable for a distributed production or print-on-demand solutions.

However, a continuous and dynamic development in the area of XML-formatting with CSS can certainly be observed; CSS is attracting many more resources and developers than XSL-FO today (Quin, 2014, p. 261). Compared to the time of testing, the *YesLogic Prince* renderer, for example, got a major update to version 10, which (according to the release notes) offers many new properties and features for the formatting of paged media with CSS3 (YesLogic, 2015b).

While the results presented in this paper do not allow any clear statements about the performance of other rendering engines, the two renderers tested here are certainly among the leading tools in this field. Therefore, similar problems might be expected even if other tools are used for the XML-to-Print rendering, but this also calls for further study. A similar test might also be interesting for rendering engines like *pdfChip* by *callas software* (Callas Software, 2015) and *pdfreactor* by RealObjects (RealObjects GmbH, 2015), which are supposed to create high-quality PDF suited for printing directly from (X)HTML.

Finally, it needs to be mentioned that the test documents used here were based on the Latin writing system only, so that the suitability of CSS3 for the formatting of print layouts for ideographic languages with a different reading direction or for documents that use annotations for pronunciation or meaning (like the Japanese furigana; cf. Sampson (1990, p. 190)) have to be analyzed separately.

5. Conclusions

The possibility to generate and format both print and digital layouts using only XML (or HTML) and CSS3 could be the key to a significant simplification of cross-media publishing: the re-use or reprocessing of media-independent data in order to automatically generate websites, e-books, and well-designed and complex print layouts from the same media-independent data in a shorter time, with less effort and at reduced costs is an attractive prospect. The W3C standard CSS3 has the potential to make this vision come true. At present, however, our tests show that only few of the corresponding paged media features are currently supported by the leading rendering engines. For the time being, CSS3 can already be used to generate works of fiction with a very simple semantic structure, but is not yet suited for the production of complex non-fiction and text books. Somewhat better results can be achieved by using non-standard, proprietary features of the respective rendering engines, thus adjusting the layout settings. But this, of course, contradicts the principle of a standards-compliant, media-independent and flexible production.

Acknowledgments

The authors would like to thank Franziska Bröckl, Amrah Gadziev, Cameron Kilborn and Nadia Nikolaus for fruitful discussions and proof reading.

References

Antenna House, 2014. *AH Formatter XSL/CSS Properties List.* [online] Available at: <http://antennahouse.com/CSSInfo/property.html> [Accessed 23 May 2015].

Bosak, J., 1998. Media-independent publishing: Four myths about XML. *Computer,* 31(10), pp. 120–122.

Callas Software, 2015. *pdfChip.* [online] Available at: <https://www.callassoftware.com/en/products/pdfchip> [Accessed 23 November 2015].

Crawford, W., 1994. Pages from the desktop: Desktop publishing today. *Library Hi Tech*, 12(3), pp. 101–119.

Ertel, A. and Laborenz, K., 2014. *Responsive Webdesign: Anpassungsfähige Websites programmieren und gestalten.* Bonn: Galileo Press (Galileo Computing).

Fellenz, G. and Fischer, T., 2013. Printlayouts mit CSS3. In: *Proceedings of the tekom Jahrestagung 2013*, Wiesbaden. [online] Available at: <http://tobias-bloggt.de/2013/11/21/printlayouts-mit-css3/> [Accessed 2 March 2016].

Goldfarb, C.F., 1990. In: Y. Rubinsky, ed., *The SGML Handbook*. Oxford: Oxford University Press.

Götz, C., 2014. *CSS 3 zur Formatierung für XML-basierte Printlayouts.* Master thesis. Hochschule für Technik, Wirtschaft und Kultur Leipzig.

Götz, C. and Nikolaus, U., 2013. EPUB 3 on current e-readers – drawbacks and opportunities. In: N. Enlund and M. Lovreček, eds., *Advances in Printing and Media Technology, Vol. XL, Proceedings of the 40th International Research Conference of iarigai, Chemnitz, 2013*. Darmstadt: iarigai, pp. 327–336.

Graham, T., 2014. Formatting from XML. In: *Proceedings of the XML Prague Conference, February 14–16 2014*. [pdf] Prague: University of Economics, Prague. Available at: <http://archive.xmlprague.cz/2014/files/xmlprague-2014-proceedings.pdf> [Accessed 2 March 2016].

Harold, E.R. and Means, W.S., 2004. *XML in a Nutshell*. Sebastopol: O'Reilly Media, Inc.

Kelly, M., 2015. *XSL-FO Is Dead, CSS Paged Media Is Prime Suspect*. [online] Available at: <http://www.rockweb.co.uk/blog/2014/06/xsl-fo-is-dead,-css-paged-media-is-prime-suspect/> [Accessed 2 March 2016].

Kleinfeld, S., 2013. The case for authoring and producing books in (X)HTML5. In: *Proceedings of Balisage: The Markup Conference 2013, August 6–9 2013, Montréal*. [online] Available at: <http://balisage.net/Proceedings/vol10/print/Kleinfeld01/BalisageVol10-Kleinfeld01.html> [Accessed 2 March 2016].

Kleinfeld, S., ed., 2016. HTMLBook: Unofficial Draft 16 February 2016. [online] O'Reilly Madia, Inc. Available at: <http://oreillymedia.github.io/ HTMLBook/> [Accessed 2 March 2016].

Knuth, D.E., 1984. *The TeXbook*. Reading, Massachusetts: Addison-Wesley.

Lamport, L., 1994. *LaTeX: A document preparation system: User's guide and reference (2nd ed.)*. Reading, Massachusetts: Addison-Wesley.

Lie, H.W., ed., 2014. *CSS Figures: Living Standard*. [online] Available at: <http://figures.spec.whatwg.org/> [Accessed 28 June 2014].

Lie, H.W., ed., 2015. *CSS Books: Living Standard*. [online] Available at: <http://books.spec.whatwg.org/> [Accessed 13 May 2015].

Lie, H.W. and Day, M., 2005. *Printing XML: Why CSS is Better than XSL*. [online] Available at: <http://www.xml.com/pub/a/2005/01/19/print.html> [Accessed 2 March 2016].

McKesson, N., 2012. *Building Books with CSS3*. [online] Available at: <http://alistapart.com/article/building-books-with-css3> [Accessed 2 March 2016].

Meyer, E.A., 2012. *CSS and Documents*. [pdf] Sebastopol: O'Reilly Media, Inc. Available at: < http://uc.irpdf.com/uploads/part1/CSS%20and%20Documents.pdf> [Accessed 2 March 2016].

Nikolaus, U., 2010. Accessibility and multi-channel publishing: Different aims, similar solutions? In: N. Enlund and M. Lovreček, eds., *Advances in Printing and Media Technology, Vol. XXXVII, Proceedings of the 37th International Research Conference of iarigai, 2010, Montréal*. Darmstadt: iarigai, pp. 13–22.

Ott, T., 2014. *Crossmediales Publizieren im Verlag*. Berlin/Boston: Walter de Gruyter GmbH.

Quin, L., 2013. *Re: [xsl] xsl 2.0?* [online] Available at: <http://www.biglist.com/lists/lists. mulberrytech.com/xsl-list/archives/201311/msg00014.html> [Accessed 2 March 2016].

Quin, L., 2014. Publishing in style with XML: Or, Why it's not XSL-FO. In: *Proceedings of the XML Prague Conference, February 14–16 2014*. [pdf] Prague: University of Economics, Prague. Available at: <http://archive.xmlprague.cz/2014/files/xmlprague-2014-proceedings.pdf> [Accessed 2 March 2016].

RealObjects GmbH, 2015. *PDFreactor*. [online] Available at: <http://www.pdfreactor.com/> [Accessed 2 Mach 2016].

Sampson, G., 1990. *Writing Systems: A Linguistic Introduction*. Stanford: Stanford University Press.

W3C, 2011a. *CSS Color Module Level 3*. [online] Available at: <http://www.w3.org/TR/2011/REC-css3-color-20110607> [Accessed 12 June 2014].

W3C, 2011b. *CSS Multi-column Layout Module*. [online] Available at: <http://www.w3.org/TR/2011/CR-css3-multicol-20110412/> [Accessed 11 June 2014].

W3C, 2011c. *Selectors Level 3*. [online] Available at: <http://www.w3.org/TR/2011/REC-css3-selectors-20110929/> [Accessed 18 March 2014].

W3C, 2012. *CSS Image Values and Replaced Content Module Level 3*. [online] Available at: <http://www.w3.org/TR/2012/CR-css3-images-20120417/> [Accessed 12 June 2012].

W3C, 2013a. *CSS Fonts Module Level 3*. [online] Available at: <http://www.w3.org/TR/2013/CR-css-fonts-3-20131003/> [Accessed 22 May 2015].

W3C, 2013b. *CSS Page Floats*. [online] Available at: <http://www.w3.org/TR/2013/ED-css3-gcpm-20130924/> [Accessed 30 June 2014].

W3C, 2013c. *CSS Paged Media Module Level 3*. [online] Available at: <http://www.w3.org/TR/2013/WD-css3-page-20130314/> [Accessed 18 March 2014].

W3C, 2013d. *CSS Text Module Level 3*. [online] Available at: <http://www.w3.org/TR/2013/WD-css-text-3-20131010/> [Accessed 9 June 2014].

W3C, 2014a. *CSS Backgrounds and Borders Module Level 3*. [online] Available at: <http://www.w3.org/TR/2014/WD-css3-background-20140204/> [Accessed 20 May 2015].

W3C, 2014b. *CSS Color Module Level 4*. [online] Available at: <http://dev.w3.org/csswg/css-color/> [Accessed 12 June 2014].

W3C, 2014c. *CSS Fragmentation Module Level 3*. [online] Available at: <http://www.w3.org/TR/2014/WD-css3-break-20140116/> [Accessed 29 May 2014].

W3C, 2014d. *CSS Generated Content for Paged Media Module*. [online] Available at: <http://www.w3.org/TR/2014/WD-css-gcpm-3-20140513/> [Accessed 26 May 2015].

W3C, 2014e. *CSS Line Grid Module Level 1*. [online] Available at: <http://www.w3.org/TR/2014/WD-css-line-grid-1-20140403/> [Accessed 12 June 2014].

YesLogic, 2015a. *Prince: CSS Properties*. [online] Available at: <http://www.princexml.com/doc/9.0/properties/> [Accessed 2 March 2016].

YesLogic, 2015b. *Release Notes for Prince 10*. [online] Available at: <http://www.princexml.com/releases/10/> [Accessed 2 March 2016].