Position Paper

by Liam Quin, Delightful Computing, Milford, Ontario, Canada.

First Things First: Support pages

The single most important thing to change, besides which everything else pales into insignificance, is to get browser developers to understand *pagination is useful*. After that, it might be helpful to have an acknowledgement that *print is not dead* (even if it doesn't wiggle about).

It should be possible to use CSS to replicate the way Web browsers print Web documents today, including margins, headings, footers (e.g. put the URL of the current document in the page header, or, if it doesn't fit, truncate it in a way that fits with interior ellipses).

It should be possible to have paginated overflow for any element anywhere in the document. As with long horizontal scroll in the middle of vertical text, for print output a different treatment is needed for nested paginated boxes, but that doesn't stop them being useful in a browser. Exploring paginated boxes within Web pages would be useful for Apps as well as helping to get the paged media specs tied down more precisely and with better interoperability.

Needs By Industry

To get people in the room, CSS for paged media needs to be seen to be listening to and catering to people with business needs for solutions. For publishing, the main segments are roughly, ordered from largest to smallest although there are disagreements in the industry):

- Newspapers, News (some representation today but no W3C focus)
- Technical Publications (again no clear focus but stronger representation)
- Journals, Higher Education, Reference Works (very limited participation but strong needs)
- Trade Publishing (relatively few needs and strong representation, yet little real visible progress
- Financial, Reports, Business (we're on a different planet)
- K12 Education

It would be useful to address industry-specific needs, but a holistic view would reduce the chance of a solution to one industry segment hurting another. For example, business needs include selecting table cells containing negative numbers, and producing running totals at the foot of each page; technical publications and reference works need better index and footnote control; journals

also need more control over floated figures and tables. Improvements to footnotes, though, should not impede academic publishing where multiple streams of footnotes is not an uncommon need, often combined with marginalia (consider a Study Bible or a critical edition).

Fix the things users can't do with CSS for Print

Parallel texts that need to be stacked but tied in position: translations, marginalia.

Index collapsing (see pages 12-14, 15, 15, 15, 16, 19, 21-21 is unacceptable, should be, See pages 12-16, 19, 21). This applies also to cross-references.

Variant text: See figure 6 (on this page, above) and figure 7 (next page); see figure 3 (opposite page). Note that the varying text may contain markup so the variants should be able to be supplied in the document. The current element() function partly works for this, but element() requires a separate version of the document fromdisplay, usually, because you have to duplicate content: it moves, rather thancopies. For a while there was a WhatWG fork of GCPM that extended this function. Maybe an extension of CSS Conditionals that allows reference to counters?

Spreads, Finishing

A *spread* is a set of one or more pages that are treated as a unit. For example, magazines often have artwork that crosses the fold between pages. In marketing and packaging, or sometimes in technical documentation or printed books, you might get folded pages that, when unfolded, show six or even more separate "pages"—this is rarely done in bound books because of cost and equipment needed, although maps sometimes work like this in travel books. In packaging and marketing it's much more common to have more than two pages participate in a spread.

In printing, outside the printed region of the page are *trim* and then *bleed* borders: the bleed border marks the outermost extent of ink, and within that the trim border marks where the page will be cut. The bleed border is usually slightly larger than the trim border to allow for mechanical inaccuracy in cutting, when the designer wants the ink to go all the way to the edge of the paper.

There are also standard ways to say how you would like your printed job finished: single-sided, double-sided, bound on which edge, stacked on pallets ready for a fork-lift truck when finished, and more; there is a standard language for this interchanged in the CIF4 Job Description Format (JDF) for this.

Probably HTML and/or CSS should allow a association with an external JDF file, and CSS should understand bleed and trim.

Page Borders and Border Images

If you look at elaborate Victorian pages you find the borders have elaborate swirls that are extended by duplicating pieces of type; the extra type is added in places that correspond to the named CSS margin areas so we are good there, except then we can't have running headers—and the border only appears in print, not on the screen.

One approach might be to add the sixteen named areas to every box; another might be to support a 17-argument version of border-image (the extra argument being for the middle); another might be to allow background-image on CSS grid areas even when they have no content.

Note that central border markers are often placed slightly above the actual middle to counteract an optical illusion. Making the central marker asymmetrical can mitigate this limitation, by including one repeated border piece below the fixed image.

Another possibility for page an region borders might be to consider nested pages, pages within other content, which would also be helpful in a Web interface where a single *div* could be paginated with surrounding context unchanged. At one point I think Opera implemented something like this, but the prototype I saw had navigation/context issues. Perhaps paginated with a scrollbar would help.

Hyphenation and line-breaking

- Want better than first-fit line-breaking but without the disadvantages of Knuth-Plass (TeX-style) for operation without review by the author; I suggested an *n*-line optimization some years ago. In addition, a hint should be added to say an element might become content-editable and should be laid out with a stable algorithm.
- Would like semi-justification (justify unless it creates excessive word space size, in which case left-justify the line with preferred – not minimum – line length); this is sometimes also called partial justification.
- Want standard way to interchange hyphenation exception dictionaries.
- Need ability to distinguish a line-break at a hyphen from an inserted hyphen (e.g. in a hyphenated URL)
- Need control over hyphenating when it creates a really short last line in a paragraph.

Copy-Fitting

XSL-FO 2 provided a way to give a list of properties to adjust in order to get text to fit in a given space. This might be better done with Houdini and JavaScript in a Web context, but then needs extensions for off-line implementations.

Position and Flow

Without concepts of *flow* and *regions*, it's hard in CSS to achieve a number of layout styles common in print.

- When a figure or table (say) spans both columns of text, do you read both columns of text *before* (in the block direction) the figure (strong spanning), or do you go to the end of the page in the first column and then come back and read all of the second column (weak spanning)? Different sorts of publications need different styles here.
- A pull-quote or an aside that appears in the middle of a single-column article, or perhaps an oval vignetted portrait, and the text flows around the vignette, is hard to do today.
- Although CSS Grid alleviates a lot of layouts problems (thank you!), if you're not using XSLT beforehand then you're going to need sub-grids often.

XSL-FO implementations may support exporting and importing an Area Tree representation that allows for a great many tweaks to be done externally; perhaps CSS OM is a move in this direction.

There should also be a way for JavaScript to discover page boundaries/fragmentation points in the text. E.g. consider someone writing a paged interface in which the last line of the previous page is repeated at the top of the next page, perhaps with a line under it and a slight space, to help the reader with the transition (much as pre-Victorian books included a *catchword* at the end of each page).

CSS *versus* XSL-FO

The XSL-FO model is that you first transform the input with XSLT (this was the original purpose of XSLT) and then format it by attaching CSS properties to a computed area tree using FO. This means that XSLT could be used to generate text, numbers, and so on e.g.with count() and format-number(). It also facilitates selecting by text content: *make negative numbers be red and have (parens) round them*.

Not all CSS print solutions today have JavaScript; using XSLT first is common in practice. And you can't easily fetch an attribute from another element in CSS – e.g. to get the parent element's attribute value—nor find properties (make the width of this box be the same as the width of…).

Custom functions in CSS are likely to allow experimentation in this area in the near future.

Conclusion

Overall, the biggest need seems to be to make paginated content of interest to browser makers. After that, maybe there might be more resources available for work on making CSS for print and paged media (including PDF) more usable. Connecting with major industry segments and working

on their needs (did you know Arabic has “drop words”?) may be a way to do that, but the CSS Working Group is already busy doing other excellent work; if browser vendors do not become interested in paged media, it might be more effective for Paged Media and GCPM to be carried on in a separate venue with strong liaison.

Liam Quin, https://www.delightfulcomputing.com/

February 2020