# PRINT CSS AT A MAJOR TRADE PUBLISHER

*Dave Cramer, Hachette Livre*
*January 31, 2020*

For almost ten years, Hachette Book Group, the fourth-largest trade publisher in the U.S., has been creating both print and digital books with HTML and CSS.

We've published several thousand different titles. We've sold perhaps a hundred million print books, and untold numbers of ebooks. Doing this work ourselves in Manhattan saves large amounts of money over offshore conversion. It's easy and natural to create digital-only titles, or even to add a print-on-demand edition to a digital original. By leaving the page layout metaphor behind, and treating print and digital as aspects of the same content, we work faster, better, and cheaper.

Before I joined Hachette, I was working for a small typesetting company in Vermont, living the usual life of people who work with XML: dealing with urgent requests for changes to our DocBook-based schema, and despairing at the "creative" markup of my offshore colleagues. Who among us hasn't seen `<para role="title">`? We did typesetting for most of the big U.S. book publishers, including Hachette. They had what was then considered a state-of-the-art "XML-First" workflow, used for novels and narrative non-fiction.

Manuscripts in Microsoft Word were converted to DocBook, some presentation-oriented markup was added in XSLT, and the result was imported into InDesign. When typesetting was finished, a PDF from InDesign was sent to the printer, and the XML was exported, reprocessed, and used as the starting point for ebook production.

It worked well enough. Print was never a problem. Hachette's ebooks were pretty good, because XML was a far better starting point than page-layout files or (the horror!) PDF. Every once in a while a stray control code from InDesign would end up in an ebook. But we weren't reaping the promised benefits of XML: the separation of content and style, and true single-source publishing. In trade publishing, a book might have a half-dozen different print editions as well as digital editions: hardcover, large print, trade paperback, mass market paperback, international mass market, and so on. This is where the whole page-layout model breaks down. In InDesign, the content and the presentation are inseparable—two different presentations of the same content means having two independent files. Even embedding XML in InDesign doesn't change anything; you still have to maintain that content in multiple files. We needed something better.

What we found came from a tiny company called Infogrid Pacific, founded by an expat New Zealander who'd lived in India for twenty years. Richard Pipe had worked in the trenches doing document conversion, typesetting, and XML, and had seen everything that could possibly go wrong. "Digital Publisher" was his response; a web application for multi-format

publishing. We call it Dante, being fond of both the typeface and the author.

Dante brought three transformative things to Hachette. First, we used HTML as our XML vocabulary. Second, we started creating print PDFs directly from HTML and CSS. Third, we insourced book production, from offshore vendors to actual employees in New York.

Dante is a web application written in Django. Although most of the program logic is in Python, most content manipulation is done with XSL. Word manuscripts are converted to HTML by using LibreOffice to export as XHTML, and then a sequence of XSL stylesheets convert the raw output to the HTML vocabulary (see appendix). The HTML book content lives in a database, split into chapters. There are design, editing and content processing tools. Critically, we can create any number of editions for a book, which essentially means a new stylesheet and new metadata. Of course we can output to PDF, various flavors of EPUB, Word—last time I checked, there were twenty-three output formats.

As an XML practitioner, the biggest shock was using HTML. At first I thought it was crazy. But it fits our content much more naturally than the DocBook customizations. It works beautifully with the CSS cascade. The ability to customize on the fly, without going through tooling changes or DTDs, is essential for books. My theory is that nearly every book has some unique element that didn't occur in the previous thousand books.

The entire HTML ecosystem is a tremendous benefit.

Experimenting and learning requires only a text editor and a browser. Years ago, I had to troubleshoot a CALS table full of row and column spans. The only way to see what was going on was to run it through the DocBook stylesheets to get a visual rendering. Compare that to the convenience and power of browser dev tools.

We use PrinceXML to create PDFs for print. Prince is a complete implementation of the web stack—HTML, CSS, SVG, MathML—but outputs to PDF rather than screens. Interestingly, it's written in a functional programming language called Mercury, invented at the University of Melbourne. The output is high quality—there's very good support for OpenType font features, and it uses the TeX justification algorithm. But it took us a while to get there. We had to find better hyphenation dictionaries and cope with hyphenation exceptions, and it took months to find an ellipse character that satisfied our editors.

The bad news is that we do a lot of manual tweaking to fix widows, orphans and bad breaks. This is done by applying word- and letter-spacing to individual paragraphs, using a separate CSS file. Some fixes require content changes, such as inserting discretionary hyphens or zero-width spaces. Since these content changes are design-specific, they are stored separately in the database, so they do not influence other editions. We are working on automating more of this page composition process by analyzing the resulting PDF—we don't yet have access to Prince's internal layout model.

Dante is still a work in progress. The design tools are awkward. The user interface can be less than intuitive. But, aside

from the business metrics of money saved and books published, I'm struck by the loyalty and affection that our users have for this eccentric little tool. Even our most die-hard InDesign user now much prefers Dante. We believe that single-source content in HTML, with styling expressed with CSS, is the best way to make our books.

## *Print CSS: The Frustrations*

We have accomplished a lot with Prince. But many aspects of print CSS are too complex or too verbose, and of course there are many features required for good page layout that no renderer yet supports.

1. Page margin boxes. We have hundreds of lines of code for these. There is a lot of duplication and little flexibility. Mixing roman and italic becomes hugely complicated.
2. Text justification. We have problems with lines of justified text with no break opportunities, such as long URLs. Our proofreaders often specify a line break, which the software cannot achieve. Hyphenation exception dictionaries are not standardized.
3. Vertical rhythm and baseline grids.
4. Initial letters.
5. Control over repeating table column heads, captions, etc.
6. Continued lines
7. Flowing text around named pages
8. Sidenotes
9. Anchored text.

## *Print CSS: The Path Forward*

Hachette joined the W3C in mid-2013. I was offered the opportunity to join the CSS Working Group, and could not believe my luck. Given our experiences making books with CSS, I wanted to help on the print-related specs, especially GCPM (Generated Content for Paged Media). Unbeknownst to myself, at almost the very moment I joined the CSSWG, the editor of GCPM and co-inventor of CSS, Häkon Wium Lie, quit the working group in dramatic fashion and attempted to take CGPM to the Whatwg. Suddenly I found myself the editor of GCPM. I have rewritten the spec entirely, tried to make it a little more rigorous, and have written a few tests. But some of the remaining work is truly beyond my abilities, especially since I do not write browser code. Most importantly, we need to specify how footnotes work. And writing CSS Layout specs is not easy.

But I absolutely believe that we should try to get GCPM to REC, based on implementations from the print renderers.