Revisiting paged media stylesheets for the web

One thing that the web is sorely lacking is the ability to create print-ready content from our web pages. CSS provides specifications for paged media but the support in browsers leaves a lot to be desired, forcing people into tools that accomplish the goals of creating high quality print content.

In many instances I will cut pieces of the code where they are not relevant. You can find the full stylesheet along with the resulting PDF from the Github Repo

One thing we need to remembebr is that for this particular project we're doing everything in the command line so download speed is not as big a concern as if we were doing this online at the same time as trying to serve our regular content.

Getting started

The first thing we do is to load the fonts using a simplified `@font-face` syntax. Rather than loading all the font formats like we would for a regular web page, we only load WOFF (compressed with Zopfli) to make our lives easier.

No, we cannot use Variable Fonts with Paged Media Processors.

In the next step we'll define some global parameters that will apply to all content.

We use the `html` element to define the global font for the document.

The h1 element is interesting. We capture the content of the element (the value) to use later as the running header for our different types of content.

Finally we define the widows and orphans for the entire document.

Widow
A paragraph-ending line that falls at the beginning of the following page or column, thus separated from the rest of the text.

Orphan
A paragraph-opening line that appears by itself at the bottom of a page or

column, thus separated from the rest of the text.

Be mindful that setting the values for widows and orphans to high can generate large blocks of empty space in your pages.

```
html {
  font-family: 'PT 'PT Serif'rif;
}

h1 {
  string-set: doctitle content();
  line-height: 1.3;
}

p {
  widows: 4;
  orphans: 4;
}
```

Next we define the global page for the document. This is what all the other pages will inherit from so we save ourselves from having to retype blocks of CSS over and over.

We define the size of the printed page to be American Letter (8.5 by 11 inches) with a 1 inch margin all around.

The margin attribute takes one to four values and follows the same rules as regular CSS:

- When **one** value is specified, it applies the same margin to **all four sides**
- When **one** value is specified, it applies the same margin to **all four sides**
- When **two** values are specified, the first margin applies to the **top and bottom**, the second to the **left and right**
- When **three** values are specified, the first margin applies to the **top**, the second to the **left and right**, the third to the **bottom**
- When **four** values are specified, the margins apply to the **top**, **right**, **bottom**, and **left** in that order (clockwise)

Next we set what we want to put in the top right corner of the document. We indicate that we want to pull the `doctitle` string from h1 element for the

corresponding section and that we want it to be 9 points (where 1pt equals $^{1}/_{72}$ of an inch)

Finally we set footnote attributes that we want to carry throughout the document.

```
// DEFINE THE DEFAULT PAGE */
@page {
  size: 8.5in 11in;
  margin: 1in;
  @top-right {
    content: string(doctitle);
    font-size: 9pt;
  }
  @footnote {
    counter-increment: footnote;
    float: bottom;
    column-span: all;
    height: auto;
  }
}
```

We use `data-` attributes to indicate what part of our book each element corresponds to. We avoid name collisions between stylesheets where, for the same specificity, the last rule wins.

We use CMYK colors rather than RGB(a) or HSL. This is not a requirement but I thought it was cool.

Once again we use points (pt) to define the default font size of the document.

```
body[data-type='book'] {
  color: cmyk(0%, 0%, 100%, 100%);
  hyphens: auto;
  font-size: 14pt;
}
```

We will rely extensively on counters so we need to define them and reset them the

first time.

The rules below saay that if the first child is a part, a chapter or an appendix to reset the counters for appendix, chapter, figures and tables.

However if there's a chapter with a sibling chapter, we don't want to reset any counters.

```
body[data-type='book'] > div[data-type='part'][data-type='part']:first-of
bo[data-type='chapter']:first-of-type,
body[data-type='book'] > section[data-type='appendix']:first-of-type {
  counter-reset: chapter;
  counter-reset: appendix;
  counter-reset: figure;
  counter-reset: table;
}

body[data-type='book'] > section[data-type='chapter'] + div[data-type='pa
  counter-reset: none;
}
```

We associate elements to pages that we'll define later. It is at this stage where we define page breaks, counter increeases and other elements that are specific to some types of pages and not others.

The code is not DRY and I'm ok with that. I've traded ease of reading and debuging for brevity.

```
// Title Page */
section[data-type='titlepage'] {
  page: titlepage;
  page-break-before: always;
  page-break-after: always;
}

// Copyright page */
section[data-type='copyright'] {
  page: copyright;
```

```
  page-break-before: always;
  page-break-after: always;
}

// Dedication */
section[data-type='dedication'] {
  page: dedication;
  page-break-before: always;
  page-break-after: always;
}

// Foreword */
section[data-type='foreword'] {
  page: foreword;
  page-break-before: always;
  page-break-after: always;
}

// Preface*/
section[data-type='preface'] {
  page: preface;
  page-break-before: always;
  page-break-after: always;
}

// Part */
div[data-type='part'] {
  page: part;
  page-break-before: always;
  page-break-after: always;
}

// Chapter */
section[data-type='chapter'] {
  counter-increment: chapter;
  page: chapter;
  page-break-before: always;
  page-break-after: always;
```

```
}

// Appendix */
section[data-type='appendix'] {
  counter-increment: appendix;
  page: appendix;
  page-break-before: always;
  page-break-after: always;
}

// Glossary */
section[data-type='glossary'] {
  page: glossary;
  page-break-before: always;
  page-break-after: always;
}

// Bibliography */
section[data-type='bibliography'] {
  page: bibliography;
  page-break-before: always;
  page-break-after: always;
}

// Index */
section[data-type='index'] {
  page: index;
  page-break-before: always;
  page-break-after: always;
}

// Colophon */
section[data-type='colophon'] {
  page: colophon;
  page-break-before: always;
  page-break-after: always;
}
```

The table of contents creates a leader for each item in the table of contents (a line of dots) pointing towards the page number at the far right of the entry.

```
// TOC */
section[data-type='toc'] {
  page: toc;
  page-break-before: always;
  page-break-after: always;
}

// Leader for toc page */
section[data-type='toc'] nav ol li a:after {
  content: leader('.') target-counter(attr(href), page);
}
```

Now that we've done the association we can defined the different pages. Because we're working with two-sided pages we have to define the items for both the right and left side pages.

All the front matter pages use lower-case roman page numbers while the rest of the content uses arabic numbers for page numbering.

```
@page toc:right {
  @bottom-right-corner {
    content: counter(page, lower-roman);
  }
  @bottom-left-corner {
    content: normal;
  }
}

@page toc:left {
  @bottom-left-corner {
    content: counter(page, lower-roman);
  }
  @bottom-right-corner {
    content: normal;
  }
```

```
}

@page chapter {
  @bottom-center {
    vertical-align: middle;
    text-align: center;
  }
}

@page chapter:right {
  @bottom-right-corner {
    content: counter(page);
  }
  @bottom-left-corner {
    content: normal;
  }
}

@page chapter:left {
  @bottom-left-corner {
    content: counter(page);
  }
  @bottom-right-corner {
    content: normal;
  }
}
```

The next block covers footnotes. These are PrinceXML specific extensions covered in their footnotes documentation

```
// Footnotes */
span.footnote {
  float: footnote;
}

::footnote-marker {
  content: counter(footnote);
```

```
  list-style-position: inside;
}

::footnote-marker::after {
  content: '. ';
}

'. '::footnote-call {
  content: counter(footnote);
  vertical-align: super;
  font-size: 65%;
}
```

Cross References are also possible using generated contnt and Prince-specific extensions.

We can target any counter available on the document and, with it, creaate interesting cross references.

```
// XReferences */
a.xref[href][href]::after {
  content: ' [See page ' target-counter(attr(href), page) ']';
}
```

One cool thing that AntennaHouse and Prince can do is generate PDF bookmarks from the headings in your document.

We are generating bookmarks for chapter, apendix glossary, bibliography and index sections.

h1 through h3 are open and will display the content of the h1 element as indicated.

```
section[data-type='chapter'] h1,
section[data-type='apendix'] h1,
section[data-type='glossary'] h1,
section[data-type='bibliography'] h1,
```

```
section[data-type='index'] h1 {
  -ah-bookmark-level: 1;
  -ah-bookmark-state: open;
  -ah-bookmark-label: content();
  prince-bookmark-level: 1;
  prince-bookmark-state: open;
  prince-bookmark-label: content();
}
```

h4 through h6 do not need to be open mostly because we want to use the bookmarks as navigation references so we just ass the bookmark without a state or a label.

```
section[data-type='chapter'] h4,
section[data-type='apendix'] h4,
section[data-type='glossary'] h4,
section[data-type='bibliography'] h4,
section[data-type='index'] h4 {
  -ah-bookmark-level: 4;
  prince-bookmark-level: 4;
}
```

Chapters and Apendices require some extra work. Because we usually have more than one chapter and can have multiple apendices, we are using counters and we want to indent all paragraphs after the first one we take some extra work to make sure it works correctly.

> For all h1 elements inside chapter sections we want to do three things:
>
> - Capture the text of the element
> - Reset the figure counter
> - Reset the tabble counter

Using the `:before` pseudo element we add a string containing the word Chapter and the current value of the chapter counter.

> We repeat the same process for the `apendix` section or sections.

```
section[data-type='chapter'] h1 {
  string-set: doctitle content();
  counter-reset: figure;
  counter-reset: table;
}

section[data-type='chapter'] h1:before {
  content: 'Chapter ' counter(chapter) '. ';
}

'Chapter 'section[data-type='chapter'] p:not(:first-of-type) {
  text-indent: 0.5in;
}

section[data-type='apendix'] h1 {
  string-set: doctitle content();
  counter-reset: figure;
  counter-reset: table;
}

section[data-type='appendix'] h1:before {
  content: 'Appendix ' counter(appendix, upper-alpha) ':  ';
}
```

Putting it all together

I build the files from the command line using iTerm in MacOS. The instructions should also work in Linux and Windows using WSL.

> Download Prince for your operating system and install it.
>
> I use (Dart) SASS to generate the stylesheets.
>
> Once it's installed run the following commands:

- **rm -rf book2.pdf** to remove the pdf file. It will return without a message if the file doesn't exist
- **sass sass:css** converts files in the sass directory to the corresponding file in

the css directory
- **prince index.html -s css/paged-media.css -o book2.pdf** runs princeXML on index.html using the css/paged-media.css stylesheet and producing book2.pdf as the final output

Looking Forward

The version of the stylesheet we've presented in this post is geared to wards books and other complex forms of printed content.

There is no reason why we couldn't simplify it to work on one or more articlles without having to carry the full book baggage.

There is a simplified article stylesheet in the Github Repo at `sass/article.scss` along with the resulting `article.pdf`.

Links and Resources

- Spec
    - CSS Paged Media
    - CSS Generated Content for Paged Media Module
- Tutorials
    - Building Books with CSS3
    - Creating Print CSS stylesheets
    - A Guide To The State Of Print Stylesheets In 2018
    - I totally forgot about print style sheets
    - Designing For Print With CSS
- Vendor Information
    - AntennaHouse
    - Prince XML User Guide