css2pdf@cloudformatter

Why Did we Develop This?

We tried to address many of the missing elements in some of the web page print solutions we found. In many solutions, they are merely converting the HTML fed to the browser, not the content from the browser DOM. We wanted to be able to process live content, that could vary based on user interaction.

We found most all solutions lacking in SVG support. You cannot actually inject the SVG into the print document, you can only insert an image of the SVG in raster form. We wanted high resolution vector-based SVG content. We wanted to leverage formatting technology far superior than a "PDF" writer and/or browser which is not designed as a print composition engine.

We also found many situations where we wanted the flexibility of just having some XML content in the page for special formatting. You can easily extend this solution to format any XML markup in your page, not only HTML.

Throughout the demos you will also see print buttons like this Out PDF located on headings. This is a demonstration of using the code to print a single <div>. In others demos, there are also checkboxes next to these headings, you can select several checkboxes and use the "Print Selected!" button to generate a document with multiple sections, one for each of the checked divs. These are all demonstrations of the power of the Javascript library and XEPOnline backend for generating PDFs.

How it Works

The XEPOnline javascript library extracts the content of a named <div> element in the HTML page. It processes that <div> and embeds all css-based styling into the HTML. You can use the library to generate a PDF of any <div>, including those generated with dynamic content as it is processing the browser DOM of the HTML at the time execution. In other words, this is not some canned HTML to print, it is the current HTML to print.

The whole solution is extensible and leverages XSL FO technology for print file generation, removing any of the complexity while maintaining ease of styling via css. The extensibility allows you to customize output in many ways, to extend and expand upon a simple File->Print operation. Because it uses XSL FO at the core, one can certainly do more than generate PDF. You could generate PostScript, AFP or XPS print files if desired.

Because HTML is **not XML**, the solution does some additional processing to ensure a well-formed XML document is exported. That well-formed XML document is generated and sent to the XEPOnline formatter via REST with a reference to a specific XSL stylesheet for processing the HTML tag content to create XSL FO.

XEPOnline accepts the REST request, attempts to format the document with RenderX XEP and returns the result. There are several options to return data from initiating a download to base64 encoding the result and inserting it into the document.

Currently Supported Elements

The XSL which processes incoming (X)HTML to XSL FO is written in XSL version 1.0. Most modern web environments make use of very few tags and control appearance through headings, div's, img's and span's with css styles. The template not only supports these core elements but also has support for many legacy structures. The following HTML structures are currently supported:

- **Block Elements** - <div>, <h1> through <h6>, <p>, <blockquote>, <pre>, <address>, <main>, <figure>, <figcaption>, <br>
- **Inline Elements** - <span>, <i>, <b>, <sup>, <sub>, <font>, <code>, <em>, <small>, <strike>, <strong>, <u>, <q>, <dfn>, <abbr>, <code>, <var>, <samp>, <kbd>, <mark>
- **Special Handling** - <a>, <img>, <svg>, <header>, <footer>
- **Tables** - Table support includes <thead>, <tbody>, <tfoot> elements as well as column and row spanning

- **Lists** - Partial support for both <ul>, <ol>

The new HTML5 tags <section> and <article> are only supported as block elements, not seperate page generating elements at this time. One <header> and one <footer> element are allowed inside the printable <div> and are used for printed document header and footer.

What's Coming?

We've done the implementation to support @media print. Would love to implement page-templates (first, last, left/right). Only Chrome supports @media print @page directives though. Guess that will need to wait a bit.

Next a little clean-up on the conversion to XSL FO, there are some issues pointed out in this document. There is certainly more work here to do, we probably did not think of everything.

We've done a start at form fields and plan to implement generating both a fillable form and a filled form.