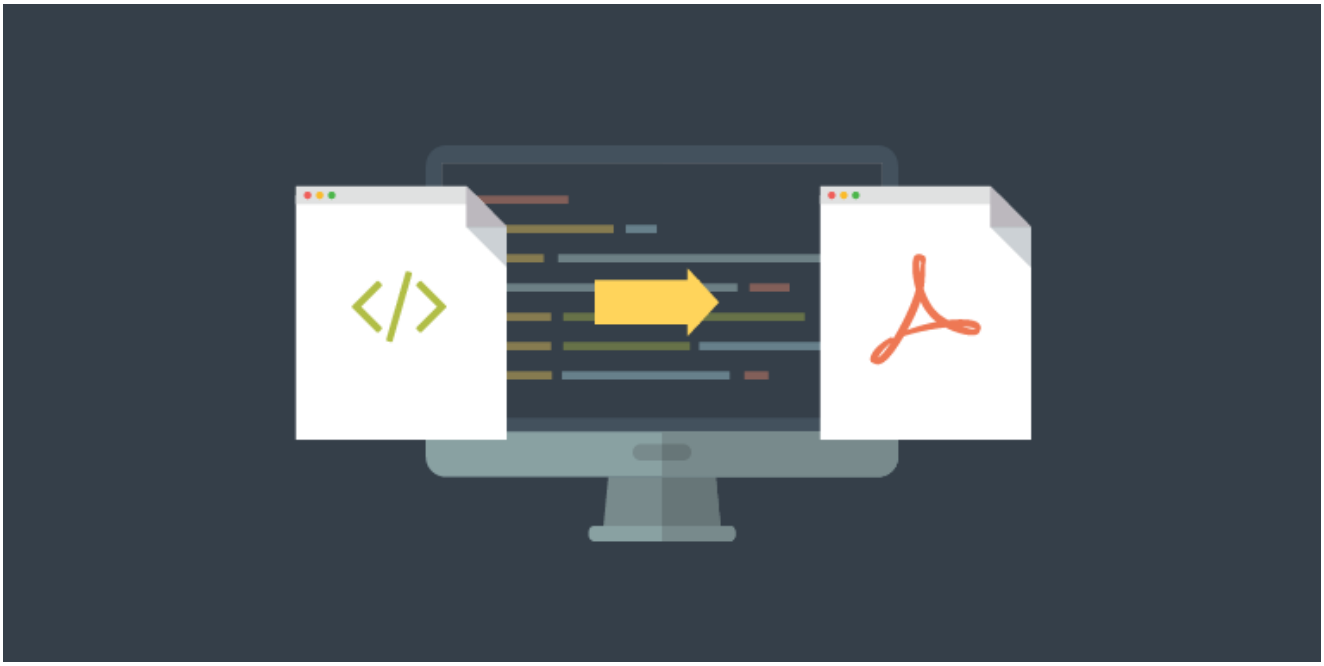
Rendering HTML into PDF using WKHTMLTOPDF

ind. industrialagency.ca/blog/rendering-html-into-pdf-using-wkhtmltodf



Rob Ferguson
Mar 30

Introduction

The PDF filetype is one of the most popular formats for sharing and providing content on both the personal and professional sides of computing. It has many advantages such as being:

- **self-contained** - able to share cross-platform with a single file while maintaining desired look and feel
- **compact** - able to compress and share large documents
- **securable** - using watermarks, passwords or encryption
- **common** - this format has become ubiquitous making it easy to share content.

There are a variety of reasons why someone would want a copy of a webpage in the form of a PDF. A few that you may be familiar with already are:

- printable receipts from online purchases
- generating invoices from online tools
- saving a hard copy "for your records" from a bank transaction or goverment website

- printable versions of online content for offline viewing

Generating PDF versions of existing web pages or exporting them such that they are styled similarly to web pages has often been a difficult task. In the past we've set this up for clients wanting a "save to pdf" version of a page. Depending on the complexity of the page in question, this has been achievable. There are plenty of libraries available to get this task done, especially in the PHP world. This sort of task isn't something we often do though.

The workflow usually starts by knowing you need to export an existing page to PDF, or create one based on an existing page. Now comes the time to pick a library to get this done. After some google catch up on some of the available solutions, you now need to settle on one. Most of the websites hosting each product don't really describe or give examples of results or how to get them within their documentation (if there is any). Relying on stackoverflow posts can also be hit and miss as each answer is different in their opinion. One person may say to use one product as "it works for me", another person may swear by some other library.

The general consensus in the past suggested that exporting to PDF from the web is like dealing with print stylesheets. The expectations you have for the output can't be that of the modern browser but more simplified in it's features and construction. This is still sort of true today still. Producing desirable results can depend on

- system fonts available
- integrity of incoming HTML markup
- features needed such as floating images or long tables that span multiple pages
- other nice to haves like a cover page, repeating header/footer areas and a table of contents

Choosing the right tool

Knowing what is going into the PDF, you'll also have to rely on the quality of the tool's output you're using and whether it provides all the requisite features. This can make for a headache when trying to come up with a workable solution.

Based on feedback seen on the web for these tools, the feeling seemed the same most of the time. Below is a comment taken from a discussion about pdf generators on stack overflow.

> "Well if you want to find a perfect XHTML+CSS to PDF converter library, forget it. It's far from possible. Because it's just like finding a perfect browser (XHTML+CSS rendering engine). Do we have one? IE or FF?
>
> I have had some success with DOMPDF. The thing is that you have to modify your HTML+CSS code to go with the way the library is meant to work. Other than that, I have pretty good results."

Our experience has been similar with using something called DOMPDF. Based on the many different solutions that were tried, this was the best. A few clients were able to get what they needed from it in the past and so recently we tried using it again on another bigger project. Unfortunately this time it had been pushed to its limit and we were unable to continue using it in the existing development. One example of this happened when trying to output a long table that spanned across multiple pages. We noticed that the rowspans in the table were breaking once reaching a new page. It turned out that DOMPDF has some open issues surrounding this problem in GitHub and users were having varied success with trying to do the same. Having reached 90% of what we needed, there was still more to be desired in terms of matching output to what we were seeing in the browser. This prompted a new search for solving the same problem in a different way.

Traditionally when googling for an answer to this problem, at least in the PHP world, you get back a list comprised of tools including but not limited to the following (some which are defunct):

- tcpdf
- dompdf
- pdflib
- htmldoc
- mpdf
- fpdf

Thankfully we came across a way better solution by using a product called WKHTMLTOPDF. Here's how they describe it on their homepage:

> wkhtmltopdf and wkhtmltoimage are open source (LGPLv3) command line tools to render HTML into PDF and various image formats using the Qt WebKit rendering engine. These run entirely "headless" and do not require a display or display service.

wkhtmltopdf is also used within popular PDF generator tools like PHP's Snappy, Ruby's wicked pdf and pdfkit.

Basically it's a command line tool with binaries available for Windows, OS X and Linux flavors. Our needs were to run it on linux and only for generating PDF's. It can also generate images as advertised but the focus here is solely on PDF's. To get it running with

images you'll also need to have additional image software packages installed on your machine for it to work.

Getting started with WKHTMLTOPDF

You can just install this using your package manager, but it'll likely be using an older version. It's best to get it right off the wkhtmltopdf website.

Then, for Linux, install the dependent packages as mentioned on the above page.

install using apt

```
sudo apt-get install openssl build-essential libssl-dev libxrender-dev git-core libx11-dev libxext-dev libfontconfig1-dev libfreetype6-dev fontconfig -y
```

install using yum

```
yum install -y urw-fonts libXext libXrender fontconfig zlib freetype
```

Now you should be able to test it out.

```
./wkhtmltopdf http://google.com google.pdf
Loading pages (1/6)
Counting pages (2/6)
Resolving links (4/6)
Loading headers and footers (5/6)
Printing pages (6/6)
Done
```

The above will give you google's homepage. Further testing with more complex websites will show you that while doing a good job, it's still not the same as what you'd get in a real browser. The other thing I noticed, which probably won't be an issue when just printing text documents, is that this tool will generally give you the mobile view of the website you're capturing.

There is a setting that will expand the viewport size so you can get closer to a desktop view if needed:

```
--viewport-size 1280x1024 --orientation Landscape
```

but even then you're still using a landscaped document. Further reading in the issue queue at the GitHub repo for this project suggests what's happening:

> If you look at the documentation of the --viewport-size option, it states "Set viewport size if you have custom scrollbars or css attribute overflow to emulate window size" -- it does not say anything of the screen resolution.
>
> On Windows, the screen resolution appears to be the actual screen resolution. On Linux, because we have to patch QT to get it running in "headless" mode the screen size is hardcoded to 800x600. Even though screen resolution could be anything, the window (or viewport size) can be changed -- which is what the option controls.
>
> If you do want to change the screen resolution, you'll have to run it under xvfb and use the --use-xserver option.

All that being said you're probably not going to want to use this tool to specifically screen grab websites. But it's important to understand it's limitations in case your custom HTML is coming out strange maybe due to a screen size issue.

Adding a PHP wrapper

The last thing we'll mention is the use of a wrapper class around this CLI tool to make it easier to use in a PHP script. The one we've chosen is called "phpwkhtmltopdf"

It provides an API to control the binary instead of having to call it directly from the PHP script using the exec() function as an example. It also simplifies things so you don't have to keep track of a long command, but instead build out the options and then run it in a few lines. To get started with it is fairly simple since it's also a composer package:

```
composer require mikehaertl/phpwkhtmltopdf
```

then a simple page example would look like this:

```
use mikehaertl\wkhtmlto\Pdf;

// You can pass a filename, a HTML string or an URL to the constructor
$pdf = new Pdf('/path/to/page.html');

if (!$pdf->saveAs('/path/to/page.pdf')) {
   echo $pdf->getError();
}
```

There are many different options for wkhtmltopdf that can be used to build out the file. Below is an example we've used to build out a file containing

- a custom footer with page numbering
- a cover page
- custom page options for footer spacing, encoding and page delay

Note I'm also using the binary option to point at the downloaded file as I haven't put it into the system path or installed it via a package manager to get a similar setup.

```
$pdf = new Pdf;
 $globalOptions = array(
  'margin-bottom'    => 20,
 );
 $pdf->setOptions($globalOptions);
 $pdf->binary = 'lib/wkhtmltox/bin/wkhtmltopdf';
 $pageOptions = array(
  'javascript-delay' => 2000,
  'encoding'         => 'UTF-8',
  'footer-line',
  'footer-left'      => strtoupper($field_product_line),
  'footer-right'     => "[page]",
  'footer-font-size' => 10,
  'footer-spacing'   => 10
 );
 $pdf->addCover('<html>'.$cover_style.$cover.'</html>', $pageOptions);
 $pdf->addPage('<html>'.$output.'</html>', $pageOptions);

 if (!$pdf->send($filename)) {
  echo $pdf->getCommand();
  throw new Exception('Could not create PDF: '.$pdf->getError());
 }
```

That translates into the below command, as seen when calling the getCommand() method:

```
wkhtmltopdf --margin-bottom '20' cover '/tmp/tmp_wkhtmlto_pdf_M3Dv2c.html'
--javascript-delay '2000'
--encoding 'UTF-8'
--footer-line
--footer-left 'ISSUES IN EMERGING HEALTH TECHNOLOGIES'
--footer-right '[page]'
--footer-font-size '10'
--footer-spacing '10' '/tmp/tmp_wkhtmlto_pdf_gPCZJI.html'
```

There is another similar wrapper available called "Snappy" which also uses wkhtmltopdf. Folks in the Laravel world may know this one better as there is a wrapper for it available.

Conclusion

While there is no silver bullet for converting to print, knowing a tool like wkhtmltopdf exists is a real stepping stone in trying to solve the problem of achieving good results from PDF generation. For most cases, as long as we can generate HTML without it breaking across pages, and can have some common features like cover pages and basic styling, that

will get you most of the way there, if not completely using this tool. No, it doesn't do pixel perfection like when viewing complex layouts in a browser. At that point though it may be worth it to look at a dedicated publishing suite to accomplish that goal, separate from the website.