OpenHTMLToPDF Template Guide

Welcome

Welcome to the template guide for OpenHTMLToPDF. This guide assumes a basic knowledge of CSS and HTML. If you find any problems with this documentation or have a question please raise an issue at the project home.

Fonts

Notes on Fonts

- Embedded fonts must be TrueType (.ttf).
- Font styles (italic, normal, etc), weights (bold, normal, etc) and variants (small-caps, normal) are not emulated. You must embed a font file for each different combination you use in the document.
- If the correct style, weight or variant is not found, a closest match will be used.
- OpenType is not supported due to PDF-BOX not supporting it.
- Comma separated font fallback is supported. This can be useful for example with Arabic text mixed with numbers.

- If no glyph is found for a character in any of the specified fonts the behavior is as follows. Control character codes will be ignored, whitespace characters will be replaced with the space character and any other character will be replaced with the replacement character (# by default).
- PDF has built in fonts (serif, sans-serif, monospace). The fonts only support a basic Western European character set and it is now usually recommended that you embed fonts that you wish to use.
- Google Fonts is a good source of open-source fonts, especially the noto family.
- Text justification is not supported for embedded fonts, also due to a lack of PDF-BOX support. See 83 - UNICODE font justification support.
- An embedded font can also be used in inline SVG images.
- Relative font weights (bolder, etc) are not implemented.
- Symbol fonts such as FontAwesome can be used, but may need tweeks (such as removing `font-face` imports other than TrueType) to their associated CSS files to get working.
- Embedded fonts are subset by default. This is typically the correct behavior, except for fonts used with form controls. You can avoid subsetting by using the `-fs-font-subset: complete-font` property.
- The `font-family` property is inherited. Therefore, you have to be very careful to make sure that form controls do not inherit a subset font.

Font Example

This example shows using all four of the Noto Serif (by Google) font types:

```
<html>
<head>
<style>
@font-face {
  font-family: 'noto';
  /* Note: No trailing format tag is allowed. */
  src: url(fonts/NotoSerif-Regular.ttf);
  font-weight: normal;
  font-style: normal;
}
@font-face {
  font-family: 'noto';
  src: url(fonts/NotoSerif-Bold.ttf);
  font-weight: bold;
  font-style: normal;
}
@font-face {
  font-family: 'noto';
  src: url(fonts/NotoSerif-BoldItalic.ttf);
  font-weight: bold;
  font-style: italic;
}
@font-face {
  font-family: 'noto';
  src: url(fonts/NotoSerif-Italic.ttf);
  font-weight: normal;
  font-style: italic;
}
```

```
.noto {
  font-family: 'noto', serif;
}
.bold {
  font-weight: bold;
}
.italic {
  font-style: italic;
}
</style>
</head>
<body>
<p class="noto">Regular Noto Font</p>
<p class="noto bold">Bold Noto Font</p>
<p class="noto italic">Italic Noto Font</p>
<p class="noto italic bold">Bold Italic Noto Font</p>
</body>
</html>
```

RTL Support

Notes on RTL Support

- The project supports RTL and bi-directional text, if configured correctly.
- Direction (`ltr`, `rtl` or `auto`) should be set in the markup rather than using CSS.
- The project supports the `bdi` element.
- When RTL is enabled, the project attempts to use the correct presentational (beginning, middle or end) glyphs. Unfortunately, not all fonts have these characters which may result in unjoined text.
- Some success has been seen for Arabic (identical output compare to Chrome web browser) but no such success has yet been reported for Persian or Hebrew languages.
- OpenType support is really needed here.
- See the Showcase Document for an example of Arabic text and 9 - RTL for more information.

Page Support

Notes on Page Support

- The default page size is A4 portrait.
- Page margin boxes such as `@top-center` are supported.
- Named pages are supported. These are useful to put sepcific content on a particular page size and setup.

Page Support Examples

This example shows all pages set to letter size, landscape with a 10% margin and 1-based page numbering:

```
<html>
<head>
<style>
@page {
  margin: 10%;
  size: letter landscape;

  @top-center {
    font-family: sans-serif;
    font-weight: bold;
    font-size: 2em;
    content: "Page " counter(page);
  }
}
</style>
</head>
<body>
...
</body>
</html>
```

The following example introduces named pages. In this example, a named page is used to make sure that table content is rendered on landscape while everything else is rendered on portrait.

```
<html>
<head>
<style>
@page {
  size: A4 portrait;
}
@page tblpage {
  size: A4 landscape;
}
table {
  page: tblpage;
}
</style>
</head>
<body>
<p>On page 1 (portrait)</p>
<table><tr><td>On page 2 (landscape)</td></tr></table>
<p>On page 3 (portrait)</p>
</body>
</html>
```

Backgrounds

Notes on Backgrounds

- Multiple background images on the one element are not supported.
- Repeat background images, especially with small images, can result in very large and slow to render PDFs.
- `background-origin`, `background-clip` and `background-attachment` are not implemented.
- The background values that cen be set are color, image, size, position and repeat.
- The `background` shorthand property currently fails to set or reset the background size.
- PNG, GIF and JPEG are supported formats.

Borders

Notes on Borders

- Border radius is implemented however the `border-radius` property only takes one value for each corner. In the unlikely situation where you need different values for horizontal and vertical radii, you can use the longhand `border-*-*-radius` properties such as `border-top-left-radius`.
- Border images are not implemented.

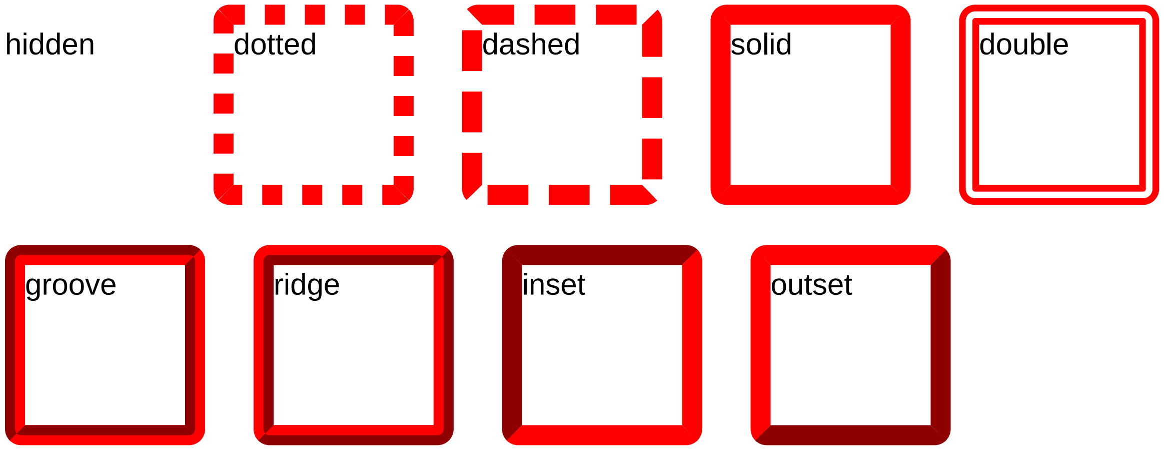
Borders Example

This example shows all border styles supported by the renderer, as well as setting the border radius:

```
<html>
<head>
<style>
.brdr {
  display: inline-block;
  width: 80px;
  height: 80px;
  margin: 10px;
  border-radius: 8px;
}
#o1 { border: 10px hidden red; }
#o2 { border: 10px dotted red; }
#o3 { border: 10px dashed red; }
#o4 { border: 10px solid red; }
#o5 { border: 10px double red; }
#o6 { border: 10px groove red; }
#o7 { border: 10px ridge red; }
#o8 { border: 10px inset red; }
#o9 { border: 10px outset red; }
</style>
<body>
<div class="brdr" id="o1">hidden</div>
<div class="brdr" id="o2">dotted</div>
<div class="brdr" id="o3">dashed</div>
```

```
<div class="brdr" id="o4">solid</div>
<div class="brdr" id="o5">double</div>
<div class="brdr" id="o6">groove</div>
<div class="brdr" id="o7">ridge</div>
<div class="brdr" id="o8">inset</div>
<div class="brdr" id="o9">outset</div>
</body>
</html>
```

The ouput of this example is shown below:



SVG Support

Notes on SVG Support

- The project supports inline SVG images.
- Embedded fonts can be used in inline SVG images.
- Importing raster images into an SVG image may be problematic as the SVG renderer does not yet use the project's URL resolver to resolve relative links.
- The SVG element must be defined as a block level element (the default) and the namespace must be included.

SVG Support Example

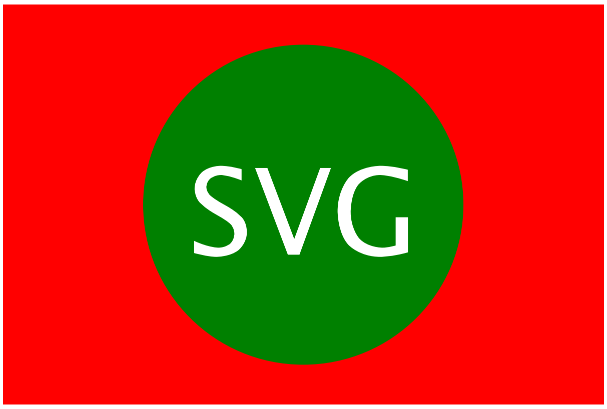
This example is taken from SVG Getting Started licensed under CC-BY-SA-2.5.

```
<html>
<body>
<svg version="1.1"
     baseProfile="full"
     width="300" height="200"
     xmlns="http://www.w3.org/2000/svg">

  <rect width="100%" height="100%" fill="red" />

  <circle cx="150" cy="100" r="80" fill="green" />
  <text x="150" y="125" font-size="60" text-anchor="middle" fill="white
</svg>
</body>
</html>
```

The output of this example is shown below:



Still to Document

- The `content` property.
- Counters.
- Tables.
- Forms.
- Project specific CSS properties.
- Lists
- General layout