wkgtk-html2pdf c++ API manual

rev V0.0.13.20260212_01

Index

1. Introduction

wkgtk-html2pdf is a powerful and easy-to-use API for converting HTML content into high-quality PDF documents using WebKitGTK. Developed as a modern alternative to legacy tools like wkhtmltopdf—which has been archived since 2023—this project fills the gap for reliable, up-to-date HTML-to-PDF conversion. It provides a clean, intuitive C++ API that simplifies HTML generation by eliminating the need for string literals and explicit closing tags, making it ideal for embedding in applications. The tool supports advanced features such as internal anchors, links, and nested sidebar indexing, enabling the creation of structured, navigable PDFs. A simple command-line interface is also available for quick, one-off conversions. With its focus on maintainability and modern development practices, wkgtk-html2pdf is designed to be a reliable choice for developers seeking a dependable solution for PDF generation.

wkgtk-html2pdf includes a set of built-in, pre-configured CSS templates for all standard ISO and US paper sizes, ensuring compatibility and consistency across different regions and use cases. These templates are designed to prevent the generation of extraneous blank pages by precisely defining page dimensions, margins, and layout constraints using CSS custom properties.

Each template uses a clean, modular structure with variables for page width, height, and margin, allowing for easy customization while maintaining reliable output. A lightweight JavaScript utility is included to monitor content overflow in real time—when content exceeds the available space, the subpage border turns red; when it fits perfectly, the border turns green. This visual feedback helps you quickly identify and resolve layout issues before conversion.

The templates also include print-specific CSS rules that ensure clean, artifact-free output when the HTML is rendered to PDF, while maintaining visual guides during development for easier debugging and refinement.

1.1 Reading this manual

This manual is a self fulfilling prophecy as it is developed entirely using wkgtk-html2pdf so the layout the code that is used to write it is in itself a tutorial. If you are uncertain about how something is done then is worth reviewing the HTML and CSS used to generate it.

2. Command line interface (CLI)

The CLI is the simplest way to get started with wkgtk-html2pdf. It allows you to convert HTML files into PDFs with minimal setup.

2.1 Basic Usage

To generate a PDF using the default settings use the following command:

```
html2pdf -i input.html -o output.pdf
```

2.2 Command line options

The following options are available for the cli.

Option	Description
-h, --help	Display the help message with all available options
-v, --verbose	Set the log level (1-7), with higher values providing more detailed output. Logs are written to the system journal
-i, --infile	Specify the source HTML file to convert
-o, --outfile	Specify the output PDF file name
-O, --orientation	Set page orientation: portrait or landscape
-s, --size	Set the page size (Default A4)
	- ISO (A): A0, A1, A2, A3, A4, A5, A6, A7, A8, A9, A10
	- ISO (B): B0, B1, B2, B3, B4, B5, B6, B7, B8, B9, B10
	- ISO (C): C0, C1, C2, C3, C4, C5, C6, C7, C8, C9, C10
	- US: Letter, Legal, Tabloid
	- ANSI: ANSIA, ANSIB, ANSIC, ANSID, ANSIE
	- Architectural: ArchA, ArchB, ArchC, ArchD, ArchE
	- Other: SRA0, SRA1, SRA2, SRA3, SRA4
--index	Create anchor points for indexing: `classic` or `enhanced` for nested sidebar indexing

3. Optimising HTML

To generate a clean, professional PDF, you must optimize your HTML to ensure it aligns with the desired page size and layout. The output is entirely governed by the CSS and HTML structure, so the quality of the final PDF depends on how well your content is designed.

3.1 Key Principles

3.1.1 Use the correct stylesheet

1. Each stylesheet is named according to the page size and orientation it supports (e.g., A4-portrait.css, ANSIA-landscape.css).
2. Link to the appropriate stylesheet in your HTML:

Example:

```
<link rel="stylesheet" href="/usr/share/wk2gtkpdf/A4-portrait.css">
```

Applying the Style Sheet for A4 portrait.

3.1.2 Use the correct classes

1. Use the **.page** class to define the overall page
2. Use the **.subpage** class to define content areas.

Example:

```
<div class="page">
    <div class="subpage">
        <!-- Your page content here -->
    </div>
</div>
```

Initialising page boundaries

3.1.3 Monitor the overflow

1. Include the JavaScript utility to monitor content overflow.
2. If overflow is detected the margin turns red.

This script provides real-time feedback, helping you identify and fix layout issues before conversion.

```
<script src="/usr/share/wkgtkpdf/overflow-monitor.js><script>
```

Declaring the javascript to monitor the overflow.

3.1.4 Avoid common issues

By designing your HTML and CSS carefully, you have full control over the final PDF output. Avoid common issues: Mismatched stylesheets, incorrect classes, or content overflow can lead to blank pages, cut-off content, or incorrect scaling.

The overflow detection script helps you identify and fix layout issues before conversion.

3.1.5 Best Practices

1. **Always test the layout** - Use the overflow detection script to ensure content fits within the page. Adjust the CSS or content as needed to avoid overflow.
2. **Use the correct stylesheet** - Ensure the stylesheet matches the CLI arguments (page size and orientation).
3. **Keep it consistent** - Use the same page size and orientation in both the CSS and CLI arguments.

3.2 Example: A4 Portrait Layout

Below is a minimal example that will create a single page PDF and utilise a compact bult in javascript function to monitor and warn of overflow.

```
<!DOCTYPE html>
<html>
<head>
    <link rel="stylesheet" href="/usr/share/wk2gtkpdf/A4-portrait.css">
</head>
<body>
    <div class="page">
        <div class="subpage">
            <h1>My Document</h1>
            <p>This content will be rendered in A4 portrait format.</p>
        </div>
    </div>
</body>
<script src="/usr/share/wkgtkpdf/overflow-monitor.js><script>
</html>
```

Minimal single page example.

Each declaration of a **.page** and **.subpage** container will create a new page. Blank pages can also be inserted by declaring a **.page** and **.subpage** container.

4. Anchors

When generating PDFs with wkgtk-html2pdf, you can choose between two indexing modes: classic and enhanced. The difference lies in which elements are included in the index.

4.1 Classic model

In classic mode, all internal links (**<a>** tags with href starting with **#**) are automatically included in the index, regardless of their container or structure.

Example:

```html
<li><a href="#reference">Reference</a></li>
<p><a href="#section1">Section 1</a></p>
<div><a href="#appendix">Appendix</a></div>
```

All these anchors are valid.

4.2 Enhanced mode

Enhanced mode (--index enhanced) is an advanced feature that allows you to selectively control which links appear in the PDF index. It only indexes elements with the **class="index-item"** class, giving you fine-grained control over the index.

Example:

```html
<!-- This will be included -->
<li class="index-item">
    <a href="#reference">
        <span>Reference</span>
        <span>A1</span>
    </a>
</li>

<!-- This will be ignored -->
<p><a href="#section1">Section 1</a></p>

<!-- This will be included (container type doesn't matter) -->
<div class="index-item">
    <a href="#appendix">Appendix</a>
</div>
```

Enhanced anchor declarations.

In this example only the first and third elements will be included in the index. While this feature may not be essential for most users, it could be useful in specific scenarios where you want to:

1. Exclude certain internal links from the index.
2. Create a more structured or semantic index.

For most users, the classic mode is sufficient and recommended.

Clickable areas marked in grey.

Save for the styling the difference is

Enhanced:

```
<li class="index-item"> <a href="#ref"> <span> Enhanced mode </span> <span>
4.1 </span> </a> </li>
```

A typical enhanced mode link.

Classic:

```
<li> <a href="#ref"> <span> Enhanced mode </span> <span> 4.1 </span> </a>
</li>
```

A typical demo_classic mode link.

As you can see from the code, theoretically the link should work across the entire line, even in classic mode but it doesn't (even in HTML). Ordinarily when you convert your link to a you lose right right hand side part of the anchor unless you enable enhanced mode

4.3 Sidebar indexing