Generating a PDF

You can generate a PDF of your Jekyll project. The PDF uses Prince XML to generate the PDF and allows you to configure which pages you want printed. The PDF output includes a table of contents for the entire guide, a mini-TOC on each section page, page numbers in cross references, and running headers and footers. The styling uses Bootstrap's CSS for print styles. You can see a sample here: Jekyll How-to Guide (PDF).

Table of Contents

Generate a PDF of All Docs in a Sidebar

1. Install Prince XML:
    - MacOS instructions
    - Windows instructions
2. Create a configuration file for the PDF output. Copy the **_config_pdf_jekyllhowto.yml** file and rename it to match your project's name (e.g., (**_config_pdf_<myproject>.yml**).
3. Open the configuration file and customize the following values:
    - `print_title` : Appears on the PDF's title page and running header.
    - `print_subtitle` : Appears on the PDF's title page.
    - `sidebar` : Used to generate the table of contents and mini-TOC.
    - `folderPath` : The folder path to your site output (for example, `/Users/tomjoht/projects/devcomm-appstore-v2/_site`). Prince needs this absolute path to access the files.
    - `copyright_notice` : The copyright statement that you want to appear in the manual.
4. Open your sidebar data file and add the following section after `folders` :

```yaml
folders:

- title: Frontmatter
  jurl: /frontmatter.html
  type: frontmatter
  pdf: true
  folderitems:

  - title: Title page
    jurl: /pdf_title_page.html
    pdf: true

  - title: Copyright page
    jurl: /pdf_copyright_page.html
    pdf: true

  - title: TOC page
    jurl: /pdf_toc_page.html
    pdf: true
```

5. Add a new property for each section title and page called `pdf: true` for all the pages you want included in the PDF. Follow the example in the previous code sample. Alternatively, look at **_data/jekyllhowto.yml** file.
6. Organize your pages into subfolders by section. (Sections are the `folders` title that contains `folderitems`, or the `subfolders` title that contains `subfolderitems`, etc.)

   For example, if your product nav has the folders "Getting Started" and "Configuration", create folders inside the _docs folder named "Getting Started" and "Configuration". Group the pages within those sections into those folders.
7. For each folder, add a new page that will serve as the mini-TOC for that section. Call it something that like **minitoc_.md**.

> **Tip:** It might be helpful to organize your pages into subfolders by section. All files get flattened into the root directory when your site builds, so it doesn't matter how many subfolders you use to organize your content.

8. Open up each mini-TOC file and add the following, customizing the frontmatter values for your own project:

```yaml
---
title: Get Started
permalink: /iap-minitoc-get-started.html
sidebar: in_app_purchasing
---

{% include pdfminitoc.html %}
```

The `title` should be the same as the section title in your sidebar table of contents. The `permalink` should match your file name. The `sidebar` corresponds to your product sidebar.

> **Note:** Be careful with your file names here. If you mistype a file name, the PDF won't build and you'll have to sort out the cause of the error later. In fact, Prince is much more exacting about only generating a PDF if all listed assets are available. You'll likely encounter some errors in the PDF generation process that stem from unavailable files or mistyped names in your sidebar data file. Think of these errors as helpful validation.

9. Open up your sidebar data file (in the _data folder) and add a `jurl` property for each **section** entry, like this:

```yaml
- title: Jekyll Project Setup
  jurl: /jekyllhowto-project-setup.html
  pdf: true
  folderitems:
```

Point the `jurl` value to your mini-TOC file.See the _data/jekyllhowto.yml file for an example.

10. Open up the first file in your table of contents (after the Frontmatter section). Add `class: first` in the page's frontmatter.

```yaml
---
title: Jekyll Project Setup
permalink: /jekyllhowto-minitoc-project-setup.html
sidebar: jekyllhowto
class: first
---
```

This property will reset the numbering so that the first page begins on "1" (after the lower-roman number numbering for the TOC and frontmatter section.)

Before running Prince, we need to build an HTML-friendly output for Prince to consume. This HTML-friendly output will apply the layout that we want reflected in the print material.

11. Create a Jekyll server shortcut file to build the PDF-friendly output that the Prince script will consume. Copy **serve_pdf_jekyllhowto.sh** and customize the file name with your own project. Open the file and customize the PDF file name to match your PDF configuration file:

```
jekyll serve --config _config_pdf_jekyllhowto.yml
```

12. Create a build shortcut file to make it easy to run Prince to generate the PDF. Copy **build_pdf_jekyllhowto.sh** and customize the file name with your own project. Open the file and customize the PDF file name (change `jekyllhowto.pdf`):

```
echo "Building the PDF ...";
prince --javascript --input-list=_site/assets/prince-list.txt -o pdf/jekyllhowto.pdf;
echo "Done. Look in the /pdf folder in your project directory.";
```

The prince-list.txt file contains scripts that iterate through your sidebar data file and gather links to all the pages to consolidate in the PDF. The `-o` parameter specifies the file name and location Prince should write the PDF file to. If you're having trouble with pages appearing, you can check prince-list.txt in your _site/assets output and make sure a valid link to the file exists on the page. The `--input-list` parameter in the above command is the input source for Prince.

Building the PDF

1. From the command line, build the HTML output that uses your PDF configuration file:

```sh
. serve_pdf_jekyllhowto.sh
```

(Use the custom Jekyll server shortcut you created earlier.)

2. When the server preview finishes, open another tab and build the PDF:

```sh
. build_pdf_jekyllhowto.sh
```

You don't need to have Jekyll server running to build the PDF. But it's helpful in case you build the PDF and notice some issues you want to fix.

If successful, you'll see a message like this:

```
Building the PDF ...
Done. Look in the /pdf folder in your project directory.
```

(Actually, the message will appear even if the build is unsuccessful — it just lets you know the process finished.)

Look in the root directory of your project (not the _/site folder), look for your PDF.

Generate a PDF of a Single Page

If you just want to create a simple PDF of one page in your docs, you can skip most of the steps in the previous section. To generate a PDF of a single doc:

1. Complete steps 1 through 3 in the previous section, Generate a PDF of All Docs in a Sidebar.
2. Build the HTML-friendly version of the site:

   ```
   jekyll serve --config _config_pdf_jekyllhowto.yml
   ```

3. From the command line, navigate to the _site folder. Then run this:

   ```
   prince --javascript jekyllhowto-content-and-formatting.html -o pdf/jekyllhowto-content-and-formatting.pdf;
   ```

   In this case, the page being converted to PDF is jekyllhowto-content-and-formatting.html. Replace this with the page you want.

Modify the Print Styles

The print styles are defined in **assets/css/pdf/printstyles.css**. To overwrite the styles:

1. Copy the contents of **printstyles.css**, which is packaged in the gem.

   To get the contents of the file, run `bundle show documentation-theme-jekyll-multioutput` from your Jekyll project directory. Go to the path shown and open the **assets/css/pdf/printstyles.css** file.

2. Copy and paste the contents of **printstyles.css** into a new file called **user_defined_pdf_styles.css**. Put **user_defined_pdf_styles.css** file into a folder called **assets/css/pdf** in your Jekyll project.

   This will overwrite the **user_defined_pdf_styles.css** file (which is blank) in the gem's files. This gem file is a placeholder intended to accommodate custom styles. It is referenced in the PDF layout files.

3. Change the styles as desired. See CSS Properties for a list of styles supported by Prince XML.

Change the PDF Styles

Does a style not look right? Do you want to customize the headers or footers for a specific page? You can do so by modifying the print stylesheet: **assets/css/pdf/printstyles.css**.

You can simply add regular CSS here as you want. See CSS Properties on the Prince XML site for supported properties.

Change the PDF's Headers or Footers

1. Follow the instructions in the previous section, Change the PDF Styles.
2. In your PDF configuration file (e.g., _config_pdf_jekyllhowto.yml), in the `defaults` section, add a `class` as a default to your **/_docs** and **pages**. For example:

```yaml
defaults:
  -
    scope:
      path: ""
      type: pages
    values:
      layout: printpdf
      class: myclass
  -
    scope:
      path: ""
      type: docs
    values:
      layout: printpdf
      class: myclass
```

This will add a default `class` property and value to your page's frontmatter, like this:

```yaml
---
class: myclass
---
```

3. Open the **user_defined_pdf_styles.css** file and define the style like this:

```css
body.myclass { page: myclass }
@page myclass {
    @top-left {
        content: " ";
    }
    @top-right {
        content: prince-script(datestamp);
    }
    @bottom-right {
        content: counter(page, lower-roman);
    }
    @bottom-left {
        content: prince-script(guideName);
        font-size: 11px;
    }
}
```

See the Page Selectors topic in the Prince XML documentation for more details.

The `datestamp` and `guideName` functions are special Prince functions defined in the printpdf.html layout (packaged in the assets/pdf folder of the gem). Other JavaScript generated content is also possible. See the Prince XML site for details.

Troubleshooting

If you see an error that Prince can't load an input file, it means one of the files in the list is incorrectly named. For example, you might see this error after running your `. build_pdf_jekyllhowto.sh` command:

```
prince: /Users/tomjoht/projects/myapp/_site/jekyllhowto-minitoc-getting-started.html: error: can't open input file: No such file or directory
```

If you see this error, look to make sure the file appears in the _site directory and is properly named. Check the `permalink` name of the file as well as its name in the sidebar menu file (e.g., _data/jekyllhowto.yml).

You might also get errors if you have JavaScript or non-allowed CSS syntax in your content.

Analyzing the Output

Look at the PDF. Check the display of your tables, images, code samples, and other formatting. Look at the running header and footer, as well as the title page, table of contents, and mini-TOC pages. Does it all look good? If so, great.

If you need to conditionalize some content so that it doesn't appear in the guide, you can use an if condition like this:

```liquid
{% unless site.format == "pdf" %}
This won't appear in the guide....
{% endunless %}
```

`unless` acts like a negative. You would read the above like this: Run this code *unless* site.format equals pdf.

You can also conditionalize your content using this syntax:

```liquid
{% if site.format == "web" %}
This won't appear in the guide....
{% endif %}
```

The configuration file for the Jekyll and Hippo outputs contains `format: web`. The configuration file for PDFs contains `format: pdf`.