Asciidoctor PDF Theming Guide

The theming system in Asciidoctor PDF is used to control the layout and styling of the PDF file Asciidoctor PDF generates from AsciiDoc. This document describes how the theming system works, how to define a custom theme in YAML and how to activate the theme when running Asciidoctor PDF.

The quickest way to create your own theme is to extend the default theme. This not only gives you a set of foundation styles to build on, it also provides a collection of bundled fonts. If you want to replace the bundled fonts with your own, you must declare the name and location of each font in the font catalog. To reuse the bundled fonts, you can either extend the default theme and/or redeclare the bundled fonts in the font catalog.

If you don't declare your own fonts (or extend the default theme), only the built-in (AFM) fonts provided by the PDF reader will be available. Using AFM fonts can result in missing functionality and warnings. See the Built-In (AFM) Fonts section to learn more about these limitations.

Table of Contents

Language Overview

The Asciidoctor PDF theme language is described using the YAML data format and incorporates many *concepts* from CSS and SASS. Therefore, if you have a background in web design, the terminology should be immediately familiar to you. **Note, however, that the theming system isn't actually CSS.**

The theme file must be named *<name>-theme.yml*, where `<name>` is the name of the theme. *We recommend **not** using the names **base** or **default** so you don't confuse it with one of the built-in themes.*

Selectors and Properties

Like CSS, themes have both selectors and properties. Selectors are the component you want to style. The properties are the style elements of that component that can be styled. All selector names are implicit (e.g., `heading`), so you customize the theme primarily by manipulating pre-defined property values (e.g., `font-size`).

The theme language in Asciidoctor PDF supports a limited subset of the properties from CSS. Some of these properties have different names from those found in CSS.

- An underscore (`_`) may be used in place of a hyphen (`-`) in all property names (so you may use `font_family` or `font-family`).
- An underscore (`_`) may be used in place of a hyphen (`-`) in all variable names (so you may use `$base_font_family` or `$base-font-family`).
- Instead of separate properties for font weight and font style, the theme language combines these settings in the `font-style` property (allowed values: `normal`, `bold`, `italic` and `bold_italic`).
- The `align` property in the theme language is roughly equivalent to the `text-align` property in CSS.
- The `font-color` property in the theme language is equivalent to the `color` property in CSS.

A theme is described in a YAML-based data format and stored in a dedicated theme file. YAML is a human-friendly data format that resembles CSS and helps to describe the theme. The theme language adds some extra features to YAML, such as variables, basic math, measurements and color values. These enhancements will be explained in detail in later sections.

Basic Theme

Here's an example of a basic theme file that extends the base theme:

basic-theme.yml

```
page:
  layout: portrait
  margin: [0.75in, 1in, 0.75in, 1in]
  size: Letter
base:
  font-color: #333333
  font-family: Times-Roman
  font-size: 12
  line-height-length: 17
  line-height: $base-line-height-length / $base-font-size
vertical-spacing: $base-line-height-length
heading:
  font-color: #262626
  font-size: 17
  font-style: bold
  line-height: 1.2
  margin-bottom: $vertical-spacing
link:
  font-color: #002FA7
outline-list:
  indent: $base-font-size * 1.5
footer:
  height: $base-line-height-length * 2.5
  line-height: 1
  recto:
    right:
      content: '{page-number}'
  verso:
    left:
      content: $footer-recto-right-content
```

When creating a new theme, you only have to define the keys you want to override from the base theme, which is loaded prior to loading your custom theme. All the available keys are documented in Keys. The converter uses the information from the theme map to help construct the PDF.

Basic Extended Theme

Instead of designing a theme from scratch, you can extend the default theme using the `extends` key as follows:

```
extends: default
base:
  font-color: #ff0000
```

You can also point the extends key at another custom theme to extend from it. If you don't want to extend any theme, including the base theme, assign the value `~` to the `extends` key (i.e., `extends: ~`).

If you start a new theme from scratch, we strongly recommend defining TrueType fonts and specifying them in the `base` and `literal` categories. Otherwise, Asciidoctor PDF will use built-in AFM fonts, which can result in missing functionality and warnings.

Instead of creating a theme from scratch, another option is to download the default-theme.yml file from the source repository. Save the file using a unique name (e.g., *custom-theme.yml*) and start hacking on it.

Alternatively, you can snag the file from your local installation using the following command:

```
$ ASCIIDOCTOR_PDF_DIR=`gem contents asciidoctor-pdf --show-install-dir`;\
    cp "$ASCIIDOCTOR_PDF_DIR/data/themes/default-theme.yml" custom-theme.yml
```

Key Nesting

Keys may be nested to an arbitrary depth to eliminate redundant prefixes (an approach inspired by SASS). Once the theme is loaded, all keys are flattened into a single map of qualified keys. Nesting is simply a shorthand way of organizing the keys. In the end, a theme is just a map of key/value pairs.

Nested keys are adjoined to their parent key with an underscore (`_`) or hyphen (`-`). This means the selector part (e.g., `link`) is combined with the property name (e.g., `font-color`) into a single, qualified key (e.g., `link_font_color` or `link-font-color`).

For example, let's assume we want to set the base (i.e., global) font size and color. These keys may be written longhand:

```
base-font-color: #333333
base-font-family: Times-Roman
base-font-size: 12
```

Or, to avoid having to type the prefix `base-` multiple times, the keys may be written as a hierarchy:

```
base:
  font-color: #333333
  font-family: Times-Roman
  font-size: 12
```

Or even:

```
base:
  font:
    color: #333333
    family: Times-Roman
    size: 12
```

Each level of nesting must be indented by two spaces from the indentation of the parent level. Also note the presence of the colon (`:`) after each key name.

Values

The value of a key may be one of the following types:

- String
  - Font family name (e.g., Roboto)
  - Font style (normal, bold, italic, bold_italic)
  - Alignment (left, center, right, justify)
  - Color as hex string (e.g., 'ff0000', #ff0000, or '#ff0000')
  - Image path
  - Enumerated type (where specified)
  - Text content (where specified)
- Null (clears any previously assigned value)
  - *empty* (i.e., no value specified)
  - null
  - ~
- Number (integer or float) with optional units (default unit is points)
- Array
  - Color as RGB array (e.g., [51, 51, 51])
  - Color CMYK array (e.g., [50, 100, 0, 0])
  - Margin (e.g., [1in, 1in, 1in, 1in])
  - Padding (e.g., [1in, 1in, 1in, 1in])
- Variable reference (e.g., $base_font_color or $base-font-color)
- Math expression

Note that keys almost always require a value of a specific type, as documented in Keys.

Inheritance

Like CSS, inheritance is a principle feature in the Asciidoctor PDF theme language. For many of the properties, if a key is not specified, the key inherits the value applied to the parent content in the content hierarchy. This behavior saves you from having to specify properties unless you want to override the inherited value.

The following keys are inherited:

- font-family
- font-color
- font-size
- font-style

- text-transform
- line-height (currently some exceptions)
- margin-bottom (if not specified, defaults to $vertical-spacing)

Heading Inheritance

Headings inherit starting from a specific heading level (e.g., `heading-h2-font-size`), then to the heading category (e.g., `heading-font-size`), then directly to the base value (e.g., `base-font-size`). Any setting from an enclosing context, such as a sidebar, is skipped.

Variables

To save you from having to type the same value in your theme over and over, or to allow you to base one value on another, the theme language supports variables. Variables consist of the key name preceded by a dollar sign (`$`) (e.g., `$base-font-size`). Any qualified key that has already been defined can be referenced in the value of another key. (In order words, as soon as the key is assigned, it's available to be used as a variable).

Variables are defined from top to bottom (i.e., in document order). Therefore, a variable must be defined before it is referenced. In other words, the path the variable refers to must be **above** the usage of that variable.

For example, once the following line is processed,

```
base:
  font-color: #333333
```

the variable `$base-font-color` will be available for use in subsequent lines and will resolve to `#333333`.

Let's say you want to make the font color of the sidebar title the same as the heading font color. Just assign the value `$heading-font-color` to the `$sidebar-title-font-color`.

```
heading:
  font-color: #191919
sidebar:
  title:
    font-color: $heading-font-color
```

You can also use variables in math expressions to use one value to build another. This is commonly done to set font sizes proportionally. It also makes it easy to test different values very quickly.

```
base:
  font-size: 12
  font-size-large: $base-font-size * 1.25
  font-size-small: $base-font-size * 0.85
```

We'll cover more about math expressions later.

Custom Variables

You can define arbitrary key names to make custom variables. This is one way to group reusable values at the top of your theme file. If you are going to do this, it's recommended that you organize the keys under a custom namespace, such as `brand`.

For instance, here's how you can define your brand colors:

```
brand:
  primary-color: #E0162B (1)
  secondary-color: '#FFFFFF' (2)
  alert-color: '0052A5' (3)
```

(1) To align with CSS, you may add `#` in front of the hex color value to coerce it to a string. A YAML preprocessor is used to ensure the value is not treated as a comment as would normally be the case in YAML.

(2) You may put quotes around the CSS-style hex value to make it friendly to a YAML editor or validation tool.

(3) The leading `#` on a hex value is entirely optional. However, we recommend that you always use either a leading `#` or surrounding quotes (or both) to prevent YAML from mangling the value (for example, 000000 would become 0, so use '000000' or #000000 instead).

You can now use these custom variables later in the theme file:

```
base:
  font-color: $brand-primary-color
```

Math Expressions & Functions

The theme language supports basic math operations to support calculated values. Like programming languages, multiply and divide take precedence over add and subtract.

The following table lists the supported operations and the corresponding operator for each.

Operation	Operator
multiply	*
divide	/
add	+
subtract	-

Operators must always be surrounded by a space on either side (e.g., 2 + 2, not 2+2).

Here's an example of a math expression with fixed values.

```
conum:
  line-height: 4 / 3
```

Variables may be used in place of numbers anywhere in the expression:

```
base:
  font-size: 12
  font-size-large: $base-font-size * 1.25
```

Values used in a math expression are automatically coerced to a float value before the operation. If the result of the expression is an integer, the value is coerced to an integer afterwards.

Numeric values less than 1 must have a 0 before the decimal point (e.g., 0.85).

The theme language also supports several functions for rounding the result of a math expression. The following functions may be used if they surround the whole value or expression for a key.

round(...)

Rounds the number to the nearest half integer.

floor(...)

Rounds the number up to the next integer.

ceil(...)

Rounds the number down the previous integer.

You might use these functions in font size calculations so that you get more exact values.

```
base:
  font-size: 12.5
  font-size-large: ceil($base-font-size * 1.25)
```

Measurement Units

Several of the keys require a value in points (pt), the unit of measure for the PDF canvas. A point is defined as 1/72 of an inch. If you specify a number without any units, the units defaults to pt.

However, us humans like to think in real world units like inches (in), centimeters (cm), or millimeters (mm). You can let the theme do this conversion for you automatically by adding a unit notation next to any number.

The following units are supported:

Unit	Suffix
Centimeter	cm
Inches	in
Millimeter	mm
Percentage[1]	%, vw, or vh
Points	pt (default)

1. A percentage with the % unit is calculated relative to the width or height of the content area. Viewport-relative percentages (vw or vh units) are calculated as a percentage of the page width or height, respectively. Currently, percentage units can only be used for placing elements on the title page or for setting the width of a block image.

Here's an example of how you can use inches to define the page margins:

```
page:
  margin: [0.75in, 1in, 0.75in, 1in]
```

The order of elements in a measurement array is the same as it is in CSS:

1. top
2. right
3. bottom
4. left

Alignments

The align subkey is used to align text and images within the parent container.

Text Alignments

Text can be aligned as follows:

- left
- center
- right
- justify (stretched to each edge)

Text Decorations

The following decorations can be applied to text:

- none (no decoration)
- underline

- line-through

Image Alignments

Images can be aligned as follows:

- left
- center
- right

Font Styles

In most cases, wherever you can specify a custom font family, you can also specify a font style. These two settings are combined to locate the font to use.

The following font styles are recognized:

- normal (no style)
- italic
- bold
- bold_italic

Text Transforms

Many places where font properties can be specified, a case transformation can be applied to the text. The following transforms are recognized:

- uppercase
- lowercase
- capitalize (each word, like CSS)
- none (clears an inherited value)

> Since Ruby 2.4, Ruby has built-in support for transforming the case of any letter defined by Unicode.
>
> If you're using Ruby < 2.4, and the text you want to transform contains characters beyond the Basic Latin character set (e.g., an accented character), you must install either the `activesupport` or the `unicode` gem in order for those characters to be transformed.
>
> ```
> $ gem install activesupport
> ```
>
> or
>
> ```
> $ gem install unicode
> ```

Colors

The theme language supports color values in three formats:

Hex

A string of 3 or 6 characters with an optional leading `#`, optional surrounding quotes, or both.

RGB

An array of numeric values ranging from 0 to 255.

CMYK

An array of numeric values ranging from 0 to 1 or from 0% to 100%.

Transparent

The special value `transparent` indicates that a color should not be used.

Hex

The hex color value is likely most familiar to web developers. The value must be either 3 or 6 characters (case insensitive) with an optional leading hash (`#`), optional surrounding quotes, or both.

To align with CSS, you may add a `#` in front of the hex color value. A YAML preprocessor is used to ensure the value is not treated as a comment as would normally be the case in YAML. That same preprocessor will also coerce a primitive value to a string if `color` is the name of the last segment in the key (e.g., `font-color`). This avoids the problem of 000 becoming 0 (and similar implicit conversions) when the theme file is parsed.

You also may put quotes around the CSS-style hex value to make it friendly to a YAML editor or validation tool. In this case, the leading `#` on a hex value is entirely optional.

Regardless, we recommend that you always use either a leading `#` or surrounding quotes (or both) to prevent YAML from mangling the value.

The following are all equivalent values for the color red:

`#ff0000`	`#FF0000`	`'ff0000'`	`'FF0000'`	`#f00`	`#F00`	`'f00'`	`'F00'`

Here's how a hex color value appears in the theme file:

```
base:
  font-color: #ff0000
```

RGB

An RGB array value must be three numbers ranging from 0 to 255. The values must be separated by commas and be surrounded by square brackets.

An RGB array is automatically converted to a hex string internally, so there's no difference between ff0000 and [255, 0, 0].

Here's how to specify the color red in RGB:

- [255, 0, 0]

Here's how a RGB color value appears in the theme file:

```
base:
  font-color: [255, 0, 0]
```

CMYK

A CMYK array value must be four numbers ranging from 0 and 1 or from 0% to 100%. The values must be separated by commas and be surrounded by square brackets.

Unlike the RGB array, the CMYK array *is not* converted to a hex string internally. PDF has native support for CMYK colors, so you can preserve the original color values in the final PDF.

Here's how to specify the color red in CMYK:

- [0, 0.99, 1, 0]
- [0, 99%, 100%, 0]

Here's how a CMYK color value appears in the theme file:

```
base:
  font-color: [0, 0.99, 1, 0]
```

Transparent

It's possible to specify no color by assigning the special value `transparent`, as shown here:

```
table:
  background-color: transparent
```

The `transparent` keyword can be used for the background or border color, but not the font color.

Images

An image is specified either as a bare image path or as an inline image macro as found in the AsciiDoc syntax. Images in the theme file are currently resolved relative to the value of the `pdf-themesdir` attribute. (If `pdf-theme` is a path that ends in `.yml`, and `pdf-themesdir` is not set, then the images are resolved relative to the directory of the path specified by `pdf-theme`).

The following image types (and corresponding file extensions) are supported:

- PNG (.png)
- JPEG (.jpg)
- SVG (.svg)

The GIF format (.gif) and BMP format (.bmp) are not supported unless you're using prawn-gmagick. See support for additional image file formats for details.

Here's how an image is specified in the theme file as a bare image path:

```
title-page:
  background-image: title-cover.png
```

Here's how the image is specified using the inline image macro:

```
title-page:
  background-image: image:title-cover.png[]
```

In either case, the image is resolved relative to the value of the `pdf-themesdir` attribute, as previously described.

Like in the AsciiDoc syntax, wrapping the value in the image macro allows you to specify other settings, such as `pdfwidth`, `fit`, and/or `align`. For example:

```
title-page:
  logo-image: image:logo.png[width=250,align=center]
```

Quoted String

Some of the keys accept a quoted string as text content. The final segment of these keys is always named `content`.

A content key accepts a string value. It's usually best to quote the string or use the YAML multi-line string syntax.

Text content may be formatted using a subset of inline HTML. You can use the well-known elements such as `<strong>`, `<em>`, `<code>`, `<a>`, `<sub>`, `<sup>`, `<del>`, and `<span>`. The `<span>` element supports the `style` attribute, which you can use to specify the `color`, `font-weight`, and `font-style` CSS properties. You can also use the `rgb` attribute on the `<color>` element to change the color or the `name` and `size` attributes on the `<font>` element to change the font properties. If you need to add an underline or strikethrough decoration to the text, you can assign the `underline` or `line-through` to the `class` attribute on any aforementioned element.

Here's an example of using formatting in the content of the menu caret:

```
menu-caret-content: " <font size=\"1.15em\"><color rgb=\"#b12146\">\u203a</color></font>
"
```

The string must be double quoted in order to use a Unicode escape code like `\u203a`.

Additionally, normal substitutions are applied to the value of content keys for running content, so you can use most AsciiDoc inline formatting (e.g., `*strong*` or `{attribute-name}`) in the values of those keys.

Fonts

You can select from built-in PDF fonts, fonts bundled with Asciidoctor PDF or custom fonts loaded from TrueType font (TTF) files. If you want to use custom fonts, you must first declare them in your theme file.

Asciidoctor has no challenge working with Unicode. In fact, it prefers Unicode and considers the entire range. However, once you convert to PDF, you have to meet the font requirements of PDF in order to preserve Unicode characters. That means you need to provide a font (at least a fallback font) that contains glyphs for all the characters you want to use. If you don't, you may notice that characters are missing (usually replaced with a box). There's nothing Asciidoctor can do to convince PDF to work with extended characters without the right fonts in play. To see which characters are missing from the font, enable verbose mode (`-v`) when running Asciidoctor PDF.

Built-In (AFM) Fonts

The names of the built-in fonts (for general-purpose text) are as follows:

Font Name	Font Family
Helvetica	sans-serif
Times-Roman	serif
Courier	monospace

Using a built-in font requires no additional files. You can use the key anywhere a `font-family` property is accepted in the theme file. For example:

```
base:
  font-family: Times-Roman
```

However, when you use a built-in font, the characters you can use in your document are limited to the characters in the WINANSI (Windows-1252) code set. WINANSI includes most of the characters needed for writing in Western languages (English, French, Spanish, etc). For anything outside of that, PDF is BYOF (Bring Your Own Font).

Even though the built-in fonts require the content to be encoded in WINANSI, *you still type your AsciiDoc document in UTF-8*. Asciidoctor PDF encodes the content into WINANSI when building the PDF.

Built-in (AFM) fonts do not use the fallback fonts. In order for the fallback font to kick in, you must use a TrueType font as the primary font.

WINANSI Encoding Behavior

When using the built-in PDF (AFM) fonts on a block of content in your AsciiDoc document, any character that cannot be encoded to WINANSI is replaced with a logic "not" glyph (¬) and you'll see the following warning in your console:

```
The following text could not be fully converted to the Windows-1252 character set:
| <string with unknown glyph>
```

This behavior differs from the default behavior in Prawn, which is to simply crash.

You'll often see this warning if you're using callouts in your document and you haven't specified a TrueType font in your theme. To prevent this warning, you need to specify a TrueType font.

When using a TrueType font, you will get no warning for a missing glyph. That's a consequence of how Prawn works and is outside of Asciidoctor PDF's control. However, you'll likely see it substituted with a box (guaranteed if you're using one of the bundled fonts).

For more information about how Prawn handles character encodings for built-in fonts, see this note in the Prawn CHANGELOG.

Bundled Fonts

Asciidoctor PDF bundles several fonts that are used by the default theme. You can also use these fonts in your custom theme by simply declaring them. These fonts provide more characters than the built-in PDF fonts, but still only a subset of UTF-8 (to reduce the size of the gem).

The family name of the fonts bundled with Asciidoctor PDF are as follows:

Noto Serif

A serif font that can be styled as normal, italic, bold or bold_italic.

M+ 1mn

A monospaced font that maps different thicknesses to the styles normal, italic, bold and bold_italic. Also provides the circuled numbers used in callouts.

M+ 1p Fallback

A sans-serif font that provides a very complete set of Unicode glyphs. Cannot be styled as italic, bold or bold_italic. Used as the fallback font in the `default-with-fallback-font` theme.

If you want to specify the location of custom fonts using the `pdf-fontsdir` attribute, yet still be able to use the bundled fonts, you need to refer to the bundled fonts using the `GEM_FONTS_DIR` token. To do so, you can either a) prefix the path of the bundled font in the theme file with the segment `GEM_FONTS_DIR` (e.g., `GEM_FONTS_DIR/mplus1p-regular-fallback.ttf`, or b) use relative paths in the theme file and include `GEM_FONT_DIR` in the value of the `pdf-fontsdir` attribute separated by the location of your custom fonts using a semi-colon (e.g., `"path/to/your/fonts;GEM_FONTS_DIR"`).

Custom Fonts

The limited character set of WINANSI, or the bland look of the built-in fonts, may motivate you to load your own font. Custom fonts can enhance the look of your PDF theme substantially.

To start, find the TTF file collection for the font you want to use. A collection typically consists of all four font styles:

- normal
- italic
- bold

- bold_italic

You'll need all four variants to support AsciiDoc content properly. Otherwise, the converter will likely crash. If you don't have one of the variants, you can simply reuse the normal variant in its place. *Asciidoctor PDF cannot italicize a font dynamically like a browser can, so the italic styles are required.*

In order for a third-party font to work properly with Prawn (and hence Asciidoctor PDF), several modifications are required. See Preparing a Custom Font to learn how to prepare your font for use with Asciidoctor PDF.

Once you've obtained the TTF files, put them in the directory inside your project where you want to store the fonts. It's recommended that you name them consistently so it's easier to type the names in the theme file.

Let's assume the name of the font is Roboto. Rename the files as follows:

- roboto-normal.ttf (*originally Roboto-Regular.ttf*)
- roboto-italic.ttf (*originally Roboto-Italic.ttf*)
- roboto-bold.ttf (*originally Roboto-Bold.ttf*)
- roboto-bold_italic.ttf (*originally Roboto-BoldItalic.ttf*)

Next, declare the font under the `font-catalog` key at the top of your theme file, giving it a unique key (e.g., `Roboto`).

```
font:
  catalog:
    Roboto:
      normal: roboto-normal.ttf
      italic: roboto-italic.ttf
      bold: roboto-bold.ttf
      bold_italic: roboto-bold_italic.ttf
```

You must declare all four variants. If you're missing the font file for one of the variants, configure it to use the same font file as the normal variant.

You can use the key that you assign to the font in the font catalog anywhere the `font-family` property is accepted in the theme file. For example, to use the Roboto font for all headings (section titles and discrete headings), use:

```
heading:
  font-family: Roboto
```

When you execute Asciidoctor PDF, specify the directory where the fonts reside using the `pdf-fontsdir` attribute:

```
$ asciidoctor-pdf -a pdf-theme=basic-theme.yml -a pdf-fontsdir=path/to/fonts
document.adoc
```

You can specify multiple directories by separating the entries with a semi-colon and enclosing the value in double quotes:

```
$ asciidoctor-pdf -a pdf-theme=basic-theme.yml -a pdf-fontsdir="path/to/fonts;path/to/more-fonts" document.adoc
```

To include the bundled fonts in the search, use the `GEM_FONTS_DIR` token:

```
$ asciidoctor-pdf -a pdf-theme=basic-theme.yml -a pdf-fontsdir="path/to/fonts;GEM_FONTS_DIR" document.adoc
```

When running Asciidoctor PDF on the JVM (perhaps using AsciidoctorJ PDF), you can refer a directory inside of any JAR file on the classpath by prefixing the path with `uri:classloader:`:

```
$ asciidoctorj -b pdf -a pdf-theme=basic-theme.yml -a pdf-fontsdir="uri:classloader:/path/to/fonts;GEM_FONTS_DIR" document.adoc
```

When Asciidoctor PDF creates the PDF, it only embeds the glyphs from the font that are needed to render the characters present in the document. Effectively, it subsets the font. While that saves space taken up by the generated PDF, you may still be storing the full font in your source repository. To minimize the size of the source font, you can use FontForge to subset the font ahead of time. Subsetting a font means remove glyphs you don't plan to use. Doing so is not a requirement, simply a personal preference.

You can add any number of fonts to the catalog. Each font must be assigned a unique key, as shown here:

```
font:
  catalog:
    Roboto:
      normal: roboto-normal.ttf
      italic: roboto-italic.ttf
      bold: roboto-bold.ttf
      bold_italic: roboto-bold_italic.ttf
    Roboto Light:
      normal: roboto-light-normal.ttf
      italic: roboto-light-italic.ttf
      bold: roboto-light-bold.ttf
      bold_italic: roboto-light-bold_italic.ttf
```

Text in SVGs will use the font catalog from your theme. We recommend that you match the font key in your theme file to the name of the font seen by the operating system. This will allow you to use the same font names (aka families) in both your graphics program and Asciidoctor PDF, thus making them portable.

Fallback Fonts

If a TrueType font is missing a character needed to render the document, such as a special symbol or emoji, you can have Asciidoctor PDF look for the character in a fallback font.

You only need to specify a single fallback font, typically one that provides a full set of symbols. If the character isn't found in the fallback font, it will mostly likely be replaced by a box (i.e., the notdef glyph), which is guaranteed if you're using the bundled fallback font.

When defining the fallback font, you **must specify all four variants** (normal, bold, italic, bold_italic), even if you use the same font file for each.

The fallback font only gets used when the primary font is a TrueType font (i.e., TTF, DFont, TTC). Any glyph missing from an AFM font is simply replaced with the "not" glyph (`¬`).

The `default` theme does not use a fallback font. However, the built-in `default-with-fallback-font` theme does. In fact, it provides two. One for general writing in non-Latin languages (M+ 1p) and another for emoji (Noto Emoji). Using the fallback font slows down PDF generation slightly because it has to analyze every single character. It's use is not recommended for large documents. Instead, it's best to select primary fonts that have all the characters you need.

Like with other custom fonts, you first need to declare the fallback font. Let's choose Droid Sans Fallback. You can map all the styles to a single font file (since bold and italic don't usually make sense for symbols).

```
font:
  catalog:
    Roboto:
      normal: roboto-normal.ttf
      italic: roboto-italic.ttf
      bold: roboto-bold.ttf
      bold_italic: roboto-bold_italic.ttf
    DroidSansFallback:
      normal: droid-sans-fallback.ttf
      italic: droid-sans-fallback.ttf
      bold: droid-sans-fallback.ttf
      bold_italic: droid-sans-fallback.ttf
```

Notice that we define all four variants for the fallback font, even though we're use the same font file for each variant. This ensures the fallback font will be used regardless of which font style is active when it gets called on.

Next, add the key name to the `fallbacks` key under the `font-catalog` key. The `fallbacks` key accepts an array of values, meaning you can specify more than one fallback font. However, we recommend using a single fallback font, if possible, as shown here:

```
font:
  catalog:
    Roboto:
      normal: roboto-normal.ttf
      italic: roboto-italic.ttf
      bold: roboto-bold.ttf
      bold_italic: roboto-bold_italic.ttf
    DroidSansFallback:
      normal: droid-sans-fallback.ttf
      italic: droid-sans-fallback.ttf
      bold: droid-sans-fallback.ttf
      bold_italic: droid-sans-fallback.ttf
  fallbacks:
  - DroidSansFallback
```

If you are using more than one fallback font, add additional lines to the `fallbacks` key.

Of course, make sure you've configured your theme to use your custom font:

```
base:
  font-family: Roboto
```

That's it! Now you're covered. If your custom TTF font is missing a glyph, Asciidoctor PDF will look in your fallback font. You don't need to reference the fallback font anywhere else in your theme file.

Here's another example that shows how to use an alternative emoji font (Symbola):

```
extends: default-with-fallback-font
font:
  catalog:
    merge: true
    Symbola: /path/to/symbola.ttf
  fallbacks: [ M+ 1p, Symbola ]
```

Now Asciidoctor PDF will look for the emoji in the Symbola font instead of the Noto Emoji font.

Keys

This section lists all the keys that are available when creating a custom theme. The keys are organized by category. Each category represents a common prefix under which the keys are typically nested.

Keys can be nested wherever an underscore (`_`) or hyphen (`-`) appears in the name. This nested structure is for organizational purposes only. All keys are flatted when the theme is loaded (e.g., `align` nested under `base` becomes `base-align`).

The converter uses the values of these keys to control how most elements are arranged and styled in the PDF. The default values listed in this section get inherited from the base theme.

The default theme has a different set of values which are not shown in this guide.

When creating a theme, all keys are optional. Required keys are provided by the base theme. Therefore, you only have to declare keys that you want to override.

Extends

A theme can extend another theme using the `extends` key. For example:

```
extends: default
base:
  font-color: #ff0000
```

The extends key accepts either a single value or an array of values. Each value is interpreted as a filename. If the filename equals `default`, it resolves to the location of the default (built-in) theme. If the filename is absolute, it's used as is. If the filename begins with `./`, it's resolved as a theme file relative to the current theme file. Otherwise, the filename is resolved as a theme file in the normal way (relative to the value of the `pdf-themesdir` attribute).

If you define the font catalog in a theme that extends from `default`, you either have to redeclare any built-in font that on which the combined theme depends, or you need to set `merge: true` above your font definitions. You can find the built-in definitions in default theme. You'll then need to include `GEM_FONTS_DIR` in the value of the `pdf-fontsdir` attribute so that the converter can find and register them. To avoid having to do this, make sure you set the font family for any element that declares a font family in the default theme.

Currently, the base theme is always loaded first. Then, the files referenced by the extends key are loaded in order. Finally, the keys in the current file are loaded. Each time a theme is loaded, the keys are overlaid onto the keys from the previous theme.

Key	Value Type	Example

extends	String or Array (default: [])	`extends:`
		`- default`
		`- ./brand-theme.yml`

Role

The keys in the `role` category define custom roles for formatting. The name of the role is the first subkey level. The role name may contain a hyphen, but **a role name cannot contain an underscore**. The keys under the role are the theming properties.

Custom roles only apply to inline phrases.

Here's an example of a role for making text red:

```
role:
  red:
    font-color: #ff0000
```

This role can be used as follows:

```
Error text is shown in [.red]#red#.
```

You can also use a role to unset a font color (to make it inherit):

```
role:
  heading-code:
    font-color: ~
```

This role can be used as follows:

```
== [.heading-code]`SELECT` clause
```

The converter provides several predefined roles, which can can all be redefined. The `big` and `small` roles map the font size to the $base-font-size-large and $base-font-size-small values, respectively. The `underline` and `line-through` roles add the underline and strikethrough decorations, respectively. The `subtitle` role is used to configure the font properties of the subtitle of a section title. The color roles (e.g., `blue`), which you may be familiar with from the HTML converter, are not mapped by default. You'll need to define these color roles in your theme if you'd like to make use of them when converting to PDF.

Key	Value Type	Example
Key Prefix: role-<name>		

background-color	Color
	(default: *not set*)
		`role:`
		`  highlight:`
		`  background-color: #ffff00`

border-color	Color
	(default: *not set*)
		`role:`
		`  found:`
		`  border-color: #cccccc`

border-offset	Number
	(default: 0)
		`role:`
		`  found:`
		`  border-offset: 2`

border-radius	Number
	(default: *not set*)
		`role:`
		`  found:`
		`  border-radius: 3`

border-width	Number
	(default: *not set*)
		`role:`
		`  found:`
		`  border-width: 0.5`

font-color	Color
	(default: *inherit*)
		`role:`
		`  red:`
		`  font-color: #ff0000`

font-family	Font family name
	(default: Courier)
		`role:`
		`  label:`
		`  font-family: M+ 1mn`

font-size	Number
	(default: *inherit*)
		`role:`
		`  large:`
		`  font-size: 12`

font-style	Font style
	(default: *inherit*)
		`role:`
		`  heavy:`
		`  font-style: bold`

text-decoration	Text decoration
	(default: none)
		`role:`
		`  deleted:`
		`  text-decoration: line-through`

text-decoration-color	Color
	(default: $role-<name>-font-color)
		`role:`
		`  deleted:`
		`  text-decoration-color: #ff0000`

text-decoration-width	Number
	(default: 1)
		`role:`
		`  underline:`
		`  text-decoration-width: 0.5`

Page

The keys in this category control the size, margins and background of each page (i.e., canvas). We recommended that you define this category before all other categories.

The background of the title page can be styled independently of other pages. See Title Page for details.

Key	Value Type	Example
Key Prefix: page		
background-color[1]	Color (default: #ffffff)	`page:` `  background-color: #fefefe`

background-image[2]	image macro[3]
	(default: *not set*)
		`page:`
		`  background-image:`
		`image:page-bg.png[]`

background-image-(recto|verso)[2]	image macro[3]
	(default: *not set*)
		`page:`
		`  background-image:`
		`  recto: image:page-bg-`
		`recto.png[]`
		`  verso: image:page-bg-`
		`verso.png[]`

foreground-image[2]	image macro[3]
	(default: *not set*)
		`page`
		`  foreground-image:`
		`image:watermark.svg[]`

initial-zoom	Fit | FitH | FitV (default: FitH)	`page:` `  initial-zoom: Fit`
layout	portrait | landscape (default: portrait)	`page:` `  layout: landscape`

margin	Measurement |
	Measurement[top,right,bottom,left]
	(default: 36)
		`page:`
		`  margin: [0.5in, 0.67in,`
		`1in, 0.67in]`

margin-inner[4]	Measurement (default: 48)	`page:` `  margin-inner: 0.75in`
margin-outer[4]	Measurement (default: 24)	`page:` `  margin-outer: 0.59in`
mode	outline | none | thumbs | fullscreen | fullscreen outline | fullscreen none | fullscreen thumbs (default: outline)	`page:` `  mode: fullscreen none`
size	Named size | Measurement[width,height] (default: A4)	`page:` `  size: Letter`
numbering-start-at[5]	title | toc | body | Integer (default: body)	`page:` `  numbering-start-at: toc`

1. To disable the background color for the page, set the value to white (i.e., FFFFFF). The color keyword `transparent` is not recognized in this context.

2. By default, page background and foreground images are automatically scaled to fit the bounds of the page (i.e., `fit=contain`) and centered (i.e., `position=center`). The size of the image can be controlled using any of the sizing attributes on the image macro (i.e., fit, pdfwidth, scaledwidth, or width) when `fit=none`. The position of the image can be controlled using the `position` attribute. If the recto (right-hand, odd-numbered pages) or verso (left-hand, even-numbered pages) background image is specified, it will be used only for that side (not available for the foreground image). If you define the keys using the flatten structure (e.g., `page-background-image-recto`), you can also set the default page background image (`page-background-image`), which will then be used as a fallback if a background image isn't specified for a given side. To disable the image, use the value `none`.
3. Target may be an absolute path or a path relative to the value of the `pdf-themesdir` attribute.
4. The margins for `recto` (right-hand, odd-numbered) and `verso` (left-hand, even-numbered) pages are calculated automatically from the margin-inner and margin-outer values. These margins and used when the value `prepress` is assigned to the `media` document attribute. If no cover is specified, the recto margin is not applied to the title page. To apply the recto margin to the title page, but not include a cover, assign the value `~` to the `front-cover-image` attribute.
5. Only works if the document uses a title page (i.e., doctype is book or `title-page` attribute is set) The `toc` value only applies if the toc is in the default location (before the first page of the body). If the toc macro is used to position the toc, the start-at behavior is the same as if the toc is not enabled. If value is an integer, page numbering will start at the specified page of the body (i.e., 1 is first page, 2 is second page, etc.)

Base

The keys in this category provide generic theme settings and are often referenced throughout the theme file as variables. We recommended that you define this category after the page category and before all other categories.

While it's common to define additional keys in this category (e.g., `base-border-radius`) to keep your theme DRY, we recommend using custom variables instead.

Key	Value Type	Example
Key Prefix: base		
align	Text alignment (default: left)	`base:` `  align: justify`
border-color	Color (default: #eeeeee)	`base:` `  border-color: #eeeeee`
border-width	Number (default: 0.5)	`base:` `  border-width: 0.5`
font-color	Color (default: #000000)	`base:` `  font-color: #333333`
font-family	Font family name (default: Helvetica)	`base:` `  font-family: Noto Serif`
font-kerning	normal | none (default: normal)	`base:` `  font-kerning: none`
font-size	Number (default: 12)	`base:` `  font-size: 10.5`
font-size-min	Number (default: 6)	`base:` `  font-size-min: $base-font-size * 0.75`
font-style	Font style (default: normal)	`base:` `  font-style: normal`
text-transform[1]	none (default: none)	`base:` `  text-transform: none`
line-height-length[2]	Number (default: *not set*)	`base:` `  line-height-length: 12`

line-height[2]	Number
	(default: 1.15)
		`base:`
		`  line-height: >`
		`  $base-line-height-length /`
		`  $base-font-size`

text-decoration-width	Number (default: 1)	`base:` `  text-decoration-width: 0.5`

1. The `text-transform` key cannot be set globally. Therefore, this key should not be used. The value of `none` is implicit and is documented here for completeness.
2. `line-height-length` is a utility property that's internal to the theme. It's used as an intermediate property for computing the `base-line-height` from the base font size and the desired line height size. For instance, if you set `base-line-height-length`, you can use `$base-line-height-length / $base-font-size` to set the value of `base-line-height`. You don't have to go about it this way in your own theme.

Vertical Spacing

The keys in this category control the general spacing between elements where a more specific setting is not designated.

Key	Value Type	Example
vertical-spacing	Number (default: 12)	`vertical-spacing: 10`

Link

The keys in this category are used to style hyperlink text.

Key	Value Type	Example
Key Prefix: link		
font-color	Color (default: #0000ee)	`link:` `  font-color: #428bca`
font-family	Font family name (default: *inherit*)	`link:` `  font-family: Roboto`
font-size	Number (default: *inherit*)	`link:` `  font-size: 9`
font-style	Font style (default: *inherit*)	`link:` `  font-style: italic`
text-decoration	Text decoration (default: none)	`link:` `  text-decoration: underline`
text-decoration-color	Color (default: $link-font-color)	`link:` `  text-decoration-color: #0000ff`
text-decoration-width	Number (default: 1)	`link:` `  text-decoration-width: 0.5`

(Inline) Literal

The keys in this category are used for inline monospaced text in prose and table cells.

Key	Value Type	Example
Key Prefix: literal		
background-color	Color (default: *not set*)	`literal:` `  background-color: #f5f5f5`
border-color[1]	Color (default: *not set*)	`literal:` `  border-color: #cccccc`
border-offset[2]	Number (default: 0)	`literal:` `  border-offset: 2`
border-radius	Number (default: *not set*)	`literal:` `  border-radius: 3`
border-width	Number (default: $base-border-width)	`literal:` `  border-width: 0.5`
font-color	Color (default: *inherit*)	`literal:` `  font-color: #b12146`
font-family	Font family name (default: Courier)	`literal:` `  font-family: M+ 1mn`
font-size	Number (default: *inherit*)	`literal:` `  font-size: 12`
font-style	Font style (default: *inherit*)	`literal:` `  font-style: bold`

1. The border is only used if a border color is specified and the border width is not explicitly set to 0. The border only works properly if the literal phrase does not have nested formatting. Otherwise, the border will be inherited, producing a less than desirable result.
2. The border offset is the amount that the background and border swells around the text. It does not affect the distance between the formatted phrase and the phrases that surround it.

Heading

The keys in this category control the style of most headings, including part titles, chapter titles, sections titles, the table of contents title and discrete headings.

Key	Value Type	Example
Key Prefix: heading		
align	Text alignment (default: $base-align)	`heading:` `  align: center`
font-color	Color (default: *inherit*)	`heading:` `  font-color: #222222`
font-family	Font family name (default: *inherit*)	`heading:` `  font-family: Noto Serif`
font-kerning	normal | none (default: *inherit*)	`heading:` `  font-kerning: none`
font-style	Font style (default: bold)	`heading:` `  font-style: bold`
text-decoration	Text decoration (default: none)	`heading:` `  text-decoration: underline`
text-decoration-color	Color (default: $heading-font-color)	`heading:` `  text-decoration-color: #cccccc`
text-decoration-width	Number (default: 1)	`heading:` `  text-decoration-width: 0.5`
text-transform	Text transform (default: *inherit*)	`heading:` `  text-transform: capitalize`
line-height	Number (default: 1.15)	`heading:` `  line-height: 1.2`
margin-top	Measurement (default: 4)	`heading:` `  margin-top: $vertical-spacing * 0.2`
margin-page-top	Measurement (default: 0)	`heading:` `  margin-page-top: $vertical-spacing`
margin-bottom	Measurement (default: 12)	`heading:` `  margin-bottom: 9.6`
min-height-after	Measurement (default: $base-font-size * $base-line-height * 1.5)	`heading:` `  min-height-after: 0.5in`

chapter-break-before	always | auto
	(default: always)
		`heading:`
		`  chapter:`
		`  break-before: auto`

part-break-before	always | auto
	(default: always)
		`heading:`
		`  part:`
		`  break-before: auto`

Key	Value Type	Example

part-break-after	always | auto
	(default: auto)
		`heading:`
		`  part:`
		`  break-after: always`

Key Prefix: heading-h<n>[1]		
align	Text alignment (default: $heading-align)	`heading:` `  h2-align: center`
font-color	Color (default: $heading-font-color)	`heading:` `  h2-font-color: [0, 99%, 100%, 0]`
font-family	Font family name (default: $heading-font-family)	`heading:` `  h4-font-family: Roboto`
font-kerning	normal | none (default: $heading-font-kerning)	`heading:` `  h3-font-kerning: none`
font-size[1]	Number (default: <1>=24; <2>=18; <3>=16; <4>=14; <5>=12; <6>=10)	`heading:` `  h6-font-size: $base-font-size * 1.7`
font-style	Font style (default: $heading-font-style)	`heading:` `  h3-font-style: bold_italic`
text-transform	Text transform (default: $heading-text-transform)	`heading:` `  h3-text-transform: uppercase`
margin-top	Measurement (default: $heading-margin-top)	`heading:` `  h2-margin-top: $vertical-spacing * 0.5`
margin-page-top	Measurement (default: $heading-margin-page-top)	`heading:` `  h2-margin-page-top: $vertical-spacing`
margin-bottom	Measurement (default: $heading-margin-bottom)	`heading:` `  h2-margin-bottom: 10`

1. `<n>` is a number ranging from 1 to 6, representing each of the six heading levels.
2. A font size is assigned to each heading level by the base theme. If you want the font size of a specific level to be inherited, you must assign the value `null` (or `~` for short).

Section

The keys in this category control the style of a section body.

Key	Value Type	Example
Key Prefix: section		
indent	Measurement | Measurement[left,right][1] (default: 0)	`section:` `  indent: [0.5in, 0]`

1. A single value gets applied to both the left and right side (e.g., `0.5in`). A two-value array configures the left and right side independently (e.g., `[0.5in, 0]`).

Title Page

The keys in this category control the style of the title page as well as the arrangement and style of the elements on it.

The title page is only enabled by default for the book doctype (e.g., `:doctype: book`). If you want to enable the title page when using a different doctype (such as the article doctype), you must define the `title-page` attribute in the document header (i.e., `:title-page:`).

Subtitle partitioning of the doctitle is only enabled when the title page is also enabled.

The title page can be disabled for the book doctype by setting the `notitle` attribute in the AsciiDoc document header (i.e., `:notitle:`). (For other doctypes, just don't set the `title-page` attribute).

Key	Value Type	Example
Key Prefix: title-page		
align	Text alignment (default: center)	`title-page:` `  align: right`
background-color[1]	Color (default: *inherit*)	`title-page:` `  background-color: #eaeaea`

background-image[2]	image macro[3]
	(default: *not set*)
		`title-page:`
		`  background-image:`
		`image:title.png[]`

font-color	Color (default: *inherit*)	`title-page:` `  font-color: #333333`
font-family	Font family name (default: *inherit*)	`title-page:` `  font-family: Noto Serif`
font-kerning	normal | none (default: *inherit*)	`title-page:` `  font-kerning: none`
font-size	Number (default: *inherit*)	`title-page:` `  font-size: 13`
font-style	Font style (default: *inherit*)	`title-page:` `  font-style: bold`
text-transform	Text transform (default: *inherit*)	`title-page:` `  text-transform: uppercase`
line-height	Number (default: 1.15)	`title-page:` `  line-height: 1`
Key Prefix: title-page-logo		

align	Image alignment
	(default: *inherit*)
		`title-page:`
		`  logo:`
		`  align: right`

image	image macro[3]
	(default: *not set*)
		`title-page:`
		`  logo:`
		`  image:`
		`image:logo.png[pdfwidth=25%]`

top	Measurement[4]
	(default: 10%)
		`title-page:`
		`  logo:`
		`  top: 25%`

Key Prefix: title-page-title		

display	none
	(default: *not set*)
		`title-page:`
		`  title:`
		`  display: none`

Key	Value Type	Example

font-color	Color
	(default: *inherit*)
		`title-page:`
		`  title:`
		`  font-color: #999999`

font-family	Font family name
	(default: *inherit*)
		`title-page:`
		`  title:`
		`  font-family: Noto Serif`

font-kerning	normal | none
	(default: *inherit*)
		`title-page:`
		`  title:`
		`  font-kerning: none`

font-size	Number
	(default: 18)
		`title-page:`
		`  title:`
		`  font-size: $heading-h1-font-size`

font-style	Font style
	(default: *inherit*)
		`title-page:`
		`  title:`
		`  font-style: bold`

text-transform	Text transform
	(default: *inherit*)
		`title-page:`
		`  title:`
		`  text-transform: uppercase`

line-height	Number
	(default: $heading-line-height)
		`title-page:`
		`  title:`
		`  line-height: 0.9`

top	Measurement[4]
	(default: 40%)
		`title-page:`
		`  title:`
		`  top: 55%`

margin-top	Measurement
	(default: 0)
		`title-page:`
		`  title:`
		`  margin-top: 13.125`

margin-bottom	Measurement
	(default: 0)
		`title-page:`
		`  title:`
		`  margin-bottom: 5`

Key Prefix: title-page-subtitle		

display	none
	(default: *not set*)
		`title-page:`
		`  subtitle:`
		`  display: none`

font-color	Color
	(default: *inherit*)
		`title-page:`
		`  subtitle:`
		`  font-color: #181818`

font-family	Font family name
	(default: *inherit*)
		`title-page:`
		`  subtitle:`
		`  font-family: Noto Serif`

font-kerning	normal | none
	(default: *inherit*)
		`title-page:`
		`  subtitle:`
		`  font-kerning: none`

Key	Value Type	Example

font-size	Number
	(default: 14)
		`title-page:`
		`  subtitle:`
		`  font-size: $heading-h3-font-size`

font-style	Font style
	(default: *inherit*)
		`title-page:`
		`  subtitle:`
		`  font-style: bold_italic`

text-transform	Text transform
	(default: *inherit*)
		`title-page:`
		`  subtitle:`
		`  text-transform: uppercase`

line-height	Number
	(default: $heading-line-height)
		`title-page:`
		`  subtitle:`
		`  line-height: 1`

margin-top	Measurement
	(default: 0)
		`title-page:`
		`  subtitle:`
		`  margin-top: 13.125`

margin-bottom	Measurement
	(default: 0)
		`title-page:`
		`  subtitle:`
		`  margin-bottom: 5`

Key Prefix: title-page-authors		

content	Quoted AsciiDoc string
	(optional subkeys: name_only, with_email, with_url)
	(default: "{author}")
		`title-page:`
		`  authors:`
		`  content:`
		`  name_only: "{author}"`
		`  with_email: "{author} <{email}>"`
		`  with_url: "{url} [{author}]"`

display	none
	(default: *not set*)
		`title-page:`
		`  authors:`
		`  display: none`

delimiter	Quoted string
	(default: ', ')
		`title-page:`
		`  authors:`
		`  delimiter: '; '`

font-color	Color
	(default: *inherit*)
		`title-page:`
		`  authors:`
		`  font-color: #181818`

font-family	Font family name
	(default: *inherit*)
		`title-page:`
		`  authors:`
		`  font-family: Noto Serif`

font-kerning	normal | none
	(default: *inherit*)
		`title-page:`
		`  authors:`
		`  font-kerning: none`

Key	Value Type	Example

font-size	Number (default: *inherit*)	`title-page:`
		`  authors:`
		`  font-size: 13`

font-style	Font style (default: *inherit*)	`title-page:`
		`  authors:`
		`  font-style: bold_italic`

text-transform	Text transform (default: *inherit*)	`title-page:`
		`  authors:`
		`  text-transform: uppercase`

margin-top	Measurement (default: 12)	`title-page:`
		`  authors:`
		`  margin-top: 13.125`

margin-bottom	Measurement (default: 0)	`title-page:`
		`  authors:`
		`  margin-bottom: 5`

Key Prefix: title-page-revision		

display	none (default: *not set*)	`title-page:`
		`  revision:`
		`  display: none`

delimiter	Quoted string (default: ', ')	`title-page:`
		`  revision:`
		`  delimiter: ': '`

font-color	Color (default: *inherit*)	`title-page:`
		`  revision:`
		`  font-color: #181818`

font-family	Font family name (default: *inherit*)	`title-page:`
		`  revision:`
		`  font-family: Noto Serif`

font-kerning	normal | none (default: *inherit*)	`title-page:`
		`  revision:`
		`  font-kerning: none`

font-size	Number (default: *inherit*)	`title-page:`
		`  revision:`
		`  font-size: $base-font-size-small`

font-style	Font style (default: *inherit*)	`title-page:`
		`  revision:`
		`  font-style: bold`

text-transform	Text transform (default: *inherit*)	`title-page:`
		`  revision:`
		`  text-transform: uppercase`

margin-top	Measurement (default: 0)	`title-page:`
		`  revision:`
		`  margin-top: 13.125`

Key	Value Type	Example

margin-bottom	Measurement (default: 0)	`title-page:`
		`  revision:`
		`  margin-bottom: 5`

1. To disable the background color for the title page, set the value to white (i.e., FFFFFF). The color keyword `transparent` is not recognized in this context.
2. By default, page background images are automatically scaled to fit the bounds of the page (i.e., `fit=contain`) and centered (i.e., `position=center`). The size of the background image can be controlled using any of the sizing attributes on the image macro (i.e., fit, pdfwidth, scaledwidth, or width) when `fit=none`. The position of the background image can be controlled using the `position` attribute.
3. Target may be an absolute path or a path relative to the value of the `pdf-themesdir` attribute.
4. % unit is relative to content height; vh unit is relative to page height.

Prose

The keys in this category control the spacing around paragraphs (paragraph blocks, paragraph content of a block, and other prose content). Typically, all the margin is placed on the bottom.

Key	Value Type	Example
Key Prefix: prose		
margin-top	Measurement (default: 0)	`prose:` `  margin-top: 0`
margin-bottom	Measurement (default: 12)	`prose:` `  margin-bottom: $vertical-spacing`
margin-inner[1]	Measurement (default: $prose-margin-bottom)	`prose:` `  margin-inner: 0`
text-indent	Measurement (default: *not set*)	`prose:` `  text-indent: 18`

1. Controls the margin between adjacent paragraphs. Useful when using indented paragraphs.

Block

The keys in this category control the spacing around block elements when a more specific setting is not designated.

Key	Value Type	Example
Key Prefix: block		
margin-top	Measurement (default: 0)	`block:` `  margin-top: 6`
margin-bottom	Measurement (default: 12)	`block:` `  margin-bottom: 6`

Block styles are applied to the following block types:

- admonition
- example
- quote
- verse
- sidebar
- image
- listing
- literal
- table

Caption

The keys in this category control the arrangement and style of block captions. In addition to the generic caption category, each of these keys can be set on the caption key nested inside the following block categories: blockquote, code, example, footnotes, image, listing, table, and verse.

Key	Value Type	Example
Key Prefix: caption		
align[1]	Text alignment (default: left)	`caption:` `  align: left`
font-color	Color (default: *inherit*)	`caption:` `  font-color: #333333`
font-family	Font family name (default: *inherit*)	`caption:` `  font-family: M+ 1mn`
font-kerning	normal | none (default: *inherit*)	`caption:` `  font-kerning: none`
font-size	Number (default: *inherit*)	`caption:` `  font-size: 11`
font-style	Font style (default: italic)	`caption:` `  font-style: italic`
text-transform	Text transform (default: *inherit*)	`caption:` `  text-transform: uppercase`
margin-inside	Measurement (default: 4)	`caption:` `  margin-inside: 3`
margin-outside	Measurement (default: 0)	`caption:` `  margin-outside: 0`

1. When nested inside the `image` key (i.e., `image-caption-align`), the value `inherit` is also accepted. The value `inherit` resolves to the alignment of the block image.

Code

The keys in this category are used to control the style of literal, listing and source blocks.

Key	Value Type	Example
Key Prefix: code		
background-color	Color (default: *not set*)	`code:` `  background-color: #f5f5f5`
border-color	Color (default: #eeeeee)	`code:` `  border-color: #cccccc`
border-radius	Number (default: *not set*)	`code:` `  border-radius: 4`
border-width	Number (default: 0.5)	`code:` `  border-width: 0.75`
font-color	Color (default: *inherit*)	`code:` `  font-color: #333333`
font-family	Font family name (default: Courier)	`code:` `  font-family: M+ 1mn`
font-size	Number (default: 10.8)	`code:` `  font-size: 11`
font-style	Font style (default: *inherit*)	`code:` `  font-style: italic`
line-height	Number (default: 1.2)	`code:` `  line-height: 1.25`
line-gap[1]	Number (default: 0)	`code:` `  line-gap: 3.8`
padding	Measurement | Measurement[top,right,bottom,left] (default: 9)	`code:` `  padding: 11`
Key Prefix: code-highlight[2]		
background-color	Color (default: #FFFFCC)	`code:` `  highlight-background-color: #ffff00`
Key Prefix: code-linenum[3]		
font-color	Color (default: #999999)	`code:` `  linenum-font-color: #ccc`

1. The line-gap property is used to tune the height of the background color applied to a span of block text highlighted using Rouge.
2. The code-highlight category only applies when using Rouge as the source highlighter. Otherwise, the styles are controlled by the source highlighter theme.
3. The code-linenum category only applies when using Pygments as the source highlighter. Otherwise, the styles are controlled by the source highlighter theme.

Callout Numbers

The keys in this category are used to control the style of callout numbers (i.e., conums) inside verbatim blocks and in callout lists (colists).

Key	Value Type	Example
Key Prefix: conum		
font-color	Color (default: *inherit*)	`conum:` `  font-color: #b12146`
font-family[1,2]	Font family name (default: *inherit*)	`conum:` `  font-family: M+ 1mn`
font-kerning[2]	normal | none (default: *inherit*)	`conum:` `  font-kerning: none`
font-size[2]	Number (default: *inherit*)	`conum:` `  font-size: $base-font-size`
font-style[2]	Font style (default: *inherit*)	`conum:` `  font-style: normal`
line-height[2]	Number (default: 1.15)	`conum:` `  line-height: 4 / 3`
glyphs[2]	circled | filled | Unicode String ranges (default: circled)	`conum:` `  glyphs: \u0031-\u0039`

1. Currently, the font must contain the circle numbers starting at glyph U+2460.
2. font-family, font-kerning, font-size, font-style, and line-height are only used for markers in a colist. These properties are inherited for conums inside a verbatim block.
3. The font must provide the required glyphs. The glyphs can be specified as a comma-separated list of ranges, where the range values are Unicode numbers (e.g., \u2460).

Button

The keys in this category apply to a button reference (generated from the inline button macro).

Key	Value Type	Example
Key Prefix: button		
background-color	Color (default: *not set*)	`button:` `  background-color: #0000ff`
border-color[1]	Color (default: *not set*)	`button:` `  border-color: #cccccc`
border-offset[2]	Number (default: 0)	`button:` `  border-offset: 1.5`
border-radius	Number (default: 0)	`button:` `  border-radius: 2`
border-width	Number (default: $base-border-width)	`button:` `  border-width: 0.5`
content[3]	Quoted string (default: "%s")	`button:` `  content: "[\u2009%s\u2009]"`
font-color	Color (default: *inherit*)	`button:` `  font-color: #ffffff`
font-family	Font family name (default: Courier)	`button:` `  font-family: M+ 1mn`
font-size	Number (default: *inherit*)	`button:` `  font-size: 12`
font-style	Font style (default: bold)	`button:` `  font-style: normal`

1. The border is only used if a border color is specified and the border width is not explicitly set to 0.
2. The border offset is the amount that the background and border swells around the text. It does not affect the distance between the formatted phrase and the phrases that surround it.
3. The character sequence `%s` in the content key gets replaced with the button label.

Key

The keys in this category apply to a key reference (generated from the inline kbd macro).

Key	Value Type	Example
Key Prefix: key		
background-color	Color (default: *not set*)	`key:` `  background-color: #fafafa`
border-color[1]	Color (default: *not set*)	`key:` `  border-color: #cccccc`
border-offset[2]	Number (default: 0)	`key:` `  border-offset: 1.5`
border-radius	Number (default: 0)	`key:` `  border-radius: 2`
border-width	Number (default: $base-border-width)	`key:` `  border-width: 0.375`
separator[3]	Quoted string (default: "+")	`key:` `  separator: "\u2009+\u2009"`
font-color	Color (default: *inherit*)	`key:` `  font-color: #000`
font-family	Font family name (default: Courier)	`key:` `  font-family: $base-font-family`
font-size	Number (default: *inherit*)	`key:` `  font-size: 10.5`
font-style	Font style (default: italic)	`key:` `  font-style: normal`

1. The border is only used if a border color is specified and the border width is not explicitly set to 0.
2. The border offset is the amount that the background and border swells around the text. It does not affect the distance between the formatted phrase and the phrases that surround it.
3. The separator is only used for multi-key sequences.

Menu

The keys in this category apply to the menu label (generated from the inline menu macro).

Key	Value Type	Example
Key Prefix: menu		
caret-content	Quoted string (default: " \u203a ")	`menu:` `  caret-content: ' > '`

Blockquote

The keys in this category control the arrangement and style of quote blocks.

Key	Value Type	Example
Key Prefix: blockquote		
background-color	Color (default: *not set*)	`blockquote:` `  background-color: #dddddd`
border-width[1]	Number (default: 0)	`blockquote:` `  border-width: 0.5`
border-left-width[1]	Number (default: 4)	`blockquote:` `  border-left-width: 5`
border-color[1]	Color (default: #eeeeee)	`blockquote:` `  border-color: #dddddd`
font-color	Color (default: *inherit*)	`blockquote:` `  font-color: #333333`
font-family	Font family name (default: *inherit*)	`blockquote:` `  font-family: Noto Serif`
font-kerning	normal | none (default: *inherit*)	`blockquote:` `  font-kerning: none`
font-size	Number (default: *inherit*)	`blockquote:` `  font-size: 13`
font-style	Font style (default: *inherit*)	`blockquote:` `  font-style: bold`
text-transform	Text transform (default: *inherit*)	`blockquote:` `  text-transform: uppercase`
padding	Measurement | Measurement[top,right,bottom,left] (default: [6, 12, -6, 14])	`blockquote:` `  padding: [5, 10, -5, 12]`
Key Prefix: blockquote-cite		

font-size	Number
	(default: *inherit*)
		`blockquote:`
		`  cite:`
		`  font-size: 9`

font-color	Color
	(default: *inherit*)
		`blockquote:`
		`  cite:`
		`  font-color: #999999`

font-family	Font family name
	(default: *inherit*)
		`blockquote:`
		`  cite:`
		`  font-family: Noto Serif`

font-kerning	normal | none
	(default: *inherit*)
		`blockquote:`
		`  cite:`
		`  font-kerning: none`

Key	Value Type	Example

font-style	Font style (default: *inherit*)	`blockquote:`
		`  cite:`
		`  font-style: bold`

text-transform	Text transform (default: *inherit*)	`blockquote:`
		`  cite:`
		`  text-transform: uppercase`

1. If border-left-width is non-zero, the border is only applied to the left side. Otherwise, if border-width is non-zero, the border is drawn around the whole block.

Verse

The keys in this category control the arrangement and style of verse blocks.

Key	Value Type	Example
Key Prefix: verse		
background-color	Color (default: *not set*)	`verse:` `  background-color: #dddddd`
border-width[1]	Number (default: 0)	`verse:` `  border-width: 0.5`
border-left-width[1]	Number (default: 4)	`verse:` `  border-left-width: 5`
border-color[1]	Color (default: #eeeeee)	`verse:` `  border-color: #dddddd`
font-color	Color (default: *inherit*)	`verse:` `  font-color: #333333`
font-family	Font family name (default: *inherit*)	`verse:` `  font-family: M+ 1mn`
font-kerning	normal | none (default: *inherit*)	`verse:` `  font-kerning: none`
font-size	Number (default: *inherit*)	`verse:` `  font-size: 10`
font-style	Font style (default: *inherit*)	`verse:` `  font-style: bold`
text-transform	Text transform (default: *inherit*)	`verse:` `  text-transform: uppercase`
padding	Measurement | Measurement[top,right,bottom,left] (default: [6, 12, -6, 14])	`verse:` `  padding: [5, 10, -5, 12]`
Key Prefix: verse-cite		

font-size	Number
	(default: *inherit*)
		`verse:`
		`  cite:`
		`  font-size: 9`

font-color	Color
	(default: *inherit*)
		`verse:`
		`  cite:`
		`  font-color: #999999`

font-family	Font family name
	(default: *inherit*)
		`verse:`
		`  cite:`
		`  font-family: Noto Serif`

font-kerning	normal | none
	(default: *inherit*)
		`verse:`
		`  cite:`
		`  font-kerning: none`

Key	Value Type	Example

font-style	Font style (default: *inherit*)	`verse:`
		`  cite:`
		`  font-style: italic`

text-transform	Text transform (default: *inherit*)	`verse:`
		`  cite:`
		`  text-transform: uppercase`

1. If border-left-width is non-zero, the border is only applied to the left side. Otherwise, if border-width is non-zero, the border is drawn around the whole block.

Sidebar

The keys in this category control the arrangement and style of sidebar blocks.

Key	Value Type	Example
Key Prefix: sidebar		
background-color	Color (default: #eeeeee)	`sidebar:` `  background-color: #eeeeee`
border-color	Color (default: *not set*)	`sidebar:` `  border-color: #ffffff`
border-radius	Number (default: *not set*)	`sidebar:` `  border-radius: 4`
border-width	Number (default: *not set*)	`sidebar:` `  border-width: 0.5`
font-color	Color (default: *inherit*)	`sidebar:` `  font-color: #262626`
font-family	Font family name (default: *inherit*)	`sidebar:` `  font-family: M+ 1p`
font-kerning	normal | none (default: *inherit*)	`sidebar:` `  font-kerning: none`
font-size	Number (default: *inherit*)	`sidebar:` `  font-size: 13`
font-style	Font style (default: *inherit*)	`sidebar:` `  font-style: italic`
text-transform	Text transform (default: *inherit*)	`sidebar:` `  text-transform: uppercase`
padding	Measurement | Measurement[top,right,bottom,left] (default: [12, 12, 0, 12])	`sidebar:` `  padding: [12, 15, 0, 15]`
Key Prefix: sidebar-title		

align	Text alignment
	(default: center)
		`sidebar:`
		`  title:`
		`  align: center`

font-color	Color
	(default: *inherit*)
		`sidebar:`
		`  title:`
		`  font-color: #333333`

font-family	Font family name
	(default: *inherit*)
		`sidebar:`
		`  title:`
		`  font-family: Noto Serif`

font-kerning	normal | none
	(default: *inherit*)
		`sidebar:`
		`  title:`
		`  font-kerning: none`

Key	Value Type	Example

font-size	Number
	(default: *inherit*)
		`sidebar:`
		`  title:`
		`  font-size: 13`

font-style	Font style
	(default: bold)
		`sidebar:`
		`  title:`
		`  font-style: bold`

text-transform	Text transform
	(default: *inherit*)
		`sidebar:`
		`  title:`
		`  text-transform: uppercase`

Example

The keys in this category control the arrangement and style of example blocks.

Key	Value Type	Example
Key Prefix: example		
background-color	Color (default: #ffffff)	`example:` `  background-color: #fffef7`
border-color	Color (default: #eeeeee)	`example:` `  border-color: #eeeeee`
border-radius	Number (default: *not set*)	`example:` `  border-radius: 4`
border-width	Number (default: 0.5)	`example:` `  border-width: 0.75`
font-color	Color (default: *inherit*)	`example:` `  font-color: #262626`
font-family	Font family name (default: *inherit*)	`example:` `  font-family: M+ 1p`
font-kerning	normal | none (default: *inherit*)	`example:` `  font-kerning: none`
font-size	Number (default: *inherit*)	`example:` `  font-size: 13`
font-style	Font style (default: *inherit*)	`example:` `  font-style: italic`
text-transform	Text transform (default: *inherit*)	`example:` `  text-transform: uppercase`
padding	Measurement | Measurement[top,right,bottom,left] (default: [12, 12, 0, 12])	`example:` `  padding: [15, 15, 0, 15]`

Admonition

The keys in this category control the arrangement and style of admonition blocks and the icon used for each admonition type.

Key	Value Type	Example
Key Prefix: admonition		
column-rule-color	Color (default: #eeeeee)	`admonition:` `  column-rule-color: #aa0000`
column-rule-style	solid | double | dashed | dotted (default: solid)	`admonition:` `  column-rule-style: double`
column-rule-width	Number (default: 0.5)	`admonition:` `  column-rule-width: 0.5`
font-color	Color (default: *inherit*)	`admonition:` `  font-color: #999999`
font-family	Font family name (default: *inherit*)	`admonition:` `  font-family: Noto Sans`
font-kerning	normal | none (default: *inherit*)	`admonition:` `  font-kerning: none`
font-size	Number (default: *inherit*)	`admonition:` `  font-size: $base-font-size-large`
font-style	Font style (default: *inherit*)	`admonition:` `  font-style: italic`
text-transform	Text transform (default: *inherit*)	`admonition:` `  text-transform: none`
padding	Measurement | Measurement[top,right,bottom,left] (default: [0, 12, 0, 12])	`admonition:` `  padding: [0, 12, 0, 12]`
Key Prefix: admonition-label		

align	Text alignment
	(default: center)
		`admonition:`
		`  label:`
		`  align: center`

min-width	Measurement
	(default: *not set*)
		`admonition:`
		`  label:`
		`  min-width: 48`

padding[1]	Measurement | Measurement[top,right,bottom,left] (default: $admonition-padding)	`admonition:` `  padding: [0, 12, 0, 12]`

vertical-align	top | middle | bottom
	(default: middle)
		`admonition:`
		`  label:`
		`  vertical-align: top`

Key Prefix: admonition-label, admonition-label-<name>[2]		

Key	Value Type	Example

font-color	Color
	(default: *inherit*)
		`admonition:`
		`  label:`
		`  font-color: #262626`

font-family	Font family name
	(default: *inherit*)
		`admonition:`
		`  label:`
		`  font-family: M+ 1p`

font-kerning	normal | none
	(default: *inherit*)
		`admonition:`
		`  label:`
		`  font-kerning: none`

font-size	Number
	(default: *inherit*)
		`admonition:`
		`  label:`
		`  font-size: 12`

font-style	Font style
	(default: bold)
		`admonition:`
		`  label:`
		`  font-style: bold_italic`

text-transform	Text transform
	(default: uppercase)
		`admonition:`
		`  label:`
		`  text-transform: lowercase`

Key Prefix: admonition-icon-<name>[2]		

name	<icon set>-<icon name>[3]
	(default: *not set*)
		`admonition:`
		`  icon:`
		`  tip:`
		`  name: fas-fire`

stroke-color	Color
	(default: caution=#bf3400; important=#bf0000; note=#19407c; tip=#111111; warning=#bf6900)
		`admonition:`
		`  icon:`
		`  important:`
		`  stroke-color: ff0000`

size	Number
	(default: 24)
		`admonition:`
		`  icon:`
		`  note:`
		`  size: 24`

1. The top and bottom padding values are ignored on admonition-label-padding.
2. `<name>` can be `note`, `tip`, `warning`, `important`, or `caution`. All icon types must be grouped under a single `icons` category. In other words, *do not* declare the `icons` category multiple times. The subkeys in the icon category cannot be flattened (e.g., `tip-name: far-lightbulb` is not valid syntax).

3. Required. See the `.yml` files in the prawn-icon repository for a list of valid icon names. The prefix (e.g., `fas-`) determines which font set to use. If the prefix is not specified, `fa-` is assumed.

Image

The keys in this category control the arrangement of block images.

Key	Value Type	Example
Key Prefix: image		
align	Image alignment (default: left)	`image:` `  align: left`
width[1]	Measurement (default: *not set*)	`image:` `  width: 100%`
border-color[2]	Color (default: *not set*)	`image:` `  border-color: #cccccc`
border-radius	Number (default: *not set*)	`image:` `  border-radius: 2`
border-width[2]	Number (default: *not set*)	`image:` `  border-width: 0.5`
border-fit[3]	content | auto (default: content)	`image:` `  border-fit: auto`
Key Prefix: image-alt		

content[4]	Quoted string
	(default: "%{link}[%{alt}]%{/link} | %{target}")
		`image:`
		`  alt:`
		`  content: "%{alt} (%{target})"`

font-color	Color
	(default: *inherit*)
		`image:`
		`  alt:`
		`  font-color: #ff000`

font-family	Font family name
	(default: *inherit*)
		`image`
		`  alt:`
		`  font-family: Courier`

font-kerning	normal | none
	(default: *inherit*)
		`image:`
		`  alt:`
		`  font-kerning: none`

font-size	Number
	(default: *inherit*)
		`image:`
		`  alt:`
		`  font-size: 9`

font-style	Font style
	(default: normal)
		`image:`
		`  alt:`
		`  font-style: italic`

caption-align	Text alignment | inherit
	(default: $caption-align)
		`image:`
		`  caption:`
		`  align: inherit`

caption-max-width[5]	fit-content | none | Measurement
	(default: none)
		`image:`
		`  caption:`
		`  max-width: fit-content`

1. Only applies to block images. If specified, this value takes precedence over the value of the `width` attribute on the image macro, but not over the value of the `pdfwidth` attribute.
2. The border is only used if a border color is specified, the border width is specified, the border width is greater than 0, and the `noborder` role is not present. The border is drawn above the image on the inside of the box reserved for the image.
3. The value `auto` means the border should expand to fit the width of the container (i.e., full width) instead of the image.
4. Use the placeholders `%{alt}`, `%{target}`, `%{link}`, and `%{/link}` to insert the alt text, image target, and link open/close tags into the content template.
5. In order for the image to be sized correctly when max-width is fit-content, a width should always be specified on the image.

SVG

The keys in this category control the SVG integration.

Key	Value Type	Example
Key Prefix: svg		
fallback_font_family[1]	Font family name (default: $base-font-family)	`svg:` `  fallback_font_family: Times-Roman`

1. The fallback font family is only used when the font family in the SVG does not map to a known font name from the font catalog.

Lead

The keys in this category control the styling of lead paragraphs.

Key	Value Type	Example
Key Prefix: lead		
font-color	Color (default: *inherit*)	`lead:` `  font-color: #262626`
font-family	Font family name (default: *inherit*)	`lead:` `  font-family: M+ 1p`
font-kerning	normal | none (default: *inherit*)	`lead:` `  font-kerning: none`
font-size	Number (default: 13.5)	`lead:` `  font-size: 13`
font-style	Font style (default: *inherit*)	`lead:` `  font-style: bold`

text-transform	Text transform
	(default: *inherit*)
		`lead:`
		`  text-transform:`
		`uppercase`

line-height	Number (default: 1.4)	`lead:` `  line-height: 1.4`

Abstract

The keys in this category control the arrangement and style of the abstract.

Key	Value Type	Example
Key Prefix: abstract		
font-color	Color (default: $base-font-color)	`abstract:` `  font-color: #5c6266`
font-size	Number (default: 13.5)	`abstract:` `  font-size: 13`
font-style	Font style (default: $base-font-style)	`abstract:` `  font-style: italic`
text-transform	Text transform (default: $base-text-transform)	`abstract:` `  text-transform: uppercase`
line-height	Number (default: 1.4)	`abstract:` `  line-height: 1.4`
padding	Measurement | Measurement[top,right,bottom,left] (default: 0)	`abstract:` `  padding: [0, 12, 0, 12]`
Key Prefix: abstract-first-line		

font-color	Color
	(default: *not set*)
		`abstract:`
		`  first-line:`
		`  font-color: #AA0000`

font-style	Font style
	(default: *not set*)
		`abstract:`
		`  first-line:`
		`  font-style: bold`

Key Prefix: abstract-title		

align	Text alignment
	(default: center)
		`abstract:`
		`  title:`
		`  align: center`

font-color	Color
	(default: $base-font-color)
		`abstract:`
		`  title:`
		`  font-color: #333333`

font-family	Font family name
	(default: $base-font-family)
		`abstract:`
		`  title:`
		`  font-family: Noto Serif`

font-kerning	normal | none
	(default: *inherit*)
		`abstract:`
		`  title:`
		`  font-kerning: none`

font-size	Number
	(default: $base-font-size)
		`abstract:`
		`  title:`
		`  font-size: 13`

font-style	Font style
	(default: bold)
		`abstract:`
		`  title:`
		`  font-style: bold`

Key	Value Type	Example

text-transform	Text transform
	(default: $base-text-transform)
		`abstract:`
		`  title:`
		`  text-transform: uppercase`

Thematic Break

The keys in this category control the style of thematic breaks (aka horizontal rules).

Key	Value Type	Example
Key Prefix: thematic-break		
border-color	Color (default: #eeeeee)	`thematic-break:` `  border-color: #eeeeee`
border-style	solid | double | dashed | dotted (default: solid)	`thematic-break:` `  border-style: dashed`
border-width	Measurement (default: 0.5)	`thematic-break:` `  border-width: 0.5`
margin-top	Measurement (default: 0)	`thematic-break:` `  margin-top: 6`
margin-bottom	Measurement (default: $vertical-spacing)	`thematic-break:` `  margin-bottom: 18`

Description List

The keys in this category control the arrangement and style of definition list items (terms and descriptions).

Asciidoctor PDF supports unordered and ordered description lists. These are defined as a description list, but get displayed as an unordered or ordered description list with the term as a subject. Only one term is supported. The subject is shown using the term font style (bold by default). The subject is stacked above the description if the "stack" role is present. Otherwise, the subject is arranged as a run-in followed by a subject stop (`:` by default). The subject stop can be customized using the `subject-stop` attribute.

```
[unordered]
alpha:: partially complete and unstable
beta:: feature complete and undergoing testing
```

Key	Value Type	Example
Key Prefix: description-list		
term-font-color	Color (default: *inherit*)	`description-list:` `  term-font-color: #AA0000`
term-font-family	Font family name (default: *inherit*)	`description-list:` `  term-font-family: Noto Serif`
term-font-kerning	normal | none (default: *inherit*)	`description-list:` `  term-font-kerning: none`
term-font-size	Number (default: *inherit*)	`description-list:` `  term-font-size: 12`
term-font-style	Font style (default: bold)	`description-list:` `  term-font-style: italic`
term-text-transform	Text transform (default: none)	`description-list:` `  term-text-transform: none`
term-line-height	Number (default: $base-line-height)	`description-list:` `  term-line-height: 1.2`
term-spacing	Measurement (default: 4)	`description-list:` `  term-spacing: 5`
description-indent	Number (default: 30)	`description-list:` `  description-indent: 15`

Outline List

The keys in this category control the arrangement and style of outline list items.

Key	Value Type	Example
Key Prefix: outline-list		
indent	Measurement (default: 30)	`outline-list:` `  indent: 40`
item-spacing	Measurement (default: 6)	`outline-list:` `  item-spacing: 4`
marker-font-color[1]	Color (default: *inherit*)	`outline-list:` `  marker-font-color: #3c763d`
text-align[2]	Text alignment (default: $base-align)	`outline-list:` `  text-align: left`

1. Controls the color of the bullet glyph that marks items in unordered lists and the number for items in ordered lists.
2. Controls the alignment of the list text only, not nested content (blocks or lists).

Unordered List

The keys in this category control the arrangement and style of unordered list items.

Key	Value Type	Example
Key Prefix: ulist-marker		

font-family	Font family name
	(default: *inherit*)
		`ulist:`
		`  marker:`
		`  font-family: Noto Serif`

font-size	Number
	(default: *inherit*)
		`ulist:`
		`  marker:`
		`  font-size: 9`

font-color	Color
	(default: $outline-list-marker-font-color)
		`ulist:`
		`  marker:`
		`  font-color: #cccccc`

line-height	Number
	(default: $base-line-height)
		`ulist:`
		`  marker:`
		`  line-height: 1.5`

Key	Value Type	Example
Key Prefix: ulist-marker-<type>[1]		

content	Quoted string	`ulist:`
		`  marker:`
		`  disc:`
		`  content: "\uf140"`

font-family	Font family name
	(default: *inherit*)
		`ulist:`
		`  marker:`
		`  disc:`
		`  font-family: fas`

font-size	Number
	(default: *inherit*)
		`ulist:`
		`  marker:`
		`  disc:`
		`  font-size: 9`

font-color	Color
	(default: *inherit*)
		`ulist:`
		`  marker:`
		`  disc:`
		`  font-color: #ff0000`

line-height	Number
	(default: *inherit*)
		`ulist:`
		`  marker:`
		`  disc:`
		`  line-height: 2`

1. <type> is one of disc, square, circle, checked, unchecked

Table

The keys in this category control the arrangement and style of tables and table cells.

Key	Value Type	Example
Key Prefix: table		
background-color	Color (default: transparent)	`table:` `  background-color: #ffffff`
border-color	Color (default: #000000)	`table:` `  border-color: #dddddd`
border-style	solid | dashed | dotted (default: solid)	`table:` `  border-style: solid`
border-width	Number (default: 0.5)	`table:` `  border-width: 0.5`
caption-align	Text alignment | inherit (default: $caption-align)	`table:` `  caption-align: inherit`
caption-side	top | bottom (default: top)	`table:` `  caption-side: bottom`
caption-max-width	fit-content | none | Measurement (default: fit-content)	`table:` `  caption-max-width: none`
font-color	Color (default: *inherit*)	`table:` `  font-color: #333333`
font-family	Font family name (default: *inherit*)	`table:` `  font-family: Helvetica`
font-kerning	normal | none (default: *inherit*)	`table:` `  font-kerning: none`
font-size	Number (default: *inherit*)	`table:` `  font-size: 9.5`
font-style	Font style (default: *inherit*)	`table:` `  font-style: italic`
grid-color	Color (default: $table-border-color)	`table:` `  grid-color: #eeeeee`
grid-style	solid | dashed | dotted (default: solid)	`table:` `  grid-style: dashed`
grid-width	Number (default: $table-border-width)	`table:` `  grid-width: 0.5`
Key Prefix: table-head		

background-color	Color
	(default: $table-background-color)
		`table:`
		`  head:`
		`  background-color: #f0f0f0`

Key	Value Type	Example

border-bottom-color	Color
	(default: $table-border-color)
		`table:`
		`  head:`
		`  border-bottom-color: #dddddd`

border-bottom-style	solid | dashed | dotted
	(default: solid)
		`table:`
		`  head:`
		`  border-bottom-style: dashed`

border-bottom-width	Number
	(default: 1.25)
		`table:`
		`  head:`
		`  border-bottom-width: 1`

font-color	Color
	(default: $table-font-color)
		`table:`
		`  head:`
		`  font-color: #333333`

font-family	Font family name
	(default: $table-font-family)
		`table:`
		`  head:`
		`  font-family: Noto Serif`

font-kerning	normal | none
	(default: *inherit*)
		`table:`
		`  head:`
		`  font-kerning: none`

font-size	Number
	(default: $table-font-size)
		`table:`
		`  head:`
		`  font-size: 10`

font-style	Font style
	(default: bold)
		`table:`
		`  head:`
		`  font-style: normal`

text-transform	Text transform
	(default: *inherit*)
		`table:`
		`  head:`
		`  text-transform: uppercase`

Key Prefix: table-body		

background-color	Color
	(default: $table-background-color)
		`table:`
		`  body:`
		`  background-color: #fdfdfd`

stripe-background-color[1]	Color
	(default: #eeeeee)
		`table:`
		`  body:`
		`  stripe-background-color: #efefef`

Key Prefix: table-foot		

background-color	Color
	(default: $table-background-color)
		`table:`
		`  foot:`
		`  background-color: #f0f0f0`

font-color	Color
	(default: $table-font-color)
		`table:`
		`  foot:`
		`  font-color: #333333`

Key	Value Type	Example

font-family	Font family name
	(default: $table-font-family)
		`table:`
		`  foot:`
		`  font-family: Noto Serif`

font-size	Number
	(default: $table-font-size)
		`table:`
		`  foot:`
		`  font-size: 10`

font-style	Font style
	(default: normal)
		`table:`
		`  foot:`
		`  font-style: italic`

Key Prefix: table-cell		

padding	Measurement |
	Measurement[top,right,bottom,left]
	(default: 2)
		`table:`
		`  cell:`
		`  padding: 3`

Key Prefix: table-header-cell		

background-color	Color
	(default: $table-head-background-color)
		`table:`
		`  header-cell:`
		`  background-color: #f0f0f0`

font-color	Color
	(default: $table-head-font-color)
		`table:`
		`  header-cell:`
		`  font-color: #1a1a1a`

font-family	Font family name
	(default: $table-head-font-family)
		`table:`
		`  header-cell:`
		`  font-family: Noto Sans`

font-size	Number
	(default: $table-head-font-size)
		`table:`
		`  header-cell:`
		`  font-size: 12`

font-style	Font style
	(default: $table-head-font-style)
		`table:`
		`  header-cell:`
		`  font-style: italic`

text-transform	Text transform
	(default: $table-head-text-transform)
		`table:`
		`  header-cell:`
		`  text-transform: uppercase`

1. This key only controls the color that is used for stripes. The appearance of stripes is controlled using the `stripes` table attribute, the `table-stripes` document attribute (since Asciidoctor 2), or the `stripes` document attribute (prior to Asciidoctor 2). Permitted attribute values are even, odd, all, and none. Prior to Asciidoctor 2, even rows are shaded by default (e.g., `stripes=even`). Since Asciidoctor 2, table stripes are not enabled by default (e.g., `stripes=none`).

Footnotes

The keys in this category control the style of the footnotes list at the end of the chapter (book) or document (otherwise). If the `footnotes-title` attribute is specified, it is styled as a block caption. The styling of the links is controlled by the global link styles.

Key	Value Type	Example
Key Prefix: footnotes		
font-color	Color (default: $base-font-color)	`footnotes:` `  font-color: #cccccc`
font-size	Number (default: 9)	`footnotes:` `  font-size: 8`
font-style	Font style (default: $base-font-style)	`footnotes:` `  font-style: italic`
item-spacing	Measurement (default: 3)	`footnotes:` `  item-spacing: 5`
margin-top	Measurement (default: 0)	`footnotes:` `  margin-top: 10`
text-transform	Text transform (default: *inherit*)	`footnotes:` `  text-transform: lowercase`

Table of Contents (TOC)

The keys in this category control the arrangement and style of the table of contents.

Key	Value Type	Example
Key Prefix: toc		
font-color	Color (default: *inherit*)	`toc:` `  font-color: #333333`
font-family	Font family name (default: *inherit*)	`toc:` `  font-family: Noto Serif`
font-kerning	normal | none (default: *inherit*)	`toc:` `  font-kerning: none`
font-size	Number (default: *inherit*)	`toc:` `  font-size: 9`
font-style	Font style (default: normal)	`toc:` `  font-style: bold`
text-decoration	Text decoration (default: none)	`toc:` `  text-decoration: underline`
text-decoration-color	Color (default: $toc-font-color)	`toc` `  text-decoration-color: #cccccc`
text-decoration-width	Number (default: 1)	`toc:` `  text-decoration-width: 0.5`
text-transform	Text transform (default: *inherit*)	`toc:` `  text-transform: uppercase`
line-height	Number (default: 1.4)	`toc:` `  line-height: 1.5`
indent	Measurement (default: 15)	`toc:` `  indent: 20`
hanging-indent	Measurement (default: *not set*)	`toc:` `  hanging-indent: 0.5in`
margin-top	Measurement (default: 0)	`toc:` `  margin-top: 0`
Key Prefix: toc-h<n>[1]		
font-color	Color (default: *inherit*)	`toc:` `  h3-font-color: #999999`
font-family	Font family name (default: *inherit*)	`toc:` `  h2-font-family: Noto Serif`
font-kerning	normal | none (default: *inherit*)	`toc:` `  h3-font-kerning: none`
font-size	Number (default: *inherit*)	`toc:` `  h3-font-size: 9`

Key	Value Type	Example
font-style	Font style (default: *inherit*)	`toc:` `  h2-font-style: italic`
text-transform	Text transform (default: *inherit*)	`toc:` `  h3-text-transform: uppercase`
Key Prefix: toc-title		

align	Text alignment
	(default: $heading-h2-align)
		`toc:`
		`  title:`
		`  align: right`

font-color	Color
	(default: $heading-h2-font-color)
		`toc:`
		`  title:`
		`  font-color: #aa0000`

font-family	Font family name
	(default: $heading-h2-font-family)
		`toc:`
		`  title:`
		`  font-family: Noto Serif`

font-kerning	normal | none
	(default: *inherit*)
		`toc:`
		`  title:`
		`  font-kerning: none`

font-size	Number
	(default: $heading-h2-font-size)
		`toc:`
		`  title:`
		`  font-size: 18`

font-style	Font style
	(default: $heading-h2-font-style)
		`toc:`
		`  title:`
		`  font-style: bold_italic`

text-transform	Text transform
	(default: $heading-h2-text-transform)
		`sidebar:`
		`  title:`
		`  text-transform: uppercase`

Key Prefix: toc-dot-leader		

content	Quoted string
	(default: '. ')
		`toc:`
		`  dot-leader:`
		`  content: ". "`

font-color[2]	Color
	(default: *inherit*)
		`toc:`
		`  dot-leader:`
		`  font-color: #999999`

font-style[2]	Font style
	(default: normal)
		`toc:`
		`  dot-leader:`
		`  font-style: bold`

levels[3]	all | none | Integers (space-separated)
	(default: all)
		`toc:`
		`  dot-leader:`
		`  levels: 2 3`

1. `<n>` is a number ranging from 1 to 6, representing each of the six heading levels.
2. The dot leader inherits all font properties except `font-style` from the root `toc` category.

3. 0-based levels (e.g., part = 0, chapter = 1). Dot leaders are only shown for the specified levels. If value is not specified, dot leaders are shown for all levels.

Running Content (Header & Footer)

The keys in this category control the arrangement and style of running header and footer content. Please note that the running content will *not* be used unless a) the periphery (header or footer) is configured and b) the height key for the periphery is assigned a value.

If the height of the running content periphery is larger than the page margin, the running content will cover the main content. To avoid this problem, reduce the height of the running content periphery or make the page margin on that side larger.

Key	Value Type	Example
Key Prefix: header		
background-color[1]	Color (default: *not set*)	`header:` `  background-color: #eeeeee`

background-image	image macro
	(default: *not set*)
		`header:`
		`  background-image:`
		`image:running-`
		`content.svg[fit=contain]`

border-color	Color (default: *not set*)	`header:` `  border-color: #dddddd`
border-style	solid | double | dashed | dotted (default: solid)	`header:` `  border-style: dashed`
border-width	Measurement (default: $base-border-width)	`header:` `  border-width: 0.25`
font-color	Color (default: *inherit*)	`header:` `  font-color: #333333`
font-family	Font family name (default: *inherit*)	`header:` `  font-family: Noto Serif`
font-kerning	normal | none (default: *inherit*)	`header:` `  font-kerning: none`
font-size	Number (default: *inherit*)	`header:` `  font-size: 9`
font-style	Font style (default: *inherit*)	`header:` `  font-style: italic`
height[2]	Measurement (default: *not set*)	`header:` `  height: 0.75in`
line-height	Number (default: $base-line-height)	`header:` `  line-height: 1.2`
padding[3]	Measurement | Measurement[top,right,bottom,left] (default: 0)	`header:` `  padding: [0, 3, 0, 3]`
image-vertical-align	top | middle | bottom | Measurement (default: *not set*)	`header:` `  image-vertical-align: 4`
sectlevels[4]	Integer (default: 2)	`header:` `  sectlevels: 3`
text-transform	Text transform (default: none)	`header:` `  text-transform: uppercase`

Key	Value Type	Example
title-style	document | toc | basic (default: document)	`header:` `  title-style: toc`
vertical-align	top | middle | bottom | [top | middle | bottom, Measurement] (default: middle)	`header:` `  vertical-align: middle`

<side>-columns[5]	Column specs triple
	(default: *not set*)
		`header:`
		`  recto:`
		`  columns: <25% =50% >25%`

<side>-<position>-content[5,6]	Quoted string
	(default: '{page-number}')
		`header:`
		`  recto:`
		`  left:`
		`  content: '\{page-number}'`

Key Prefix: footer		
background-color[1]	Color (default: *not set*)	`footer:` `  background-color: #eeeeee`
background-image	image macro (default: *not set*)	`footer:` `  background-image: image:running-content.svg[fit=contain]`
border-color	Color (default: *not set*)	`footer:` `  border-color: #dddddd`
border-style	solid | double | dashed | dotted (default: solid)	`footer:` `  border-style: dashed`
border-width	Measurement (default: $base-border-width)	`footer:` `  border-width: 0.25`
font-color	Color (default: *inherit*)	`footer:` `  font-color: #333333`
font-family	Font family name (default: *inherit*)	`footer:` `  font-family: Noto Serif`
font-kerning	normal | none (default: *inherit*)	`footer:` `  font-kerning: none`
font-size	Number (default: *inherit*)	`footer:` `  font-size: 9`
font-style	Font style (default: *inherit*)	`footer:` `  font-style: italic`
height[2]	Measurement (default: *not set*)	`footer:` `  height: 0.75in`
line-height	Number (default: $base-line-height)	`footer:` `  line-height: 1.2`

Key	Value Type	Example
padding[3]	Measurement | Measurement[top,right,bottom,left] (default: 0)	`footer:` `  padding: [0, 3, 0, 3]`
image-vertical-align	top | middle | bottom | Measurement (default: *not set*)	`footer:` `  image-vertical-align: 4`
sectlevels[4]	Integer (default: 2)	`footer:` `  sectlevels: 3`
text-transform	Text transform (default: none)	`footer:` `  text-transform: uppercase`
title-style	document | toc | basic (default: document)	`footer:` `  title-style: toc`
vertical-align	top | middle | bottom | [top | middle | bottom, Measurement] (default: middle)	`footer:` `  vertical-align: top`

<side>-columns[5]	Column specs triple (default: *not set*)	`footer:`
		`  verso:`
		`  columns: <50% =0% <50%`

<side>-<position>-content[5,6]	Quoted string (default: '{page-number}')	`footer:`
		`  verso:`
		`  center:`
		`  content: '\{page-number}'`

Key Prefix: running-content		
start-at[7]	title | toc | body | Integer (default: body)	`running-content:` `  start-at: toc`

1. The background color spans the width of the page, as does the border when a background color is specified.
2. **If the height is not set, the running content at this periphery is disabled.**
3. If the side padding is negative, the content will bleed into the margin of the page.
4. The maximum section level considered when assigning the implicit `section-title` attribute (and related) available to the running content.
5. `<side>` can be `recto` (right-hand, odd-numbered pages) or `verso` (left-hand, even-numbered pages). The `columns` key can also be defined one level up (on `header` or `footer`), in which case the setting will be inherited. Where the page sides fall in relation to the physical or printed page number is controlled using the `pdf-folio-placement` attribute (except when `media=prepress`, which implies `physical`).
6. `<position>` can be `left`, `center` or `right`.

7. Only works if the document uses a title page (i.e., doctype is book or `title-page` attribute is set) The `toc` value only applies if the toc is in the default location (before the first page of the body). If the toc macro is used to position the toc, the start-at behavior is the same as if the toc is not enabled. If value is an integer, the running content will start at the specified page of the body (i.e., 1 is first page, 2 is second page, etc.)

If you don't specify a height for either the header or footer key, it effectively disables the content at that periphery.

Although not listed in the table above, you can control the font settings (font-family, font-size, font-color, font-style, text-transform) that get applied to the running content in each column position for each page side (e.g., `footer-<side>-<position>-font-color`). For example, you can set the font color used for the right-hand column on recto pages by setting `footer-recto-right-font-color: 6CC644`.

Disabling

If you define running header and footer content in your theme (including the height), you can still disable this content per document by setting the `noheader` and `nofooter` attributes in the AsciiDoc document header, respectively.

If you extend either the base or default theme, and don't specify content for the footer, the current page number will be added to the right side on recto pages and the left side on verso pages. To disable this behavior, you can use the following snippet:

```
extends: default
footer:
  recto:
    right:
      content: ~
  verso:
    left:
      content: ~
```

Instead of erasing the content (which is what the `~` does), you can specify content of your choosing.

Replacing

If you want to replace the alternating page numbers with a centered page number, then you can restrict the footer to a single column and specify the content for the center position.

```
extends: default
footer:
  columns: =100%
  recto:
    center:
      content: '{page-number}'
  verso:
    center:
      content: '{page-number}'
```

In the last two examples, the recto and verso both have the same content. In this case, you can reduce the amount of configuring using a YAML reference. For example:

```
extends: default
footer:
  columns: =100%
  recto: &shared_footer
    center:
      content: '{page-number}'
  verso: *shared_footer
```

The `&shared_footer` assigns an ID to the YAML subtree under the `recto` key and the `*shared_footer` outputs a copy of it under the `verso` key. This technique can be used throughout the theme file as it's a core feature of YAML.

Attribute References

You can use *any* attribute defined in your AsciiDoc document (such as `doctitle`) in the content of the running header and footer. In addition, the following attributes are also available when defining the content keys in the footer:

- page-count
- page-number (only set if the `pagenums` attribute is set on the document, which it is by default)
- document-title
- document-subtitle
- part-title
- chapter-title
- section-title
- section-or-chapter-title

If you reference an attribute which is not defined, all the text on that same line in the running content will be dropped. This feature allows you to have alternate lines that are selected when all the attribute references are satisfied. One case where this is useful is when referencing the `page-number` attribute. If you unset the `pagenums` attribute on the document, any line in the running content that makes reference to `{page-number}` will be dropped.

You can also use built-in AsciiDoc text replacements like `(C)`, numeric character references like `&#169;`, hexadecimal character references like `&#x20ac;`, and inline formatting (e.g., bold, italic, monospace).

Here's an example that shows how attributes and replacements can be used in the running footer:

```
header:
  height: 0.75in
  line-height: 1
  recto:
    center:
      content: '(C) ACME -- v{revnumber}, {docdate}'
  verso:
    center:
      content: $header-recto-center-content
footer:
  height: 0.75in
  line-height: 1
  recto:
    right:
      content: '{section-or-chapter-title} | *{page-number}*'
  verso:
    left:
      content: '*{page-number}* | {chapter-title}'
```

Multiple Lines

You can split the content value across multiple lines using YAML's multiline string syntax. In this case, the single quotes around the string are not necessary. To force a hard line break in the output, add `+` to the end of the line in normal AsciiDoc fashion.

```
footer:
  height: 0.75in
  line-height: 1.2
  recto:
    right:
      content: |
        Section Title - Page Number +
        {section-or-chapter-title} - {page-number}
  verso:
    left:
      content: |
        Page Number - Chapter Title +
        {page-number} - {chapter-title}
```

You can use most AsciiDoc inline formatting in the values of these keys. For instance, to make the text bold, surround it in asterisks (as shown above). One exception to this rule are inline images, which are described in the next section.

Images

You can add an image to the running header or footer using the AsciiDoc inline image syntax. The image target is resolved relative to the value of the `pdf-themesdir` attribute. If the image macro is the whole value for a column position, you can use the `position` and `fit` attributes to align and scale it relative to the column box. Otherwise, the image is treated like a normal inline image, for which you can only adjust the width.

Here's an example of how to use an image in the running header (which also applies for the footer).

```
header:
  height: 0.75in
  image-vertical-align: 2 (1)
  recto:
    center:
      content: image:footer-logo.png[width=80]
  verso:
    center:
      content: $header-recto-center-content
```

(1) You can use the `image-vertical-align` key to slightly nudge the image up or down.

By default, the image must fit in the allotted space for the running header or footer. Otherwise, you will run into layout issues. Adjust the width attribute accordingly using the `pdfwidth` attribute. Alternatively, you can set the `fit` attribute to `scale-down` (e.g., `fit=scale-down`) to reduce the image size to fit in the available space or `contain` (i.e., `fit=contain`) to scale the image (up or down) to fit the available space.

Applying Your Theme

After creating a theme, you'll need to tell Asciidoctor PDF where to find it. This is done using AsciiDoc attributes.

There are three AsciiDoc attributes that tell Asciidoctor PDF how to locate and apply your theme.

pdf-theme (or pdf-style)

The name of the YAML theme file to load. If the name ends with `.yml`, it's assumed to be the complete name of a file and is resolved relative to `pdf-themesdir`, if specified, otherwise the current directory. Otherwise, `-theme.yml` is appended to the name to make the file name (i.e., `<name>-theme.yml`) and is resolved relative to `pdf-themesdir`, if specified, otherwise the built-in themes dir.

pdf-themesdir (or pdf-stylesdir)

The directory where the theme file is located. *Specifying an absolute path is recommended.*

If you use images in your theme, image paths are resolved relative to this directory. If `pdf-theme` ends with `.yml`, and `pdf-themesdir` is not specified, then `pdf-themesdir` defaults to the directory of the path specified by `pdf-theme`.

pdf-fontsdir

The directory or directories where the fonts used by your theme, if any, are located. Multiple entries must be separated by a semi-colon. To reference a file inside a JAR file on the classpath, prefix with the path with `uri:classloader:` (AsciidoctorJ only). *Specifying an absolute path is recommended.*

Let's assume that you've put your theme files inside a directory named `resources` with the following layout:

```
document.adoc
resources/
```

```
themes/
    basic-theme.yml
  fonts/
    roboto-normal.ttf
    roboto-italic.ttf
    roboto-bold.ttf
    roboto-bold_italic.ttf
```

Here's how you'd load your theme when calling Asciidoctor PDF:

```
$ asciidoctor-pdf -a pdf-themesdir=resources/themes -a pdf-theme=basic -a pdf-
fontsdir=resources/fonts
```

If all goes well, Asciidoctor PDF should run without an error or warning.

You only need to specify the `pdf-fontsdir` if you're using custom fonts in your theme.

You can skip setting the `pdf-themesdir` attribute and just pass the absolute path of your theme file to the `pdf-theme` attribute.

```
$ asciidoctor-pdf -a pdf-theme=resources/themes/basic-theme.yml -a pdf-
fontsdir=resources/fonts
```

However, in this case, image paths in your theme won't be resolved properly.

Paths are resolved relative to the current directory. However, in the future, this may change so that paths are resolved relative to the base directory (typically the document's directory). Therefore, it's recommend that you specify absolute paths for now to future-proof your configuration.

```
$ asciidoctor-pdf -a pdf-themesdir=/path/to/resources/themes -a pdf-theme=basic -a pdf-
fontsdir=/path/to/resources/fonts
```

As usual, you can also use build tools like Maven and Gradle to build a themed PDF. The only thing you need to add to an existing build is the attributes mentioned above.

- Maven Example
- Gradle Example

Speaking of Java, you can bundle and distribute your theme and fonts in a jar file. To reference the theme file and/or directory of fonts from inside the jar, refer to their location on the classpath using the `uri:classloader:` prefix. Here's how you'd load both the theme and fonts from the classpath:

```
$ asciidoctorj -b pdf -a pdf-theme="uri:classloader:/path/to/themes/my-theme.yml" -a
pdf-fontsdir="uri:classloader:/path/to/fonts" document.adoc
```

This only works when running Asciidoctor PDF on the JVM.

Theme-Related Document Attributes

There are various settings in the theme you control using document attributes. These settings override equivalent keys defined in the theme file, where applicable.

Attribute	Value Type	Example
autofit-option	flag (default: *not set*)	`:autofit-option:`
chapter-label	string (default: Chapter)	`:chapter-label: Chapitre`
<face>-cover-image[1]	path[2] | image macro[3] (format can be image or PDF)	`:front-cover-image: image:front-cover.pdf[]`
hyphens[7]	language code | *blank* to default to en_us (default: *not set*)	`:hyphens: de`
media	screen | print | prepress	`:media: prepress`
compress	flag (default: *not set*)	`:compress:`
optimize	screen | ebook | printer | prepress | default (default: default)	`:optimize: prepress`
outlinelevels[10]	Integer | Integer:Integer (default: same as *toclevels*)	`:outlinelevels: 2`
page-background-image[4]	path[2] | image macro[3]	`:page-background-image: image:bg.jpg[]`
page-background-image-(recto|verso)[4]	path[2] | image macro[3]	`:page-background-image-recto: image:bg-recto.jpg[]`
page-foreground-image	path[2] | image macro[3]	`:page-foreground-image: image:watermark.svg[]`
pagenums[5]	flag (default: *set*)	`:pagenums:`
pdf-page-layout	portrait | landscape	`:pdf-page-layout: landscape`
pdf-page-margin	Measurement | Measurement[top,right,bottom,left]	`:pdf-page-margin: [1in, 0.5in]`
pdf-page-mode	outline | none | thumbs | fullscreen | fullscreen outline | fullscreen none | fullscreen thumbs (default: outline)	`:pdf-page-mode: fullscreen none`
pdf-page-size	Named size | Measurement[width, height]	`:pdf-page-size: [6in, 9in]`
pdf-folio-placement	virtual | virtual-inverted | physical | physical-inverted	`:pdf-folio-placement: physical`
pdf-version	1.3 | 1.4 | 1.5 | 1.6 | 1.7 (default: 1.4)	`:pdf-version: 1.7`
pdfmark[6]	flag (default: *not set*)	`:pdfmark:`

Attribute	Value Type	Example
scripts[7]	cjk (default: *not set*)	`:scripts: cjk`
text-align[8]	Text alignment	`:text-align: left`
title-logo-image	path[2] | image macro[3]	`:title-logo-image: image:logo.png[top=25%, align=center, pdfwidth=0.5in]`
title-page[9]	flag (default: *not set*)	`:title-page:`
title-page-background-image	path[2] | image macro[3]	`:title-page-background-image: image:title-bg.jpg[]`
toc-max-pagenum-digits[10]	Integer (default: 3)	`:toc-max-pagenum-digits: 4`

1. `<face>` can be `front` or `back`.
2. The path is resolved relative to base_dir.
3. The target of the image macro is resolved relative to `imagesdir`. If the image macro syntax is not used, the value is resolved relative to the base directory, which defaults to the document directory.
4. By default, page background images are automatically scaled to fit the bounds of the page (i.e., `fit=contain`) and centered (i.e., `position=center`). The size of the background image can be controlled using any of the sizing attributes on the image macro (i.e., fit, pdfwidth, scaledwidth, or width) when `fit=none`. The position of the background image can be controlled using the `position` attribute. If the recto (right-hand, odd-numbered pages) or verso (left-hand, even-numbered pages) background is specified, it will be used only for that side. If a background image isn't specified for a side, the converter will use the default page background image (`page-background-image`), if specified. To disable the background image for a side, use the value `none`.
5. Controls whether the implicit `page-number` attribute is to the running header and footer content specified in the theme file. Instead of disabling page numbers, you can use the `noheader` and `nofooter` attributes to disable the running header and footer, respectively.
6. Enables generation of the pdfmark file, which contains metadata that is fed to Ghostscript when optimizing the PDF file.
7. Activates line break rules for CJK languages (specifically Chinese and Japanese). Chinese and Japanese are written without spaces (and may not use spaces when mixing with English words either). This setting allows a line break to be placed between any two CJK characters to accommodate wrapping. These languages also use different punctuation for pause, full stop, and dash, which are taken into account when breaking lines.
8. *(Experimental)* The `text-align` document attribute is intended as a simple way to toggle text justification. The value of this attribute overrides the `base-align` key set by the theme. For more fine-grained control, you should customize using the theme.
9. The title page is only enabled by default for the book doctype. To force the title page to be used for other doctypes, set the `title-page` attribute in the document header.

10. If the TOC overlaps the first page of content, increase this number.
11. The second number in the value of `outlinelevels` is the number of levels of the outline to expand (e.g., `3:1`). If the second number is not present, all levels are expanded.

Publishing Mode

Asciidoctor PDF provides the following features to assist with publishing:

- Double-sided (mirror) page margins
- Automatic facing pages

These features are activated when you set the `media` attribute to `prepress` in the header of your AsciiDoc document or from the CLI or API. The following sections describe the behaviors that this setting activates.

Double-Sided Page Margins

The page margins for the recto (right-hand, odd-numbered) and verso (left-hand, even-numbered) pages are automatically calculated by replacing the side page margins with the values of the `page-margin-inner` and `page-margin-outer` keys.

For example, let's assume you've defined the following settings in your theme:

```
page:
  margin: [0.5in, 0.67in, 0.67in, 0.67in]
  margin-inner: 0.75in
  margin-outer: 0.59in
```

The page margins for the recto and verso pages will be resolved as follows:

recto page margin

[0.5in, **0.59in**, 0.67in, **0.75in**]

verso page margin

[0.5in, **0.75in**, 0.67in, **0.59in**]

The page margins alternate between recto and verso. The first page in the document (after the cover) is a recto page.

If no cover is specified, the recto margin is not applied to the title page. To apply the recto margin to the title page, but not include a cover, assign the value `~` to the `front-cover-image` attribute.

Automatic Facing Pages

When converting the book doctype using the prepress media setting, a blank page will be inserted when necessary to ensure the following elements start on a recto page:

- Title page
- Table of contents

- First page of body
- Parts and chapters

Other “facing” pages may be added in the future.

It’s possible to disable the automatic facing feature for a given part or chapter. This can be done by adding the nonfacing option to the section node. When the nonfacing option is present, the part or chapter title will be placed on the next adjacent page rather than the next facing page.

```
[%nonfacing]
= Minor Chapter

content
```

For documents that use the article doctype, Asciidoctor PDF incorrectly places the document title and table of contents on their own pages. This can result in the page numbering and the page facing to be out of sync. As a workaround, Asciidoctor PDF inserts a blank page, if necessary, to ensure the first page of body content is a recto-facing page.

You can check on the status of this defect by following issue #95.

Source Highlighting Theme

You can define and apply your own source highlighting theme to source blocks when using Rouge as the source highlighter. This section explains how.

A custom theme for Rouge is defined using a Ruby class. Start by creating a Ruby source file to define your theme. Name the file according to the name of your theme and put the file in a folder of your choice (e.g., *rouge_themes/custom.rb*). The name of the Ruby class doesn’t matter, though it’s customary to name it according to the name of the theme as well.

rouge_themes/custom.rb

```
require 'rouge' unless defined? ::Rouge.version

module Rouge; module Themes
  class Custom < CSSTheme
    name 'custom'

    style Comment,           fg: '#008800', italic: true
    style Error,             fg: '#a61717', bg: '#e3d2d2'
    style Str,               fg: '#0000ff'
    style Str::Char,         fg: '#800080'
    style Num,               fg: '#0000ff'
    style Keyword,           fg: '#000080', bold: true
    style Operator::Word,    bold: true
    style Name::Tag,         fg: '#000080', bold: true
    style Name::Attribute,   fg: '#ff0000'
     style Generic::Deleted,  fg: '#000000', bg: '#ffdddd', inline_block: true, extend:
true
     style Generic::Inserted, fg: '#000000', bg: '#ddffdd', inline_block: true, extend:
true
    style Text, {}
  end
end; end
```

Each style declaration accepts the following properties:

- `fg` - sets the foreground (text) color
- `bg` - sets the background color
- `bold` - change the font weight to bold
- `italic` - change the font style to italic
- `underline` - add an underline to the text
- `inline_block` - fill the background color to the height of the line (Asciidoctor PDF only)
- `extend` - extend the background color to the end of the line for a line-oriented match (Asciidoctor PDF only)

Colors are defined using hexadecimal format (e.g., #ff0000 for red).

Use the `Text` token to set the background color of the source block and the default text color.

The complete list of tokens can be found in the token.rb file from Rouge. Refer to the bundled themes to find more examples.

Once you've defined your theme, you need to enable it use it using the `rouge-style` document attribute, which you specify in the document header or via the Asciidoctor CLI or API.

```
:source-highlighter: rouge
:rouge-style: custom
```

Finally, you need to active your theme by requiring the theme file when you invoke Asciidoctor.

```
$ asciidoctor -r ./rouge_themes/custom.rb sample.adoc
```

You should now see that the source code is highlighted to your liking. For more information about source highlighting with Rouge, refer to the Rouge project page.

Extending the Converter

This converter uses Prawn under the covers to generate the PDF. Prawn is a low-level PDF writer that can load fonts, ink text, embed images, add graphics, and draw lines. With those operations alone, this converter manages to produce a PDF from an AsciiDoc document. This section explains the role of theming in this process and how to extend the converter to take it further.

Going Beyond Theming

While creating the PDF document, there are thousands of small decisions the converter must make about how to instruct Prawn to layout the content elements on the page (so-called "hackable typesetting"). But once these elements are written, they can't be moved or styled (as is the case with HTML and CSS). To help influence those decisions—and thus prevent the converter from becoming too opinionated, a theming system was introduced. That theming system is described in this document.

The theme support is there to provide basic customizations (fonts, colors, borders, spacing, etc.). But it can only go so far. At some point, it becomes necessary to extend the converter to meet advanced design requirements.

Before you dive into extending this converter, you'll need to understand how to use Prawn. The article Hackable PDF Typesetting in Ruby with Prawn gives a crash course in how to create your first PDF document containing text, graphics, and images with Prawn. That article is essential reading for working with Asciidoctor PDF, which uses many of the same operations (as well as many helpful add-ons). Once you feel comfortable with Prawn, you're ready to extend the converter.

Tailoring Conversion

The methods on a converter class that handle conversion follow the pattern `convert_<name>` for block elements (e.g., `convert_example`) and `convert_inline_<name>` for inline elements (e.g., `convert_inline_anchor`), where `<name>` is the name of the element.

When you override a block element, you write PDF objects directly to the Prawn Document (the current context). When you override an inline element, you return pseudo-HTML, which is then parsed and converted into PDF objects.

The pseudo-HTML in Asciidoctor PDF evolved from the inline formatting in Prawn, now supporting the following elements: a, br, button, code, color, del, em, font, img, key, mark, span, strong, sub, sup. All decimal and hexadecimal character references are supported, as well as the named entities amp, apos, gt, lt, nbsp, and quot (e.g., `&apos;`). You can change the font color using the `rgb` attribute on the `color` element (e.g., `<color rgb="#ff0000">`) and the font family and size using the `name` and `size` attributes on the `font` element, respectively (e.g., `<font name="sans" size="12">`). You can also use the `style` attribute on `span` to control the font color, weight, and style using the relevant CSS property names. The pseudo-HTML in Asciidoctor PDF also supports the `class` attribute on any element for applying

roles from the theme. (Though not recommended, you can pass this pseudo-HTML straight through to Prawn using an inline passthrough in AsciiDoc).

Defining the Extended Converter

Starting with Asciidoctor 2, defining an extending converter and registering it in place of the original is very straightforward.

custom-pdf-converter.rb

```
class CustomPDFConverter < (Asciidoctor::Converter.for 'pdf')
  register_for 'pdf'

  # overrides go here
end
```

As it stands, the converter doesn't do anything differently than the primary converter because we haven't yet overridden any of its methods. Let's do that now, starting by overriding the thematic break (aka horizontal rule) to make it render like a ribbon:

```
  def convert_thematic_break node
    theme_margin :thematic_break, :top
    stroke_horizontal_rule 'FF0000', line_width: 0.5, line_style: :solid
    move_down 1
    stroke_horizontal_rule 'FF0000', line_width: 1, line_style: :solid
    move_down 1
    stroke_horizontal_rule 'FF0000', line_width: 0.5, line_style: :solid
    theme_margin :thematic_break, :bottom
  end
```

This converter will replace the thematic break with a red ribbon.

Another way to override the converter is to modify the node, then delegate back to the primary converter. Let's put a page break before all paragraphs unless the cursor is at the top of the page. We'll call `super` to let the primary converter handle the work of rendering the paragraph.

```
  def convert_paragraph node
    bounds.move_past_bottom unless at_page_top?
    super
  end
```

Now let's look at how to modify an inline element. Let's say we want to override the kbd element.

```
  def convert_inline_kbd node
    %(<strong><color rgb="AA0000">#{(node.attr 'keys').join ' + '}</color></strong>)
  end
```

Refer to the primary converter to discover the pseudo-HTML you can use for inline elements.

So far we've just been biting around the edges. A more realistic use case is to customize the part title page in a multi-part book. Since this is a specialized section element, there's a dedicated method named

`layout_part_title` that you'll need to override.

Let's customize the part title page by making the background orange, making the font white, centering the title on the page, and disabling the running content. (You don't need to start a new page before and after the part title since that's already done for you).

```
  def layout_part_title node, title, opts = {}
    fill_absolute_bounds 'E64C3D'
    move_down 20
     typeset_text title, (calc_line_metrics 1.5), color: 'FFFFFF', inline_format: true,
align: :center, size: 42
    page.instance_variable_set :@imported_page, true
  end
```

The method `typeset_text` and `calc_line_metrics` are provided by Asciidoctor PDF to make writing text easier. If you wanted, you could just use the low-level `text` method provided by Prawn.

To find all the available methods to override, consult the API docs. For deeper examples of how to override the behavior of the converter, refer to the extended converter in the InfoQ Mini-Book template.

Now that you've seen some examples of how to extend the converter, let's look at how to use it.

Using the Extended Converter

To use this converter, register it by passing the path to the `-r` flag when calling the `asciidoctor-pdf` command:

```
$ asciidoctor-pdf -r ./custom-pdf-converter.rb document.adoc
```

That's all there is too it. Now you're hacking the typesetting!

Appendix A: Preparing a Custom Font

Any TTF font can be used with Prawn—and hence Asciidoctor PDF—without modifications (unless, of course, it's corrupt or contains errors). However, you may discover that kerning is disabled and certain required glyphs are missing (or replaced with boxes). To address these problems, you need to prepare the font using a font program such as FontForge.

Validate the Font

Before using the font, you may want to check that the font is valid. To do so, create the following script, which will verify that the TTF font is free from errors.

validate-font.rb

```
require 'ttfunk'
require 'ttfunk/subset_collection'

ttf_subsets = TTFunk::SubsetCollection.new TTFunk::File.open ARGV[0]
(0...(ttf_subsets.instance_variable_get           :@subsets).size).each           {|idx|
ttf_subsets[idx].encode }
```

Run the script on your font as follows:

```
$ ruby validate-font.rb path/to/font.ttf
```

If this script fails, the font will not work with Asciidoctor PDF. To repair it, open the font in FontForge and resave it using File Generate Fonts… Generate. Dismiss any warning dialogs.

Resaving the font in FontForge will usually resolve any errors in the font. (If not, you may need to find another font, or at least another copy of it).

Modifying the Font

To ready your font for use with Asciidoctor PDF, you'll need to modify it using a font program. We recommend using FontForge. But don't let this scare you off. FontForge essentially works like a vector-drawing tool, in which each character is a separate canvas. You can find a crash course in how to use the program on the FontForge project site.

Here are the modifications you need to apply to a custom font for it to work best with Asciidoctor PDF:

- Convert the font to TTF (only required if the font is not already a TTF, such as an OTF or TTC).
- Add the glyphs for the required characters if missing from the font (optional if using a fallback font).
- Subset the font to exclude unused characters to reduce the file size (optional).
- Save the file using the old-style kern table to activate kerning.

Technically, subsetting the font (i.e., removing glyphs) is not required since Prawn only embeds the characters from the font used in the document (i.e., it automatically subsets the font). However, if you plan to commit the font to a repository, subsetting helps keep the file size down.

Most fonts do not provide glyphs for all the Unicode character ranges (i.e., scripts). (A glyph is the corresponding vector image for a Unicode character). In fact, many fonts only include glyphs for Latin (Basic, Supplement, and Extended) and a few other scripts (e.g., Cyrillic, Greek). That means certain glyphs Asciidoctor PDF relies on may be missing from the font.

Below are the non-Latin characters (identified by Unicode code point) on which Asciidoctor PDF relies that are often missing from fonts. Unless you're using a fallback font that fills in the missing glyphs, you need to ensure these glyphs are present in your font (and add them if not).

- \u00a0 - no-break space
- \ufeff - zero width no-break space
- \u200b - zero width space (used for line break hints)

- \u000a - line feed character (zero width)
- \u2009 - thin spaced (used in the button UI element)
- \u202f - narrow no-break space (used in the keybinding UI element)
- \u2011 - non-breaking hyphen
- \u2022 - disc (used for first-level unordered list level)
- \u25e6 - circle (used for second-level unordered list level)
- \u25aa - square (used for third-level unordered list level)
- \u2611 - ballot box checked (used for checked list item)
- \u2610 - ballot box unchecked (used for unchecked list item)
- \u2014 - em-dash (used in quote attribute)
- \u203a - single right-pointing quotation mark (used in the menu UI element)
- \u25ba - right pointer (used for media play icon when icon fonts are disabled)
- .notdef - used as the default glyph when a requested character is missing from the font (usually a box)

If the .notdef glyph is non-empty (i.e., contains splines), it will be used as the default glyph when the document requests a character that is missing from the font. Unlike other glyphs, the .notdef glyph is referenced by name only. It does not have a designated Unicode code point.

If you're preparing a font for use in verbatim blocks (e.g., a listing block), you'll also need this range of characters:

- \u2460 to \u2468 - circled numbers

One way to get these glyphs is to steal them from another font (or from another character in the same font). To do so, open the other font in FontForge, select the character, press `Ctrl`+`c`, switch back to your font, select the character again, and press `Ctrl`+`v`. You may need to scale the glyph so it fits properly in the art box.

If you're copying a non-visible character, be sure to set the width to 0 using Metrics Set Width..., enter 0 into **Set Width To**, then click [**OK**].

When you're done, save the font with the old-style kern table enabled. To do so, select File Generate Fonts..., click [**Options**], and make sure only the following options are selected (equivalent to the flags 0x90 + 0x08):

OpenType

Old style 'kern'

Then click [**Generate**] to generate and save the font.

Your font file is now ready to be used with Asciidoctor PDF.

Scripting the Font Modifications

Performing all this font modification manually can be tedious (not to mention hard to reproduce). Fortunately, FontForge provides a scripting interface, which you can use to automate the process.

In fact, that's what we use to prepare the fonts that are bundled with Asciidoctor PDF. You can find that FontForge script, the Bash script that calls it, and the Docker image in which it is run in the scripts directory of this project. You can use that script as a starting point or reference for your own font preparation / modification script.