Asciidoctor PDF Theming Guide

Dan Allen

Table of Contents

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.

If you're creating a custom theme, you're expected to supply your own fonts. We recognize this can be a major obstacle when you're starting out. Therefore, your other option is to simply redeclare the fonts from the default theme in the font catalog. Asciidoctor PDF will then resolve the fonts that are bundled with the gem.

If you don't declare your own fonts, the built-in (AFM) fonts declared in base theme will be used instead. Using AFM fonts can result in missing functionality and warnings. See the Built-In (AFM) Fonts section to learn more about these limitations.

Language Overview

The theme language in Asciidoctor PDF is based on the YAML data format and incorporates many concepts from CSS and SASS. Therefore, if you have a background in web design, the theme language should be immediately familiar to you.

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.

- Underscores (`_`) can be used in place of hyphens (`-`) for all property names in the theme language.
- 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 `text_align` property from CSS is the `align` property in the theme language.
- The `color` property from CSS is the `font_color` property in the theme language.

A theme (or style) 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.

The theme file must be named *<name>-theme.yml*, where `<name>` is the name of the theme.

Here's an example of a basic theme file:

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
```

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.

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
```

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 (_). 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`).

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 hierarchically:

```
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 more spaces of indentation than 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., #ffffff)
  - 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)
- 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: #E0162B # ①
  secondary: '#FFFFFF' # ②
  alert: '0052A5' # ③
```

① 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 it 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.

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

```
base:
  font_color: $brand_primary
```

Math Expressions & Functions

The theme language supports basic math operations to support calculated values. Like programming languages, multiple 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.

Numbers with more than two digits should be written as a float (e.g., 100.0), a math expression (e.g, 1 * 100), or with a unit (e.g., 100pt). Otherwise, the value may be misinterpreted as a hex color (e.g., '100') and could cause the converter to crash.

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)

Image Alignments

Images can be aligned as follows:

- left
- center
- right

Font Styles

In most cases, whereever 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
- 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 it would normally be the case in YAML.

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:

```
base:
  background_color: transparent
```

Images

An image is specified either as a bare image path or as an inline image macro as found in the AsciiDoc syntax. Images are currently resolved relative to the value of the `pdf-stylesdir` attribute.

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

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

The GIF format (.gif) is not supported.

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

```
title_page:
  background_image: title-cover.png
```

In this case, the image is resolved relative to theme directory.

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

```
title_page:
  background_image: image:title-cover.png[]
```

In this case, the image is resolved relative to the value of the `imagesdir` attribute. Wrapping the value in the image macro sends a hint to the converter to resolve it just like other images.

Like in the AsciiDoc syntax, the inline image macro allows you to supply set the width of the image and the alignment:

```
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. There's nothing Asciidoctor can do to convince PDF to work with extended characters without the right fonts in play.

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 fonts do not use the fallback fonts. In order for the fallback font to kick in, you must be using a TrueType 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 simply crashes.

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.

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.

At the time of this writing, you cannot use the bundled fonts if you change the value of the `pdf-fontsdir` attribute (and thus define your own custom fonts). This limitation may be lifted in the future.

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

- normal
- italic
- bold
- bold_italic

You'll need all four styles to support AsciiDoc content properly. *Asciidoctor PDF cannot italicize a font dynamically like a browser can, so you need the italic style.*

Once you've obtained the TTF files, put them into a directory in 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. Name 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 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 instance, to use the Roboto font for all headings, you'd use:

```
heading:
  font_family: Roboto
```

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

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

Currently, all fonts referenced by the theme need to be present in the directory specified by the `pdf-fontsdir` attribute.

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. In other words, Asciidoctor PDF automatically subsets the font. However, if you're storing the fonts in a repository, you may want to subset the font (for instance, by using FontForge) to reduce the space the font occupies in that storage. This is 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 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.

Fallback Fonts

If a TrueType font is missing a character needed to render the document, such as a special symbol, 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.

The fallback font is only 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 (¬).

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. Keep in mind that the default theme currently uses a fallback font, though this may change in the future.

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
```

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 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.

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 (_) 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.

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. See Title Page for details.

Key	Value Type	Example
Key Prefix: page		
background_color[1]	Color (default: #ffffff)	`page:` `  background_color: #fefefe`
background_image[1]	Inline image macro[2] (default: *not set*)	`page:` `  background_image: image:page-bg.png[]`
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[3]	Measurement (default: 48)	`page:` `  margin_inner: 0.75in`
margin_outer[3]	Measurement (default: 24)	`page:` `  margin_outer: 0.59in`

Key	Value Type	Example
size	Named size | Measurement[width,height] (default: A4)	`page:` `  size: Letter`

1. Page background images are automatically scaled to fit within the bounds of the page.

Page backgrounds do not currently work when using AsciidoctorJ PDF. This limitation is due to a bug in Prawn 1.3.1. The limitation will remain until AsciidoctorJ PDF upgrades to Prawn 2.x (an upgrade that is waiting on AsciidoctorJ to migrate to JRuby 9000). For more details, see this thread.

2. Target may be an absolute path or a path relative to the value of the `pdf-stylesdir` attribute.
3. 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.

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_size	Number (default: 12)	`base:` `  font_size: 10.5`
font_size_min	Number (default: 9)	`base:` `  font_size_min: 6`
font_style	Font style (default: normal)	`base:` `  font_style: normal`

Key	Value Type	Example
text_transform[1]	none (default: none)	`base:` `  text_transform: none`
line_height_length[2]	Number (default: 13.8)	`base:` `  line_height_length: 12`

line_height[2]	Number
	(default: 1.15)
		`base:`
		`  line_height: >`
		`  $base_line_height_length /`
		`  $base_font_size`

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. You should set one of `line_height` or `line_height_length`, then derive the value of the other using a calculation as these are correlated values. For instance, if you set `line_height_length`, then use `$base_line_height_length / $base_font_size` as the value of `line_height`.

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	none | underline | line-through (default: none)	`link:` `  text_decoration: underline`

(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		
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`

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: $base_font_family)	`heading:` `  font_family: Noto Serif`
font_style	Font style (default: bold)	`heading:` `  font_style: bold`
text_transform	Text transform (default: *inherit*)	`heading:` `  text_transform: uppercase`
line_height	Number (default: 1.15)	`heading:` `  line_height: 1.2`
margin_top	Measurement (default: 4)	`heading:` `  margin_top: $vertical_spacing * 0.2`
margin_bottom	Measurement (default: 12)	`heading:` `  margin_bottom: 9.6`
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`

Key	Value Type	Example
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:` `  text_transform: lowercase`

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).

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 can be disabled from the document by setting the `notitle` attribute in the AsciiDoc document header.

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[1]	Inline image macro[2] (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_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	Value Type	Example
Key Prefix: title_page_logo		

align	Image alignment (default: *inherit*)	`title_page:`
		`  logo:`
		`  align: right`

image	Inline image macro[2] (default: *not set*)	`title_page:`
		`  logo:`
		`  image: image:logo.png[pdfwidth=25%]`

top	Percentage[3] (default: 10%)	`title_page:`
		`  logo:`
		`  top: 25%`

Key Prefix: title_page_title		

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_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	Percentage[3] (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		

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_size	Number (default: 14)	`title_page:`
		`  subtitle:`
		`  font_size: $heading_h3_font_size`

Key	Value Type	Example

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		

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_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		

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`

Key	Value Type	Example

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`

margin_bottom	Measurement (default: 0)	`title_page:`
		`  revision:`
		`  margin_bottom: 5`

1. Page background images are automatically scaled to fit within the bounds of the page.

Page backgrounds do not currently work when using AsciidoctorJ PDF. This limitation is due to a bug in Prawn 1.3.1. The limitation will remain until AsciidoctorJ PDF upgrades to Prawn 2.x (an upgrade that is waiting on AsciidoctorJ to migrate to JRuby 9000). For more details, see this thread.

2. Target may be an absolute path or a path relative to the value of the `pdf-stylesdir` attribute.
3. Percentage unit can be % (relative to content height) or vh (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`

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		

Key	Value Type	Example
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.

Key	Value Type	Example
Key Prefix: caption		
align	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_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`

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		

Key	Value Type	Example
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: 10.5)	`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_linenum[2]		
font_color	Color (default: #999999)	`code:` `  linenum_font_color: #ccc`

1. The line_gap is used to tune the height of the background color applied to a span of block text highlighted using Rouge.
2. The code_linenum category only applies when using Pygments as the source highlighter. Otherwise, the style is controlled by the source highlighter theme.

Callout Numbers

The keys in this category are used to control the style of callout numbers (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`

Key	Value Type	Example
font_family[1,2]	Font family name (default: *inherit*)	`conum:` `  font_family: M+ 1mn`
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`

1. Currently, the font must contain the circle numbers starting at glyph U+2460.
2. font_family, 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.

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		
border_width[1]	Number (default: 4)	`blockquote:` `  border_width: 5`
border_color[1]	Color (default: #eeeeee)	`blockquote:` `  border_color: #eeeeee`
font_color	Color (default: *inherit*)	`blockquote:` `  font_color: #333333`
font_family	Font family name (default: *inherit*)	`blockquote:` `  font_family: Noto Serif`
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`

Key	Value Type	Example
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_style	Font style (default: *inherit*)	`blockquote:`
		`  cite:`
		`  font_style: bold`

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

1. Only applies to the left side.

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_size	Number (default: *inherit*)	`sidebar:` `  font_size: 13`
font_style	Font style (default: *inherit*)	`sidebar:` `  font_style: italic`

Key	Value Type	Example
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_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_size	Number (default: *inherit*)	`example:` `  font_size: 13`

Key	Value Type	Example
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_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`

Key	Value Type	Example
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]		

font_color	Color (default: *inherit*)	`admonition:`
		`  label:`
		`  font_color: #262626`

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

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: fa-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`. The subkeys in the icon category cannot be flattened (e.g., `tip_name: fa-lightbulb-o` 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., `fa-`) determines which font set to use.

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%`

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.

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_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`

Key	Value Type	Example
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_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_size	Number
	(default: $base_font_size)
		`abstract:`
		`  title:`
		`  font_size: 13`

font_style	Font style
	(default: bold)
		`abstract:`
		`  title:`
		`  font_style: bold`

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).

Key	Value Type	Example
Key Prefix: description_list		
term_font_style	Font style (default: bold)	`description_list:` `  term_font_style: italic`
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`

Key	Value Type	Example

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: fa`

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`

Key	Value Type	Example
caption_side	top | bottom (default: top)	table: caption_side: bottom
font_color	Color (default: *inherit*)	table: font_color: #333333
font_family	Font family name (default: *inherit*)	table: font_family: Helvetica
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

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_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

Key	Value Type	Example

font_color	Color
	(default: $table_font_color)
		table:
		  foot:
		  font_color: #333333

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. Applied to even rows by default; controlled using `stripes` attribute (even, odd, all, none) on table.

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		

Key	Value Type	Example
font_color	Color (default: *inherit*)	`toc:` `  font_color: #333333`
font_family	Font family name (default: *inherit*)	`toc:` `  font_family: Noto Serif`
font_size	Number (default: *inherit*)	`toc:` `  font_size: 9`
font_style	Font style (default: normal)	`toc:` `  font_style: bold`
text_decoration	none | underline (default: none)	`toc:` `  text_decoration: underline`
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`
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:` `  font_family: Noto Serif`
font_size	Number (default: *inherit*)	`toc:` `  font_size: 9`
font_style	Font style (default: *inherit*)	`toc:` `  font_style: italic`
text_decoration	none | underline (default: *inherit*)	`toc:` `  text_decoration: none`
text_transform	Text transform (default: *inherit*)	`toc:` `  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`

Key	Value Type	Example

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.

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

Key	Value Type	Example
font_family	Font family name (default: *inherit*)	`header:` `  font_family: Noto Serif`
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`
vertical_align	top | middle | bottom (default: middle)	`header:` `  vertical_align: center`

<side>_columns[4]	Column specs triple
	(default: *not set*)
		`header:`
		`  recto:`
		`  columns: <25% =50% >25%`

<side>_<position>_content[4,5]	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`
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_size	Number (default: *inherit*)	`footer:` `  font_size: 9`
font_style	Font style (default: *inherit*)	`footer:` `  font_style: italic`

Key	Value Type	Example
height[2]	Measurement (default: *not set*)	`footer:` `  height: 0.75in`
line_height	Number (default: $base_line_height)	`footer:` `  line_height: 1.2`
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`
vertical_align	top | middle | bottom (default: middle)	`footer:` `  vertical_align: top`

<side>_columns[4]	Column specs triple
	(default: *not set*)
		`footer:`
		`  verso:`
		`  columns: <50% =0% <50%`

<side>_<position>_content[4,5]	Quoted string
	(default: '{page-number}')
		`footer:`
		`  verso:`
		`  center:`
		`  content: '\{page-number}'`

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. `<side>` can be `recto` (right-hand, odd-numbered pages) or `verso` (left-hand, even-numbered pages). 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`).
5. `<position>` can be `left`, `center` or `right`.

You must define a height for the running header or footer, respectively, or it will not be shown.

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

If content is not specified for the running footer, the page number (i.e., `{page-number}`) is shown on the left on verso pages and the right on recto pages. You can disable this behavior by defining the attribute `nofooter` in the AsciiDoc document header or by defining the key `footer_<side>_content: none` in the theme.

Although not listed in the table above, you can control the font properties used for running content for each column position on 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`.

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
- document-title
- document-subtitle
- part-title
- chapter-title
- section-title
- section-or-chapter-title

You can also built-in AsciiDoc text replacements like `(C)`, numeric character references like `&#169;` 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}'
```

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. Note that the image must be the whole value for a given position (left, center or right). It cannot be combined with text.

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 `footer_vertical_align` attribute to slighly 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` (e.g., `fit=contain`) to resize the image to the maximum size that will fit.

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-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.

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. Otherwise, `-theme.yml` is appended to the name to make the file name (i.e., `<name>-theme.yml`).

pdf-fontsdir

The directory where the fonts used by your theme, if any, are located. *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-stylesdir=resources/themes -a pdf-style=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 are using custom fonts in your theme.

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

```
$ asciidoctor-pdf -a pdf-style=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-stylesdir=/path/to/resources/themes -a pdf-style=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

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[]`
media	screen | print | prepress	`:media: prepress`
page-background-image[4]	path[2] | image macro[3]	`:page-background-image: image:bg.jpg[]`
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-size	Named size | Measurement[width, height]	`:pdf-page-size: 6in x 9in`
pdf-folio-placement	virtual | virtual-inverted | physical | physical-inverted	`:pdf-folio-placement: physical`
pdfmark[6]	flag (default: *not set*)	`:pdfmark:`
text-alignment[7]	Text alignment	`:text-alignment: left`
title-logo-image	path[2] | image macro[3]	`:title-logo-image: image:logo.png[top=25%, align=center, pdfwidth=0.5in]`
title-page[8]	flag (default: *not set*)	`:title-page:`
title-page-background-image	path[2] | image macro[3]	`:title-page-background-image: image:title-bg.jpg[]`

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. Page background images are automatically scaled to fit within the bounds of the page.

Page backgrounds do not currently work when using AsciidoctorJ PDF. This limitation is due to a bug in Prawn 1.3.1. The limitation will remain until AsciidoctorJ PDF upgrades to Prawn 2.x (an upgrade that is waiting on AsciidoctorJ to migrate to JRuby 9000). For more details, see this thread.

5. Controls whether the `page-number` attribute is accessible to the running header and footer content specified in the theme file. Use the `noheader` and `nofooter` attributes to disable the running header and footer, respectively, from the document.
6. Enables generation of the pdfmark file, which contains metadata that is fed to Ghostscript when optimizing the PDF file.
7. *(Experimental)* The `text-alignment` 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.
8. Force a title page to be added even when the doctype is not book.

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 is a recto page.

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 following 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 hexidecimal 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.