rinohtype Documentation

Release 0.3.1.dev0

Brecht Machiels

November 28, 2016

Contents

Release v0.3.1.dev0. (*Release History*)

Rinohtype is a high-level PDF library for Python. It helps automating the creation of any type of document, ranging from invoices to long, complex technical documents.

Paired with Sphinx, rinohtype offers a modern alternative to LaTeX . Sphinx helps writing large structured documents and supports a multitude of different output formats including searchable HTML. Rinohtype provides a Sphinx backend that allows generating professionally typeset PDF documents.

Here is a list of rinohtype's main features:

- a powerful page layout system supporting columns, running headers/footers, floatable elements and footnotes
- figures, large tables and automatically generated table of contents
- automatic numbering and cross-referencing of headings, figures and tables
- use one of the included document templates or create your own
- an intuitive style sheet system inspired by CSS
- modular design allowing for multiple frontends (such as reStructuredText, Markdown, DocBook, ...) and backends (PDF, SVG, bitmap, ...)
- handles OpenType, TrueType and Type1 fonts with support for advanced typographic features such as kerning, ligatures and small capitals
- embeds PDF, PNG and JPEG images, preserving transparency and color profiles
- easy to deploy; pure-Python with few dependencies
- built on Unicode; ready for non-latin languages

Rinohtype is currently in a beta phase. We are working toward a first stable release.

Rinohtype is open source software licensed under the GNU AGPL 3.0. Practically, this means you are free to use it in open-source software, but not in (commercial) closed-source software. For this purpose, you need a commercial license (see http://www.opqode.com/). We are also available for consultancy projects involving rinohtype, so please don't hesitate to contact us.

CHAPTER 1

Introduction

rinohtype was initially conceived as a modern replacement for LaTeX. An important goal in the design of rinohtype is for documents to be much easier to customize than in LaTeX. By today's standards, the arcane TeX macro language upon which LaTeX is built makes customization unnecessarily difficult for one. Simply being built with Python makes rinohtype already much easier to approach than TeX. Additionally, rinohtype is built around the following core concepts to ensure customizability:

Document templates These determine the page layout and (for longer documents) the different parts of your document.

Style sheets The CSS-inspired style sheets determine the look of individual document elements. A style sheet assigns style attributes to each type of document element. For example, a paragraph's style is determined by the typeface, font weight, size and color, horizontal alignment of text etc.

Structured Input rinohtype renders a document from a document tree that does not describe any style aspects but only semantics. The style sheet maps specific style properties to the elements in this document tree. The document tree can be generated from a structured document format such as reStructuredText and DocBook using one of the included frontends, or it can be built manually.

rinohtype is implemented as a Python package and doubles as a high-level PDF library. Its modular design makes it easy to to customize and extend for specific applications. Moreover, because rinotype's source code is open, all of its internals can be inspected and even modified, making it customizable at all levels.

1.1 Usage Examples

rinohtype supports three modes of operation, which are discussed in more detail in the *Quickstart* guide. For each of these modes, you can choose to use one of the document templates included with rinohtype or a third-party template available from PyPI and optionally customize it to your needs. Or you can create a custom template from scratch. The same goes for the style sheet used to style the document elements.

1.1.1 reStructuredText Renderer

rinohtype includes the **`rinoh`** command-line tool that can render reStructuredText documents. Rendering the reStructuredText demonstration article demo.txt (using the standard article template and style sheet) generates `demo.pdf`.

1.1.2 Sphinx Builder

Configuring rinohtype as a builder for Sphinx allows rendering a Sphinx project to PDF without the need for a LaTeX installation. This documentation was rendered by rinohtype to `rinohtype.pdf`.

1.1.3 High-level PDF library

rinohtype can be used as a Python library to generate PDF documents. Just like with **`rinoh`** and the Sphinx builder, you can select which document template and style sheet to use.

Additionally, you need to supply a document tree. This document tree can be parsed from a structured document format such as reStructuredText by using one of the provided frontends or built manually using building blocks provided by rinohtype. You can also write a frontend for a custom format such as an XML dialect.

All of these approaches allow for parts of the content to be fetched from a database or other data sources. When parsing the document tree from a structured document format, a templating engine like Jinja2 can be used.

CHAPTER 2

Installation

Rinohtype supports Python 3.3 and up. Depending on demand, it might be back-ported to Python 2.7, however[1].

Use pip to install the latest version of rinohtype and its dependencies:

```
pip install rinohtype
```

If you plan on using rinohtype as an alternative to LaTeX, you will want to install Sphinx as well:

```
pip install Sphinx
```

See *Sphinx Builder* in the *Quickstart* guide on how to render Sphinx documents with rinohtype.

2.1 Dependencies

For parsing reStructuredText documents, rinohtype depends on docutils. For parsing PNG images the pure-Python PurePNG package is required. pip takes care of these requirements automatically when you install rinohtype.

If you want to include images other than PDF, PNG or JPEG, you will need to install Pillow additionally.

[1] Be sure to contact us if you are interested in running rinohtype on Python 2.7.

CHAPTER 3

Quickstart

This section gets you started quickly, discussing each of the three modes of operation introduced in *Introduction*. Additionally, the basics of style sheets and document templates are explained.

3.1 reStructuredText Renderer

Installing rinohtype places the **`rinoh`** script in the `PATH`. This can be used to render reStructuredText documents such as demo.txt:

```
rinoh demo.txt
```

After rendering finishes, you will find `demo.pdf` alongside the input file.

`rinoh` allows specifying the document template and style sheet to use when rendering the reStructuredText document. See its *command-line options* for details. Two rendering passes are required to make sure that cross-references to page numbers are correct. After a document has been rendered, rinohtype will save the page reference data to a *.rtc* file. Provided the document (or the template or style sheet) doesn't change a lot, this can prevent the need to perform a second rendering pass.

3.2 Sphinx Builder

To use rinohtype to render Sphinx documents, at a minimum you need to add `'rinoh.frontend.sphinx'` to the `extensions` list in the Sphinx project's `conf.py`.

If your Sphinx project is already configured for rendering with LaTeX, rinohtype will happily interpret `latex_documents` and other options for the LaTeX builder. Otherwise, you need to set the *`rinoh_documents`* configuration option:

```
rinoh_documents = [('index',              # top-level file (index.rst)
                    'target',             # output (target.pdf)
                    'Document Title',     # document title
                    'John A. Uthor')]     # document author
```

Other configuration variables are optional and allow configuring the style of the generated PDF document. See *Sphinx Builder* for details.

When building the documentation, select the *rinoh* builder by passing it to `sphinx-build -b` option:

```
sphinx-build -b rinoh . _build/rinoh
```

Just like the **`rinoh`** command line tool, the Sphinx builder requires two *rendering passes*.

3.3 High-level PDF Library

Note: The focus of rinohtype development is currently on the **rinoh** tool and Sphinx builder. Use as a Python library is possible, but documentation may be lacking. Please be patient.

The most basic way to use rinohtype in an application is to hook up an included frontend, a document template and a style sheet:

```
from rinoh.frontend.rst import ReStructuredTextReader
from rinoh.templates import Article

# the parser builds a rinohtype document tree
parser = ReStructuredTextReader()
with open('my_document.rst') as file:
    document_tree = parser.parse(file)

# render the document to 'my_document.pdf'
document = Article(document_tree)
document.render('my_document')
```

This basic application can be customized to your specific requirements by customizing the document template, the style sheet and the way the document's content tree is built. The basics of document templates and style sheets are covered the the sections below.

The document tree returned by the `ReStructuredTextReader` in the example above can also be built manually. A `DocumentTree` is simply a list of `Flowables`, which can have child elements. These children in turn can also have children, and so on; together they form a tree.

Here is an example document tree of a short article:

```
from rinoh.document import DocumentTree
from rinoh.styleds import *

document_tree = DocumentTree(
                    [Paragraph('My Document', style='title'), # metadata!
                     Section([Heading('First Section'),
                              Paragraph('This is a paragraph with some '
                                        + StyledText('emphasized text',
                                                     style='emphasis')
                                        + ' and an '
                                        + InlineImage('image.pdf')),
                              Section([Heading('A subsection'),
                                       Paragraph('Another paragraph')
                                      ])
                             ]),
                     Section([Heading('Second Section'),
                              List([Paragraph('a list item'),
                                    Paragraph('another list item')
                                   ])
                             ])
                    ])
```

It is clear that this type of content is best parsed from a structured document format such as reStructuredText or XML. Manually building a document tree is well suited for short, custom documents however.

3.4 Style Sheets

A style sheet defines the look of each elements in a document. For each type of document element, the style sheet assign values to the style properties available for that element. Style sheets are stored in plain text files using the Windows INI[1] format with the `.rts` extension. Below is an excerpt from the `sphinx_stylesheet` included with rinohtype.

```
[STYLESHEET]
name=Sphinx
description=Mostly a copy of the LaTeX style included with Sphinx
pygments_style=friendly

[VARIABLES]
mono_typeface=TeX Gyre Cursor
serif_typeface=TeX Gyre Pagella
sans_typeface=Tex Gyre Heros
fallback_typeface=DejaVu Serif
thin_black_stroke=0.5pt,#000
blue=#20435c

[default:Paragraph]
typeface=$(serif_typeface)
font_weight=REGULAR
font_size=10pt
line_spacing=fixed(12pt, leading(0))
indent_first=0
space_above=0
space_below=0
text_align=JUSTIFY
kerning=True
ligatures=True
hyphen_lang=en_US
hyphen_chars=4

[fallback]
typeface=$(fallback_typeface)

[body]
base=default
space_above=5pt
space_below=0
text_align=justify

[emphasis]
font_slant=italic

[strong]
font_weight=BOLD

[literal emphasis]
base=emphasis
typeface=$(mono_typeface)
hyphenate=False
ligatures=False

[literal strong]
```

[1] see *Supported INI File Structure* in `configparser`

```
base=strong
typeface=$(mono_typeface)
hyphenate=False
ligatures=False

[quote]
font_slant=italic
```

Except for [STYLESHEET] and [VARIABLES], each configuration section in a style sheet determines the style of a particular type of document element. The emphasis style, for example, determines the look of emphasized text, which is displayed in an italic font. This is similar to how HTML's cascading style sheets work. In rinohtype however, document elements are identified by means of a descriptive label (such as *emphasis*) instead of a cryptic selector. rinohtype also makes use of selectors, but these are collected in a *StyledMatcher* which maps them to descriptive names to be used by many style sheets. Unless you are using rinohtype as a PDF library to create custom documents, the *default matcher* should cover your needs.

The following two subsections briefly illustrate how to extend an existing style sheet and how to create a new, independent style sheet. For more in-depth information on style sheets, please refer to the *Styling Document Elements* section in *Advanced Topics*.

3.4.1 Extending an Existing Style Sheet

Starting from an existing style sheet, it is easy to make small changes to the style of individual document elements. The following style sheet file is based on the Sphinx stylesheet included with rinohtype.

```
[STYLESHEET]
name=My Style Sheet
description=Small tweaks made to the Sphinx style sheet
base=sphinx

[VARIABLES]
mono_typeface=Courier

[emphasis]
font_color=#00a

[strong]
base=DEFAULT_STYLE
font_color=#a00
```

By default, styles defined in a style sheet *extend* the corresponding style from the base style sheet. In this example, emphasized text will be set in an italic font (as configured in the *base style sheet*) and colored blue (#00a).

It is also possible to completely override the style definition. This can be done by setting the base of a style definition to DEFAULT_STYLE as illustrated by the *strong* style. This causes strongly emphasised text to be displayed in red (#a00) but **not** in a bold font as defined in the *base style sheet* (the default for font_weight is *Medium*; see *TextStyle*). Refer to *The Default Matcher* to find out which style attributes each style accepts (by following the hyperlink to the style class's documentation).

The style sheet also redefines the mono_typeface variable. This variable is used in the *base style sheet* in all style definitions where a monospaced font is desired. Redefining the variable in the derived style sheet, affects all of these style definitions.

3.4.2 Starting from Scratch

If you don't specify a base style sheet in the [STYLESHEET] section, you create an independent style sheet. You should do this if you want to create a document style that is not based on an existing style sheet. If the style definition for a particular document element is not included in the style sheet, the default values for its style properties are used.

Unless a custom *StyledMatcher* is passed to *StyleSheetFile*, the *default matcher* is used. Providing your own matcher offers even more customizability, but it is unlikely you will need this. See the *Matchers* section in *Advanced Topics* for details.

Note: In the future, rinohtype will be able to generate an empty INI style sheet, listing all styles defined in the matcher with the supported style attributes along with the default values as comments. This generated style sheet can serve as a good starting point for developing a custom style sheet from scratch.

3.5 Document Templates

As with style sheets, you can choose to make use of a template provided by rinohtype and optionally customize it or you can create a custom template from scratch.

3.5.1 Configuring a Template

rinohtype provides a number of *Standard Document Templates*. These can be customized by means of a template configuration file; a plain text file in the INI[1] format with the .rtt extension. The example from *High-level PDF Library* above can be customized using the following template configuration, for example:

```
[TEMPLATE_CONFIGURATION]
name = my article configuration
template = article

parts =
    title
    ;front_matter
    contents
stylesheet = sphinx_base14
language = fr
abstract_location = title

[SectionTitles]
contents = 'Contents'

[AdmonitionTitles]
caution = 'Careful!'
warning = 'Please be warned'

[VARIABLES]
paper_size = A5

[title]
page_number_format = lowercase roman
end_at_page = left

[contents]
page_number_format = number
```

```
[title_page]
top_margin = 2cm
```

The `TEMPLATE_CONFIGURATION` sections collects global template options. Set *name* to provide a short label for your template configuration. *template* identifies the *document template* to configure.

All document templates consist of a number of document parts. The *`Article`* template defines three parts: *`title`*, *`front_matter`* and *`contents`*. The order of these parts can be changed (although that makes little sense for the article template), and individual parts can optionally be hidden by setting the *`parts`* configuration option. The configuration above hides the front matter part (commented out using a semicolon), for example.

The template configuration also specifies which style sheet is used for styling document elements. The `DocumentTemplate.stylesheet` option takes the name of an installed style sheet (see *`rinoh --list-stylesheets`*) or the filename of a stylesheet file (`.rts`).

The `language` option sets the default language for the document. It determines which language is used for standard document strings such as section and admonition titles.

The *`Article`* template defines two custom template options. The `abstract_location` option determines where the (optional) article abstract is placed, on the title page or in the front matter part. `table_of_contents` allows hiding the table of contents section. Empty document parts will not be included in the document. When the table of contents section is suppressed and there is no abstract in the input document or `abstract_location` is set to title, the front matter document part will not appear in the PDF.

The standard document strings configured by the `language` option described above can be overridden by user-defined strings in the `SectionTitles` and `AdmonitionTitles` sections of the configuration file. For example, the default title for the table of contents section (*Table of Contents*) is replaced with *Contents*. The configuration also sets custom titles for the caution and warning admonitions.

The others sections in the configuration file are the `VARIABLES` section, followed by document part and page template sections. Similar to style sheets, the variables can be referenced in the template configuration sections. Here, the `paper_size` variable is set, which is being referenced by by all page templates in `Article` (although indirectly through the `page` base page template).

For document part templates, `page_number_format` determines how page numbers are formatted. When a document part uses the same page number format as the preceding part, the numbering is continued.

The `DocumentPartTemplate.end_at_page` option controls at which page the document part ends. This is set to `left` for the title part in the example configuration to make the contents part start on a right page.

Each document part finds page templates by name. They will first look for specific left/right page templates by appending `_left_page` or `_right_page` to the document part name. If these page templates have not been defined in the template, it will look for the more general `<document part name>_page` template. Note that, if left and right page templates have been defined by the template (such as the book template), the configuration will need to override these, as they will have priority over the general page template defined in the configuration.

The example configuration only adjusts the top margin for the *`TitlePageTemplate`*, but many more aspects of the page templates are configurable. Refer to *Standard Document Templates* for details.

A template configuration file can be specified when rendering using the command-line **`rinoh`** tool or *Sphinx Builder* by passing it to the `--`*`template`* command-line option or setting the *`rinoh_template`* option in `conf.py`, respectively. To render a document using this template configuration programatically, load the template file using `TemplateConfigurationFile`.

```
from rinoh.frontend.rst import ReStructuredTextReader
from rinoh.template import TemplateConfigurationFile

# the parser builds a rinohtype document tree
parser = ReStructuredTextReader()
```

```
with open('my_document.rst') as file:
    document_tree = parser.parse(file)

# load the article template configuration file
config = TemplateConfigurationFile('my_article.rtt')

# render the document to 'my_document.pdf'
document = config.document(document_tree)
document.render('my_document')
```

The `TemplateConfigurationFile.document()` method creates a document instance with the template configuration applied. So if you want to render your document using a different template configuration, it suffices to load the new configuration file.

Refer to the *Article* documentation to discover all of the options accepted by it and the document part and page templates.

3.5.2 Customizing a Template

If you need to customize a template beyond what is possible by configuration, you can subclass the `Article` (or other) template class and extend it, or override document part and page templates with custom templates.

```
from rinoh.attribute import OverrideDefault
from rinoh.template import DocumentPartTemplate, PageTemplate
from rinoh.templates import Article


class BibliographyPartTemplate(DocumentPartTemplate):
    ...


class MyTitlePageTemplate(PageTemplate):
    ...


class MyArticle(Article):
    parts = OverrideDefault(['title', 'contents', 'bibliography'])

    # default document part templates
    bibliography = BibliographyPartTemplate()

    # default page templates
    title_page = MyTitlePageTemplate(base='page')
    bibliography_page = PageTemplate(base='page')
```

`MyArticle` extends the *Article* template, adding the extra `bibliography` document part, along with the page template `bibliography_page`. The new document part is included in *parts*, while also leaving out *front_matter* by default. Finally, the template also replaces the title page template with a custom one.

3.5.3 Creating a Custom Template

A new template can be created from scratch by subclassing *DocumentTemplate*, defining all document parts, their templates and page templates.

The *Article* and *Book* templates are examples of templates that inherit directly from *DocumentTemplate*. We will briefly discuss the article template here. The *Article* template overrides the default style sheet and defines the two custom template attributes discussed in *Configuring a Template* above. The document parts *`title`*, *`front_matter`* and *`contents`* are listed the in *`parts`* attribute and templates for each of them are provided along with page templates:

```
class Article(DocumentTemplate):
    stylesheet = OverrideDefault(sphinx_article)
    table_of_contents = Attribute(Bool, True,
                                  'Show or hide the table of contents')
    abstract_location = Attribute(AbstractLocation, 'front matter',
                                  'Where to place the abstract')

    parts = OverrideDefault(['title', 'front_matter', 'contents'])

    # default document part templates
    title = TitlePartTemplate()
    front_matter = ArticleFrontMatter()
    contents = ContentsPartTemplate()

    # default page templates
    page = PageTemplate(page_size=Var('paper_size'),
                        chapter_title_flowables=None)
    title_page = TitlePageTemplate(base='page',
                                   top_margin=8*CM)
    front_matter_page = PageTemplate(base='page')
    contents_page = PageTemplate(base='page')
```

The custom `ArticleFrontMatter` template reads the values for the two custom template attributes defined in *`Article`* to determine which flowables are included in the front matter:

```
class ArticleFrontMatter(DocumentPartTemplate):
    toc_section = TableOfContentsSection()

    def _flowables(self, document):
        meta = document.metadata
        abstract_loc = document.get_option('abstract_location')
        if ('abstract' in meta
                and abstract_loc == AbstractLocation.FRONT_MATTER):
            yield meta['abstract']
        if document.get_option('table_of_contents'):
            yield self.toc_section
```

Have a look at the `src/book` for an example of a slightly more complex template that defines separate templates for left and right pages.

CHAPTER 4

Command-line Renderer

4.1 rinoh

Render a structured document to PDF.

```
usage: rinoh [-h] [-f FORMAT] [-o OPTION=VALUE] [-t NAME OR FILENAME]
                 [-s NAME OR FILENAME] [-p PAPER] [--list-templates]
                 [--list-stylesheets] [--list-fonts [FILENAME]] [--list-formats]
                 [--list-options FRONTEND] [--version]
                 [input]
```

input
    the reStructuredText document to render

-h, --help
    show this help message and exit

-f <format>, **--format** <format>
    the format of the input file (default: autodetect)

-o <option=value>, **--option** <option=value>
    options to be passed to the input file reader

-t <name or filename>, **--template** <name or filename>
    the document template or template configuration file to use (default: article)

-s <name or filename>, **--stylesheet** <name or filename>
    the style sheet used to style the document elements (default: the template's default)

-p <paper>, **--paper** <paper>
    the paper size to render to (default: the template's default)

--list-templates
    list the installed document templates and exit

--list-stylesheets
    list the installed style sheets and exit

--list-fonts <filename>
    list the installed fonts or, if FILENAME is given, write a PDF file displaying all the fonts

--list-formats
    list the supported input formats and exit

--list-options <frontend>
    list the options supported by the given frontend and exit

--version
show program's version number and exit

CHAPTER 5

Sphinx Builder

The `rinoh.frontent.sphinx` is a Sphinx extension module. It provides a Sphinx builder with the name *rinoh*. The builder recognizes the following `conf.py` options. Of these, only *`rinoh_documents`* (or `latex_documents`) is required:

`rinoh_documents`
Determines how to group the document tree into PDF output files. Its format is identical to that of `latex_documents`, with the exception that *targetname* should specify the name of the PDF file without the extension. If it is not specified, the value of `latex_documents` is used instead (with the `.tex` extension stripped from the *targetname*).

`rinoh_template`
Determines the template used to render the document. It takes:

- the filename of a template configuration file,
- a *`TemplateConfiguration`* instance,
- the name of an installed template (see *`rinoh --list-templates`*)
- a *`DocumentTemplate`* subclass

The default is `'book'`, which resolves to the *`Book`* template.

`rinoh_stylesheet`
If `rinoh_template_configuration` does not specify a style sheet, this variable specifies the style sheet used to style the document elements. It can be a `StyleSheet` instance or a string identifying an installed style sheet. Default: the default style sheet for the chosen document template.

If `pygments_style` is specified, it overrides the code highlighting style for the specified or default style sheet.

`rinoh_paper_size`
This overrides paper size defined in the template or template configuration (*`rinoh_template`*). This should be a `Paper` instance. A set of predefined paper sizes can be found in the *`rinoh.paper`* module. If not specified, the value of the `'papersize'` entry in `latex_elements` is converted to the equivalent `Paper`. If this is not specified either, the value specified for `latex_paper_size` is used.

Note: Since the interactions between *`rinoh_template`*, *`rinoh_paper_size`*, *`rinoh_stylesheet`* and `pygments_style` are fairly complex, this behavior may be changed (simplified) in the future.

`rinoh_logo`
Path (relative to the configuration directory) to an image file to use at the top of the title page. If not specified, the `latex_logo` value is used.

`rinoh_domain_indices`

Controls the generation of domain-specific indices. Identical to `latex_domain_indices`, which is used when *`rinoh_domain_indices`* is not specified.

CHAPTER 6

Standard Document Templates

Rinohtype includes a number of document templates. These are configurable and therefore should cater for most documents.

6.1 Article

The article template consists of a title page, an optional table of contents and the body text. By default, it uses a single numbering style for all pages and the same template for even and odd pages.

class `rinoh.templates.article.`**`Article`**(*document_tree*, *configuration=None*, *backend=None*)

`stylesheet`

`StyleSheet`

Overrides the default set in *`DocumentTemplate`*

Accepts: the name of an *installed style sheet* or the filename of a stylesheet file (with the `.rts` extension)

Default: `sphinx_article` (= *`rinoh.stylesheets.sphinx_article`*)

`table_of_contents`

`Bool`

Show or hide the table of contents

Accepts: `true` or `false`

Default: `true`

`abstract_location`

`AbstractLocation`

Where to place the abstract

Accepts: `title`, `front matter`

Default: `front matter`

`parts`

`PartsList`

Overrides the default set in *`DocumentTemplate`*

Accepts: a space-separated list of document part template names

Default: *`title front_matter contents`*

language
Language

(*DocumentTemplate*) The main language of the document

Accepts: the code of one of the *supported languages*

Default: *EN* (English)

strings
Strings

(*DocumentTemplate*) Strings to override standard element names

Accepts: strings need to be entered in INI sections named after the *StringCollection* subclasses

Default: `none`

title
TitlePartTemplate

base: `None`

front_matter
`ArticleFrontMatter`

base: `None`

contents
ContentsPartTemplate

base: `None`

page
PageTemplate

base: `None`

Overrides these defaults:

- *chapter_title_flowables* = `None`
- *page_size* = `$(paper_size)`

title_page
TitlePageTemplate

base: *page*

Overrides these defaults:

- *top_margin* = `8cm`

front_matter_page
PageTemplate

base: *page*

contents_page
PageTemplate

base: *page*

class `rinoh.templates.article.`**AbstractLocation**
Where to place the article's abstract

Accepts: `title`, `front matter`

6.2 Book

The book template consists of a title page, the table of contents, the body text and an index. The front matter pages are numbered using lowercase roman numerals. The template uses different templates for even and odd pages.

class `rinoh.templates.book.`**`Book`**(*document_tree*, *configuration=None*, *backend=None*)

> **`stylesheet`**
>
> > *`StyleSheet`*
> >
> > Overrides the default set in *`DocumentTemplate`*
> >
> > Accepts: the name of an *installed style sheet* or the filename of a stylesheet file (with the `.rts` extension)
> >
> > Default: `sphinx` (= *`rinoh.stylesheets.sphinx`*)
>
> **`parts`**
>
> > *`PartsList`*
> >
> > Overrides the default set in *`DocumentTemplate`*
> >
> > Accepts: a space-separated list of document part template names
> >
> > Default: *`title front_matter contents back_matter`*
>
> **`language`**
>
> > *`Language`*
> >
> > (*`DocumentTemplate`*) The main language of the document
> >
> > Accepts: the code of one of the *supported languages*
> >
> > Default: *`EN`* (English)
>
> **`strings`**
>
> > *`Strings`*
> >
> > (*`DocumentTemplate`*) Strings to override standard element names
> >
> > Accepts: strings need to be entered in INI sections named after the *`StringCollection`* subclasses
> >
> > Default: `none`
>
> **`cover`**
>
> > *`FixedDocumentPartTemplate`*
> >
> > base: `None`
> >
> > Overrides these defaults:
> >
> > - *`end_at_page`* = `left`
> > - *`drop_if_empty`* = `False`
> > - *`page_number_format`* = `None`
>
> **`title`**
>
> > *`TitlePartTemplate`*
> >
> > base: `None`
> >
> > Overrides these defaults:
> >
> > - *`end_at_page`* = `left`
> > - *`page_number_format`* = `number`

front_matter
FixedDocumentPartTemplate

base: None

Overrides these defaults:

- *end_at_page* = left
- *flowables* = [TableOfContentsSection()]
- *page_number_format* = lowercase roman

contents
ContentsPartTemplate

base: None

Overrides these defaults:

- *end_at_page* = left
- *page_number_format* = number

back_matter
BackMatterTemplate

base: None

Overrides these defaults:

- end_at_page = left
- page_number_format = number

page
PageTemplate

base: None

Overrides these defaults:

- *top_margin* = 1in
- *page_size* = $(paper_size)
- *right_margin* = 1in
- *left_margin* = 1in
- *bottom_margin* = 1in

cover_page
PageTemplate

base: *page*

title_page
TitlePageTemplate

base: *page*

front_matter_page
PageTemplate

base: *page*

Overrides these defaults:

- *header_text* = None
- *header_footer_distance* = 0

front_matter_right_page
PageTemplate

base: *front_matter_page*

Overrides these defaults:

- *chapter_footer_text* = '\t' '\t' '{PAGE_NUMBER}'
- *footer_text* = '\t' '\t' '{PAGE_NUMBER}'
- *chapter_title_flowables* = [Paragraph[Field({SECTION_TITLE})] (style=front matter section title)]
- *chapter_header_text* = None
- *chapter_title_height* = 2.5in

front_matter_left_page
PageTemplate

base: *front_matter_page*

Overrides these defaults:

- *footer_text* = '{PAGE_NUMBER}'

contents_page
PageTemplate

base: *page*

Overrides these defaults:

- *header_footer_distance* = 0

contents_right_page
PageTemplate

base: *contents_page*

Overrides these defaults:

- *chapter_title_flowables* = [Paragraph[MixedStyledText[MixedStyledText[StringField('rinoh.structure.SectionTitles'>, 'chapter'), SingleStyledText(' ', style=None)] (style=None), Field({SECTION_NUMBER})] (style=None)] (style=body matter chapter label), Paragraph[Field({SECTION_TITLE})] (style=body matter chapter title)]
- *header_text* = '\t' '\t' '{DOCUMENT_TITLE}' ', ' '{DOCUMENT_SUBTITLE}'
- *chapter_header_text* = None
- *chapter_title_height* = 2.4in
- *chapter_footer_text* = '\t' '\t' '{PAGE_NUMBER}'
- *footer_text* = '{SECTION_NUMBER}' '. ' '{SECTION_TITLE}' '\t' '\t' '{PAGE_NUMBER}'

contents_left_page

PageTemplate

base: *contents_page*

Overrides these defaults:

- *header_text* = ’{DOCUMENT_TITLE}’ ’, ’ ’{DOCUMENT_SUBTITLE}’
- *footer_text* = ’{PAGE_NUMBER}’ ’\t’ ’\t’ ’{SectionTitles.chapter}’ ’ ’ ’{SECTION_NUMBER}’ ’. ’ ’{SECTION_TITLE}’

back_matter_page

PageTemplate

base: *page*

Overrides these defaults:

- *header_footer_distance* = 0
- *columns* = 2

back_matter_right_page

PageTemplate

base: *back_matter_page*

Overrides these defaults:

- *chapter_title_flowables* = [Paragraph[Field({SECTION_TITLE})](style=front matter section title)]
- *header_text* = ’\t’ ’\t’ ’{DOCUMENT_TITLE}’ ’, ’ ’{DOCUMENT_SUBTITLE}’
- *chapter_header_text* = None
- *chapter_title_height* = 2.5in
- *chapter_footer_text* = ’\t’ ’\t’ ’{PAGE_NUMBER}’
- *footer_text* = ’{SECTION_TITLE}’ ’\t’ ’\t’ ’{PAGE_NUMBER}’

back_matter_left_page

PageTemplate

base: *back_matter_page*

Overrides these defaults:

- *header_text* = ’{DOCUMENT_TITLE}’ ’, ’ ’{DOCUMENT_SUBTITLE}’
- *footer_text* = ’{PAGE_NUMBER}’ ’\t’ ’\t’ ’{SECTION_TITLE}’

CHAPTER 7

Advanced Topics

This section serves as a guide to the internal workings of rinohtype, and more specifically on how element styling and document templates work. This will be helpful if you want to create custom style sheets, design a new document template or develop a new frontend.

7.1 Flowables and Inline Elements

A *Flowable* is a document element that is placed on a page. It is usually a part of a document tree. Flowables at one level in a document tree are rendered one below the other.

Here is schematic representation of an example document tree:

```
|- Section
|   |- Paragraph
|   \- Paragraph
\- Section
    |- Paragraph
    |- List
    |   |- ListItem
    |   |   |- Paragraph (item label; a number or bullet symbol)
    |   |   \- StaticGroupedFlowables (item body)
    |   |       \- Paragraph
    |   \- ListItem
    |       \- Paragraph
    |   |   \- StaticGroupedFlowables
    |   |       \- List
    |   |           |- ListItem
    |   |           |   \- ...
    |   |           \- ...
    \- Paragraph
```

This represents a document consisting of two sections. The first section contains two paragraphs. The second section contains a paragraph followed by a list and another paragraph. All of the elements in this tree are instances of *Flowable* subclasses.

Section and `List` are subclasses of *GroupedFlowables*; they group a number of flowables. In the case of *List*, these are always of the `ListItem` type. Each list item contains an item number (ordered list) or a bullet symbol (unordered list) and an item body. For simple lists, the item body is typically a single *Paragraph*. The second list item contains a nested *List*.

A *Paragraph* does not have any *Flowable* children. It is however the root node of a tree of *inline elements*. This is an example paragraph in which several text styles are combined:

```
Paragraph
 |- SingleStyledText('Text with ')
 |- MixedStyledText(style='emphasis')
 |   |- SingleStyledText('multiple ')
 |   \- MixedStyledText(style='strong')
 |       |- SingleStyledText('nested ')
 |       \- SingleStyledText('styles', style='small caps')
 \- SingleStyledText('.')
```

The visual representation of the words in this paragraph is determined by the applied style sheet. Read more about how this works in the next section.

Besides `SingleStyledText` and `MixedStyledText` elements (subclasses of `StyledText`), paragraphs can also contain `InlineFlowable`s. Currently, the only inline flowable is `InlineImage`.

The common superclass for flowable and inline elements is `Styled`, which indicates that these elements can be styled using the style sheets which are discussed next.

7.2 Styling Document Elements

Rinohtype's style sheets are heavily inspired by CSS, but add some functionality that CSS lacks. Similar to CSS, rinohtype makes use of so-called *selectors* to select document elements (flowables and inline elements) to style.

Unlike CSS however, these selectors are not directly specified in a style sheet. Instead, all selectors are collected in a *matcher* where they are mapped to descriptive labels for the selected elements. A *style sheets* assigns style properties to these labels. Besides the usefulness of having these labels instead of the more cryptic selectors, a matcher can be reused by multiple style sheets, avoiding duplication.

7.2.1 Selectors

Selectors in rinohtype select elements of a particular type. The *class* of a document element serves as a selector for all instances of the class (and its subclasses). The `Paragraph` class is a selector that matches all paragraphs in the document, for example:

```
Paragraph
```

As with CSS selectors, elements can also be matched based on their context. For example, the following matches any paragraph that is a direct child of a list item (the list item label):

```
ListItem / Paragraph
```

Python's ellipsis can be used to match any number of levels of elements in the document tree. The following selector matches paragraphs at any level inside a table cell:

```
TableCell / ... / Paragraph
```

To help avoid duplicating selector definitions, context selectors can reference other selectors defined in the same *matcher* using `SelectorByName`:

```
SelectorByName('definition term') / ... / Paragraph
```

Selectors can select all instances of `Styled` subclasses. These include `Flowable` and `StyledText`, but also `TableSection`, `TableRow`, `Line` and `Shape`. Elements of some of the latter classes only appear as children of other flowables (such as `Table`).

Similar to a HTML element's *class* attribute, `Styled` elements can have an optional *style* attribute which can be used when constructing a selector. This one selects all styled text elements with the *emphasis* style, for example:

```
StyledText.like('emphasis')
```

The `Styled.like()` method can also match **arbitrary attributes** of elements by passing them as keyword arguments. This can be used to do more advanced things such as selecting the background objects on all odd rows of a table, limited to the cells not spanning multiple rows:

```
TableCell.like(row_index=slice(0, None, 2), rowspan=1) / TableCellBackground
```

The argument passed as *row_index* is a slice object that is used for extended indexing[1]. To make this work, `TableCell.row_index` is an object with a custom `__eq__()` that allows comparison to a slice.

Rinohtype borrows CSS's concept of specificity to determine the "winning" selector when multiple selectors match a given document element. Each part of a selector adds to the specificity of a selector. Roughly stated, the more specific selector will win. For example:

```
ListItem / Paragraph                       # specificity (0, 0, 0, 0, 2)
```

wins over:

```
Paragraph                                  # specificity (0, 0, 0, 0, 1)
```

since it matches two elements instead of just one.

Specificity is represented as a 5-tuple. The last four elements represent the number of *location* (currently not used), *style*, *attribute* and *class* matches. Here are some selectors along with their specificity:

```
StyledText.like('emphasis')                # specificity (0, 0, 1, 0, 1)
TableCell / ... / Paragraph                # specificity (0, 0, 0, 0, 2)
TableCell.like(row_index=2, rowspan=1)     # specificity (0, 0, 0, 2, 1)
```

Specificity ordering is the same as tuple ordering, so (0, 0, 1, 0, 0) wins over (0, 0, 0, 5, 0) and (0, 0, 0, 0, 3) for example. Only when the number of style matches are equal, the attributes match count is compared and so on.

In practice, the class match count is dependent on the element being matched. If the class of the element exactly matches the selector, the right-most specificity value is increased by 2. If the element's class is a subclass of the selector, it is only increased by 1.

The first element of the specificity tuple is the *priority* of the selector. For most selectors, the priority will have the default value of 0. The priority of a selector only needs to be set in some cases. For example, we want the `CodeBlock` selector to match a `CodeBlock` instance. However, because `CodeBlock` is a `Paragraph` subclass, another selector with a higher specificity will also match it:

```
CodeBlock                                  # specificity (0, 0, 0, 0, 2)
DefinitionList / Definition / Paragraph    # specificity (0, 0, 0, 0, 3)
```

To make sure the `CodeBlock` selector wins, we increase the priority of the `CodeBlock` selector by prepending it with a + sign:

```
+CodeBlock                                 # specificity (1, 0, 0, 0, 2)
```

In general, you can use multiple + or − signs to adjust the priority:

```
++CodeBlock                                # specificity (2, 0, 0, 0, 2)
---CodeBlock                               # specificity (-3, 0, 0, 0, 2)
```

[1] Indexing a list like this `lst[slice(0, None, 2)]` is equivalent to `lst[0::2]`.

7.2.2 Matchers

At the most basic level, a *StyledMatcher* is a dictionary that maps labels to selectors:

```
matcher = StyledMatcher()
...
matcher['emphasis'] = StyledText.like('emphasis')
matcher['chapter'] = Section.like(level=1)
matcher['list item number'] = ListItem / Paragraph
matcher['nested line block'] = (GroupedFlowables.like('line block')
                                / GroupedFlowables.like('line block'))
...
```

Rinohtype currently includes one matcher which defines labels for all common elements in documents:

```
from rinoh.stylesheets import matcher
```

7.2.3 Style Sheets

A *StyleSheet* takes a *StyledMatcher* to provide element labels to assign style properties to:

```
styles = StyleSheet('IEEE', matcher=matcher)
...
styles['strong'] = TextStyle(font_weight=BOLD)
styles('emphasis', font_slant=ITALIC)
styles('nested line block', margin_left=0.5*CM)
...
```

Each *Styled* has a *Style* class associated with it. For *Paragraph*, this is *ParagraphStyle*. These style classes determine which style attributes are accepted for the styled element. Because the style class can automatically be determined from the selector, it is possible to simply pass the style properties to the style sheet by calling the *StyleSheet* instance as shown above.

Style sheets are usually loaded from a *.rts* file using *StyleSheetFile*. An example style sheet file is shown in *Style Sheets* in the *Quickstart* guide.

A style sheet file contains a number of sections, denoted by a section title enclosed in square brackets. There are two special sections:

- `[STYLESHEET]` describes global style sheet information (see *StyleSheetFile* for details)
- `[VARIABLES]` collects variables that can be referenced elsewhere in the style sheet

Other sections define the style for a document elements. The section titles correspond to the labels associated with selectors in the *StyledMatcher*. Each entry in a section sets a value for a style attribute. The style for enumerated lists is defined like this, for example:

```
[enumerated list]
margin_left=8pt
space_above=5pt
space_below=5pt
ordered=true
flowable_spacing=5pt
number_format=NUMBER
label_suffix=')'
```

Since this is an enumerated list, *ordered* is set to `true`. *number_format* and *label_suffix* are set to produce list items labels of the style *1)*, *2)*, ... Other entries control margins and spacing. See *ListStyle* for the full list of accepted style attributes.

Note: The supported attributes and format of attribute values have not not yet been fully documented. Please look at the included style sheets for now.

Base Styles

It is possible to define styles which are not linked to a selector. These can be useful to collect common attributes in a base style for a set of style definitions. For example, the Sphinx style sheet defines the *header_footer* style to serve as a base for the *header* and *footer* styles:

```
[header_footer : Paragraph]
base=default
typeface=$(sans_typeface)
font_size=10pt
font_weight=BOLD
indent_first=0pt
tab_stops=50% CENTER, 100% RIGHT

[header]
base=header_footer
padding_bottom=2pt
border_bottom=$(thin_black_stroke)
space_below=24pt

[footer]
base=header_footer
padding_top=4pt
border_top=$(thin_black_stroke)
space_above=18pt
```

Because there is no selector associated with *header_footer*, the element type needs to be specified manually. This is done by adding the name of the relevant `Styled` subclass to the section name, using a colon (`:`) to separate it from the style name, optionally surrounded by spaces.

Custom Selectors

It is also possible to define new selectors directly in a style sheet file. This allows making tweaks to an existing style sheet without having to create a new `StyledMatcher`. However, this should be used sparingly. If a great number of custom selectors are required, it is better to create a new `StyledMatcher`

The syntax for specifying a selector for a style is similar to that when constructing selectors in a Python source code (see *Matchers*), but with a number of important differences. A `Styled` subclass name followed by parentheses represents a simple class selector (without context). Arguments to be passed to `Styled.like()` can be included within the parentheses.

```
[special text : StyledText('special')]
font_color=#FF00FF

[accept button : InlineImage(filename='images/ok_button.png')]
baseline=20%
```

Even if no arguments are passed to the class selector, it is important that the class name is followed by parentheses. If the parentheses are omitted, the selector is not registered with the matcher and the style can only be used as a base style for other style definitions (see *Base Styles*).

As in Python source code, context selectors are constructed using forward slashes (`/`) and the ellipsis (`...`). Another selector can be referenced in a context selector by enclosing its name in single or double quotes.

```
[admonition title colon : Admonition / ... / StyledText('colon')]
font_size=10pt

[chapter title : LabeledFlowable('chapter title')]
label_spacing=1cm
align_baselines=false

[chapter title number : 'chapter title' / Paragraph('number')]
font_size=96pt
text_align=right
```

Variables

Variables can be used for values that are used in multiple style definitions. This example declares a number of typefaces to allow easily replacing the fonts in a style sheet:

```
[VARIABLES]
mono_typeface=TeX Gyre Cursor
serif_typeface=TeX Gyre Pagella
sans_typeface=Tex Gyre Heros
thin_black_stroke=0.5pt,#000
blue=#20435c
```

It also defines the *thin_black_stroke* line style for use in table and frame styles, and a specific color labelled *blue*. These variables can be referenced in style definitions as follows:

```
[code block]
typeface=$(mono_typeface)
font_size=9pt
text_align=LEFT
indent_first=0
space_above=6pt
space_below=4pt
border=$(thin_black_stroke)
padding_left=5pt
padding_top=1pt
padding_bottom=3pt
```

Another stylesheet can inherit (see below) from this one and easily replace fonts in the document by overriding the variables.

Style Attribute Resolution

The style system makes a distinction between text (inline) elements and flowables with respect to how attribute values are resolved.

Text elements by default inherit the properties from their parent. Take for example the *emphasis* style definition from the example above. The value for style properties other than *font_slant* (which is defined in the *emphasis* style itself) will be looked up in the style definition corresponding to the parent element. That can be another *`StyledText`* instance, or a *`Paragraph`*. If that style definition neither defines the style attribute, the lookup proceeds recursively, moving up in the document tree.

For **flowables**, there is no fall-back to the parent's style by default. A base style can be explicitly specified however. If a style attribute is not present in a particular style definition, it is looked up in the base style.

This can also help avoid duplication of style information and the resulting maintenance difficulties. In the following example, the *unnumbered heading level 1* style inherits all properties from *heading level 1*, overriding only the *number_format* attribute:

```
[heading level 1]
typeface=$(sans_typeface)
font_weight=BOLD
font_size=16pt
font_color=$(blue)
line_spacing=SINGLE
space_above=18pt
space_below=12pt
number_format=NUMBER
label_suffix=' '

[unnumbered heading level 1]
base=heading level 1
number_format=None
```

When a value for a particular style attribute is set nowhere in the style definition lookup hierarchy its default value is returned. The default values for all style properties are defined in the class definition for each of the `Style` subclasses.

For text elements, it is possible to override the default behavior of falling back to the parent's style. Setting *base* to the label of a `TextStyle` or `ParagraphStyle` prevents fallback to the parent element's style.

7.2.4 Style Logs

When rendering a document, rinohtype will create a style log. It is written to disk using the same base name as the output file, but with a *.stylelog* extension. The information logged in the style log is invaluable when debugging your style sheet. It tells you which style maps to each element in the document.

The style log lists the document elements (as a tree) that have been rendered to each page, and for each element all matching styles are listed together with their specificity. No styles are listed when there aren't any selectors matching an element and the default values are used. The winning style is indicated with a > symbol. Styles that are not defined in the style sheet or its base(s) are marked with an x. If none of the styles are defined, rinohtype falls back to using the default style.

Here is an example excerpt from a style log:

```
...
  Paragraph('January 03, 2012', style='title page date')
       > (0,0,1,0,2) title page date
         (0,0,0,0,2) body
    SingleStyledText('January 03, 2012')
---------------------------------- page 3 ----------------------------------
#### ChainedContainer('column1')
  DocumentTree()
    Section(id='structural-elements')             demo.txt:62 <section>
         > (0,0,0,1,2) chapter
      Heading('1 Structural Elements')            demo.txt:62 <title>
           > (0,0,0,1,2) heading level 1
             (0,0,0,0,2) other heading levels
          MixedStyledText('1 Structural Elements')
            SingleStyledText('1')
            MixedStyledText(' ')
              SingleStyledText(' ')
            SingleStyledText('Structural Elements')
      Paragraph('A paragraph.')                   demo.txt:64 <paragraph>
```

```
           > (0,0,0,0,2) body
        MixedStyledText('A paragraph.')
          SingleStyledText('A paragraph.')
      List(style='bulleted')                        demo.txt:66 <bullet_list>
           > (0,0,1,0,2) bulleted list
        ListItem()
             x (0,0,1,0,4) bulleted list item
             > fallback to default style
          ListItemLabel('•')
               > (0,0,1,0,6) bulleted list item label
                 (0,0,0,0,2) list item label
              MixedStyledText('•')
                SingleStyledText('')
                SingleStyledText('•')
          StaticGroupedFlowables()                  demo.txt:66 <list_item>
               > (0,0,0,0,3) list item body
...
```

7.3 Document Templates

Note: This section still needs to be written. For now, please refer to the *Document Templates* section in the *Quickstart*, which discusses the configuration of existing templates and the creation of custom templates.

CHAPTER 8

API Documentation

Note: The API documentation is still incomplete.

8.1 Dimension (`rinoh.dimension`)

Classes for expressing dimensions: lengths, widths, line thickness, etc.

Each dimension is expressed in terms of a unit. Several common units are are defined here as constants. To create a new dimension, multiply number with a unit:

```
height = 100*PT
width = 50*PERCENT
```

Fractional dimensions are evaluated within the context they are defined in. For example, the width of a `Flowable` is evaluated with respect to the total width available to it.

class `rinoh.dimension.`**`Dimension`**(*value=0*, *unit=None*)

A simple dimension

Parameters

- **`value`** (*`int or float`*) – the magnitude of the dimension
- **`unit`** (*`DimensionUnit`*) – the unit this dimension is expressed in. Default: *`PT`*.

`grow`(*value*)

Grow this dimension (in-place)

The `value` is interpreted as a magnitude expressed in the same unit as this dimension.

Parameters **`value`** (*`int or float`*) – the amount to add to the magnitude of this dimension

Returns *`Dimension`* – this (growed) dimension itself

`rinoh.dimension.`**`PT`** **= DimensionUnit(1.0, 'pt')**

PostScript points

`rinoh.dimension.`**`INCH`** **= DimensionUnit(72.0, 'in')**

imperial/US inch

`rinoh.dimension.`**`PICA`** **= DimensionUnit(12.0, 'pc')**

computer pica

rinoh.dimension.**MM = DimensionUnit(2.8346456692913384, 'mm')**
millimeter

rinoh.dimension.**CM = DimensionUnit(28.346456692913385, 'cm')**
centimeter

rinoh.dimension.**PERCENT = FractionUnit(100, '%')**
fraction of 100

rinoh.dimension.**QUARTERS = FractionUnit(4, '/4')**
fraction of 4

8.2 Document (rinoh.document)

class rinoh.document.**DocumentTree** (*flowables*, *source_file=None*, *options=None*)
Holds the document's contents as a tree of flowables

Parameters

- **flowables** (*list [Flowable]*) – the list of top-level flowables
- **source_file** (*str*) – absolute path of the source file, used to locate images and include in logging and error and warnings.
- **options** (*Reader*) – frontend-specific options

class rinoh.document.**Document** (*document_tree*, *stylesheet*, *language*, *strings=None*, *backend=None*)
Renders a document tree to pages

Parameters

- **document_tree** (DocumentTree) – a tree of the document's contents
- **stylesheet** (StyleSheet) – style sheet used to style document elements
- **language** (Language) – the language to use for standard strings
- **strings** (Strings) – overrides localized strings provided by *language*
- **backend** – the backend used for rendering the document

render (*filename_root=None*, *file=None*)
Render the document repeatedly until the output no longer changes due to cross-references that need some iterations to converge.

create_outlines ()
Create an outline in the output file that allows for easy navigation of the document. The outline is a hierarchical tree of all the sections in the document.

class rinoh.document.**DocumentPart** (*template*, *document*, *flowables*)
Part of a document.

Parameters

- **template** (DocumentPartTemplate) – the template that determines the contents and style of this document part
- **document** (Document) – the document this part belongs to
- **flowables** (*list [Flowable]*) – the flowables to render in this document part

add_page (*page*)
Append *page* (*Page*) to this *DocumentPart*.

new_page (*page_number*, *new_chapter*, ***kwargs*)
Called by `render()` with the `Chain`s that need more :class:`Container`s. This method should create a new :class:`Page` which contains a container associated with *chain*.

8.2.1 Pages

class `rinoh.document.`**`Page`** (*document_part*, *number*, *paper*, *orientation='portrait'*)
A single page in a document.

A *`Page`* is a `Container`, so other containers can be added as children.

Parameters

- **`document_part`** (`DocumentPart`) – the document part this page is part of
- **`number`** (*`int`*) – the 1-based index of this page in the document part
- **`paper`** (`Paper`) – determines the dimensions of this page
- **`orientation`** (`PageOrientation`) – the orientation of this page

`page`
Returns the page itself.

class `rinoh.document.`**`PageOrientation`**
Accepts: `portrait`, `landscape`

class `rinoh.document.`**`PageType`**
Accepts: `left`, `right`, `any`

8.3 Drawing Primitives (`rinoh.draw`)

class `rinoh.draw.`**`LineStyle`** (*base=None*, ***attributes*)

`stroke`
`Stroke`

Width and color used to draw the line

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `1pt, #000`

8.4 Flowable (`rinoh.flowable`)

class `rinoh.flowable.`**`Flowable`** (*id=None*, *style=None*, *parent=None*)
A document element that can be "flowed" into a container on the page.

A flowable can adapt to the width of the container, or it can horizontally align itself in the container (see *`HorizontallyAlignedFlowable`*).

class `rinoh.flowable.`**`FlowableStyle`** (*base=None*, ***attributes*)
The `Style` for *`Flowable`* objects.

space_above

Dimension

Vertical space preceding the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

space_below

Dimension

Vertical space following the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

margin_left

Dimension

Left margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

margin_right

Dimension

Right margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_left

Dimension

Left padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_right

Dimension

Right padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_top

Dimension

Top padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_bottom

Dimension

Bottom padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`keep_with_next`

`Bool`

Keep this flowable and the next on the same page

Accepts: `true` or `false`

Default: `false`

`border`

`Stroke`

Border surrounding the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_left`

`Stroke`

Border left of the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_right`

`Stroke`

Border right of the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_top`

`Stroke`

Border above the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_bottom`

`Stroke`

Border below the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`background_color`

`Color`

Color of the area within the flowable's borders

Accepts: HEX string with optional alpha component (`#RRGGBB`, `#RRGGBBAA`, `#RGB` or `#RGBA`)

Default: `none`

class `rinoh.flowable.`**`FlowableState`**(*flowable*, *_initial=True*)

Stores a flowable's rendering state, which can be copied.

This enables saving the rendering state at certain points in the rendering process, so rendering can later be resumed at those points, if needed.

8.4.1 No-Output Flowables

These flowables do not directly place anything on the page. All except *DummyFlowable* do have side-effects however. Some of these side-effects affect the rendering of the document in an indirect way.

class `rinoh.flowable.`**`DummyFlowable`**(*id=None*, *parent=None*)

A flowable that does not directly place anything on the page.

Subclasses can produce side-effects to affect the output in another way.

class `rinoh.flowable.`**`AnchorFlowable`**(*id=None*, *parent=None*)

A dummy flowable that registers a destination anchor.

Places a destination for the flowable's ID at the current cursor position.

class `rinoh.flowable.`**`SetMetadataFlowable`**(*parent=None*, ***metadata*)

A dummy flowable that stores metadata in the document.

The metadata is passed as keyword arguments. It will be available to other flowables during the rendering stage.

class `rinoh.flowable.`**`WarnFlowable`**(*message*, *parent=None*)

A dummy flowable that emits a warning during the rendering stage.

Parameters **`message`** (*str*) – the warning message to emit

class `rinoh.flowable.`**`PageBreak`**(*id=None*, *style=None*, *parent=None*)

A flowable that optionally triggers a page break before rendering.

If this flowable's *page_break* style attribute is not `None`, it breaks to the page of the type indicated by *page_break* before starting rendering.

8.4.2 Labeled Flowables

class `rinoh.flowable.`**`LabeledFlowable`**(*label*, *flowable*, *id=None*, *style=None*, *parent=None*)

A flowable with a label.

The flowable and the label are rendered side-by-side. If the label exceeds the *label_max_width* style attribute value, the flowable is rendered below the label.

Parameters

- **`label`** (`Flowable`) – the label for the flowable
- **`flowable`** (`Flowable`) – the flowable to label

`style_class`

alias of *LabeledFlowableStyle*

class `rinoh.flowable.`**`LabeledFlowableStyle`**(*base=None*, ***attributes*)

`label_min_width`

Dimension

Minimum label width

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `12pt`

label_max_width

Dimension

Maximum label width

Accepts: a numeric value followed by a unit (pt, in, pc, mm, cm, %, /4)

Default: 80pt

label_spacing

Dimension

Spacing between a label and the labeled flowable

Accepts: a numeric value followed by a unit (pt, in, pc, mm, cm, %, /4)

Default: 3pt

align_baselines

Bool

Line up the baselines of the label and the labeled flowable

Accepts: true or false

Default: true

wrap_label

Bool

Wrap the label at *label_max_width*

Accepts: true or false

Default: false

space_above

Dimension

(*FlowableStyle*) Vertical space preceding the flowable

Accepts: a numeric value followed by a unit (pt, in, pc, mm, cm, %, /4)

Default: 0

space_below

Dimension

(*FlowableStyle*) Vertical space following the flowable

Accepts: a numeric value followed by a unit (pt, in, pc, mm, cm, %, /4)

Default: 0

margin_left

Dimension

(*FlowableStyle*) Left margin

Accepts: a numeric value followed by a unit (pt, in, pc, mm, cm, %, /4)

Default: 0

margin_right

Dimension

(*FlowableStyle*) Right margin

Accepts: a numeric value followed by a unit (pt, in, pc, mm, cm, %, /4)

Default: `0`

`padding_left`

`Dimension`

(*`FlowableStyle`*) Left padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`padding_right`

`Dimension`

(*`FlowableStyle`*) Right padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`padding_top`

`Dimension`

(*`FlowableStyle`*) Top padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`padding_bottom`

`Dimension`

(*`FlowableStyle`*) Bottom padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`keep_with_next`

`Bool`

(*`FlowableStyle`*) Keep this flowable and the next on the same page

Accepts: `true` or `false`

Default: `false`

`border`

`Stroke`

(*`FlowableStyle`*) Border surrounding the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_left`

`Stroke`

(*`FlowableStyle`*) Border left of the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_right`

`Stroke`

(*`FlowableStyle`*) Border right of the flowable

Accepts: the width (*Dimension*) and color (Color) of the stroke, separated by a comma (,)

Default: none

border_top
Stroke

(*FlowableStyle*) Border above the flowable

Accepts: the width (*Dimension*) and color (Color) of the stroke, separated by a comma (,)

Default: none

border_bottom
Stroke

(*FlowableStyle*) Border below the flowable

Accepts: the width (*Dimension*) and color (Color) of the stroke, separated by a comma (,)

Default: none

background_color
Color

(*FlowableStyle*) Color of the area within the flowable's borders

Accepts: HEX string with optional alpha component (#RRGGBB, #RRGGBBAA, #RGB or #RGBA)

Default: none

class rinoh.flowable.**LabeledFlowableState**(*flowable*, *content_flowable_state*, *_initial=True*)

8.4.3 Grouping Flowables

class rinoh.flowable.**GroupedFlowables**(*id=None*, *style=None*, *parent=None*)
Groups a list of flowables and renders them one below the other.

Makes sure that a flowable for which *keep_with_next* is enabled is not seperated from the flowable that follows it.

Subclasses should implement *flowables()*.

style_class
alias of *GroupedFlowablesStyle*

flowables(*container*)
Generator yielding the *Flowable*s to group

class rinoh.flowable.**GroupedFlowablesStyle**(*base=None*, ***attributes*)

title
StyledText

Title to precede the flowables

Accepts: a list of styled text strings, separated by spaces. A styled text string is a quoted string (' or "), optionally followed by a style name enclosed in braces: 'text string' (style name)

Default: (no value)

flowable_spacing
Dimension

Spacing between flowables

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`space_above`
`Dimension`

(*`FlowableStyle`*) Vertical space preceding the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`space_below`
`Dimension`

(*`FlowableStyle`*) Vertical space following the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`margin_left`
`Dimension`

(*`FlowableStyle`*) Left margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`margin_right`
`Dimension`

(*`FlowableStyle`*) Right margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`padding_left`
`Dimension`

(*`FlowableStyle`*) Left padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`padding_right`
`Dimension`

(*`FlowableStyle`*) Right padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`padding_top`
`Dimension`

(*`FlowableStyle`*) Top padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_bottom
Dimension

(*FlowableStyle*) Bottom padding

Accepts: a numeric value followed by a unit (pt, in, pc, mm, cm, %, /4)

Default: 0

keep_with_next
Bool

(*FlowableStyle*) Keep this flowable and the next on the same page

Accepts: true or false

Default: false

border
Stroke

(*FlowableStyle*) Border surrounding the flowable

Accepts: the width (*Dimension*) and color (Color) of the stroke, separated by a comma (,)

Default: none

border_left
Stroke

(*FlowableStyle*) Border left of the flowable

Accepts: the width (*Dimension*) and color (Color) of the stroke, separated by a comma (,)

Default: none

border_right
Stroke

(*FlowableStyle*) Border right of the flowable

Accepts: the width (*Dimension*) and color (Color) of the stroke, separated by a comma (,)

Default: none

border_top
Stroke

(*FlowableStyle*) Border above the flowable

Accepts: the width (*Dimension*) and color (Color) of the stroke, separated by a comma (,)

Default: none

border_bottom
Stroke

(*FlowableStyle*) Border below the flowable

Accepts: the width (*Dimension*) and color (Color) of the stroke, separated by a comma (,)

Default: none

background_color
Color

(*FlowableStyle*) Color of the area within the flowable's borders

Accepts: HEX string with optional alpha component (#RRGGBB, #RRGGBBAA, #RGB or #RGBA)

Default: `none`

class `rinoh.flowable.`**`GroupedFlowablesState`**(*groupedflowables*, *flowables*, *first_flowable_state=None*, *_initial=True*)

class `rinoh.flowable.`**`StaticGroupedFlowables`**(*flowables*, *id=None*, *style=None*, *parent=None*)

Groups a static list of flowables.

Parameters **`flowables`** (*`iterable[Flowable]`*) – the flowables to group

class `rinoh.flowable.`**`GroupedLabeledFlowables`**(*id=None*, *style=None*, *parent=None*)

Groups a list of labeled flowables, lining them up.

8.4.4 Horizontally Aligned Flowables

class `rinoh.flowable.`**`HorizontallyAlignedFlowable`**(**args*, *align=None*, *width=None*, ***kwargs*)

A flowable with configurable width and horizontal alignment.

The *width* and *horizontal_align* control the width and horizontal alignment of the flowable.

`style_class`

alias of *`HorizontallyAlignedFlowableStyle`*

class `rinoh.flowable.`**`HorizontallyAlignedFlowableStyle`**(*base=None*, ***attributes*)

`width`

`Dimension`

The width of the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `none`

`horizontal_align`

`HorizontalAlignment`

Horizontal alignment of the flowable

Accepts: `left`, `right`, `center`

Default: `left`

`space_above`

`Dimension`

(*`FlowableStyle`*) Vertical space preceding the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`space_below`

`Dimension`

(*`FlowableStyle`*) Vertical space following the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

margin_left
Dimension

(*FlowableStyle*) Left margin

Accepts: a numeric value followed by a unit (pt, in, pc, mm, cm, %, /4)

Default: 0

margin_right
Dimension

(*FlowableStyle*) Right margin

Accepts: a numeric value followed by a unit (pt, in, pc, mm, cm, %, /4)

Default: 0

padding_left
Dimension

(*FlowableStyle*) Left padding

Accepts: a numeric value followed by a unit (pt, in, pc, mm, cm, %, /4)

Default: 0

padding_right
Dimension

(*FlowableStyle*) Right padding

Accepts: a numeric value followed by a unit (pt, in, pc, mm, cm, %, /4)

Default: 0

padding_top
Dimension

(*FlowableStyle*) Top padding

Accepts: a numeric value followed by a unit (pt, in, pc, mm, cm, %, /4)

Default: 0

padding_bottom
Dimension

(*FlowableStyle*) Bottom padding

Accepts: a numeric value followed by a unit (pt, in, pc, mm, cm, %, /4)

Default: 0

keep_with_next
Bool

(*FlowableStyle*) Keep this flowable and the next on the same page

Accepts: true or false

Default: false

border
Stroke

(*FlowableStyle*) Border surrounding the flowable

Accepts: the width (*Dimension*) and color (Color) of the stroke, separated by a comma (,)

Default: `none`

`border_left`
`Stroke`

(*`FlowableStyle`*) Border left of the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_right`
`Stroke`

(*`FlowableStyle`*) Border right of the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_top`
`Stroke`

(*`FlowableStyle`*) Border above the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_bottom`
`Stroke`

(*`FlowableStyle`*) Border below the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`background_color`
`Color`

(*`FlowableStyle`*) Color of the area within the flowable's borders

Accepts: HEX string with optional alpha component (`#RRGGBB`, `#RRGGBBAA`, `#RGB` or `#RGBA`)

Default: `none`

class `rinoh.flowable.`**`HorizontallyAlignedFlowableState`**(*flowable*, *_initial=True*)

8.4.5 Floating Flowables

class `rinoh.flowable.`**`Float`**(*id=None*, *style=None*, *parent=None*)
A flowable that can optionally be placed elsewhere on the page.

If this flowable's *float* style attribute is set to `True`, it is not flowed in line with the surrounding flowables, but it is instead flowed into another container pointed to by the former's `Container.float_space` attribute.

This is typically used to place figures and tables at the top or bottom of a page, instead of in between paragraphs.

class `rinoh.flowable.`**`FloatStyle`**(*base=None*, ***attributes*)

`float`
`Bool`

Float the flowable to the top or bottom of the page

Accepts: `true` or `false`

Default: `false`

`space_above`
`Dimension`

(*`FlowableStyle`*) Vertical space preceding the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`space_below`
`Dimension`

(*`FlowableStyle`*) Vertical space following the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`margin_left`
`Dimension`

(*`FlowableStyle`*) Left margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`margin_right`
`Dimension`

(*`FlowableStyle`*) Right margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`padding_left`
`Dimension`

(*`FlowableStyle`*) Left padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`padding_right`
`Dimension`

(*`FlowableStyle`*) Right padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`padding_top`
`Dimension`

(*`FlowableStyle`*) Top padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_bottom
`Dimension`

(*FlowableStyle*) Bottom padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

keep_with_next
`Bool`

(*FlowableStyle*) Keep this flowable and the next on the same page

Accepts: `true` or `false`

Default: `false`

border
`Stroke`

(*FlowableStyle*) Border surrounding the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

border_left
`Stroke`

(*FlowableStyle*) Border left of the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

border_right
`Stroke`

(*FlowableStyle*) Border right of the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

border_top
`Stroke`

(*FlowableStyle*) Border above the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

border_bottom
`Stroke`

(*FlowableStyle*) Border below the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

background_color
`Color`

(*FlowableStyle*) Color of the area within the flowable’s borders

Accepts: HEX string with optional alpha component (`#RRGGBB`, `#RRGGBBAA`, `#RGB` or `#RGBA`)

Default: `none`

8.5 Fonts and Typefaces (`rinoh.font`)

Classes for fonts and typefaces.

class `rinoh.font.`**`Font`** (*filename*, *weight='medium'*, *slant='upright'*, *width='normal'*)

A collection of glyphs in a particular style

This is a base class for classes that parse different font formats. See `rinoh.font.type1` and `rinoh.font.opentype`.

Parameters

- **`filename`** (*`str`*) – filename of the font file to load
- **`weight`** (*`FontWeight`*) – weight of the font
- **`slant`** (*`FontSlant`*) – slant of the font
- **`width`** (*`FontWidth`*) – width of the font

`get_glyph` (*char*, *variant*)

Return the glyph for a particular character

If the glyph of requested font variant is not present in the font, the *normal* variant is returned instead. If that is not present either, an exception is raised.

Parameters

- **`char`** (*`str of length 1`*) – the character for which to find the glyph
- **`variant`** (*`FontVariant`*) – the variant of the glyph to return

Returns *GlyphMetrics* – the requested glyph

Raises `MissingGlyphException` – when the requested glyph is not present in the font

`get_ligature` (*glyph*, *successor_glyph*)

Return the ligature to replace the given glyphs

If no ligature is defined in the font for the given glyphs, return `None`.

Parameters

- **`glyph`** (*`GlyphMetrics`*) – the first of the glyphs to combine
- **`successor_glyph`** (*`GlyphMetrics`*) – the second of the glyphs to combine

Returns *GlyphMetrics or None* – the ligature to replace the given glyphs

`get_kerning` (*a*, *b*)

Look up the kerning for two glyphs

Parameters

- **`a`** (*`GlyphMetrics`*) – the first of the glyphs
- **`b`** (*`GlyphMetrics`*) – the second of the glyphs

Returns *float* – the kerning value in font units

class `rinoh.font.`**`Typeface`** (*name*, **fonts*)

A set of fonts that share common design features

The fonts collected in a typeface differ in weight, width and/or slant.

Parameters ***fonts** (*Font*) – the fonts that make up this typeface

fonts()

Generator yielding all fonts of this typeface

Yields *Font* – the next font in this typeface

get_font(*weight='medium'*, *slant='upright'*, *width='normal'*)

Return the font matching or closest to the given style

If a font with the given weight, slant and width is available, return it. Otherwise, return the font that is closest in style.

Parameters

- **weight** (*FontWeight*) – weight of the font
- **slant** (*FontSlant*) – slant of the font
- **width** (*FontWidth*) – width of the font

Returns *Font* – the requested font

8.6 Images and Figures (rinoh.image)

class rinoh.image.**Scale**

Scaling factor for images

Can be a numerical value where a value of 1 keeps the image's original size or one of `values`.

Accepts: `fit`, `fill` or a float larger than 0

class rinoh.image.**BackgroundImageArgs**(*base=None*, ***attributes*)

Arguments accepted by attributes of type *BackgroundImage*

scale

Scale

Scaling factor for the image

Accepts: `fit`, `fill` or a float larger than 0

Default: `fit`

width

Dimension

The width to scale the image to

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `none`

height

Dimension

The height to scale the image to

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `none`

dpi

`Integer`

Overrides the DPI value set in the image

Accepts: a natural number (positive integer)

Default: (no value)

rotate

`Integer`

The angle in degrees to rotate the image

Accepts: a natural number (positive integer)

Default: (no value)

limit_width

Dimension

Limit the image width when none of *scale*, *width* and *height* are given and the image would be wider than the container

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `none`

align

`HorizontalAlignment`

How to align the image within the page

Accepts: `left`, `right`, `center`

Default: `left`

class `rinoh.image.`**`BackgroundImage`**(*filename_or_file*, *scale=1.0*, *width=None*, *height=None*, *dpi=None*, *rotate=0*, *limit_width=None*, *align=None*, *id=None*, *style=None*, *parent=None*)

Image to place in the background of a page

Takes the same arguments as `Image`. *BackgroundImageArgs* details how to set the keyword arguments when specifying a background image in a template configuration.

class `rinoh.image.`**`CaptionStyle`**(*base=None*, ***attributes*)

numbering_level

`Integer`

At which section level to restart numbering

Accepts: a natural number (positive integer)

Default: `1`

number_separator

StyledText

Characters inserted between the section number and the caption number

Accepts: a list of styled text strings, separated by spaces. A styled text string is a quoted string (`'` or `"`), optionally followed by a style name enclosed in braces: `'text string' (style name)`

Default: `'.'`

indent_first
Dimension

(*ParagraphStyle*) Indentation of the first line of text

Accepts: a numeric value followed by a unit (pt, in, pc, mm, cm, %, /4)

Default: 0pt

line_spacing
LineSpacing

(*ParagraphStyle*) Spacing between the baselines of two successive lines of text

Accepts: a *predefined line spacing* type or the name of a line spacing type followed by comma-separated arguments enclosed in braces: `<spacing type>(<arg1>, <arg2>, ...)`

Default: default

text_align
TextAlign

(*ParagraphStyle*) Alignment of text to the margins

Accepts: left, right, center, justify

Default: justify

tab_stops
TabStopList

(*ParagraphStyle*) List of tab positions

Accepts: a comma-seperated list of tab stops. A tab stop is specified as `<position> [align] [fill string]`, where `<position>` (*Dimension*) is required and `align` (`TabAlign`) and `fill string` (string enclosed in quotes) are optional.

Default: (no value)

space_above
Dimension

(*FlowableStyle*) Vertical space preceding the flowable

Accepts: a numeric value followed by a unit (pt, in, pc, mm, cm, %, /4)

Default: 0

space_below
Dimension

(*FlowableStyle*) Vertical space following the flowable

Accepts: a numeric value followed by a unit (pt, in, pc, mm, cm, %, /4)

Default: 0

margin_left
Dimension

(*FlowableStyle*) Left margin

Accepts: a numeric value followed by a unit (pt, in, pc, mm, cm, %, /4)

Default: 0

`margin_right`
`Dimension`

(*`FlowableStyle`*) Right margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`padding_left`
`Dimension`

(*`FlowableStyle`*) Left padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`padding_right`
`Dimension`

(*`FlowableStyle`*) Right padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`padding_top`
`Dimension`

(*`FlowableStyle`*) Top padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`padding_bottom`
`Dimension`

(*`FlowableStyle`*) Bottom padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`keep_with_next`
`Bool`

(*`FlowableStyle`*) Keep this flowable and the next on the same page

Accepts: `true` or `false`

Default: `false`

`border`
`Stroke`

(*`FlowableStyle`*) Border surrounding the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_left`
`Stroke`

(*`FlowableStyle`*) Border left of the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_right`
`Stroke`

(*`FlowableStyle`*) Border right of the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_top`
`Stroke`

(*`FlowableStyle`*) Border above the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_bottom`
`Stroke`

(*`FlowableStyle`*) Border below the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`background_color`
`Color`

(*`FlowableStyle`*) Color of the area within the flowable's borders

Accepts: HEX string with optional alpha component (`#RRGGBB`, `#RRGGBBAA`, `#RGB` or `#RGBA`)

Default: `none`

`typeface`
`Typeface`

(*`TextStyle`*) Typeface to set the text in

Accepts: the name of a typeface. See *`rinoh --list-fonts`* for a list of the available typefaces.

Default: `Times`

`font_weight`
`FontWeight`

(*`TextStyle`*) Thickness of character outlines relative to their height

Accepts: `hairline`, `thin`, `ultra-light`, `extra-light`, `light`, `book`, `regular`, `medium`, `demi-bold`, `bold`, `extra-bold`, `heavy`, `black`, `extra-black`, `ultra-black`

Default: `medium`

`font_slant`
`FontSlant`

(*`TextStyle`*) Slope style of the font

Accepts: `upright`, `oblique`, `italic`

Default: `upright`

font_width
FontWidth

(*TextStyle*) Stretch of the characters

Accepts: normal, condensed, extended

Default: normal

font_size
Dimension

(*TextStyle*) Height of characters

Accepts: a numeric value followed by a unit (pt, in, pc, mm, cm, %, /4)

Default: 10pt

font_color
Color

(*TextStyle*) Color of the font

Accepts: HEX string with optional alpha component (#RRGGBB, #RRGGBBAA, #RGB or #RGBA)

Default: #000

font_variant
FontVariant

(*TextStyle*) Variant of the font

Accepts: normal, small capital, oldstyle figures

Default: normal

position
TextPosition

(*TextStyle*) Vertical text position

Accepts: normal, superscript, subscript

Default: normal

kerning
Bool

(*TextStyle*) Improve inter-letter spacing

Accepts: true or false

Default: true

ligatures
Bool

(*TextStyle*) Run letters together where possible

Accepts: true or false

Default: true

hyphenate
Bool

(*TextStyle*) Allow words to be broken over two lines

Accepts: true or false

Default: `true`

`hyphen_chars`
`Integer`

(*`TextStyle`*) Minimum number of characters in a hyphenated part of a word

Accepts: a natural number (positive integer)

Default: `2`

`hyphen_lang`
`Locale`

(*`TextStyle`*) Language to use for hyphenation. Accepts locale codes such as 'en_US'

Accepts: locale identifier in the `<language ID>_<region ID>` format

Default: `en_US`

`number_format`
`NumberFormat`

(`NumberStyle`) How numbers are formatted

Accepts: `none`, `number`, `symbol`, `lowercase character`, `uppercase character`, `lowercase roman`, `uppercase roman`

Default: `number`

`label_prefix`
`StyledText`

(`LabelStyle`) Text to prepend to the label

Accepts: a list of styled text strings, separated by spaces. A styled text string is a quoted string (`'` or `"`), optionally followed by a style name enclosed in braces: `'text string' (style name)`

Default: (no value)

`label_suffix`
`StyledText`

(`LabelStyle`) Text to append to the label

Accepts: a list of styled text strings, separated by spaces. A styled text string is a quoted string (`'` or `"`), optionally followed by a style name enclosed in braces: `'text string' (style name)`

Default: (no value)

`custom_label`
`Bool`

(`LabelStyle`) Use a custom label if specified

Accepts: `true` or `false`

Default: `false`

class `rinoh.image.`**`FigureStyle`**(*base=None*, ***attributes*)

`float`
`Bool`

(*`FloatStyle`*) Float the flowable to the top or bottom of the page

Accepts: `true` or `false`

Default: `false`

`space_above`
`Dimension`

(*`FlowableStyle`*) Vertical space preceding the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`space_below`
`Dimension`

(*`FlowableStyle`*) Vertical space following the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`margin_left`
`Dimension`

(*`FlowableStyle`*) Left margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`margin_right`
`Dimension`

(*`FlowableStyle`*) Right margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`padding_left`
`Dimension`

(*`FlowableStyle`*) Left padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`padding_right`
`Dimension`

(*`FlowableStyle`*) Right padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`padding_top`
`Dimension`

(*`FlowableStyle`*) Top padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`padding_bottom`
`Dimension`

(*`FlowableStyle`*) Bottom padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

keep_with_next
`Bool`

(*FlowableStyle*) Keep this flowable and the next on the same page

Accepts: `true` or `false`

Default: `false`

border
`Stroke`

(*FlowableStyle*) Border surrounding the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

border_left
`Stroke`

(*FlowableStyle*) Border left of the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

border_right
`Stroke`

(*FlowableStyle*) Border right of the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

border_top
`Stroke`

(*FlowableStyle*) Border above the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

border_bottom
`Stroke`

(*FlowableStyle*) Border below the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

background_color
`Color`

(*FlowableStyle*) Color of the area within the flowable's borders

Accepts: HEX string with optional alpha component (`#RRGGBB`, `#RRGGBBAA`, `#RGB` or `#RGBA`)

Default: `none`

title
StyledText

(*GroupedFlowablesStyle*) Title to precede the flowables

Accepts: a list of styled text strings, separated by spaces. A styled text string is a quoted string (' or "), optionally followed by a style name enclosed in braces: `'text string' (style name)`

Default: (no value)

flowable_spacing
Dimension

(*GroupedFlowablesStyle*) Spacing between flowables

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

space_above
Dimension

(*FlowableStyle*) Vertical space preceding the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

space_below
Dimension

(*FlowableStyle*) Vertical space following the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

margin_left
Dimension

(*FlowableStyle*) Left margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

margin_right
Dimension

(*FlowableStyle*) Right margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_left
Dimension

(*FlowableStyle*) Left padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_right
Dimension

(*FlowableStyle*) Right padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_top

Dimension

(*FlowableStyle*) Top padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_bottom

Dimension

(*FlowableStyle*) Bottom padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

keep_with_next

`Bool`

(*FlowableStyle*) Keep this flowable and the next on the same page

Accepts: `true` or `false`

Default: `false`

border

`Stroke`

(*FlowableStyle*) Border surrounding the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

border_left

`Stroke`

(*FlowableStyle*) Border left of the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

border_right

`Stroke`

(*FlowableStyle*) Border right of the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

border_top

`Stroke`

(*FlowableStyle*) Border above the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

border_bottom
`Stroke`

(*FlowableStyle*) Border below the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

background_color
`Color`

(*FlowableStyle*) Color of the area within the flowable's borders

Accepts: HEX string with optional alpha component (`#RRGGBB`, `#RRGGBBAA`, `#RGB` or `#RGBA`)

Default: `none`

8.7 Index (`rinoh.index`)

class `rinoh.index.`**IndexStyle**(*base=None*, ***attributes*)

initials
`Bool`

Group index entries based on their first letter

Accepts: `true` or `false`

Default: `true`

title
StyledText

(*GroupedFlowablesStyle*) Title to precede the flowables

Accepts: a list of styled text strings, separated by spaces. A styled text string is a quoted string (`'` or `"`), optionally followed by a style name enclosed in braces: `'text string' (style name)`

Default: (no value)

flowable_spacing
Dimension

(*GroupedFlowablesStyle*) Spacing between flowables

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

space_above
Dimension

(*FlowableStyle*) Vertical space preceding the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

space_below
Dimension

(*FlowableStyle*) Vertical space following the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`margin_left`
`Dimension`

(*`FlowableStyle`*) Left margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`margin_right`
`Dimension`

(*`FlowableStyle`*) Right margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`padding_left`
`Dimension`

(*`FlowableStyle`*) Left padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`padding_right`
`Dimension`

(*`FlowableStyle`*) Right padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`padding_top`
`Dimension`

(*`FlowableStyle`*) Top padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`padding_bottom`
`Dimension`

(*`FlowableStyle`*) Bottom padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`keep_with_next`
`Bool`

(*`FlowableStyle`*) Keep this flowable and the next on the same page

Accepts: `true` or `false`

Default: `false`

border
`Stroke`

(*FlowableStyle*) Border surrounding the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

border_left
`Stroke`

(*FlowableStyle*) Border left of the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

border_right
`Stroke`

(*FlowableStyle*) Border right of the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

border_top
`Stroke`

(*FlowableStyle*) Border above the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

border_bottom
`Stroke`

(*FlowableStyle*) Border below the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

background_color
`Color`

(*FlowableStyle*) Color of the area within the flowable's borders

Accepts: HEX string with optional alpha component (`#RRGGBB`, `#RRGGBBAA`, `#RGB` or `#RGBA`)

Default: `none`

8.8 Language (`rinoh.language`)

class `rinoh.language.`**`Language`**(*code*, *name*)
Collects localized strings for a particular language

Parameters

- **code** (*str*) – short code identifying the language
- **name** (*str*) – native name of the language

`languages`
Dictionary mapping codes to *`Language`*s

The following languages are supported:

`rinoh.language.`**`EN`** **= Language('en', 'English')**
Localized strings for English

SectionTitles

Contents	Table of Contents
Chapter	Chapter
Index	Index

AdmonitionTitles

Attention	Attention!
Caution	Caution!
Danger	!DANGER!
Error	Error
Hint	Hint
Important	Important
Note	Note
Tip	Tip
Warning	Warning
Seealso	See also

`rinoh.language.`**`FR`** **= Language('fr', 'Français')**
Localized strings for Français

SectionTitles

Contents	Table des Matières
Chapter	Chapitre
Index	Index

AdmonitionTitles

Attention	Attention!
Caution	Prudence!
Danger	!DANGER!
Error	Erreur
Hint	Conseil

Important Important

Note Note

Tip Astuce

Warning Avertissement

Seealso Voir aussi

`rinoh.language.`**`IT`** **= Language('it', 'Italiano')**

Localized strings for Italiano

SectionTitles

Contents Contenuti

Chapter Capitolo

Index Indice

AdmonitionTitles

Attention Attenzione!

Caution Prudenza!

Danger !PERICOLO!

Error Errore

Hint Consiglio

Important Importante

Note Nota

Tip Suggerimento

Warning Avvertimento

Seealso Vedi anche

`rinoh.language.`**`NL`** **= Language('nl', 'Nederlands')**

Localized strings for Nederlands

SectionTitles

Contents Inhoudsopgave

Chapter Hoofdstuk

Index Index

AdmonitionTitles

Attention Opgelet!

Caution Pas op!

Danger !GEVAAR!

Error Fout

Hint Hint

Important Belangrijk

Note Noot

Tip Tip

Warning Waarschuwing

Seealso Zie ook

8.9 Paper sizes (`rinoh.paper`)

The *Paper* class and a number of predefined paper formats.

class `rinoh.paper.`**`Paper`** (*name*, *width*, *height*)

Defines a paper size.

Parameters

- **`name`** (*str*) – the name of this paper type
- **`width`** (`Dimension`) – the (portrait) width of this paper type
- **`height`** (`Dimension`) – the (portrait) height of this paper type

`rinoh.paper.`**`A0`** **= Paper('A0', width=841*MM, height=1189*MM)**

`rinoh.paper.`**`A1`** **= Paper('A1', width=594*MM, height=841*MM)**

`rinoh.paper.`**`A2`** **= Paper('A2', width=420*MM, height=594*MM)**

`rinoh.paper.`**`A3`** **= Paper('A3', width=297*MM, height=420*MM)**

`rinoh.paper.`**`A4`** **= Paper('A4', width=210*MM, height=297*MM)**

`rinoh.paper.`**`A5`** **= Paper('A5', width=148*MM, height=210*MM)**

`rinoh.paper.`**`A6`** **= Paper('A6', width=105*MM, height=148*MM)**

`rinoh.paper.`**`A7`** **= Paper('A7', width=74*MM, height=105*MM)**

`rinoh.paper.`**`A8`** **= Paper('A8', width=52*MM, height=74*MM)**

`rinoh.paper.`**`A9`** **= Paper('A9', width=37*MM, height=52*MM)**

`rinoh.paper.`**`A10`** **= Paper('A10', width=26*MM, height=37*MM)**

`rinoh.paper.`**`LETTER`** **= Paper('letter', width=8.5*INCH, height=11*INCH)**

`rinoh.paper.`**`LEGAL`** **= Paper('legal', width=8.5*INCH, height=14*INCH)**

`rinoh.paper.`**`JUNIOR_LEGAL`** **= Paper('junior legal', width=8*INCH, height=5*INCH)**

`rinoh.paper.`**`LEDGER`** **= Paper('ledger', width=17*INCH, height=11*INCH)** rinoh.paper.**TABLOID = Paper('tabloid', width=11*INCH, height=17*INCH)**

8.10 Paragraph (rinoh.paragraph)

class rinoh.paragraph.**ParagraphBase** (*id=None*, *style=None*, *parent=None*)

Base class for paragraphs

A paragraph is a collection of mixed-styled text that can be flowed into a container.

style_class

alias of *ParagraphStyle*

render (*container*, *descender*, *state*, *first_line_only=False*)

Typeset the paragraph

The paragraph is typeset in the given container starting below the current cursor position of the container. When the end of the container is reached, the rendering state is preserved to continue setting the rest of the paragraph when this method is called with a new container.

Parameters

- **container** (*Container*) – the container to render to
- **descender** (*float or None*) – descender height of the preceeding line
- **state** (ParagraphState) – the state where rendering will continue
- **first_line_only** (*bool*) – typeset only the first line

class rinoh.paragraph.**Paragraph** (*text_or_items*, *id=None*, *style=None*, *parent=None*)

A paragraph of static text

Parameters

- **text_or_items** – see *MixedStyledText*
- **style** – see *Styled*
- **parent** – see DocumentElement

class rinoh.paragraph.**ParagraphStyle** (*base=None*, ***attributes*)

The Style for *Paragraph* objects

indent_first

Dimension

Indentation of the first line of text

Accepts: a numeric value followed by a unit (pt, in, pc, mm, cm, %, /4)

Default: 0pt

line_spacing

LineSpacing

Spacing between the baselines of two successive lines of text

Accepts: a *predefined line spacing* type or the name of a line spacing type followed by comma-separated arguments enclosed in braces: <spacing type>(<arg1>, <arg2>, ...)

Default: default

text_align

`TextAlign`

Alignment of text to the margins

Accepts: `left`, `right`, `center`, `justify`

Default: `justify`

tab_stops

`TabStopList`

List of tab positions

Accepts: a comma-seperated list of tab stops. A tab stop is specified as `<position> [align] [fill string]`, where `<position>` (*`Dimension`*) is required and `align` (`TabAlign`) and `fill string` (string enclosed in quotes) are optional.

Default: (no value)

space_above

`Dimension`

(*`FlowableStyle`*) Vertical space preceding the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

space_below

`Dimension`

(*`FlowableStyle`*) Vertical space following the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

margin_left

`Dimension`

(*`FlowableStyle`*) Left margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

margin_right

`Dimension`

(*`FlowableStyle`*) Right margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_left

`Dimension`

(*`FlowableStyle`*) Left padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_right

`Dimension`

(*FlowableStyle*) Right padding

Accepts: a numeric value followed by a unit (pt, in, pc, mm, cm, %, /4)

Default: 0

padding_top
Dimension

(*FlowableStyle*) Top padding

Accepts: a numeric value followed by a unit (pt, in, pc, mm, cm, %, /4)

Default: 0

padding_bottom
Dimension

(*FlowableStyle*) Bottom padding

Accepts: a numeric value followed by a unit (pt, in, pc, mm, cm, %, /4)

Default: 0

keep_with_next
Bool

(*FlowableStyle*) Keep this flowable and the next on the same page

Accepts: true or false

Default: false

border
Stroke

(*FlowableStyle*) Border surrounding the flowable

Accepts: the width (*Dimension*) and color (Color) of the stroke, separated by a comma (,)

Default: none

border_left
Stroke

(*FlowableStyle*) Border left of the flowable

Accepts: the width (*Dimension*) and color (Color) of the stroke, separated by a comma (,)

Default: none

border_right
Stroke

(*FlowableStyle*) Border right of the flowable

Accepts: the width (*Dimension*) and color (Color) of the stroke, separated by a comma (,)

Default: none

border_top
Stroke

(*FlowableStyle*) Border above the flowable

Accepts: the width (*Dimension*) and color (Color) of the stroke, separated by a comma (,)

Default: none

border_bottom
`Stroke`

(*FlowableStyle*) Border below the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

background_color
`Color`

(*FlowableStyle*) Color of the area within the flowable's borders

Accepts: HEX string with optional alpha component (`#RRGGBB`, `#RRGGBBAA`, `#RGB` or `#RGBA`)

Default: `none`

typeface
Typeface

(*TextStyle*) Typeface to set the text in

Accepts: the name of a typeface. See *rinoh --list-fonts* for a list of the available typefaces.

Default: `Times`

font_weight
`FontWeight`

(*TextStyle*) Thickness of character outlines relative to their height

Accepts: `hairline`, `thin`, `ultra-light`, `extra-light`, `light`, `book`, `regular`, `medium`, `demi-bold`, `bold`, `extra-bold`, `heavy`, `black`, `extra-black`, `ultra-black`

Default: `medium`

font_slant
`FontSlant`

(*TextStyle*) Slope style of the font

Accepts: `upright`, `oblique`, `italic`

Default: `upright`

font_width
`FontWidth`

(*TextStyle*) Stretch of the characters

Accepts: `normal`, `condensed`, `extended`

Default: `normal`

font_size
Dimension

(*TextStyle*) Height of characters

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `10pt`

font_color
`Color`

(*TextStyle*) Color of the font

Accepts: HEX string with optional alpha component (`#RRGGBB`, `#RRGGBBAA`, `#RGB` or `#RGBA`)

Default: `#000`

font_variant
`FontVariant`

(*TextStyle*) Variant of the font

Accepts: `normal`, `small capital`, `oldstyle figures`

Default: `normal`

position
`TextPosition`

(*TextStyle*) Vertical text position

Accepts: `normal`, `superscript`, `subscript`

Default: `normal`

kerning
`Bool`

(*TextStyle*) Improve inter-letter spacing

Accepts: `true` or `false`

Default: `true`

ligatures
`Bool`

(*TextStyle*) Run letters together where possible

Accepts: `true` or `false`

Default: `true`

hyphenate
`Bool`

(*TextStyle*) Allow words to be broken over two lines

Accepts: `true` or `false`

Default: `true`

hyphen_chars
`Integer`

(*TextStyle*) Minimum number of characters in a hyphenated part of a word

Accepts: a natural number (positive integer)

Default: `2`

hyphen_lang
`Locale`

(*TextStyle*) Language to use for hyphenation. Accepts locale codes such as 'en_US'

Accepts: locale identifier in the `<language ID>_<region ID>` format

Default: `en_US`

class `rinoh.paragraph.`**`ParagraphState`**(*paragraph*, *words*, *nested_flowable_state=None*, *_first_word=None*, *_initial=True*)

8.10.1 Styled Text (`rinoh.text`)

class `rinoh.text.`**`StyledText`**(*id=None*, *style=None*, *parent=None*)

Base class for text that has a *`TextStyle`* associated with it.

`style_class`

alias of *`TextStyle`*

`__add__`(*other*)

Return the concatenation of this styled text and *other*. If *other* is *None*, this styled text itself is returned.

`__radd__`(*other*)

Return the concatenation of *other* and this styled text. If *other* is *None*, this styled text itself is returned.

`__iadd__`(*other*)

Return the concatenation of this styled text and *other*. If *other* is *None*, this styled text itself is returned.

`to_string`(*flowable_target*)

Return the text content of this styled text.

`is_script`(*container*)

Returns *True* if this styled text is super/subscript.

`script_level`(*container*)

Nesting level of super/subscript.

`height`(*container*)

Font size after super/subscript size adjustment.

`y_offset`(*container*)

Vertical baseline offset (up is positive).

`items`

The list of items in this StyledText.

`spans`(*container*)

Generator yielding all spans in this styled text, one item at a time (used in typesetting).

class `rinoh.text.`**`TextStyle`**(*base=None*, ***attributes*)

The `Style` for *`StyledText`* objects

`typeface`

`Typeface`

Typeface to set the text in

Accepts: the name of a typeface. See *`rinoh --list-fonts`* for a list of the available typefaces.

Default: `Times`

`font_weight`

`FontWeight`

Thickness of character outlines relative to their height

Accepts: `hairline`, `thin`, `ultra-light`, `extra-light`, `light`, `book`, `regular`, `medium`, `demi-bold`, `bold`, `extra-bold`, `heavy`, `black`, `extra-black`, `ultra-black`

Default: `medium`

`font_slant`

`FontSlant`

Slope style of the font

Accepts: `upright`, `oblique`, `italic`

Default: `upright`

font_width
`FontWidth`

Stretch of the characters

Accepts: `normal`, `condensed`, `extended`

Default: `normal`

font_size
Dimension

Height of characters

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `10pt`

font_color
`Color`

Color of the font

Accepts: HEX string with optional alpha component (`#RRGGBB`, `#RRGGBBAA`, `#RGB` or `#RGBA`)

Default: `#000`

font_variant
`FontVariant`

Variant of the font

Accepts: `normal`, `small capital`, `oldstyle figures`

Default: `normal`

position
`TextPosition`

Vertical text position

Accepts: `normal`, `superscript`, `subscript`

Default: `normal`

kerning
`Bool`

Improve inter-letter spacing

Accepts: `true` or `false`

Default: `true`

ligatures
`Bool`

Run letters together where possible

Accepts: `true` or `false`

Default: `true`

hyphenate
`Bool`

Allow words to be broken over two lines

Accepts: `true` or `false`

Default: `true`

hyphen_chars
`Integer`

Minimum number of characters in a hyphenated part of a word

Accepts: a natural number (positive integer)

Default: `2`

hyphen_lang
`Locale`

Language to use for hyphenation. Accepts locale codes such as 'en_US'

Accepts: locale identifier in the `<language ID>_<region ID>` format

Default: `en_US`

class `rinoh.text.`**`SingleStyledText`**(*text*, *style=None*, *parent=None*)

class `rinoh.text.`**`MixedStyledText`**(*text_or_items*, *style=None*, *parent=None*)
Concatenation of styled text

Parameters

- **`text_or_items`** (*`str, StyledText or iterable of these`*) – mixed styled text
- **`style`** – see *`Styled`*
- **`parent`** – see `DocumentElement`

`__repr__`()
Return a representation of this mixed-styled text; its children along with a representation of its *`TextStyle`*.

`append`(*item*)
Append *item* (*`StyledText`* or `str`) to the end of this mixed-styled text.

The parent of *item* is set to this mixed-styled text.

8.10.2 Inline Elements (`rinoh.inline`)

class `rinoh.inline.`**`InlineFlowable`**(*baseline=None*, *id=None*, *style=None*, *parent=None*)

style_class
alias of *`InlineFlowableStyle`*

class `rinoh.inline.`**`InlineFlowableStyle`**(*base=None*, ***attributes*)

baseline
`Dimension`

The location of the flowable's baseline relative to the bottom edge

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0pt`

space_above
Dimension

(*FlowableStyle*) Vertical space preceding the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

space_below
Dimension

(*FlowableStyle*) Vertical space following the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

margin_left
Dimension

(*FlowableStyle*) Left margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

margin_right
Dimension

(*FlowableStyle*) Right margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_left
Dimension

(*FlowableStyle*) Left padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_right
Dimension

(*FlowableStyle*) Right padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_top
Dimension

(*FlowableStyle*) Top padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_bottom
Dimension

(*FlowableStyle*) Bottom padding

Accepts: a numeric value followed by a unit (pt, in, pc, mm, cm, %, /4)

Default: 0

keep_with_next
Bool

(*FlowableStyle*) Keep this flowable and the next on the same page

Accepts: true or false

Default: false

border
Stroke

(*FlowableStyle*) Border surrounding the flowable

Accepts: the width (*Dimension*) and color (Color) of the stroke, separated by a comma (,)

Default: none

border_left
Stroke

(*FlowableStyle*) Border left of the flowable

Accepts: the width (*Dimension*) and color (Color) of the stroke, separated by a comma (,)

Default: none

border_right
Stroke

(*FlowableStyle*) Border right of the flowable

Accepts: the width (*Dimension*) and color (Color) of the stroke, separated by a comma (,)

Default: none

border_top
Stroke

(*FlowableStyle*) Border above the flowable

Accepts: the width (*Dimension*) and color (Color) of the stroke, separated by a comma (,)

Default: none

border_bottom
Stroke

(*FlowableStyle*) Border below the flowable

Accepts: the width (*Dimension*) and color (Color) of the stroke, separated by a comma (,)

Default: none

background_color
Color

(*FlowableStyle*) Color of the area within the flowable's borders

Accepts: HEX string with optional alpha component (#RRGGBB, #RRGGBBAA, #RGB or #RGBA)

Default: `none`

8.10.3 Styling Properties

Line Spacing

class `rinoh.paragraph.`**`LineSpacing`**

Base class for line spacing types

Line spacing is defined as the distance between the baselines of two consecutive lines of text in a paragraph.

`advance`(*line*, *last_descender*, *container*)

Return the distance between the descender of the previous line and the baseline of the current line.

class `rinoh.paragraph.`**`DefaultSpacing`**

The default line spacing as specified by the font.

class `rinoh.paragraph.`**`ProportionalSpacing`**(*factor*)

Line spacing proportional to the line height

Parameters **`factor`** (`float`) – amount by which the line height is multiplied to obtain the line spacing

class `rinoh.paragraph.`**`FixedSpacing`**(*pitch*, *minimum=ProportionalSpacing(1.0)*)

Fixed line spacing, with optional minimum spacing

Parameters

- **`pitch`** (`Dimension`) – the distance between the baseline of two consecutive lines of text
- **`minimum`** (`LineSpacing`) – the minimum line spacing to prevents lines with large fonts (or inline elements) from overlapping; set to `None` if no minimum is required, set to *None*

class `rinoh.paragraph.`**`Leading`**(*leading*)

Line spacing determined by the space in between two lines

Parameters **`leading`** (`Dimension`) – the space between the bottom of a line and the top of the next line of text

The following standard line spacings have been predefined:

`rinoh.paragraph.`**`DEFAULT` = DefaultSpacing()**

The default line spacing as specified by the font.

`rinoh.paragraph.`**`STANDARD` = ProportionalSpacing(1.2)**

Line spacing of 1.2 times the line height.

`rinoh.paragraph.`**`SINGLE` = ProportionalSpacing(1.0)**

Line spacing equal to the line height (no leading).

`rinoh.paragraph.`**`DOUBLE` = ProportionalSpacing(2.0)**

Line spacing of double the line height.

Tabulation

class `rinoh.paragraph.`**`TabStop`**(*position*, *align='left'*, *fill=None*)

Horizontal position for aligning text of successive lines.

`get_position`(*line_width*)

Return the absolute position of this tab stop.

8.10.4 Rendering Internals

class `rinoh.paragraph.`**`Glyph`**`(`*glyph*, *width*, *char*`)`

class `rinoh.paragraph.`**`GlyphsSpan`**`(`*span*, *chars_to_glyphs*, *glyphs_and_widths=[]*`)`

8.10.5 Miscellaneous Internals

class `rinoh.paragraph.`**`HyphenatorStore`**

8.11 Cross-References and Fields (`rinoh.reference`)

class `rinoh.reference.`**`ReferenceType`**

Accepts: `reference`, `number`, `title`, `page`

class `rinoh.reference.`**`ReferencingParagraphStyle`**`(`*base=None*, ***attributes*`)`

`text`

`ReferenceText`

The text content of this paragraph

Accepts: a list of styled text strings, separated by spaces. A styled text string is a quoted string (`'` or `"`), optionally followed by a style name enclosed in braces: `'text string' (style name)`

Default: `'{TITLE}'`

`indent_first`

`Dimension`

(*`ParagraphStyle`*) Indentation of the first line of text

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0pt`

`line_spacing`

`LineSpacing`

(*`ParagraphStyle`*) Spacing between the baselines of two successive lines of text

Accepts: a *predefined line spacing* type or the name of a line spacing type followed by comma-separated arguments enclosed in braces: `<spacing type>(<arg1>, <arg2>, ...)`

Default: `default`

`text_align`

`TextAlign`

(*`ParagraphStyle`*) Alignment of text to the margins

Accepts: `left`, `right`, `center`, `justify`

Default: `justify`

`tab_stops`

`TabStopList`

(*`ParagraphStyle`*) List of tab positions

Accepts: a comma-seperated list of tab stops. A tab stop is specified as `<position> [align] [fill string]`, where `<position>` (*Dimension*) is required and `align` (`TabAlign`) and `fill string` (string enclosed in quotes) are optional.

Default: (no value)

space_above
Dimension

(*FlowableStyle*) Vertical space preceding the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

space_below
Dimension

(*FlowableStyle*) Vertical space following the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

margin_left
Dimension

(*FlowableStyle*) Left margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

margin_right
Dimension

(*FlowableStyle*) Right margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_left
Dimension

(*FlowableStyle*) Left padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_right
Dimension

(*FlowableStyle*) Right padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_top
Dimension

(*FlowableStyle*) Top padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_bottom
`Dimension`

(`FlowableStyle`) Bottom padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

keep_with_next
`Bool`

(`FlowableStyle`) Keep this flowable and the next on the same page

Accepts: `true` or `false`

Default: `false`

border
`Stroke`

(`FlowableStyle`) Border surrounding the flowable

Accepts: the width (`Dimension`) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

border_left
`Stroke`

(`FlowableStyle`) Border left of the flowable

Accepts: the width (`Dimension`) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

border_right
`Stroke`

(`FlowableStyle`) Border right of the flowable

Accepts: the width (`Dimension`) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

border_top
`Stroke`

(`FlowableStyle`) Border above the flowable

Accepts: the width (`Dimension`) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

border_bottom
`Stroke`

(`FlowableStyle`) Border below the flowable

Accepts: the width (`Dimension`) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

background_color
`Color`

(`FlowableStyle`) Color of the area within the flowable's borders

Accepts: HEX string with optional alpha component (`#RRGGBB`, `#RRGGBBAA`, `#RGB` or `#RGBA`)

Default: `none`

typeface
Typeface

(*TextStyle*) Typeface to set the text in

Accepts: the name of a typeface. See *rinoh --list-fonts* for a list of the available typefaces.

Default: `Times`

font_weight
`FontWeight`

(*TextStyle*) Thickness of character outlines relative to their height

Accepts: `hairline`, `thin`, `ultra-light`, `extra-light`, `light`, `book`, `regular`, `medium`, `demi-bold`, `bold`, `extra-bold`, `heavy`, `black`, `extra-black`, `ultra-black`

Default: `medium`

font_slant
`FontSlant`

(*TextStyle*) Slope style of the font

Accepts: `upright`, `oblique`, `italic`

Default: `upright`

font_width
`FontWidth`

(*TextStyle*) Stretch of the characters

Accepts: `normal`, `condensed`, `extended`

Default: `normal`

font_size
Dimension

(*TextStyle*) Height of characters

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `10pt`

font_color
`Color`

(*TextStyle*) Color of the font

Accepts: HEX string with optional alpha component (`#RRGGBB`, `#RRGGBBAA`, `#RGB` or `#RGBA`)

Default: `#000`

font_variant
`FontVariant`

(*TextStyle*) Variant of the font

Accepts: `normal`, `small capital`, `oldstyle figures`

Default: `normal`

position
`TextPosition`

(*TextStyle*) Vertical text position

Accepts: `normal`, `superscript`, `subscript`

Default: `normal`

kerning
`Bool`

(*TextStyle*) Improve inter-letter spacing

Accepts: `true` or `false`

Default: `true`

ligatures
`Bool`

(*TextStyle*) Run letters together where possible

Accepts: `true` or `false`

Default: `true`

hyphenate
`Bool`

(*TextStyle*) Allow words to be broken over two lines

Accepts: `true` or `false`

Default: `true`

hyphen_chars
`Integer`

(*TextStyle*) Minimum number of characters in a hyphenated part of a word

Accepts: a natural number (positive integer)

Default: `2`

hyphen_lang
`Locale`

(*TextStyle*) Language to use for hyphenation. Accepts locale codes such as 'en_US'

Accepts: locale identifier in the `<language ID>_<region ID>` format

Default: `en_US`

class `rinoh.reference.`**NoteMarkerStyle**(*base=None*, ***attributes*)

typeface
Typeface

(*TextStyle*) Typeface to set the text in

Accepts: the name of a typeface. See *rinoh --list-fonts* for a list of the available typefaces.

Default: `Times`

font_weight
FontWeight

(*TextStyle*) Thickness of character outlines relative to their height

Accepts: `hairline`, `thin`, `ultra-light`, `extra-light`, `light`, `book`, `regular`, `medium`, `demi-bold`, `bold`, `extra-bold`, `heavy`, `black`, `extra-black`, `ultra-black`

Default: `medium`

font_slant
FontSlant

(*TextStyle*) Slope style of the font

Accepts: `upright`, `oblique`, `italic`

Default: `upright`

font_width
FontWidth

(*TextStyle*) Stretch of the characters

Accepts: `normal`, `condensed`, `extended`

Default: `normal`

font_size
Dimension

(*TextStyle*) Height of characters

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `10pt`

font_color
Color

(*TextStyle*) Color of the font

Accepts: HEX string with optional alpha component (`#RRGGBB`, `#RRGGBBAA`, `#RGB` or `#RGBA`)

Default: `#000`

font_variant
FontVariant

(*TextStyle*) Variant of the font

Accepts: `normal`, `small capital`, `oldstyle figures`

Default: `normal`

position
TextPosition

(*TextStyle*) Vertical text position

Accepts: `normal`, `superscript`, `subscript`

Default: `normal`

kerning
Bool

(*TextStyle*) Improve inter-letter spacing

Accepts: `true` or `false`

Default: `true`

`ligatures`
`Bool`

(*`TextStyle`*) Run letters together where possible

Accepts: `true` or `false`

Default: `true`

`hyphenate`
`Bool`

(*`TextStyle`*) Allow words to be broken over two lines

Accepts: `true` or `false`

Default: `true`

`hyphen_chars`
`Integer`

(*`TextStyle`*) Minimum number of characters in a hyphenated part of a word

Accepts: a natural number (positive integer)

Default: `2`

`hyphen_lang`
`Locale`

(*`TextStyle`*) Language to use for hyphenation. Accepts locale codes such as 'en_US'

Accepts: locale identifier in the `<language ID>_<region ID>` format

Default: `en_US`

`number_format`
`NumberFormat`

(`NumberStyle`) How numbers are formatted

Accepts: `none`, `number`, `symbol`, `lowercase character`, `uppercase character`, `lowercase roman`, `uppercase roman`

Default: `number`

`label_prefix`
`StyledText`

(`LabelStyle`) Text to prepend to the label

Accepts: a list of styled text strings, separated by spaces. A styled text string is a quoted string (`'` or `"`), optionally followed by a style name enclosed in braces: `'text string' (style name)`

Default: (no value)

`label_suffix`
`StyledText`

(`LabelStyle`) Text to append to the label

Accepts: a list of styled text strings, separated by spaces. A styled text string is a quoted string (`'` or `"`), optionally followed by a style name enclosed in braces: `'text string' (style name)`

Default: (no value)

custom_label

`Bool`

(`LabelStyle`) Use a custom label if specified

Accepts: `true` or `false`

Default: `false`

8.12 Structure (`rinoh.structure`)

8.12.1 Sections

class `rinoh.structure.`**`Section`** (*flowables*, *id=None*, *style=None*, *parent=None*)

A subdivision of a document

A section usually has a heading associated with it, which is optionally numbered.

style_class

alias of *`SectionStyle`*

class `rinoh.structure.`**`SectionStyle`** (*base=None*, ***attributes*)

show_in_toc

`Bool`

List this section in the table of contents

Accepts: `true` or `false`

Default: `true`

title

`StyledText`

(*`GroupedFlowablesStyle`*) Title to precede the flowables

Accepts: a list of styled text strings, separated by spaces. A styled text string is a quoted string (`'` or `"`), optionally followed by a style name enclosed in braces: `'text string' (style name)`

Default: (no value)

flowable_spacing

`Dimension`

(*`GroupedFlowablesStyle`*) Spacing between flowables

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

space_above

`Dimension`

(*`FlowableStyle`*) Vertical space preceding the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

space_below

Dimension

(*FlowableStyle*) Vertical space following the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

margin_left

Dimension

(*FlowableStyle*) Left margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

margin_right

Dimension

(*FlowableStyle*) Right margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_left

Dimension

(*FlowableStyle*) Left padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_right

Dimension

(*FlowableStyle*) Right padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_top

Dimension

(*FlowableStyle*) Top padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_bottom

Dimension

(*FlowableStyle*) Bottom padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

keep_with_next

`Bool`

(*FlowableStyle*) Keep this flowable and the next on the same page

Accepts: `true` or `false`

Default: `false`

`border`
`Stroke`

(*`FlowableStyle`*) Border surrounding the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_left`
`Stroke`

(*`FlowableStyle`*) Border left of the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_right`
`Stroke`

(*`FlowableStyle`*) Border right of the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_top`
`Stroke`

(*`FlowableStyle`*) Border above the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_bottom`
`Stroke`

(*`FlowableStyle`*) Border below the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`background_color`
`Color`

(*`FlowableStyle`*) Color of the area within the flowable's borders

Accepts: HEX string with optional alpha component (`#RRGGBB`, `#RRGGBBAA`, `#RGB` or `#RGBA`)

Default: `none`

`page_break`
`Break`

(`PageBreakStyle`) Type of page break to insert before rendering this flowable

Accepts: `none`, `any`, `left`, `right`

Default: `None`

`space_above`
`Dimension`

(*`FlowableStyle`*) Vertical space preceding the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`space_below`

`Dimension`

(*`FlowableStyle`*) Vertical space following the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`margin_left`

`Dimension`

(*`FlowableStyle`*) Left margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`margin_right`

`Dimension`

(*`FlowableStyle`*) Right margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`padding_left`

`Dimension`

(*`FlowableStyle`*) Left padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`padding_right`

`Dimension`

(*`FlowableStyle`*) Right padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`padding_top`

`Dimension`

(*`FlowableStyle`*) Top padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`padding_bottom`

`Dimension`

(*`FlowableStyle`*) Bottom padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`keep_with_next`
`Bool`

(*`FlowableStyle`*) Keep this flowable and the next on the same page

Accepts: `true` or `false`

Default: `false`

`border`
`Stroke`

(*`FlowableStyle`*) Border surrounding the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_left`
`Stroke`

(*`FlowableStyle`*) Border left of the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_right`
`Stroke`

(*`FlowableStyle`*) Border right of the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_top`
`Stroke`

(*`FlowableStyle`*) Border above the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_bottom`
`Stroke`

(*`FlowableStyle`*) Border below the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`background_color`
`Color`

(*`FlowableStyle`*) Color of the area within the flowable's borders

Accepts: HEX string with optional alpha component (`#RRGGBB`, `#RRGGBBAA`, `#RGB` or `#RGBA`)

Default: `none`

class `rinoh.structure.`**`Heading`**(*title*, *custom_label=None*, *id=None*, *style=None*, *parent=None*)
The title for a section

Parameters

- **`title`** (`StyledText`) – the title text

- **custom_label** (StyledText) – a frontend can supply a custom label to use instead of an automatically determined section number

style_class
alias of *HeadingStyle*

class rinoh.structure.**HeadingStyle**(*base=None*, ***attributes*)

number_separator
StyledText

Characters inserted between the number label of the parent section and this section. If None, only show this section's number label.

Accepts: a list of styled text strings, separated by spaces. A styled text string is a quoted string (' or "), optionally followed by a style name enclosed in braces: 'text string' (style name)

Default: '.'

keep_with_next
Bool

Overrides the default set in *FlowableStyle*

Accepts: true or false

Default: true

indent_first
Dimension

(*ParagraphStyle*) Indentation of the first line of text

Accepts: a numeric value followed by a unit (pt, in, pc, mm, cm, %, /4)

Default: 0pt

line_spacing
LineSpacing

(*ParagraphStyle*) Spacing between the baselines of two successive lines of text

Accepts: a *predefined line spacing* type or the name of a line spacing type followed by comma-separated arguments enclosed in braces: <spacing type>(<arg1>, <arg2>, ...)

Default: default

text_align
TextAlign

(*ParagraphStyle*) Alignment of text to the margins

Accepts: left, right, center, justify

Default: justify

tab_stops
TabStopList

(*ParagraphStyle*) List of tab positions

Accepts: a comma-seperated list of tab stops. A tab stop is specified as <position> [align] [fill string], where <position> (*Dimension*) is required and align (TabAlign) and fill string (string enclosed in quotes) are optional.

Default: (no value)

space_above
Dimension

(*FlowableStyle*) Vertical space preceding the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

space_below
Dimension

(*FlowableStyle*) Vertical space following the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

margin_left
Dimension

(*FlowableStyle*) Left margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

margin_right
Dimension

(*FlowableStyle*) Right margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_left
Dimension

(*FlowableStyle*) Left padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_right
Dimension

(*FlowableStyle*) Right padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_top
Dimension

(*FlowableStyle*) Top padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_bottom
Dimension

(*FlowableStyle*) Bottom padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`border`
`Stroke`

(*`FlowableStyle`*) Border surrounding the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_left`
`Stroke`

(*`FlowableStyle`*) Border left of the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_right`
`Stroke`

(*`FlowableStyle`*) Border right of the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_top`
`Stroke`

(*`FlowableStyle`*) Border above the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_bottom`
`Stroke`

(*`FlowableStyle`*) Border below the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`background_color`
`Color`

(*`FlowableStyle`*) Color of the area within the flowable's borders

Accepts: HEX string with optional alpha component (`#RRGGBB`, `#RRGGBBAA`, `#RGB` or `#RGBA`)

Default: `none`

`typeface`
`Typeface`

(*`TextStyle`*) Typeface to set the text in

Accepts: the name of a typeface. See *`rinoh --list-fonts`* for a list of the available typefaces.

Default: `Times`

`font_weight`
`FontWeight`

(*`TextStyle`*) Thickness of character outlines relative to their height

Accepts: `hairline`, `thin`, `ultra-light`, `extra-light`, `light`, `book`, `regular`, `medium`, `demi-bold`, `bold`, `extra-bold`, `heavy`, `black`, `extra-black`, `ultra-black`

Default: `medium`

`font_slant`
`FontSlant`

(*`TextStyle`*) Slope style of the font

Accepts: `upright`, `oblique`, `italic`

Default: `upright`

`font_width`
`FontWidth`

(*`TextStyle`*) Stretch of the characters

Accepts: `normal`, `condensed`, `extended`

Default: `normal`

`font_size`
`Dimension`

(*`TextStyle`*) Height of characters

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `10pt`

`font_color`
`Color`

(*`TextStyle`*) Color of the font

Accepts: HEX string with optional alpha component (`#RRGGBB`, `#RRGGBBAA`, `#RGB` or `#RGBA`)

Default: `#000`

`font_variant`
`FontVariant`

(*`TextStyle`*) Variant of the font

Accepts: `normal`, `small capital`, `oldstyle figures`

Default: `normal`

`position`
`TextPosition`

(*`TextStyle`*) Vertical text position

Accepts: `normal`, `superscript`, `subscript`

Default: `normal`

`kerning`
`Bool`

(*`TextStyle`*) Improve inter-letter spacing

Accepts: `true` or `false`

Default: `true`

ligatures

Bool

(*TextStyle*) Run letters together where possible

Accepts: `true` or `false`

Default: `true`

hyphenate

Bool

(*TextStyle*) Allow words to be broken over two lines

Accepts: `true` or `false`

Default: `true`

hyphen_chars

Integer

(*TextStyle*) Minimum number of characters in a hyphenated part of a word

Accepts: a natural number (positive integer)

Default: `2`

hyphen_lang

Locale

(*TextStyle*) Language to use for hyphenation. Accepts locale codes such as 'en_US'

Accepts: locale identifier in the `<language ID>_<region ID>` format

Default: `en_US`

number_format

NumberFormat

(`NumberStyle`) How numbers are formatted

Accepts: `none`, `number`, `symbol`, `lowercase character`, `uppercase character`, `lowercase roman`, `uppercase roman`

Default: `number`

label_prefix

StyledText

(`LabelStyle`) Text to prepend to the label

Accepts: a list of styled text strings, separated by spaces. A styled text string is a quoted string (`'` or `"`), optionally followed by a style name enclosed in braces: `'text string' (style name)`

Default: (no value)

label_suffix

StyledText

(`LabelStyle`) Text to append to the label

Accepts: a list of styled text strings, separated by spaces. A styled text string is a quoted string (`'` or `"`), optionally followed by a style name enclosed in braces: `'text string' (style name)`

Default: (no value)

custom_label
Bool

(LabelStyle) Use a custom label if specified

Accepts: `true` or `false`

Default: `false`

class `rinoh.structure.`**`SectionTitles`**(***strings*)
Collection of localized titles for common sections

contents
String

Title for the table of contents section

chapter
String

Label for top-level sections

index
String

Title for the index section

8.12.2 Lists

class `rinoh.structure.`**`List`**(*flowables*, *id=None*, *style=None*, *parent=None*)

style_class
alias of *ListStyle*

class `rinoh.structure.`**`ListStyle`**(*base=None*, ***attributes*)

ordered
Bool

This list is ordered or unordered

Accepts: `true` or `false`

Default: `false`

bullet
StyledText

Bullet to use in unordered lists

Accepts: a list of styled text strings, separated by spaces. A styled text string is a quoted string (`'` or `"`), optionally followed by a style name enclosed in braces: `'text string' (style name)`

Default: `'•'`

title
StyledText

(*GroupedFlowablesStyle*) Title to precede the flowables

Accepts: a list of styled text strings, separated by spaces. A styled text string is a quoted string (`'` or `"`), optionally followed by a style name enclosed in braces: `'text string' (style name)`

Default: (no value)

flowable_spacing

Dimension

(*GroupedFlowablesStyle*) Spacing between flowables

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

space_above

Dimension

(*FlowableStyle*) Vertical space preceding the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

space_below

Dimension

(*FlowableStyle*) Vertical space following the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

margin_left

Dimension

(*FlowableStyle*) Left margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

margin_right

Dimension

(*FlowableStyle*) Right margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_left

Dimension

(*FlowableStyle*) Left padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_right

Dimension

(*FlowableStyle*) Right padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_top

Dimension

(*FlowableStyle*) Top padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`padding_bottom`

`Dimension`

(*`FlowableStyle`*) Bottom padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`keep_with_next`

`Bool`

(*`FlowableStyle`*) Keep this flowable and the next on the same page

Accepts: `true` or `false`

Default: `false`

`border`

`Stroke`

(*`FlowableStyle`*) Border surrounding the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_left`

`Stroke`

(*`FlowableStyle`*) Border left of the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_right`

`Stroke`

(*`FlowableStyle`*) Border right of the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_top`

`Stroke`

(*`FlowableStyle`*) Border above the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_bottom`

`Stroke`

(*`FlowableStyle`*) Border below the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

background_color
`Color`

(*FlowableStyle*) Color of the area within the flowable's borders

Accepts: HEX string with optional alpha component (`#RRGGBB`, `#RRGGBBAA`, `#RGB` or `#RGBA`)

Default: `none`

number_format
`NumberFormat`

(`NumberStyle`) How numbers are formatted

Accepts: `none`, `number`, `symbol`, `lowercase character`, `uppercase character`, `lowercase roman`, `uppercase roman`

Default: `number`

label_prefix
StyledText

(`LabelStyle`) Text to prepend to the label

Accepts: a list of styled text strings, separated by spaces. A styled text string is a quoted string (`'` or `"`), optionally followed by a style name enclosed in braces: `'text string' (style name)`

Default: (no value)

label_suffix
StyledText

(`LabelStyle`) Text to append to the label

Accepts: a list of styled text strings, separated by spaces. A styled text string is a quoted string (`'` or `"`), optionally followed by a style name enclosed in braces: `'text string' (style name)`

Default: (no value)

custom_label
`Bool`

(`LabelStyle`) Use a custom label if specified

Accepts: `true` or `false`

Default: `false`

class `rinoh.structure.`**`ListItemLabelStyle`**(*base=None*, ***attributes*)

indent_first
Dimension

(*ParagraphStyle*) Indentation of the first line of text

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0pt`

line_spacing
LineSpacing

(*ParagraphStyle*) Spacing between the baselines of two successive lines of text

Accepts: a *predefined line spacing* type or the name of a line spacing type followed by comma-separated arguments enclosed in braces: `<spacing type>(<arg1>, <arg2>, ...)`

Default: `default`

`text_align`

`TextAlign`

(*`ParagraphStyle`*) Alignment of text to the margins

Accepts: `left`, `right`, `center`, `justify`

Default: `justify`

`tab_stops`

`TabStopList`

(*`ParagraphStyle`*) List of tab positions

Accepts: a comma-seperated list of tab stops. A tab stop is specified as `<position> [align] [fill string]`, where `<position>` (*`Dimension`*) is required and `align` (`TabAlign`) and `fill string` (string enclosed in quotes) are optional.

Default: (no value)

`space_above`

`Dimension`

(*`FlowableStyle`*) Vertical space preceding the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`space_below`

`Dimension`

(*`FlowableStyle`*) Vertical space following the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`margin_left`

`Dimension`

(*`FlowableStyle`*) Left margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`margin_right`

`Dimension`

(*`FlowableStyle`*) Right margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`padding_left`

`Dimension`

(*`FlowableStyle`*) Left padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_right
Dimension

(*FlowableStyle*) Right padding

Accepts: a numeric value followed by a unit (pt, in, pc, mm, cm, %, /4)

Default: 0

padding_top
Dimension

(*FlowableStyle*) Top padding

Accepts: a numeric value followed by a unit (pt, in, pc, mm, cm, %, /4)

Default: 0

padding_bottom
Dimension

(*FlowableStyle*) Bottom padding

Accepts: a numeric value followed by a unit (pt, in, pc, mm, cm, %, /4)

Default: 0

keep_with_next
Bool

(*FlowableStyle*) Keep this flowable and the next on the same page

Accepts: true or false

Default: false

border
Stroke

(*FlowableStyle*) Border surrounding the flowable

Accepts: the width (*Dimension*) and color (Color) of the stroke, separated by a comma (,)

Default: none

border_left
Stroke

(*FlowableStyle*) Border left of the flowable

Accepts: the width (*Dimension*) and color (Color) of the stroke, separated by a comma (,)

Default: none

border_right
Stroke

(*FlowableStyle*) Border right of the flowable

Accepts: the width (*Dimension*) and color (Color) of the stroke, separated by a comma (,)

Default: none

border_top
Stroke

(*FlowableStyle*) Border above the flowable

Accepts: the width (*Dimension*) and color (Color) of the stroke, separated by a comma (,)

Default: `none`

`border_bottom`
`Stroke`

(*`FlowableStyle`*) Border below the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`background_color`
`Color`

(*`FlowableStyle`*) Color of the area within the flowable's borders

Accepts: HEX string with optional alpha component (`#RRGGBB`, `#RRGGBBAA`, `#RGB` or `#RGBA`)

Default: `none`

`typeface`
`Typeface`

(*`TextStyle`*) Typeface to set the text in

Accepts: the name of a typeface. See *`rinoh --list-fonts`* for a list of the available typefaces.

Default: `Times`

`font_weight`
`FontWeight`

(*`TextStyle`*) Thickness of character outlines relative to their height

Accepts: `hairline`, `thin`, `ultra-light`, `extra-light`, `light`, `book`, `regular`, `medium`, `demi-bold`, `bold`, `extra-bold`, `heavy`, `black`, `extra-black`, `ultra-black`

Default: `medium`

`font_slant`
`FontSlant`

(*`TextStyle`*) Slope style of the font

Accepts: `upright`, `oblique`, `italic`

Default: `upright`

`font_width`
`FontWidth`

(*`TextStyle`*) Stretch of the characters

Accepts: `normal`, `condensed`, `extended`

Default: `normal`

`font_size`
`Dimension`

(*`TextStyle`*) Height of characters

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `10pt`

font_color
Color

(*TextStyle*) Color of the font

Accepts: HEX string with optional alpha component (`#RRGGBB`, `#RRGGBBAA`, `#RGB` or `#RGBA`)

Default: `#000`

font_variant
FontVariant

(*TextStyle*) Variant of the font

Accepts: `normal`, `small capital`, `oldstyle figures`

Default: `normal`

position
TextPosition

(*TextStyle*) Vertical text position

Accepts: `normal`, `superscript`, `subscript`

Default: `normal`

kerning
Bool

(*TextStyle*) Improve inter-letter spacing

Accepts: `true` or `false`

Default: `true`

ligatures
Bool

(*TextStyle*) Run letters together where possible

Accepts: `true` or `false`

Default: `true`

hyphenate
Bool

(*TextStyle*) Allow words to be broken over two lines

Accepts: `true` or `false`

Default: `true`

hyphen_chars
Integer

(*TextStyle*) Minimum number of characters in a hyphenated part of a word

Accepts: a natural number (positive integer)

Default: `2`

hyphen_lang
Locale

(*TextStyle*) Language to use for hyphenation. Accepts locale codes such as ‘en_US’

Accepts: locale identifier in the `<language ID>_<region ID>` format

Default: `en_US`

`label_prefix`

`StyledText`

(`LabelStyle`) Text to prepend to the label

Accepts: a list of styled text strings, separated by spaces. A styled text string is a quoted string (`'` or `"`), optionally followed by a style name enclosed in braces: `'text string' (style name)`

Default: (no value)

`label_suffix`

`StyledText`

(`LabelStyle`) Text to append to the label

Accepts: a list of styled text strings, separated by spaces. A styled text string is a quoted string (`'` or `"`), optionally followed by a style name enclosed in braces: `'text string' (style name)`

Default: (no value)

`custom_label`

`Bool`

(`LabelStyle`) Use a custom label if specified

Accepts: `true` or `false`

Default: `false`

class `rinoh.structure.`**`DefinitionList`**(*flowables*, *id=None*, *style=None*, *parent=None*)

8.12.3 Table of Contents

class `rinoh.structure.`**`TableOfContentsSection`**

class `rinoh.structure.`**`TableOfContents`**(*local=False*, *id=None*, *style=None*, *parent=None*)

`style_class`

alias of *`TableOfContentsStyle`*

class `rinoh.structure.`**`TableOfContentsStyle`**(*base=None*, ***attributes*)

`depth`

`Integer`

The number of section levels to include in the table of contents

Accepts: a natural number (positive integer)

Default: `3`

`title`

`StyledText`

(*`GroupedFlowablesStyle`*) Title to precede the flowables

Accepts: a list of styled text strings, separated by spaces. A styled text string is a quoted string (`'` or `"`), optionally followed by a style name enclosed in braces: `'text string' (style name)`

Default: (no value)

flowable_spacing
: *Dimension*

(*GroupedFlowablesStyle*) Spacing between flowables

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

space_above
: *Dimension*

(*FlowableStyle*) Vertical space preceding the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

space_below
: *Dimension*

(*FlowableStyle*) Vertical space following the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

margin_left
: *Dimension*

(*FlowableStyle*) Left margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

margin_right
: *Dimension*

(*FlowableStyle*) Right margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_left
: *Dimension*

(*FlowableStyle*) Left padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_right
: *Dimension*

(*FlowableStyle*) Right padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_top
: *Dimension*

(*FlowableStyle*) Top padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`padding_bottom`
`Dimension`

(*`FlowableStyle`*) Bottom padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`keep_with_next`
`Bool`

(*`FlowableStyle`*) Keep this flowable and the next on the same page

Accepts: `true` or `false`

Default: `false`

`border`
`Stroke`

(*`FlowableStyle`*) Border surrounding the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_left`
`Stroke`

(*`FlowableStyle`*) Border left of the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_right`
`Stroke`

(*`FlowableStyle`*) Border right of the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_top`
`Stroke`

(*`FlowableStyle`*) Border above the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_bottom`
`Stroke`

(*`FlowableStyle`*) Border below the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`background_color`
`Color`

(*`FlowableStyle`*) Color of the area within the flowable's borders

Accepts: HEX string with optional alpha component (`#RRGGBB`, `#RRGGBBAA`, `#RGB` or `#RGBA`)

Default: `none`

`indent_first`

`Dimension`

(*`ParagraphStyle`*) Indentation of the first line of text

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0pt`

`line_spacing`

`LineSpacing`

(*`ParagraphStyle`*) Spacing between the baselines of two successive lines of text

Accepts: a *predefined line spacing* type or the name of a line spacing type followed by comma-separated arguments enclosed in braces: `<spacing type>(<arg1>, <arg2>, ...)`

Default: `default`

`text_align`

`TextAlign`

(*`ParagraphStyle`*) Alignment of text to the margins

Accepts: `left`, `right`, `center`, `justify`

Default: `justify`

`tab_stops`

`TabStopList`

(*`ParagraphStyle`*) List of tab positions

Accepts: a comma-seperated list of tab stops. A tab stop is specified as `<position> [align] [fill string]`, where `<position>` (*`Dimension`*) is required and `align` (`TabAlign`) and `fill string` (string enclosed in quotes) are optional.

Default: (no value)

`space_above`

`Dimension`

(*`FlowableStyle`*) Vertical space preceding the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`space_below`

`Dimension`

(*`FlowableStyle`*) Vertical space following the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`margin_left`

`Dimension`

(*`FlowableStyle`*) Left margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

margin_right
Dimension

(*FlowableStyle*) Right margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_left
Dimension

(*FlowableStyle*) Left padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_right
Dimension

(*FlowableStyle*) Right padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_top
Dimension

(*FlowableStyle*) Top padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_bottom
Dimension

(*FlowableStyle*) Bottom padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

keep_with_next
`Bool`

(*FlowableStyle*) Keep this flowable and the next on the same page

Accepts: `true` or `false`

Default: `false`

border
`Stroke`

(*FlowableStyle*) Border surrounding the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

border_left
`Stroke`

(*FlowableStyle*) Border left of the flowable

Accepts: the width (*Dimension*) and color (Color) of the stroke, separated by a comma (,)

Default: none

border_right
Stroke

(*FlowableStyle*) Border right of the flowable

Accepts: the width (*Dimension*) and color (Color) of the stroke, separated by a comma (,)

Default: none

border_top
Stroke

(*FlowableStyle*) Border above the flowable

Accepts: the width (*Dimension*) and color (Color) of the stroke, separated by a comma (,)

Default: none

border_bottom
Stroke

(*FlowableStyle*) Border below the flowable

Accepts: the width (*Dimension*) and color (Color) of the stroke, separated by a comma (,)

Default: none

background_color
Color

(*FlowableStyle*) Color of the area within the flowable's borders

Accepts: HEX string with optional alpha component (#RRGGBB, #RRGGBBAA, #RGB or #RGBA)

Default: none

typeface
Typeface

(*TextStyle*) Typeface to set the text in

Accepts: the name of a typeface. See *rinoh --list-fonts* for a list of the available typefaces.

Default: Times

font_weight
FontWeight

(*TextStyle*) Thickness of character outlines relative to their height

Accepts: hairline, thin, ultra-light, extra-light, light, book, regular, medium, demi-bold, bold, extra-bold, heavy, black, extra-black, ultra-black

Default: medium

font_slant
FontSlant

(*TextStyle*) Slope style of the font

Accepts: upright, oblique, italic

Default: upright

font_width
FontWidth

(*TextStyle*) Stretch of the characters

Accepts: normal, condensed, extended

Default: normal

font_size
Dimension

(*TextStyle*) Height of characters

Accepts: a numeric value followed by a unit (pt, in, pc, mm, cm, %, /4)

Default: 10pt

font_color
Color

(*TextStyle*) Color of the font

Accepts: HEX string with optional alpha component (#RRGGBB, #RRGGBBAA, #RGB or #RGBA)

Default: #000

font_variant
FontVariant

(*TextStyle*) Variant of the font

Accepts: normal, small capital, oldstyle figures

Default: normal

position
TextPosition

(*TextStyle*) Vertical text position

Accepts: normal, superscript, subscript

Default: normal

kerning
Bool

(*TextStyle*) Improve inter-letter spacing

Accepts: true or false

Default: true

ligatures
Bool

(*TextStyle*) Run letters together where possible

Accepts: true or false

Default: true

hyphenate
Bool

(*TextStyle*) Allow words to be broken over two lines

Accepts: true or false

Default: `true`

`hyphen_chars`

`Integer`

(*`TextStyle`*) Minimum number of characters in a hyphenated part of a word

Accepts: a natural number (positive integer)

Default: `2`

`hyphen_lang`

`Locale`

(*`TextStyle`*) Language to use for hyphenation. Accepts locale codes such as 'en_US'

Accepts: locale identifier in the `<language ID>_<region ID>` format

Default: `en_US`

class `rinoh.structure.`**`TableOfContentsEntry`**(*flowable*, *id=None*, *style=None*, *parent=None*)

`style_class`

alias of *`TableOfContentsEntryStyle`*

class `rinoh.structure.`**`TableOfContentsEntryStyle`**(*base=None*, ***attributes*)

`text`

`ReferenceText`

Overrides the default set in *`ReferencingParagraphStyle`*

Accepts: a list of styled text strings, separated by spaces. A styled text string is a quoted string (`'` or `"`), optionally followed by a style name enclosed in braces: `'text string' (style name)`

Default: `'{NUMBER}' '\t' '{TITLE}' '\t' '{PAGE}'`

`indent_first`

`Dimension`

(*`ParagraphStyle`*) Indentation of the first line of text

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0pt`

`line_spacing`

`LineSpacing`

(*`ParagraphStyle`*) Spacing between the baselines of two successive lines of text

Accepts: a *predefined line spacing* type or the name of a line spacing type followed by comma-separated arguments enclosed in braces: `<spacing type>(<arg1>, <arg2>, ...)`

Default: `default`

`text_align`

`TextAlign`

(*`ParagraphStyle`*) Alignment of text to the margins

Accepts: `left`, `right`, `center`, `justify`

Default: `justify`

tab_stops
`TabStopList`

(*ParagraphStyle*) List of tab positions

Accepts: a comma-seperated list of tab stops. A tab stop is specified as `<position> [align] [fill string]`, where `<position>` (*Dimension*) is required and `align` (`TabAlign`) and `fill string` (string enclosed in quotes) are optional.

Default: (no value)

space_above
Dimension

(*FlowableStyle*) Vertical space preceding the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

space_below
Dimension

(*FlowableStyle*) Vertical space following the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

margin_left
Dimension

(*FlowableStyle*) Left margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

margin_right
Dimension

(*FlowableStyle*) Right margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_left
Dimension

(*FlowableStyle*) Left padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_right
Dimension

(*FlowableStyle*) Right padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_top
Dimension

(*FlowableStyle*) Top padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_bottom
Dimension

(*FlowableStyle*) Bottom padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

keep_with_next
`Bool`

(*FlowableStyle*) Keep this flowable and the next on the same page

Accepts: `true` or `false`

Default: `false`

border
`Stroke`

(*FlowableStyle*) Border surrounding the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

border_left
`Stroke`

(*FlowableStyle*) Border left of the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

border_right
`Stroke`

(*FlowableStyle*) Border right of the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

border_top
`Stroke`

(*FlowableStyle*) Border above the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

border_bottom
`Stroke`

(*FlowableStyle*) Border below the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

background_color
`Color`

(*FlowableStyle*) Color of the area within the flowable's borders

Accepts: HEX string with optional alpha component (`#RRGGBB`, `#RRGGBBAA`, `#RGB` or `#RGBA`)

Default: `none`

typeface
Typeface

(*TextStyle*) Typeface to set the text in

Accepts: the name of a typeface. See *rinoh --list-fonts* for a list of the available typefaces.

Default: `Times`

font_weight
`FontWeight`

(*TextStyle*) Thickness of character outlines relative to their height

Accepts: `hairline`, `thin`, `ultra-light`, `extra-light`, `light`, `book`, `regular`, `medium`, `demi-bold`, `bold`, `extra-bold`, `heavy`, `black`, `extra-black`, `ultra-black`

Default: `medium`

font_slant
`FontSlant`

(*TextStyle*) Slope style of the font

Accepts: `upright`, `oblique`, `italic`

Default: `upright`

font_width
`FontWidth`

(*TextStyle*) Stretch of the characters

Accepts: `normal`, `condensed`, `extended`

Default: `normal`

font_size
Dimension

(*TextStyle*) Height of characters

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `10pt`

font_color
`Color`

(*TextStyle*) Color of the font

Accepts: HEX string with optional alpha component (`#RRGGBB`, `#RRGGBBAA`, `#RGB` or `#RGBA`)

Default: `#000`

font_variant
`FontVariant`

(*TextStyle*) Variant of the font

Accepts: `normal`, `small capital`, `oldstyle figures`

Default: `normal`

`position`
`TextPosition`

(*`TextStyle`*) Vertical text position

Accepts: `normal`, `superscript`, `subscript`

Default: `normal`

`kerning`
`Bool`

(*`TextStyle`*) Improve inter-letter spacing

Accepts: `true` or `false`

Default: `true`

`ligatures`
`Bool`

(*`TextStyle`*) Run letters together where possible

Accepts: `true` or `false`

Default: `true`

`hyphenate`
`Bool`

(*`TextStyle`*) Allow words to be broken over two lines

Accepts: `true` or `false`

Default: `true`

`hyphen_chars`
`Integer`

(*`TextStyle`*) Minimum number of characters in a hyphenated part of a word

Accepts: a natural number (positive integer)

Default: `2`

`hyphen_lang`
`Locale`

(*`TextStyle`*) Language to use for hyphenation. Accepts locale codes such as 'en_US'

Accepts: locale identifier in the `<language ID>_<region ID>` format

Default: `en_US`

8.12.4 Adminitions

class `rinoh.structure.`**`Admonition`**(*flowables*, *title=None*, *type=None*, *id=None*, *style=None*, *parent=None*)

`style_class`
alias of *`AdmonitionStyle`*

class `rinoh.structure.`**`AdmonitionStyle`**(*base=None*, ***attributes*)

`inline_title`

`Bool`

Show the admonition's title inline with the body text, if possible

Accepts: `true` or `false`

Default: `true`

`title`

`StyledText`

(*`GroupedFlowablesStyle`*) Title to precede the flowables

Accepts: a list of styled text strings, separated by spaces. A styled text string is a quoted string (`'` or `"`), optionally followed by a style name enclosed in braces: `'text string' (style name)`

Default: (no value)

`flowable_spacing`

`Dimension`

(*`GroupedFlowablesStyle`*) Spacing between flowables

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`space_above`

`Dimension`

(*`FlowableStyle`*) Vertical space preceding the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`space_below`

`Dimension`

(*`FlowableStyle`*) Vertical space following the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`margin_left`

`Dimension`

(*`FlowableStyle`*) Left margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`margin_right`

`Dimension`

(*`FlowableStyle`*) Right margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_left
Dimension

(*FlowableStyle*) Left padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_right
Dimension

(*FlowableStyle*) Right padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_top
Dimension

(*FlowableStyle*) Top padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_bottom
Dimension

(*FlowableStyle*) Bottom padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

keep_with_next
`Bool`

(*FlowableStyle*) Keep this flowable and the next on the same page

Accepts: `true` or `false`

Default: `false`

border
`Stroke`

(*FlowableStyle*) Border surrounding the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

border_left
`Stroke`

(*FlowableStyle*) Border left of the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

border_right
`Stroke`

(*FlowableStyle*) Border right of the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_top`

`Stroke`

(*`FlowableStyle`*) Border above the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_bottom`

`Stroke`

(*`FlowableStyle`*) Border below the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`background_color`

`Color`

(*`FlowableStyle`*) Color of the area within the flowable's borders

Accepts: HEX string with optional alpha component (`#RRGGBB`, `#RRGGBBAA`, `#RGB` or `#RGBA`)

Default: `none`

class `rinoh.structure.`**`AdmonitionTitles`**`(`*****strings***`)`

Collection of localized titles for common admonitions

`attention`

`String`

Title for attention admonitions

`caution`

`String`

Title for caution admonitions

`danger`

`String`

Title for danger admonitions

`error`

`String`

Title for error admonitions

`hint`

`String`

Title for hint admonitions

`important`

`String`

Title for important admonitions

`note`

`String`

Title for note admonitions

tip
String

Title for tip admonitions

warning
String

Title for warning admonitions

seealso
String

Title for see-also admonitions

8.12.5 Horizontal Rule

class `rinoh.structure.`**`HorizontalRule`**(*id=None*, *style=None*, *parent=None*)

style_class
alias of *HorizontalRuleStyle*

class `rinoh.structure.`**`HorizontalRuleStyle`**(*base=None*, ***attributes*)

space_above
Dimension

(*FlowableStyle*) Vertical space preceding the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

space_below
Dimension

(*FlowableStyle*) Vertical space following the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

margin_left
Dimension

(*FlowableStyle*) Left margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

margin_right
Dimension

(*FlowableStyle*) Right margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_left
Dimension

(*FlowableStyle*) Left padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_right
Dimension

(*FlowableStyle*) Right padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_top
Dimension

(*FlowableStyle*) Top padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_bottom
Dimension

(*FlowableStyle*) Bottom padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

keep_with_next
`Bool`

(*FlowableStyle*) Keep this flowable and the next on the same page

Accepts: `true` or `false`

Default: `false`

border
`Stroke`

(*FlowableStyle*) Border surrounding the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

border_left
`Stroke`

(*FlowableStyle*) Border left of the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

border_right
`Stroke`

(*FlowableStyle*) Border right of the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

border_top
Stroke

(*FlowableStyle*) Border above the flowable

Accepts: the width (*Dimension*) and color (Color) of the stroke, separated by a comma (,)

Default: none

border_bottom
Stroke

(*FlowableStyle*) Border below the flowable

Accepts: the width (*Dimension*) and color (Color) of the stroke, separated by a comma (,)

Default: none

background_color
Color

(*FlowableStyle*) Color of the area within the flowable's borders

Accepts: HEX string with optional alpha component (#RRGGBB, #RRGGBBAA, #RGB or #RGBA)

Default: none

stroke
Stroke

(*LineStyle*) Width and color used to draw the line

Accepts: the width (*Dimension*) and color (Color) of the stroke, separated by a comma (,)

Default: 1pt, #000

8.13 Strings (rinoh.strings)

class rinoh.strings.**String**(*description*)
Descriptor used to describe a configurable string

Parameters **description** (*str*) – a short description for this string

class rinoh.strings.**StringCollection**(***strings*)
A collection of related configurable strings

Subclasses

- *AdmonitionTitles*
- *SectionTitles*

class rinoh.strings.**Strings**(**string_collections*)
Stores several *StringCollection*s

class rinoh.strings.**StringField**(*strings_class*, *key*, *case=None*, *style=None*, *parent=None*)
Styled text that will be substituted with a configured string

The configured string is either the localized string as determined by the language set for the document or the user-supplied string passed to the TemplateConfiguration

8.14 Style (`rinoh.style`)

class `rinoh.style.`**`Styled`**(*id=None*, *style=None*, *parent=None*)

A document element who's style can be configured.

Parameters **`style`** (*`str, Style`*) – the style to associate with this element. If *style* is a string, the corresponding style is lookup up in the document's style sheet by means of selectors.

`style_class`

The *`Style`* subclass that corresponds to this *`Styled`* subclass.

class `rinoh.style.`**`Style`**(*base=None*, ***attributes*)

Dictionary storing style attributes.

The style attributes associated with this *`Style`* are specified as class attributes of type `Attribute`.

Style attributes can also be accessed as object attributes.

`__repr__`()

Return a textual representation of this style.

`__getitem__`(*attribute*)

Return the value of *attribute*.

If the attribute is not specified in this *`Style`*, find it in this style's base styles (hierarchically), or ultimately raise a `DefaultValueException`.

8.14.1 Style Sheet

class `rinoh.style.`**`StyleSheet`**(*name*, *matcher=None*, *base=None*, *description=None*, *pygments_style=None*, ***user_options*)

Dictionary storing a collection of related styles by name.

*`Style`*s stored in a *`StyleSheet`* can refer to their base style by name.

Parameters

- **`name`** (*`str`*) – a label for this style sheet
- **`matcher`** (`StyledMatcher`) – the matcher providing the selectors the styles contained in this style sheet map to. If no matcher is given and *base* is specified, the *base*'s matcher is used. If *base* is not set, the default matcher is used.
- **`base`** (*`StyleSheet or str`*) – the style sheet to extend
- **`description`** (*`str`*) – a short string describing this style sheet
- **`pygments_style`** (*`str`*) – the Pygments style to use for styling code blocks

`get_selector`(*name*)

Find a selector mapped to a style in this or a base style sheet.

Parameters **`name`** (*`str`*) – a style name

Returns `Selector` – the selector mapped to the style *name*

Raises `KeyError` – if the style *name* was not found in this or a base style sheet

class `rinoh.style.`**`StyleSheetFile`**(*filename*, *base=None*, ***kwargs*)

Loads styles defined in a *.rts* file (INI format).

Parameters **`filename`** (*`str`*) – the path to the style sheet file

StyleSheetFile takes the same optional arguments as *StyleSheet*. These can also be specified in the `[STYLESHEET]` section of the style sheet file. If an argument is specified in both places, the one passed as an argument overrides the one specified in the style sheet file.

class `rinoh.style.`**`StyledMatcher`**(*mapping_or_iterable=None*, ***kwargs*)

Dictionary mapping labels to selectors.

This matcher can be initialized in the same way as a `dict` by passing a mapping, an interable and/or keyword arguments.

Included Style Sheets

These style sheets are included with rinohtype:

`rinoh.stylesheets.`**`sphinx`**

Mostly a copy of the LaTeX style included with Sphinx

Entry point name: `sphinx`

`rinoh.stylesheets.`**`sphinx_article`**

The Sphinx stylesheet adjusted for the article template

Entry point name: `sphinx_article`

`rinoh.stylesheets.`**`sphinx_base14`**

The Sphinx stylesheet, but using the fonts from the PDF core set (yielding smaller PDF files)

Entry point name: `sphinx_base14`

Additional style sheets can be installed from PyPI. The installed style sheets can be listed using *rinoh --list-stylesheets*.

The Default Matcher

`rinoh.stylesheets.`**`matcher`**

The default matcher defines the following styles

- bold: *TextStyle*
- title page author: *ParagraphStyle*
- title page extra: *ParagraphStyle*
- command: *TextStyle*
- emphasis: *TextStyle*
- body matter chapter title: *ParagraphStyle*
- code with variable: *TextStyle*
- fallback: *TextStyle*
- monospaced: *TextStyle*
- menu cascade: *TextStyle*
- UI control: *TextStyle*
- title page subtitle: *ParagraphStyle*
- strong: *TextStyle*
- literal emphasis: *TextStyle*

- man page: *TextStyle*
- program: *TextStyle*
- literal strong: *TextStyle*
- mail header: *TextStyle*
- window title: *TextStyle*
- title page logo: *HorizontallyAlignedFlowableStyle*
- title page date: *ParagraphStyle*
- title page title: *ParagraphStyle*
- quote: *TextStyle*
- keystrokes: *TextStyle*
- body matter chapter label: *ParagraphStyle*
- body matter chapter number: *TextStyle*
- UI control accelerator: *TextStyle*
- file path: *TextStyle*
- title page rule: *HorizontalRuleStyle*
- regular expression: *TextStyle*
- error: *TextStyle*
- title reference: *TextStyle*
- draft comment: *TextStyle*
- MIME type: *TextStyle*
- front matter section title: *ParagraphStyle*
- make variable: *TextStyle*
- newsgroup: *TextStyle*
- italic: *TextStyle*
- linked reference: *TextStyle*
- unlinked reference: *TextStyle*
- internal hyperlink: *TextStyle*
- external hyperlink: *TextStyle*
- broken hyperlink: *TextStyle*
- body: *ParagraphStyle*
- code block: `CodeBlockStyle`
- attribution: *ParagraphStyle*
- centered: *ParagraphStyle*
- block quote: *GroupedFlowablesStyle*
- line block: *GroupedFlowablesStyle*
- nested line block: *GroupedFlowablesStyle*

- line block line: *ParagraphStyle*
- title: *ParagraphStyle*
- subtitle: *ParagraphStyle*
- date: *ParagraphStyle*
- author: *ParagraphStyle*
- affiliation: *ParagraphStyle*
- chapter: *SectionStyle*
- heading level 1: *HeadingStyle*
- unnumbered heading level 1: *HeadingStyle*
- heading level 2: *HeadingStyle*
- unnumbered heading level 2: *HeadingStyle*
- heading level 3: *HeadingStyle*
- unnumbered heading level 3: *HeadingStyle*
- heading level 4: *HeadingStyle*
- unnumbered heading level 4: *HeadingStyle*
- heading level 5: *HeadingStyle*
- unnumbered heading level 5: *HeadingStyle*
- other heading levels: *HeadingStyle*
- appendix: *SectionStyle*
- appendix heading level 1: *HeadingStyle*
- appendix heading level 2: *HeadingStyle*
- appendix heading level 3: *HeadingStyle*
- appendix heading level 4: *HeadingStyle*
- appendix heading level 5: *HeadingStyle*
- prerequisites: *GroupedFlowablesStyle*
- prerequisites title: *ParagraphStyle*
- post requirement: *GroupedFlowablesStyle*
- abstract: *GroupedFlowablesStyle*
- abstract paragraph: *ParagraphStyle*
- example: *GroupedFlowablesStyle*
- example title: *ParagraphStyle*
- topic: *GroupedFlowablesStyle*
- topic title: *ParagraphStyle*
- rubric: *ParagraphStyle*
- sidebar: *GroupedFlowablesStyle*
- sidebar title: *ParagraphStyle*

- sidebar subtitle: `ParagraphStyle`
- list item label: `ListItemLabelStyle`
- enumerated list: `ListStyle`
- enumerated list item: `LabeledFlowableStyle`
- enumerated list item label: `ListItemLabelStyle`
- nested enumerated list: `ListStyle`
- (table) enumerated list: `ListStyle`
- (table) enumerated list item: `LabeledFlowableStyle`
- (table) enumerated list item label: `ListItemLabelStyle`
- bulleted list: `ListStyle`
- compact bulleted list: `ListStyle`
- bulleted list item: `LabeledFlowableStyle`
- bulleted list item label: `ListItemLabelStyle`
- nested bulleted list: `ListStyle`
- (table) bulleted list: `ListStyle`
- (table) bulleted list item: `LabeledFlowableStyle`
- (table) bulleted list item label: `ListItemLabelStyle`
- steps list: `ListStyle`
- steps list title: `ParagraphStyle`
- steps list item: `LabeledFlowableStyle`
- steps list item label: `ListItemLabelStyle`
- choices list: `ListStyle`
- choices list item: `LabeledFlowableStyle`
- choices list item label: `ListItemLabelStyle`
- list item body: `GroupedFlowablesStyle`
- list item paragraph: `ParagraphStyle`
- definition list: `GroupedFlowablesStyle`
- definition list item: `LabeledFlowableStyle`
- definition term: `GroupedFlowablesStyle`
- definition term paragraph: `ParagraphStyle`
- definition term classifier: `TextStyle`
- definition: `GroupedFlowablesStyle`
- definition paragraph: `ParagraphStyle`
- related links: `GroupedFlowablesStyle`
- related links section title: `ParagraphStyle`
- related links list: `ListStyle`

- related links list item: *LabeledFlowableStyle*
- related links list item label: *ListItemLabelStyle*
- related links list item paragraph: *ReferencingParagraphStyle*
- related link title reference: *TextStyle*
- related link page reference: *TextStyle*
- related link number reference: *TextStyle*
- related link reference: *TextStyle*
- versionmodified: *TextStyle*
- object description: *LabeledFlowableStyle*
- object signatures: *GroupedFlowablesStyle*
- object signature: *ParagraphStyle*
- object name: *TextStyle*
- additional name part: *TextStyle*
- object type: *TextStyle*
- object returns: *TextStyle*
- object parentheses: *TextStyle*
- object parameter list: *TextStyle*
- object parameter: *TextStyle*
- object parameter (no emphasis): *TextStyle*
- object brackets: *TextStyle*
- object optional parameter: *TextStyle*
- object annotation: *TextStyle*
- object description content: *GroupedFlowablesStyle*
- object description content paragraph: *ParagraphStyle*
- production list: *GroupedFlowablesStyle*
- production: *LabeledFlowableStyle*
- token name: *ParagraphStyle*
- token definition: *ParagraphStyle*
- field list: *GroupedFlowablesStyle*
- field list item: *LabeledFlowableStyle*
- field name: *ParagraphStyle*
- option list: *GroupedFlowablesStyle*
- option list item: *LabeledFlowableStyle*
- option: *ParagraphStyle*
- option string: *TextStyle*
- option argument: *TextStyle*

- admonition: *AdmonitionStyle*
- admonition title: *ParagraphStyle*
- admonition inline title: *TextStyle*
- attention admonition: *AdmonitionStyle*
- attention admonition title: *ParagraphStyle*
- attention admonition inline title: *TextStyle*
- caution admonition: *AdmonitionStyle*
- caution admonition title: *ParagraphStyle*
- caution admonition inline title: *TextStyle*
- danger admonition: *AdmonitionStyle*
- danger admonition title: *ParagraphStyle*
- danger admonition inline title: *TextStyle*
- error admonition: *AdmonitionStyle*
- error admonition title: *ParagraphStyle*
- error admonition inline title: *TextStyle*
- hint admonition: *AdmonitionStyle*
- hint admonition title: *ParagraphStyle*
- hint admonition inline title: *TextStyle*
- important admonition: *AdmonitionStyle*
- important admonition title: *ParagraphStyle*
- important admonition inline title: *TextStyle*
- note admonition: *AdmonitionStyle*
- note admonition title: *ParagraphStyle*
- note admonition inline title: *TextStyle*
- tip admonition: *AdmonitionStyle*
- tip admonition title: *ParagraphStyle*
- tip admonition inline title: *TextStyle*
- warning admonition: *AdmonitionStyle*
- warning admonition title: *ParagraphStyle*
- warning admonition inline title: *TextStyle*
- seealso admonition: *AdmonitionStyle*
- seealso admonition title: *ParagraphStyle*
- seealso admonition inline title: *TextStyle*
- header: *ParagraphStyle*
- footer: *ParagraphStyle*
- footnote marker: *NoteMarkerStyle*

- citation marker: *NoteMarkerStyle*
- footnote paragraph: *ParagraphStyle*
- footnote label: *ParagraphStyle*
- figure: *FigureStyle*
- image: *HorizontallyAlignedFlowableStyle*
- inline image: *InlineFlowableStyle*
- caption: *CaptionStyle*
- figure legend: *GroupedFlowablesStyle*
- figure legend paragraph: *ParagraphStyle*
- table of contents section: *SectionStyle*
- table of contents title: *HeadingStyle*
- table of contents: *TableOfContentsStyle*
- toc level 1: *TableOfContentsEntryStyle*
- toc level 2: *TableOfContentsEntryStyle*
- toc level 3: *TableOfContentsEntryStyle*
- L3 toc level 3: *TableOfContentsEntryStyle*
- toc linked reference: *TextStyle*
- table: *TableStyle*
- table with caption: *GroupedFlowablesStyle*
- choices table: *TableStyle*
- table cell: *TableCellStyle*
- table body cell background on even row: *TableCellBackgroundStyle*
- table body cell background on odd row: *TableCellBackgroundStyle*
- table body cell paragraph: *ParagraphStyle*
- table first column paragraph: *ParagraphStyle*
- table body cell list item number: *ParagraphStyle*
- table head cell: *TableCellStyle*
- table head cell paragraph: *ParagraphStyle*
- table cell left border: *TableCellBorderStyle*
- table cell top border: *TableCellBorderStyle*
- table cell right border: *TableCellBorderStyle*
- table cell bottom border: *TableCellBorderStyle*
- table top border: *TableCellBorderStyle*
- table bottom border: *TableCellBorderStyle*
- table left border: *TableCellBorderStyle*
- table right border: *TableCellBorderStyle*

- table head cell left border: *TableCellBorderStyle*
- table head cell right border: *TableCellBorderStyle*
- table head bottom border: *TableCellBorderStyle*
- table head left border: *TableCellBorderStyle*
- table head right border: *TableCellBorderStyle*
- table body top border: *TableCellBorderStyle*
- table body left border: *TableCellBorderStyle*
- table body right border: *TableCellBorderStyle*
- horizontal rule: *HorizontalRuleStyle*
- index: *IndexStyle*
- index section label: *ParagraphStyle*
- level 1 index entry: *ParagraphStyle*
- level 2 index entry: *ParagraphStyle*
- level 3 index entry: *ParagraphStyle*
- level 4 index entry: *ParagraphStyle*
- domain index entry name: *TextStyle*

8.15 Tables (`rinoh.table`)

class `rinoh.table.`**`TableStyle`**(*base=None*, ***attributes*)

`split_minimum_rows`

`Integer`

The minimum number of rows to display when the table is split across pages

Accepts: a natural number (positive integer)

Default: (no value)

`repeat_head`

`Bool`

Repeat the head when the table is split across pages

Accepts: `true` or `false`

Default: `false`

`width`

Dimension

(*HorizontallyAlignedFlowableStyle*) The width of the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `none`

horizontal_align

`HorizontalAlignment`

(*HorizontallyAlignedFlowableStyle*) Horizontal alignment of the flowable

Accepts: `left`, `right`, `center`

Default: `left`

space_above

Dimension

(*FlowableStyle*) Vertical space preceding the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

space_below

Dimension

(*FlowableStyle*) Vertical space following the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

margin_left

Dimension

(*FlowableStyle*) Left margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

margin_right

Dimension

(*FlowableStyle*) Right margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_left

Dimension

(*FlowableStyle*) Left padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_right

Dimension

(*FlowableStyle*) Right padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_top

Dimension

(*FlowableStyle*) Top padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`padding_bottom`

`Dimension`

(*`FlowableStyle`*) Bottom padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`keep_with_next`

`Bool`

(*`FlowableStyle`*) Keep this flowable and the next on the same page

Accepts: `true` or `false`

Default: `false`

`border`

`Stroke`

(*`FlowableStyle`*) Border surrounding the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_left`

`Stroke`

(*`FlowableStyle`*) Border left of the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_right`

`Stroke`

(*`FlowableStyle`*) Border right of the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_top`

`Stroke`

(*`FlowableStyle`*) Border above the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`border_bottom`

`Stroke`

(*`FlowableStyle`*) Border below the flowable

Accepts: the width (*`Dimension`*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

`background_color`

`Color`

(*`FlowableStyle`*) Color of the area within the flowable's borders

Accepts: HEX string with optional alpha component (`#RRGGBB`, `#RRGGBBAA`, `#RGB` or `#RGBA`)

Default: `none`

class `rinoh.table.`**`VerticalAlign`**

Accepts: `top`, `middle`, `bottom`

class `rinoh.table.`**`TableCellStyle`**(*base=None*, ***attributes*)

`vertical_align`

`VerticalAlign`

Vertical alignment of the cell contents within the available space

Accepts: `top`, `middle`, `bottom`

Default: `middle`

`title`

`StyledText`

(*`GroupedFlowablesStyle`*) Title to precede the flowables

Accepts: a list of styled text strings, separated by spaces. A styled text string is a quoted string (`'` or `"`), optionally followed by a style name enclosed in braces: `'text string' (style name)`

Default: (no value)

`flowable_spacing`

`Dimension`

(*`GroupedFlowablesStyle`*) Spacing between flowables

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`space_above`

`Dimension`

(*`FlowableStyle`*) Vertical space preceding the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`space_below`

`Dimension`

(*`FlowableStyle`*) Vertical space following the flowable

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`margin_left`

`Dimension`

(*`FlowableStyle`*) Left margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

`margin_right`

`Dimension`

(*FlowableStyle*) Right margin

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_left
Dimension

(*FlowableStyle*) Left padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_right
Dimension

(*FlowableStyle*) Right padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_top
Dimension

(*FlowableStyle*) Top padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

padding_bottom
Dimension

(*FlowableStyle*) Bottom padding

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `0`

keep_with_next
`Bool`

(*FlowableStyle*) Keep this flowable and the next on the same page

Accepts: `true` or `false`

Default: `false`

border
`Stroke`

(*FlowableStyle*) Border surrounding the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

border_left
`Stroke`

(*FlowableStyle*) Border left of the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

border_right
`Stroke`

(*FlowableStyle*) Border right of the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

border_top
`Stroke`

(*FlowableStyle*) Border above the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

border_bottom
`Stroke`

(*FlowableStyle*) Border below the flowable

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

background_color
`Color`

(*FlowableStyle*) Color of the area within the flowable's borders

Accepts: HEX string with optional alpha component (`#RRGGBB`, `#RRGGBBAA`, `#RGB` or `#RGBA`)

Default: `none`

class `rinoh.table.`**TableCellBorderStyle**(*base=None*, ***attributes*)

stroke
`Stroke`

Overrides the default set in *LineStyle*

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

class `rinoh.table.`**TableCellBackgroundStyle**(*base=None*, ***attributes*)

fill_color
`Color`

Overrides the default set in `ShapeStyle`

Accepts: HEX string with optional alpha component (`#RRGGBB`, `#RRGGBBAA`, `#RGB` or `#RGBA`)

Default: `none`

stroke
`Stroke`

Overrides the default set in *LineStyle*

Accepts: the width (*Dimension*) and color (`Color`) of the stroke, separated by a comma (`,`)

Default: `none`

8.16 Templates (`rinoh.template`)

Document templates are created by subclassing *`DocumentTemplate`*, just like the *standard templates* shipped with rinohtype.

class `rinoh.template.`**`DocumentTemplate`**(*document_tree*, *configuration=None*, *backend=None*)

Template for documents

Parameters

- **`document_tree`** (`DocumentTree`) – a tree of the document's contents
- **`configuration`** (`TemplateConfiguration`) – configuration for this template
- **`backend`** – the backend used for rendering the document

`language`

`Language`

The main language of the document

Accepts: the code of one of the *supported languages*

Default: *`EN`* (English)

`strings`

`Strings`

Strings to override standard element names

Accepts: strings need to be entered in INI sections named after the *`StringCollection`* subclasses

Default: `none`

`stylesheet`

`StyleSheet`

The stylesheet to use for styling document elements

Accepts: the name of an *installed style sheet* or the filename of a stylesheet file (with the `.rts` extension)

Default: `sphinx` (= *`rinoh.stylesheets.sphinx`*)

`parts`

`PartsList`

The parts making up this document

Accepts: a space-separated list of document part template names

Default: (empty list)

Document templates can be customized by setting values for the configuration attributes defined in a *`DocumentTemplate`* subclass in a *`TemplateConfiguration`*. An template configuration can be passed as *configuration* on template instantiation. However, it is better to make use of the *`document`* method, however.

class `rinoh.template.`**`TemplateConfiguration`**(*name*, *base=None*, *template=None*, *description=None*, ***options*)

Stores a configuration for a *`DocumentTemplate`*

Parameters

- **`name`** (*`str`*) – a label for this template configuration
- **`base`** (`TemplateConfiguration`) – the template configuration to extend

- **template** (*DocumentTemplateMeta or str*) – the document template to configure
- **description** (*str*) – a short string describing this style sheet
- ****options** – configuration values for the configuration attributes defined by the document *template*

template = **None**

The *DocumentTemplate* subclass to configure

document (*document_tree*, *backend=None*)

Create a *DocumentTemplate* object based on the given document tree and this template configuration

Parameters

- **document_tree** (DocumentTree) – tree of the document's contents
- **backend** – the backend to use when rendering the document

class rinoh.template.**PartsList** (**parts*)

Stores the names of the document part templates making up a document

Parameters ***parts** (*list[str]*) – the names of the document parts

8.16.1 Document Parts

The document part templates which are listed by name in *DocumentTemplate.parts* are looked up as attributes of the *DocumentTemplate* subclass. They are instances of *DocumentPartTemplate* subclasses:

class rinoh.template.**DocumentPartTemplate** (*base=None*, ***attributes*)

A template that produces a document part

The document part is created given a set of flowables, and page templates. The latter are looked up in the *TemplateConfiguration* where this part template was.

page_number_format

NumberFormat

The format for page numbers in this document part. If it is different from the preceding part's number format, numbering restarts at 1

Accepts: none, number, symbol, lowercase character, uppercase character, lowercase roman, uppercase roman

Default: number

end_at_page

PageType

The type of page to end this document part on

Accepts: left, right, any

Default: any

drop_if_empty

Bool

Exclude this part from the document if it is empty (no flowables)

Accepts: true or false

Default: true

The following document part templates are used in the standard document templates:

class `rinoh.template.`**`TitlePartTemplate`**(*base=None*, ***attributes*)

The title page of a document.

`drop_if_empty`

`Bool`

Overrides the default set in *`DocumentPartTemplate`*

Accepts: `true` or `false`

Default: `false`

`page_number_format`

`NumberFormat`

(*`DocumentPartTemplate`*) The format for page numbers in this document part. If it is different from the preceding part's number format, numbering restarts at 1

Accepts: `none`, `number`, `symbol`, `lowercase character`, `uppercase character`, `lowercase roman`, `uppercase roman`

Default: `number`

`end_at_page`

`PageType`

(*`DocumentPartTemplate`*) The type of page to end this document part on

Accepts: `left`, `right`, `any`

Default: `any`

class `rinoh.template.`**`ContentsPartTemplate`**(*base=None*, ***attributes*)

The body of a document.

Renders all of the content present in the *`DocumentTree`* passed to the *`DocumentTemplate`*.

`page_number_format`

`NumberFormat`

(*`DocumentPartTemplate`*) The format for page numbers in this document part. If it is different from the preceding part's number format, numbering restarts at 1

Accepts: `none`, `number`, `symbol`, `lowercase character`, `uppercase character`, `lowercase roman`, `uppercase roman`

Default: `number`

`end_at_page`

`PageType`

(*`DocumentPartTemplate`*) The type of page to end this document part on

Accepts: `left`, `right`, `any`

Default: `any`

`drop_if_empty`

`Bool`

(*`DocumentPartTemplate`*) Exclude this part from the document if it is empty (no flowables)

Accepts: `true` or `false`

Default: `true`

class `rinoh.template.`**`FixedDocumentPartTemplate`**`(`*base=None*, ***attributes*`)`

A document part template that renders a fixed list of flowables

`flowables`

`FlowablesList`

The list of flowables to include in this document part

Accepts: Python source code that represents a list of *`Flowable`*s

Default: `[]`

`page_number_format`

`NumberFormat`

(*`DocumentPartTemplate`*) The format for page numbers in this document part. If it is different from the preceding part's number format, numbering restarts at 1

Accepts: `none`, `number`, `symbol`, `lowercase character`, `uppercase character`, `lowercase roman`, `uppercase roman`

Default: `number`

`end_at_page`

`PageType`

(*`DocumentPartTemplate`*) The type of page to end this document part on

Accepts: `left`, `right`, `any`

Default: `any`

`drop_if_empty`

`Bool`

(*`DocumentPartTemplate`*) Exclude this part from the document if it is empty (no flowables)

Accepts: `true` or `false`

Default: `true`

8.16.2 Page Templates

The document templates make use of page templates:

class `rinoh.template.`**`PageTemplate`**`(`*base=None*, ***attributes*`)`

`header_footer_distance`

`Dimension`

Distance of the header and footer to the content area

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `14pt`

`columns`

`Integer`

The number of columns for the body text

Accepts: a natural number (positive integer)

Default: `1`

`column_spacing`

`Dimension`

The spacing between columns

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `1cm`

`header_text`

`StyledText`

The text to place in the page header

Accepts: a list of styled text strings, separated by spaces. A styled text string is a quoted string (' or "), optionally followed by a style name enclosed in braces: `'text string' (style name)`

Default: `'{SECTION_NUMBER}' ' ' '{SECTION_TITLE}'`

`footer_text`

`StyledText`

The text to place in the page footer

Accepts: a list of styled text strings, separated by spaces. A styled text string is a quoted string (' or "), optionally followed by a style name enclosed in braces: `'text string' (style name)`

Default: `'\t' '{PAGE_NUMBER}' '/' '{NUMBER_OF_PAGES}'`

`chapter_header_text`

`StyledText`

The text to place in the header on a page that starts a new chapter

Accepts: a list of styled text strings, separated by spaces. A styled text string is a quoted string (' or "), optionally followed by a style name enclosed in braces: `'text string' (style name)`

Default: (no value)

`chapter_footer_text`

`StyledText`

The text to place in the footer on a page that starts a new chapter

Accepts: a list of styled text strings, separated by spaces. A styled text string is a quoted string (' or "), optionally followed by a style name enclosed in braces: `'text string' (style name)`

Default: (no value)

`chapter_title_flowables`

`FlowablesList`

Generator that yields the flowables to represent the chapter title

Accepts: Python source code that represents a list of *`Flowable`*s

Default: `[Paragraph[StringField(<class 'rinoh.structure.SectionTitles'>, 'chapter')] (style=None), Paragraph[Field({SECTION_NUMBER})] (style=None), Paragraph[Field({SECTION_TITLE})] (style=None)]`

`chapter_title_height`

`Dimension`

The height of the container holding the chapter title

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `150pt`

`page_size`

`Paper`

(*`PageTemplateBase`*) The format of the pages in the document

Accepts: the name of a *predefined paper format* or `<width> * <height>` where `width` and `height` are *`Dimension`*s

Default: `A4`

`page_orientation`

`PageOrientation`

(*`PageTemplateBase`*) The orientation of pages in the document

Accepts: `portrait`, `landscape`

Default: `portrait`

`left_margin`

`Dimension`

(*`PageTemplateBase`*) The margin size on the left of the page

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `3cm`

`right_margin`

`Dimension`

(*`PageTemplateBase`*) The margin size on the right of the page

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `3cm`

`top_margin`

`Dimension`

(*`PageTemplateBase`*) The margin size at the top of the page

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `3cm`

`bottom_margin`

`Dimension`

(*`PageTemplateBase`*) The margin size at the bottom of the page

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `3cm`

`background`

`BackgroundImage`

(*`PageTemplateBase`*) An image to place in the background of the page

Accepts: filename of an image file enclosed in quotes, optionally followed by space-delimited keyword arguments (`<keyword>=<value>`) that determine how the image is displayed

Default: `none`

after_break_background
BackgroundImage

(*PageTemplateBase*) An image to place in the background after a page break

Accepts: filename of an image file enclosed in quotes, optionally followed by space-delimited keyword arguments (`<keyword>=<value>`) that determine how the image is displayed

Default: `none`

class `rinoh.template.`**`TitlePageTemplate`**(*base=None*, ***attributes*)

show_date
`Bool`

Show or hide the document's date

Accepts: `true` or `false`

Default: `true`

show_author
`Bool`

Show or hide the document's author

Accepts: `true` or `false`

Default: `true`

extra
StyledText

Extra text to include on the title page below the title

Accepts: a list of styled text strings, separated by spaces. A styled text string is a quoted string (`'` or `"`), optionally followed by a style name enclosed in braces: `'text string' (style name)`

Default: (no value)

page_size
Paper

(*PageTemplateBase*) The format of the pages in the document

Accepts: the name of a *predefined paper format* or `<width> * <height>` where `width` and `height` are *Dimension*s

Default: `A4`

page_orientation
PageOrientation

(*PageTemplateBase*) The orientation of pages in the document

Accepts: `portrait`, `landscape`

Default: `portrait`

left_margin
Dimension

(*PageTemplateBase*) The margin size on the left of the page

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `3cm`

right_margin
Dimension

(*PageTemplateBase*) The margin size on the right of the page

Accepts: a numeric value followed by a unit (pt, in, pc, mm, cm, %, /4)

Default: 3cm

top_margin
Dimension

(*PageTemplateBase*) The margin size at the top of the page

Accepts: a numeric value followed by a unit (pt, in, pc, mm, cm, %, /4)

Default: 3cm

bottom_margin
Dimension

(*PageTemplateBase*) The margin size at the bottom of the page

Accepts: a numeric value followed by a unit (pt, in, pc, mm, cm, %, /4)

Default: 3cm

background
BackgroundImage

(*PageTemplateBase*) An image to place in the background of the page

Accepts: filename of an image file enclosed in quotes, optionally followed by space-delimited keyword arguments (<keyword>=<value>) that determine how the image is displayed

Default: none

after_break_background
BackgroundImage

(*PageTemplateBase*) An image to place in the background after a page break

Accepts: filename of an image file enclosed in quotes, optionally followed by space-delimited keyword arguments (<keyword>=<value>) that determine how the image is displayed

Default: none

The base class for these collects the common options:

class rinoh.template.**PageTemplateBase**(*base=None*, ***attributes*)

page_size
Paper

The format of the pages in the document

Accepts: the name of a *predefined paper format* or <width> * <height> where width and height are *Dimension*s

Default: A4

page_orientation
PageOrientation

The orientation of pages in the document

Accepts: portrait, landscape

Default: `portrait`

`left_margin`

`Dimension`

The margin size on the left of the page

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `3cm`

`right_margin`

`Dimension`

The margin size on the right of the page

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `3cm`

`top_margin`

`Dimension`

The margin size at the top of the page

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `3cm`

`bottom_margin`

`Dimension`

The margin size at the bottom of the page

Accepts: a numeric value followed by a unit (`pt`, `in`, `pc`, `mm`, `cm`, `%`, `/4`)

Default: `3cm`

`background`

`BackgroundImage`

An image to place in the background of the page

Accepts: filename of an image file enclosed in quotes, optionally followed by space-delimited keyword arguments (`<keyword>=<value>`) that determine how the image is displayed

Default: `none`

`after_break_background`

`BackgroundImage`

An image to place in the background after a page break

Accepts: filename of an image file enclosed in quotes, optionally followed by space-delimited keyword arguments (`<keyword>=<value>`) that determine how the image is displayed

Default: `none`

CHAPTER 9

Frequenty Asked Questions

PDFs produced by rinohtype contain mostly empty pages. What's up? Old versions of some PDF viewers do not support the way rinohtype embeds fonts in a PDF (see issue 2). PDF viewers that are known to be affected are:

- pre-37.0 Firefox's built-in PDF viewer (pdf.js)
- pre-0.41 poppler-based applications such as Evince

CHAPTER 10

Release History

10.1 Release 0.3.1.dev

Changed:

- more closely mimic the Sphinx LaTeX builder's title page (issue #60)

10.2 Release 0.3.0 (2016-11-23)

New Features:

- support localization of standard document strings (en, fr, it, nl) (#53)
- localized strings can be overridden in the document template configuration
- make use of a fallback typeface when a glyph is not available (#55) (the 'fallback' style in the Sphinx stylesheet sets the fallback typeface)
- template configuration (INI) files: specify which document parts to include, configure document part and page templates, customize localized strings, ...
- support specifying more complex selectors directly in a style sheet file
- (figure and table) captions support hierarchical numbering (see CaptionStyle)
- make the frontends independent of the current working directory
- reStructuredText: support the table :widths: option (upcoming docutils 0.13)
- Sphinx frontend: provide styles for Sphinx's inline markup roles
- rinoh (command line renderer):
  - support template configuration files
  - support file formats for which a frontend is installed (see –list-formats)
  - accept options to configure the frontend (see –list-options)
  - option to list the installed fonts (on the command line or in a PDF file)
- show the current page number as part of the rendering progress indicator
- Book template: support for setting a cover page
- frontends: raise a more descriptive exception when a document tree node is not mapped

- validate the default value passed to an Attribute
- preliminary support for writing a style sheet to an INI file, listing default values for non-specified attributes (#23)

Changed:

- rinoh: the output PDF is now placed in the current directory, not in the same directory as the input file
- Sphinx builder configuration: replace the `rinoh_document_template` and `rinoh_template_configuration` options with `rinoh_template`
- if no base is given for a style, style attribute lookup proceeds to look in the style of the same name in the base style sheet (#66)
- DEFAULT_STYLE can be used as a base style to prevent style attribute lookup in the style of the same name in the base style sheet
- rename FieldList to DefinitionList and use it to replace uses (docutils and Sphinx frontends) of the old DefinitionList (#54)
- the new DefinitionList (FieldList) can be styled like the old DefinitionList by setting max_label_width to None, 0 or a 0-valued Dimension
- figures are now non-floating by default (float placement needs more work)
- hide the index chapter when there are no index entries (#51)
- style sheets: use the default matcher if none is specified
- Sphinx style sheet: copy the admonition style from the Sphinx LaTeX builder
- Sphinx style sheet: keep the admonition title together with the body
- Sphinx style sheet: color linked references as in the LaTeX output (#62)
- Sphinx style sheet: disable hyphenation/ligatures for literal strong text
- no more DocumentSection; a document now consists of parts (containing pages)
- template configuration:
  - refer to document part templates by name so that they can be replaced
  - the list of document parts can be changed in the template configuration
  - document parts take the 'end_at_page' option (left, right, or any)
  - find (left/right) page templates via the document part name they belong to
  - fall back to <doc_part>_page when the right or left template is not found
  - each template configuration requires a name
- DocumentTree: make the `source_file` argument optional
- don't abort when the document section hierarchy is missing levels (#67)
- use the PDF backend by default (no need to specify it)
- store the unit with Dimension instances (better printing)
- rename the *float* module to *image*

Fixed:

- improve compatibility with Windows: Windows path names and file encoding
- crash if a StyledText is passed to HeadingStyle.number_separator

- GroupedLabeledFlowables label width could be unnecessarily wide
- fix and improve automatic table column sizing
- Figures can now be referenced using the ‘reference’ format (“Figure 1.2”)
- HorizontallyAlignedFlowable: make more robust
- make document elements referenceable by secondary IDs
- reStructuredText: only the first classifier for a definition term was shown
- Sphinx frontend: support the ‘centered’ directive
- Sphinx frontend: basic support for the ‘hlist’ directive
- Sphinx frontend: handle :abbr: without explanation
- Sphinx frontend: support nested inline nodes (guilabel & samp roles)
- PDF backend: fix writing of Type 1 fonts from a parsed PDF file
- PDF reader: handle multi-page PDFs (#71)
- PDF reader: fix parsing of XRef streams
- PDF reader: fix writing of parsed files

10.3 Release 0.2.1 (2016-08-18)

New Features:

- optionally limit the width of large images and make use of this to simulate the Sphinx LaTeX builder behavior (#46)
- reStructuredText/Sphinx: support for images with hyperlinks (#49)
- record the styled page numbers in the PDF as page labels (#41)
- unsupported Python versions: prevent installation where possible (sdist) or exit on import (wheel)
- support Python 3.6

Bugfixes:

- make StyleSheet objects picklable so the Sphinx builder’s rinoh_stylesheet option can actually be used
- Fix #47: ClassNotFound exception in Literal_Block.lexer_getter()
- Fix #45: Images that don’t fit are still placed on the page
- don’t warn about duplicate style matches that resolve to the same style

10.4 Release 0.2.0 (2016-08-10)

Styling:

- generate a style log (show matching styles) to help style sheet development
- keep_with_next style attribute: prevent splitting two flowables across pages
- stylesheets can be loaded from files in INI format
- check the type of attributes passed to styles

- source code highlighting using Pygments
- table of contents entries can be styled more freely
- allow hiding the section numbers of table of contents entries
- allow for custom chapter titles
- selectors can now also select based on document part/section
- various small tweaks to selectors and matchers
- various fixes relating to style sheets

Templates:

- configurable standard document templates: article and book
- a proper infrastructure for creating custom document templates
- support for left/right page templates
- make the Article template more configurable
- pages now have background, content and header/footer layers
- support for generating an index
- make certain strings configurable (for localization, for example)

Frontends:

- Sphinx: interpret the LaTeX configuration variables if the corresponding rinohtype variable is not set
- Sphinx: roughly match the LaTeX output (document template and style sheet)
- added a CommonMark frontend based on recommonmark
- added basic ePUB and DocBook frontends
- XML frontends: fix whitespace handling
- frontends now return generators yielding flowables (more flexible)

Command-line Renderer (rinoh):

- allow specifying a template and style sheet
- automatically install typefaces used in the style sheet from PyPI

Fonts:

- typefaces are discovered/loaded by entry point
- more complete support for OpenType fonts
- fix support for the 14 base Type 1 fonts

Images:

- more versatile image sizing: absolute width/height & scaling
- allow specifying the baseline for inline images
- several fixes in the JPEG reader

Miscellaneous:

- reorganize the Container class hierarchy
- fixes in footnote handling

- drop Python 3.2 support (3.3, 3.4 and 3.5 are supported)

10.5 Release 0.1.3 (2015-08-04)

- recover from the slow rendering speed caused by a bugfix in 0.1.2 (thanks to optimized element matching in the style sheets)
- other improvements and bugfixes related to style sheets

10.6 Release 0.1.2 (2015-07-31)

- much improved Sphinx support (we can now render the Sphinx documentation)
- more complete support for reStructuredText (docutils) elements
- various fixes related to footnote placement
- page break option when starting a new section
- fixes in handling of document sections and parts
- improvements to section/figure/table references
- native support for PNG and JPEG images (drops PIL/Pillow requirement, but adds PurePNG 0.1.1 requirement)
- new ‘sphinx’ stylesheet used by the Sphinx builder (~ Sphinx LaTeX style)
- restores Python 3.2 compatibility

10.7 Release 0.1.1 (2015-04-12)

First preview release

Python Module Index

Index