

beaulivre

WRITE YOUR BOOKS IN
A COLORFUL WAY

Corresponding to: beaulivre 2021/08/08

JINWEN XU

August 2021, Beijing

Preface

`beaulivre` is a member of the `colorist` class series. Its name is taken from French words “beau” (for “beautiful”) and “livre” (for “book”). The entire collection includes `colorart` and `lebhart` for typesetting articles and `colorbook` and `beaulivre` for typesetting books. My original intention in designing this series was to write drafts and notes that look colorful yet not dazzling.

`beaulivre` has multi-language support, including Chinese (simplified and traditional), English, French, German, Italian, Japanese, Portuguese (European and Brazilian), Russian and Spanish. These languages can be switched seamlessly in a single document. Due to the usage of custom fonts, `lebhart` requires \LaTeX or \Lua\TeX to compile.

This documentation is typeset using `beaulivre` (with the option `allowbf`). You can think of it as a short introduction and demonstration.

TIP

Multi-language support, theorem-like environments, draft marks and some other features are provided by the [ProjLib](#) toolkit. Here we only briefly discuss how to use it with this document class. For more detailed information, you can refer to the documentation of [ProjLib](#).

Contents

I INSTRUCTION

Before you start	3
1 Usage and examples	5
1.1 How to load it	5
1.2 Example - A complete document	5
1.2.1 Initialization	6
1.2.2 Set the language	6
1.2.3 Draft marks	6
1.2.4 Theorem-like environments	6
2 On the default fonts	7
3 The options	9
4 Instructions by topic	11
4.1 Language configuration	11
4.2 Theorems and how to reference them	12
4.3 Define a new theorem-like environment	13
4.4 Draft mark	14
5 Known issues	15

II DEMONSTRATION

6 Heading on Level 0 (chapter)	19
6.1 Heading on Level 1 (section)	19
6.1.1 Heading on Level 2 (subsection)	19

6.2 Lists 20

6.2.1 Example for list (itemize) 20

6.2.2 Example for list (enumerate) 20

6.2.3 Example for list (description) 20

PART I

INSTRUCTION

You can add some introductory text here via `\parttext<text>`.

Before you start

In order to use the package or classes described here, you need to:

- install TeX Live or MikTeX of the latest possible version, and make sure that `colorist` and `projlib` are correctly installed in your TeX system.
- download and install the required fonts, see the section “On the default fonts”.
- be familiar with the basic usage of L^AT_EX, and knows how to compile your document with pdfL^AT_EX, X_YL^AT_EX or LuaL^AT_EX.

1

Usage and examples

1.1 How to load it

One only needs to put

```
\documentclass{beaulivre}
```

as the first line to use the beaulivre class.

ATTENTION

You need to use either Xe_{La}TeX or Lua_{La}TeX engine to compile.

1.2 Example - A complete document

Let's first look at a complete document.

```
1 \documentclass{colorbook}
2 \usepackage{ProjLib}
3
4 \UseLanguage{French}
5
6 \begin{document}
7
8 \frontmatter
9
10 \begin{titlepage}
11     <code for titlepage>
12 \end{titlepage}
13
14 \tableofcontents
15
16 \mainmatter
17
18 \part{<part title>}
19 \parttext{<text after part title>}
20
21 \chapter{<chapter title>}
22
23 \section{<section title>}
24
25 \dnf<Plus de contenu est nécessaire.>
```

```

26
27 \begin{theorem}\label{thm:abc}
28     Ceci est un théorème.
29 \end{theorem}
30 Référence du théorème: \cref{thm:abc}
31
32 \backmatter
33
34 ...
35
36 \end{document}

```

If you find this example a little complicated, don't worry. Let's now look at this example piece by piece.

1.2.1 Initialization

```

\documentclass{beaulivre}
\usepackage{ProjLib}

```

Initialization is straightforward. The first line loads the document class `beaulivre`, and the second line loads the `ProjLib` toolkit to obtain some additional functionalities.

1.2.2 Set the language

```

\UseLanguage{French}

```

This line indicates that French will be used in the document (by the way, if only English appears in your article, then there is no need to set the language). You can also switch the language in the same way later in the middle of the text. Supported languages include Simplified Chinese, Traditional Chinese, Japanese, English, French, German, Spanish, Portuguese, Brazilian Portuguese and Russian.

For detailed description of this command and more related commands, please refer to the section on the multi-language support.

1.2.3 Draft marks

```

\dntf<(some hint)>

```

When you have some places that have not yet been finished yet, you can mark them with this command, which is especially useful during the draft stage.

1.2.4 Theorem-like environments

```

\begin{theorem}\label{thm:abc}
    Ceci est un théorème.
\end{theorem}
Référence du théorème: \cref{thm:abc}

```

Commonly used theorem-like environments have been pre-defined. Also, when referencing a theorem-like environment, it is recommended to use `\cref{<label>}` — in this way, there is no need to explicitly write down the name of the corresponding environment every time.

2

On the default fonts

By default, this document class uses Palatino Linotype as the English main font; Source Han Serif, Source Han Sans and Source Han Mono as the Chinese main font, sans serif font and typewriter font; and partially uses Neo Euler as the math font. You need to download and install these fonts by yourself. The Source Han font series can be downloaded at <https://github.com/adobe-fonts> (It is recommended to download the Super-OTC version, so that the download size is smaller). Neo Euler can be downloaded at <https://github.com/khaledhosny/euler-otf>. When the corresponding font is not installed, the font that comes with TeX Live will be used instead, and the effect may be discounted.

In addition, Source Code Pro is used as the English sans serif font, New Computer Modern Mono as the English monospace font, as well as some symbols in the mathematical fonts of Asana Math, Tex Gyre Pagella Math, and Latin Modern Math. These fonts are already available in TeX Live or MikTeX, which means you don't need to install them yourself.

- English main font. English sans serif font. English typewriter font.
- 简体中文主要字体，简体中文无衬线字体，简体中文等宽字体
- 繁體中文主要字體，繁體中文無襯線字體，繁體中文等寬字體
- 日本語のメインフォント、日本語のサンセリフフォント、日本語の等幅フォント
- Math demonstration: $\alpha, \beta, \gamma, \delta, 1, 2, 3, 4, a, b, c, d,$

$$\text{li}(x) := \int_2^x \frac{1}{\log t} dt$$

3

The options

beaulivre offers the following options:

- The language options EN / english / English, FR / french / French, etc.
 - For the option names of a specific language, please refer to *<language name>* in the next section. The first specified language will be used as the default language.
 - The language options are optional, mainly for increasing the compilation speed. Without them the result would be the same, only slower.
- draft or fast
 - The option fast enables a faster but slightly rougher style, main differences are:
 - * Use simpler math font configuration;
 - * Do not use hyperref;
 - * Enable the fast mode of ProjLib toolkit.

TIP

During the draft stage, it is recommended to use the fast option to speed up compilation. When in fast mode, there will be a watermark “DRAFT” to indicate that you are currently in the draft mode.

- a4paper or b5paper
 - Paper size options. The default paper size is 8.5in × 11in.
- palatino, times, garamond, noto, biolinum | useosf
 - Font options. As the name suggest, font with corresponding name will be loaded.
 - The useosf option is used to enable the old-style figures.
- allowbf
 - Allow boldface. When this option is enabled, the main title, the titles of all levels and the names of theorem-like environments will be bolded.
- runin
 - Use the “runin” style for `\subsubsection`
- puretext or nothms
 - Pure text mode. Does not load theorem-like environments.
- nothmnum, thmnum or thmnum=<counter>
 - Theorem-like environments will not be numbered / numbered in order 1, 2, 3... / numbered within *<counter>*. Here, *<counter>* should be a built-in counter (such as subsection) or a custom counter defined in the preamble. If no option is used, they will be numbered within chapter (book) or section (article).

- `regionalref`, `originalref`
 - When referencing, whether the name of the theorem-like environment changes with the current language. The default is `regionalref`, *i.e.*, the name corresponding to the current language is used; for example, when referencing a theorem-like environment in English context, the names “Theorem, Definition...” will be used no matter which language context the original environment is in. If `originalref` is enabled, then the name will always remain the same as the original place; for example, when referencing a theorem written in the French context, even if one is currently in the English context, it will still be displayed as “Théorème”.
 - In fast mode, the option `originalref` will have no effect.

In addition, the commonly used `oneside` and `twoside` options are also available. Two-page layout is used by default.

4

Instructions by topic

4.1 Language configuration

beaulivre has multi-language support, including Chinese (simplified and traditional), English, French, German, Italian, Japanese, Portuguese (European and Brazilian), Russian and Spanish. The language can be selected by the following macros:

- `\UseLanguage{⟨language name⟩}` is used to specify the language. The corresponding setting of the language will be applied after it. It can be used either in the preamble or in the main body. When no language is specified, “English” is selected by default.
- `\UseOtherLanguage{⟨language name⟩}{⟨content⟩}`, which uses the specified language settings to typeset `⟨content⟩`. Compared with `\UseLanguage`, it will not modify the line spacing, so line spacing would remain stable when CJK and Western texts are mixed.

`⟨language name⟩` can be (it is not case sensitive, for example, French and french have the same effect):

- Simplified Chinese: CN, Chinese, SChinese or SimplifiedChinese
- Traditional Chinese: TC, TChinese or TraditionalChinese
- English: EN or English
- French: FR or French
- German: DE, German or ngerman
- Italian: IT or Italian
- Portuguese: PT or Portuguese
- Portuguese (Brazilian): BR or Brazilian
- Spanish: ES or Spanish
- Japanese: JP or Japanese
- Russian: RU or Russian

In addition, you can also add new settings to selected language:

- `\AddLanguageSetting{⟨settings⟩}`
 - Add `⟨settings⟩` to all supported languages.
- `\AddLanguageSetting(⟨language name⟩){⟨settings⟩}`
 - Add `⟨settings⟩` to the selected language `⟨language name⟩`.

For example, `\AddLanguageSetting(German){\color{orange}}` can make all German text displayed in orange (of course, one then need to add `\AddLanguageSetting{\color{black}}` in order to correct the color of the text in other languages).

4.2 Theorems and how to reference them

Environments such as `definition` and `theorem` have been preset and can be used directly.

More specifically, preset environments include: `assumption`, `axiom`, `conjecture`, `convention`, `corollary`, `definition`, `definition-proposition`, `definition-theorem`, `example`, `exercise`, `fact`, `hypothesis`, `lemma`, `notation`, `observation`, `problem`, `property`, `proposition`, `question`, `remark`, `theorem`, and the corresponding unnumbered version with an asterisk `*` in the name. The titles will change with the current language. For example, `theorem` will be displayed as “Theorem” in English mode and “Théorème” in French mode.

When referencing a theorem-like environment, it is recommended to use `\cref{<label>}`. In this way, there is no need to explicitly write down the name of the corresponding environment every time.

EXAMPLE

```
\begin{definition}[Strange things] \label{def: strange} ...
```

will produce

DEFINITION 4.1 (Strange things) This is the definition of some strange objects. There is approximately an one-line space before and after the theorem environment, and there will be a symbol to mark the end of the environment.

`\cref{def: strange}` will be displayed as: **DEFINITION 4.1**.

After using `\UseLanguage{French}`, a theorem will be displayed as:

THÉORÈME 4.2 (Inutile) Un théorème en français.

By default, when referenced, the name of the theorem matches the current context. For example, the definition above will be displayed in French in the current French context: **DÉFINITION 4.1** et **THÉORÈME 4.2**. If you want the name of the theorem to always match the language of the context in which the theorem is located, you can add `originalref` to the global options.

The following are the main styles of theorem-like environments:

THEOREM 4.3 Theorem style: theorem, proposition, lemma, corollary, ...

Proof | Proof style



Remark style



CONJECTURE 4.4 Conjecture style

EXAMPLE Example style: example, fact, ...

PROBLEM 4.5 Problem style: problem, question, ...

For aesthetics, adjacent definitions will be connected together automatically:

DEFINITION 4.6 First definition.

DEFINITION 4.7 Second definition.

4.3 Define a new theorem-like environment

If you need to define a new theorem-like environment, you must first define the name of the environment in the language to use:

- `\NameTheorem[⟨language name⟩]{⟨name of environment⟩}{⟨name string⟩}`

For *⟨language name⟩*, please refer to the section on language configuration. When *⟨language name⟩* is not specified, the name will be set for all supported languages. In addition, environments with or without asterisk share the same name, therefore, `\NameTheorem{envname*}{...}` has the same effect as `\NameTheorem{envname}{...}`.

And then define this environment in one of following five ways:

- `\CreateTheorem*{⟨name of environment⟩}`
 - Define an unnumbered environment *⟨name of environment⟩*
- `\CreateTheorem{⟨name of environment⟩}`
 - Define a numbered environment *⟨name of environment⟩*, numbered in order 1,2,3,...
- `\CreateTheorem{⟨name of environment⟩}[⟨numbered like⟩]`
 - Define a numbered environment *⟨name of environment⟩*, which shares the counter *⟨numbered like⟩*
- `\CreateTheorem{⟨name of environment⟩}<⟨numbered within⟩>`
 - Define a numbered environment *⟨name of environment⟩*, numbered within the counter *⟨numbered within⟩*
- `\CreateTheorem{⟨name of environment⟩}(⟨existed environment⟩)`
`\CreateTheorem*{⟨name of environment⟩}(⟨existed environment⟩)`
 - Identify *⟨name of environment⟩* with *⟨existed environment⟩* or *⟨existed environment⟩**.
 - This method is usually useful in the following two situations:
 1. To use a more concise name. For example, with `\CreateTheorem{thm}(theorem)`, one can then use the name `thm` to write theorem.
 2. To remove the numbering of some environments. For example, one can remove the numbering of the `remark` environment with `\CreateTheorem{remark}(remark*)`.

TIP

This macro utilizes the feature of `amsthm` internally, so the traditional `theoremstyle` is also applicable to it. One only needs declare the style before the relevant definitions.

Here is an example. The following code:

```
\NameTheorem[EN]{proofidea}{Idea}  
\CreateTheorem*{proofidea*}  
\CreateTheorem{proofidea}<section>
```

defines an unnumbered environment `proofidea*` and a numbered environment `proofidea` (numbered within section) respectively. They can be used in English context. The effect is as follows:

Idea | The `proofidea*` environment. ☐

Idea 4.3.1 | The `proofidea` environment. ☐

4.4 Draft mark

You can use `\dnf` to mark the unfinished part. For example:

- `\dnf` or `\dnf<...>`. The effect is: `To be finished #1` or `To be finished #2: ...`.

The prompt text changes according to the current language. For example, it will be displayed as `Pas encore fini #3` in French mode.

Similarly, there is `\needgraph`:

- `\needgraph` or `\needgraph<...>`. The effect is:

`A graph is needed here #1`

or

`A graph is needed here #2: ...`

The prompt text changes according to the current language. For example, in French mode, it will be displayed as

`Il manque une image ici #3`

5

Known issues

- The font settings are still not perfect.
- The design is not complete yet, including the TOC, part, and chapter.
- Since many features are based on the [ProjLib](#) toolkit, `colorist` (and hence `colorart`, `lebhart` and `colorbook`, `beaulivre`) inherits all its problems. For details, please refer to the “Known Issues” section of the [ProjLib](#) documentation.
- The error handling mechanism is incomplete: there is no corresponding error prompt when some problems occur.
- There are still many things that can be optimized in the code.

PART II

DEMONSTRATION

6

Heading on Level 0 (chapter)

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

6.1 Heading on Level 1 (section)

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

6.1.1 Heading on Level 2 (subsection)

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Heading on Level 3 (subsubsection)

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

HEADING ON LEVEL 4 (PARAGRAPH) Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

6.2 Lists

6.2.1 Example for list (itemize)

- First item in a list
- Second item in a list
- Third item in a list
- Fourth item in a list
- Fifth item in a list

Example for list (4*itemize)

- First item in a list
 - First item in a list
 - * First item in a list
 - First item in a list
 - Second item in a list
 - * Second item in a list
 - Second item in a list
- Second item in a list

6.2.2 Example for list (enumerate)

1. First item in a list
2. Second item in a list
3. Third item in a list
4. Fourth item in a list
5. Fifth item in a list

Example for list (4*enumerate)

1. First item in a list
 - a) First item in a list
 - (i) First item in a list
 - A. First item in a list
 - B. Second item in a list
 - (ii) Second item in a list
 - b) Second item in a list
2. Second item in a list

6.2.3 Example for list (description)

First item in a list
Second item in a list
Third item in a list
Fourth item in a list
Fifth item in a list

Example for list (4*description)

First item in a list
 First item in a list
 First item in a list
 First item in a list

Second item in a list

Second item in a list

Second item in a list

Second item in a list