ProjLib

beaulivre

Write your books in a colorful way

Corresponding to: beaulivre 2021/06/07

Jinwen XU

June 2021, Beijing

This page is intentionally left blank

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, beaulivre requires XeLaTeX or LuaLaTeX to compile.

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

Since the main body of colorist is modified from the minimalist series, some elements have not been completely redesigned yet, especially the TOC, part and chapter style. These will be added gradually in the future versions.

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.

This page is intentionally left blank

Contents

This page is intentionally left blank

1 Initialization

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 XeLaTeX or LuaLaTeX engine to compile.

1.2 Options

beaulivre offers the following options:

- `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.
- `a4paper` or `b5paper`
  - Optional paper size. The default paper size is 8.5in × 11in.
- `palatino`, `times`, `garamond`, `biolinum`
  - Font options. As the name suggest, font with corresponding name will be loaded.
- `allowbf`
  - Allow boldface. When this option is enabled, the title, titles of all levels and names of theorem-like environments will be bolded.

Tip

- During the draft stage, it is recommended to use the `fast` option to speed up compilation. At the end, one should remove the "fast" option to get the final version. When in `fast` mode, there will be a watermark "DRAFT" to indicate that you are currently in the draft mode.

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

This page is intentionally left blank

2 On the fonts

By default, beaulivre uses Palatino Linotype as the English font, FounderType's YouSong and YouHei GBK as the Chinese fonts, and partially uses Neo Euler as the math font. Among them, Neo Euler can be downloaded at `https://github.com/khaledhosny/euler-otf`. The other fonts are not free, you need to purchase and install them on your own. (For the Chinese fonts, visit FounderType's website for detail: `https://www.foundertype.com`).

FONT DEMONSTRATION

- English main font. English sans serif font. `English typewriter font`.
- 中文主要字体，中文无衬线字体
- 数学示例: $\alpha, \beta, \gamma, \delta, 1, 2, 3, 4, a, b, c, d,$

$$\operatorname{li}(x) \coloneqq \int_2^\infty \frac{1}{\log t}\,\mathrm{d}t$$

When the corresponding font is not installed, fonts that comes with TeX Live will be used instead. In this case, the experience might be reduced.

This page is intentionally left blank

3 Some instructions

Many of the features described next are provided by the ProjLib toolkit. Only the basic usage is mentioned here. For more details, please refer to its user documentation.

3.1 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).

3.2 Theorems and how to reference them

Environments such as `definition` and `theorem` have been pre-defined 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 3.1 (Strange things) This is the definition of some strange objects.

`\cref{def: strange}` will be displayed as: **DEFINITION 3.1**.
After using `\UseLanguage{French}`, a theorem will be displayed as:

THÉORÈME 3.1 (Inutile) Un théorème en français.

By default, when referenced, the name of the theorem always matches the language of the context in which the theorem is located. For example, the definition above is still displayed in English in the current French mode : **DEFINITION 3.1** and **THÉORÈME 3.1**. If you want the name of the theorem to match the current context when referencing, you can add `regionalref` to the global options.

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

THEOREM 3.2 Theorem style: theorem, proposition, lemma, corollary, ...

Proof | Proof style

Remark style

CONJECTURE 3.1 Conjecture style

EXAMPLE Example style: example, fact, ...

PROBLEM 3.1 Problem style: problem, question, ...

For aesthetics, adjacent definitions will be connected together automatically:

DEFINITION 3.2 First definition.

DEFINITION 3.3 Second definition.

3.3 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. 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 3.3.1 | The `proofidea` environment.

3.4 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`

3.5 Known issues

- The font settings are still not perfect.
- Since many features are based on the ProjLib toolkit, 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.

4 Document templates

4.1 The standard way

If you want to write in the standard way, you can refer to the following example:

```
\documentclass{beaulivre}
\usepackage{PJLtoolkit} % Load ProjLib toolkit

\UseLanguage{French} % Use French from here

\begin{document}

\title{Le Titre}
\author{Auteur}
\date{\PJLdate{2022-04-01}}

\maketitle

\chapter{Un théorème}

%% Theorem-like environments can be used directly
\begin{theorem}\label{thm:abc}
    Ceci est un théorème.
\end{theorem}

Référence du théorème: \cref{thm:abc}
    % It is recommended to use clever reference

\end{document}
```

If you wish to switch to the standard class later, just replace the first two lines with:

```
\documentclass{article}
\usepackage[a4paper,margin=1in]{geometry}
\usepackage[hidelinks]{hyperref}
\usepackage[palatino]{PJLtoolkit} % Load ProjLib toolkit
```

4.2 The $\mathcal{AMS}$ way

If you intend to switch to the journal template in the future and thus want to use the writing style as in the $\mathcal{AMS}$ classes, you can refer to the following example:

```
\documentclass{beaulivre}
\usepackage{PJLtoolkit} % Load ProjLib toolkit

\UseLanguage{French} % Use French from here

\begin{document}

\title{Le Titre}
\author{Auteur 1}
\address{Adresse 1}
\email{\href{Courriel 1}{Courriel 1}}
\author{Auteur 1}
\address{Adresse 1}
\email{\href{Courriel 2}{Courriel 2}}
\date{\PJLdate{2022-04-01}}
\subjclass{*****}
\keywords{...}

\maketitle

\chapter{Première section}

%% Theorem-like environments can be used directly
\begin{theorem}\label{thm:abc}
    Ceci est un théorème.
\end{theorem}

Référence du théorème: \cref{thm:abc}
    % It is recommended to use clever reference

\end{document}
```

In this way, if you wish to switch to $\mathcal{AMS}$ class later, just replace the first two lines with:

```
\documentclass{amsart}
\usepackage[a4paper,margin=1in]{geometry}
\usepackage[hidelinks]{hyperref}
\usepackage[palatino]{PJLtoolkit} % Load ProjLib toolkit
```

5 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.

5.1 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.

5.1.1 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.

5.2 Lists

5.2.1 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

5.2.2 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

5.2.3 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