

brilliant-cv

Documentation

<i>Authors:</i> yunanwg
<i>Build Date:</i> 2026-01-22
<i>Version:</i> 3.1.1

Contents

1. Introduction	2
2. Setup	2
3. Migration from v1 to v2	3
4. Configuration via metadata.toml	4
5. Functions	6

1. Introduction

Brilliant CV is a Typst template for making Résumé, CV or Cover Letter inspired by the famous LaTeX CV template Awesome-CV.

2. Setup

Step 1: Install Fonts

In order to make Typst render correctly, you will have to install the required fonts [Roboto](#) and [Source Sans Pro](#) (or Source Sans 3) in your local system.

Step 2: Check Documentation

You are reading this documentation now, woah!

Step 3: Bootstrap Template

In your local system, just working like `git clone`, bootstrap the template using this command:

```
1 typst init @preview/brilliant-cv:<version>
```

bash

Replace the `<version>` with the latest or any releases (after 2.0.0).

Step 4: Compile Files

Adapt the `metadata.toml` to suit your needs, then `typst c cv.typ` to get your first CV!

Step 5: Go beyond

It is recommended to:

1. Use `git` to manage your project, as it helps trace your changes and version control your CV.
2. Use `typstyle` and `pre-commit` to help you format your CV.
3. Use `typos` to check typos in your CV if your main locale is English.
4. (Advanced) Use LTex in your favorite code editor to check grammars and get language suggestions.

3. Migration from v1 to v2

With an existing CV project using the v1 version of the template, a migration is needed, including replacing some files / some content in certain files.

1. Delete brilliant-cv folder, .gitmodules. (Future package management will directly be managed by Typst)
2. Migrate all the config on metadata.typ by creating a new metadata.toml. Follow the example toml file in the repo,

it is rather straightforward to migrate.

3. For cv.typ and letter.typ, copy the new files from the repo, and adapt the modules you have in your project.
4. For the module files in /modules_* folders:
 - a. Delete the old import #import "../brilliant-CV/template.typ": *, and replace it by the import statements in the new template files.
 - b. Due to the Typst path handling mechanism, one cannot directly pass the path string to some functions anymore. This concerns, for example, the logo argument in cvEntry, but also on cvPublication as well. Some parameter names were changed, but most importantly, you should pass a function instead of a string (i.e. image("logo.png") instead of "logo.png"). Refer to new template files for reference.
 - c. You might need to install Roboto and Source Sans Pro on your local system now, as new Typst package discourages including these large files.
 - d. Run typst c cv.typ without passing the font-path flag. All should be good now, congrats!

Feel free to raise an issue for more assistance should you encounter a problem that you cannot solve on your own :)

4. Configuration via `metadata.toml`

The `metadata.toml` file is the main configuration file for your CV. By changing the key-value pairs in the config file, you can setup the names, contact information, and other details that will be displayed in your CV.

You can also override the language set in `metadata.toml` via the CLI:

```
1 typst compile cv.typ --input language=fr
```

bash

Here is an example of a `metadata.toml` file:

```
1 # INFO: value must matches folder suffix; i.e "zh" -> "./modules_zh"
2 language = "en"
3
4 [layout]
5 # Optional values: skyblue, red, nephritis, concrete, darknight
6 awesome_color = "skyblue"
7
8 # Skips are for controlling the spacing between sections and entries
9 before_section_skip = "1pt"
10 before_entry_skip = "1pt"
11 before_entry_description_skip = "1pt"
12
13 # Font size for the content text
14 font_size = "9pt"
15
16 [layout.header]
17 # Optional values: left, center, right
18 header_align = "left"
19
20 # Decide if you want to display profile photo or not
21 display_profile_photo = true
22 # Radius in % to clip profile photo at
23 display_profile_photo = "50%"
24 profile_photo_path = "template/src/avatar.png"
25
26 [layout.entry]
27 # Decide if you want to put your company in bold or your position in bold
28 display_entry_society_first = true
29
30 # Decide if you want to display organisation logo or not
31 display_logo = true
32
33 [inject]
34 # Decide if you want to inject AI prompt or not
35 inject_ai_prompt = false
36
37 # Decide if you want to inject keywords or not
```

toml

```

38 inject_keywords = true
39 injected_keywords_list = ["Data Analyst", "GCP", "Python", "SQL", "Tableau"]
40
41 [personal]
42 first_name = "John"
43 last_name = "Doe"
44
45 # The order of this section will affect how the entries are displayed
46 # The custom value is for any additional information you want to add
47 [personal.info]
48 github = "yunanwg"
49 phone = "+33 6 12 34 56 78"
50 email = "john.doe@me.org"
51 linkedin = "johndoe"
52 # gitlab = "yunanwg"
53 # homepage: "jd.me.org"
54 # orcid = "0000-0000-0000-0000"
55 # researchgate = "John-Doe"
56 # extraInfo = "I am a cool kid"
57 # custom-1 = (icon: "", text: "example", link: "https://example.com")
58
59 # add a new section if you want to include the language of your choice
60 # i.e. [[lang.ru]]
61 # each section must contains the following fields
62 [lang.en]
63 header_quote = "Experienced Data Analyst looking for a full time job starting from now"
64 cv_footer = "Curriculum vitae"
65 letter_footer = "Cover letter"
66
67 [lang.fr]
68 header_quote = "Analyste de données expérimenté à la recherche d'un emploi à temps plein disponible dès maintenant"
69 cv_footer = "Résumé"
70 letter_footer = "Lettre de motivation"
71
72 [lang.zh]
73 header_quote = "数据分析专家，寻求全职工作机会"
74 cv_footer = "简历"
75 letter_footer = "求职信"
76
77 # For languages that are not written in Latin script
78 # Currently supported non-latin language codes: ("zh", "ja", "ko", "ru")
79 [lang.non_latin]
80 name = "姓名"
81 font = "Heiti SC"

```

5. Functions

cv-entry

Add an entry to the CV.

- title (str): The title of the entry.
- society (str): The society of the entry (company, university, etc.).
- date (str | content): The date(s) of the entry.
- location (str): The location of the entry.
- description (array): The description of the entry. It can be a string or an array of strings.
- logo (image): The logo of the society. If empty, no logo will be displayed.
- tags (array): The tags of the entry.
- metadata (array): (optional) the metadata read from the TOML file.
- awesome-colors (array): (optional) the awesome colors of the CV.

Parameters

```
cv-entry(  
    title,  
    society,  
    date,  
    location,  
    description,  
    logo,  
    tags,  
    metadata,  
    awesome-colors,  
    awesomeColors  
) -> content
```

cv-entry-start

Add the start of an entry to the CV.

- society (str): The society of the entry (company, university, etc.).
- location (str): The location of the entry.
- logo (image): The logo of the society. If empty, no logo will be displayed.
- metadata (array): (optional) the metadata read from the TOML file.
- awesome-colors (array): (optional) the awesome colors of the CV.

Parameters

```
cv-entry-start(  
    society,  
    location,  
    logo,  
    metadata,  
    awesome-colors,  
    awesomeColors  
) -> content
```

cv-honor

Add a Honor to the CV.

- date (str): The date of the honor.
- title (str): The title of the honor.
- issuer (str): The issuer of the honor.
- url (str): The URL of the honor.
- location (str): The location of the honor.
- awesome-colors (array): (optional) The awesome colors of the CV.
- metadata (array): (optional) The metadata read from the TOML file.

Parameters

```
cv-honor(  
    date,  
    title,  
    issuer,  
    url,  
    location,  
    awesome-colors,  
    metadata  
) -> content
```

cv-publication

Add the publications to the CV by reading a bib file.

- bib (bibliography): The bibliography object with the path to the bib file.
- keyList (list): The list of keys to include in the publication list.
- refStyle (str): The reference style of the publication list.
- refFull (bool): Whether to show the full reference or not.

Parameters

```
cv-publication(  
    bib,  
    ref-style,  
    ref-full,  
    key-list,  
    refStyle,  
    refFull,  
    keyList  
) -> content
```

cv-section

Add the title of a section.

NOTE: If the language is non-Latin, the title highlight will not be sliced.

- title (str): The title of the section.
- highlighted (bool): Whether the first n letters will be highlighted in accent color.
- letters (int): The number of first letters of the title to highlight.

- metadata (array): (optional) the metadata read from the TOML file.
- awesome-colors (array): (optional) the awesome colors of the CV.

Parameters

```
cv-section(
    title,
    highlighted,
    letters,
    metadata,
    awesome-colors,
    awesomeColors
) -> content
```

cv-skill

Add a skill to the CV.

- type (str): The type of the skill. It is displayed on the left side.
- info (str | content): The information about the skill. It is displayed on the right side. Items can be separated by #hbar().

Parameters

```
cv-skill(
    type,
    info
) -> content
```

cv-skill-tag

Add a skill tag to the CV.

- skill (str | content): The skill to be displayed.

Parameters

```
cv-skill-tag(skill) -> content
```

cv-skill-with-level

Add a skill with a level to the CV.

- type (str): The type of the skill. It is displayed on the left side.
- level (int): The level of the skill. It is displayed in as circles in the middle. The minimum level is 0 and the maximum level is 5.
- info (str | content): The information about the skill. It is displayed on the right side.

Parameters

```
cv-skill-with-level(  
    type,  
    level,  
    info  
) -> content
```

cv-metadata

Metadata state to avoid passing metadata to every function