

# Catppuccin for Typst

🐾 Soothing pastel theme for Typst



v1.0.0      November 24, 2024

<https://github.com/catppuccin/typst>

TimeTravelPenguin

## Abstract

The **catppuccin** package provides colourful **Catppuccin** aesthetics for **Typst** documents. It provides four soothing pastel themes that is easy on the eyes. This manual provides a detailed documentation of the package.

## Contents

1. Overview .....	2
1.1. About .....	2
1.2. Basic Usage .....	2
2. Modules .....	3
2.1. Catppuccin .....	3
2.1.1. catppuccin .....	3
3. Flavors .....	3
3.1. Flavor Schema .....	3
3.1.1. get-flavor .....	4
3.1.2. parse-flavor .....	5
3.1.3. flavors .....	5
3.2. Tidy Styles .....	5
3.2.1. get-tidy-colors .....	5
3.3. Version .....	5
3.3.1. version .....	5

# 1. Overview

## 1.1. About

This document provides a detailed documentation of the **catppuccin** package for Typst. Inspired by the [L<sup>A</sup>T<sub>E</sub>X Catppuccin package](#), this package hopes to make writing in Typst more pleasurable and easy to use.

As someone who has done a lot of L<sup>A</sup>T<sub>E</sub>X, I found myself spending a lot of time writing in dark themes (usually by inverting the document colors). Eventually I found the Catppuccin package for L<sup>A</sup>T<sub>E</sub>X, and I incorporated it into my custom preamble to allow me to enable, disable, or configure the enabled theme. When I finished, I would submit my work with the theme disabled, without explicitly removing code!

I have plans for the future of this package, such as added styling and perhaps integration with other packages (if that ever becomes easier to do without making a new package).

## 1.2. Basic Usage

Using this package is simple. See Listing 1 for an example of how to use the package.

```
1 #import "catppuccin.typ": catppuccin, flavors
2
3 #show: catppuccin.with(flavor: flavors.mocha)
4
5 // The rest of your document
```

Listing 1: Example usage of the Catppuccin package

You can disable the theme by commenting out or deleting the show block.

## 2. Modules

### 2.1. Catppuccin

- `catppuccin()`

#### 2.1.1. `catppuccin`

Configure your document to use a Catppuccin flavor.

Example:

```
1 #import "@preview/catppuccin": catppuccin, flavors
2
3 #show: catppuccin.with(flavors.mocha)
```

This should be used at the top of your document.

- `flavor` (string | flavor): The flavor to set.

#### Parameters

```
catppuccin(
  flavor,
  body: content
) -> content
```

**body** `content`

The content to apply the flavor to.

## 3. Flavors

The Catppuccin package comes with four flavors: **Latte**, **Frappe**, **Macchiato**, and **Mocha**. Each flavor has its own unique color palette that is easy on the eyes. You can choose a flavor by setting the `flavor` parameter in the `catppuccin.with` function.

In this package, we refer to the dictionary related to each flavor with the type alias `flavor`.

### 3.1. Flavor Schema

Here we describe the schema for the `flavor` dictionary. Use `get-flavor()` function to

- **name** `string` — The name of the flavor (e.g. Frappé)
- **emoji** `string` — The emoji associated with the flavor.
- **order** `integer` — The order of the flavor in the Catppuccin lineup.
- **dark** `boolean` — Whether the flavor is a dark theme.
- **light** `boolean` — Whether the flavor is a light theme.
- **colors** `dictionary` — A dictionary of colors used in the flavor. Keys are the color names as a `string` and values are dictionaries with the following keys:
  - **name** `string` — The name of the color.
  - **order** `integer` — The order of the color in the palette.
  - **hex** `string` — The hex value of the color.
  - **rgb** `string` — The RGB value of the color.
  - **accent** `boolean` — Whether the color is an accent color.

- get-flavor()
- parse-flavor()

#### Variables:

- flavors

#### 3.1.1. get-flavor

Get the palette for the given flavor.

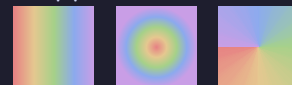
#### Example

```
1      #let      items      = 
2      flavors.values().map(flavor => [
3          "red", "yellow", "green",
4          "blue", "mauve",
5      ].map(c => flavor.colors.at(c).rgb)
6
7      #let fills = (
8          gradient.linear(..rainbow),
9          gradient.radial(..rainbow),
10         gradient.conic(..rainbow),
11     )
12
13     #stack(
14         dir: ttb,
15         spacing: 4pt,
16         text(flavor.name + ":"),
17         stack(
18             dir: ltr,
19             spacing: 3mm,
20             ..fills.map(fill => square(fill: fill))
21         )
22     )
23 ]
24
25     #grid(columns: 1, gutter: 1em, ..items)
```

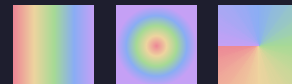
Latte:



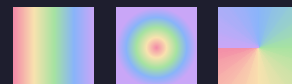
Frappé:



Macchiato:



Mocha:



#### Parameters

get-flavor(flavor: **string**) -> **dictionary**

**flavor** **string**

The flavor name as a string to get the flavor for. This function is provided as a helper for anyone requiring dynamic resolution of a flavor.

### 3.1.2. parse-flavor

Parse a flavor. If the flavor is a string, get the flavor from the dictionary. Otherwise, assert that the flavor is a valid flavor.

- flavor (string | dictionary): The flavor to parse.

#### Parameters

parse-flavor(flavor) -> dictionary

### 3.1.3. flavors dictionary

The available flavors for Catppuccin. Given simply by the dictionary

```
1 #let flavors = (  
2   latte: { ... },  
3   frappe: { ... },  
4   macchiato: { ... },  
5   mocha: { ... },  
6 )
```

## 3.2. Tidy Styles

- get-tidy-colors()

### 3.2.1. get-tidy-colors

#### Parameters

get-tidy-colors(flavor: string) -> dictionary

flavor string

The name of the flavor to use.

Default: flavors.mocha

## 3.3. Version

Variables:

- version

### 3.3.1. version version

The package version of Catppuccin.

Example:

```
1 This package's version is  
  #version.
```

This package's version is 1.0.0.