

# Catppuccin for Typst

Soothing pastel theme for Typst



v0.1.0      August 20, 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
2.1.2. get-palette .....	3
2.1.3. themes .....	4
3. Flavors .....	5
3.1. Flavor Schema .....	5
3.1.1. latte .....	5
3.1.2. frappe .....	5
3.1.3. macchiato .....	5
3.1.4. mocha .....	6
3.2. Tidy Styles .....	6
3.2.1. get-tidy-colors .....	6
3.3. Version .....	6
3.3.1. version .....	6

# 1. Overview

## 1.1. About

This document provides a detailed documentation of the **catppuccin** package for Typst. Inspired by the  $\text{\LaTeX}$  [Catppuccin package](#), this package hopes to make writing in Typst more pleasurable and easy to use.

As someone who has done a lot of  $\text{\LaTeX}$ , 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  $\text{\LaTeX}$ , 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.

```
#import "catppuccin.typ": catppuccin, themes

#show: catppuccin.with(
  flavor: themes.mocha,
  code-block: true,
  code-syntax: true,
)

// 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. Just note that if you are manually accessing palettes via the [get-palette](#)(flavor) function, you will need to manually account for those changes. It is planned to make this easier in the future be it though a redesign or simple helper functions.

## 2. Modules

### 2.1. Catppuccin

- [catppuccin\(\)](#)
- [get-palette\(\)](#)

Variables:

- [themes](#)

#### 2.1.1. catppuccin

Configure your document to use a Catppuccin flavor.

Example:

```
#import "@preview/catppuccin": catppuccin, themes

#show: catppuccin.with(themes.mocha, code-block: true, code-syntax: true)
```

This should be used at the top of your document.

Parameters

```
catppuccin(  
  theme: string,  
  code-block: boolean,  
  code-syntax: boolean,  
  body: content  
) -> content
```

**theme** string

The flavor to set.

**code-block** boolean

Whether to stylise code blocks.

Default: true

**code-syntax** boolean

Whether to use Catppuccin syntax highlighting in code blocks.

Default: true

**body** content

The content to apply the flavor to.

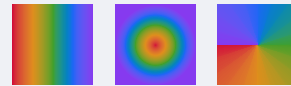
#### 2.1.2. get-palette

Get the color palette for the given theme. The returned dictionary has keys as defined in Flavor Schemas 3.1..

## Example

```
#let items = themes.values().map(theme => [  
  #let palette = get-palette(theme)  
  #let rainbow = (  
    "red", "yellow", "green",  
    "blue", "mauve",  
  ).map(c => palette.colors.at(c).rgb)  
  
  #let fills = (  
    gradient.linear(..rainbow),  
    gradient.radial(..rainbow),  
    gradient.conic(..rainbow),  
  )  
  
  #stack(  
    dir: ttb,  
    spacing: 4pt,  
    text(palette.name + ":"),  
    stack(  
      dir: ltr,  
      spacing: 3mm,  
      ..fills.map(fill => square(fill: fill))  
    )  
  )  
])  
  
#grid(columns: 1, gutter: 1em, ..items)
```

Latte:



Frappé:



Macchiato:



Mocha:



## Parameters

`get-palette(theme: string) -> dictionary`

**theme** `string`

The theme to get the palette for. The dict `themes` can be used to simplify this.

### 2.1.3. themes `dictionary`

The available flavors for Catppuccin. Given simply by the dictionary

```
#let themes = (  
  latte: "latte",  
  frappe: "frappe",  
  macchiato: "macchiato",  
  mocha: "mocha",  
)
```

These names are used to set the theme of the document. To access the accented names, you can use `get-palette()` and access the `name` key.

## 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-palette()` 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.

Variables:

- `latte`
- `frappe`
- `macchiato`
- `mocha`

#### 3.1.1. latte `flavor`

The Latte color palette.

Example

```
#let theme = themes.latte
#let palette = get-palette(theme)
Selected theme: #palette.name #palette.emoji
```

Selected theme: Latte

#### 3.1.2. frappe `flavor`

The Frappé color palette.

Example

```
#let theme = themes.frappe
#let palette = get-palette(theme)
Selected theme: #palette.name #palette.emoji
```

Selected theme: Frappé

#### 3.1.3. macchiato `flavor`

The Macchiato color palette.

### Example

```
#let theme = themes.macchiato
#let palette = get-palette(theme)
Selected theme: #palette.name #palette.emoji
```

Selected theme: Macchiato

#### 3.1.4. mocha flavor

The Mocha color palette.

### Example

```
#let theme = themes.mocha
#let palette = get-palette(theme)
Selected theme: #palette.name #palette.emoji
```

Selected theme: Mocha

## 3.2. Tidy Styles

- [get-tidy-colors\(\)](#)

### 3.2.1. get-tidy-colors

A style that can be used to generate documentation using [Tidy](#) for the Catppuccino theme. The returned dictionary is a tidy styles dictionary with some additional keys, most importantly `ctp-palette` whose value is the associated flavor.

#### Parameters

`get-tidy-colors(theme: string) -> dictionary`

**theme** string

The name of the theme to use.

Default: `themes.mocha`

## 3.3. Version

### Variables:

- [version](#)

#### 3.3.1. version version

The package version of Catppuccin.

### Example:

This package's version is `#version`.

This package's version is 0.1.0.