Catppuccin for Typst

🪶 Soothing pastel theme for Typst

v1.0.0 May 15, 2025

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

1.1. About

This document provides a detailed documentation of the **catppuccin** package for Typst. Inspired by the 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 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 LaTeX, and I incorporated it into my custom preable 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!

1.2. Basic Usage

Using this package is simple. The following is an example of how to use the package.

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

#show: catppuccin.with(flavor: flavors.mocha)

// The rest of your document
```

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

2. Modules

2.1. Catppuccin

2.1.1. catppuccin

Configure your document to use a Catppuccin flavor.

Example

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

#show: catppuccin.with(flavors.mocha)
```

This should be used at the top of your document.

Parameters

```
catppuccin(
  flavor: string flavor,
  code-syntax: bool,
  body
) -> content
```

flavor `string` or `flavor`

The flavor to set

code-syntax `bool`

Whether or not to use Catppuccin's theme for code syntaxing

Default: `true`

2.1.2. config-code-blocks

Configures the appearance of code syntax to match the Catppuccin theme.

Parameters

```
config-code-blocks(
  flavor: string flavor,
  body
) -> content
```

flavor `string` or `flavor`

The flavor to set

3. Flavor Schema

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`.

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

- **name** `string` — The name of the flavor (e.g. Frappé)
- **identifier** `string` — The identifier of the flavor (e.g. frappe)
- **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.

3.1. Flavors

Variables

3.1.1. get-flavor

Get the palette for the given flavor.

Example

```
#let items = flavors.values().map(flavor => [
  #let rainbow = (
    "red", "yellow", "green",
    "blue", "mauve",
  ).map(c => flavor.colors.at(c).rgb)

  #let fills = (
    gradient.linear(..rainbow),
    gradient.radial(..rainbow),
    gradient.conic(..rainbow),
  )

  #stack(
```

```
    dir: ttb,
    spacing: 4pt,
    text(flavor.name + ":"),
    stack(
      dir: ltr,
      spacing: 3mm,
      ..fills.map(fill => square(fill: fill))
    )
  )
])

#grid(columns: 1, gutter: 1em, ..items)
```

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. get-or-validate-flavor

Get the flavor for the given flavor name or validate the given flavor. This function is provided as a helper for anyone requiring dynamic resolution of a flavor.

Parameters

```
get-or-validate-flavor(flavor: string dictionary flavor) -> flavor
```

flavor `string` or `dictionary` or `flavor`

The flavor name as a string to get the flavor for

3.1.3. validate-flavor

Validate that the given dictionary is a valid flavor.

Parameters

```
validate-flavor(flavor: dictionary flavor) -> flavor
```

flavor `dictionary` or `flavor`

The flavor to validate

3.1.4. color-names `dictionary`

The available color names for Catppuccin. Given simply by the dictionary

```
#let color-names = (
   rosewater: "Rosewater",
   flamingo: "Flamingo",
   pink: "Pink",
   // ...
 )
```

3.1.5. flavors `dictionary`

The available flavors for Catppuccin. Given simply by the dictionary

```
 #let flavors = (
   latte: { ... },
   frappe: { ... },
   macchiato: { ... },
   mocha: { ... },
 )
```

Variables

- frappe
- latte
- macchiato
- mocha

3.1.6. frappe `flavor`

The Frappé flavor and palette.

Example

```
#let flavor = flavors.frappe
Selected flavor: #flavor.name #flavor.emoji
```

```
Selected flavor: Frappé 🪴
```

3.1.7. latte `flavor`

Example

```
#let flavor = flavors.latte
Selected flavor: #flavor.name #flavor.emoji
```

```
Selected flavor: Latte 🌻
```

3.1.8. macchiato `flavor`

Example

```
#let flavor = flavors.macchiato
Selected flavor: #flavor.name #flavor.emoji
```

```
Selected flavor: Macchiato 🌺
```

3.1.9. mocha `flavor`

Example

```
#let flavor = flavors.mocha
Selected flavor: #flavor.name #flavor.emoji
```

Selected flavor: Mocha 🌿

4. Miscellaneous

4.1. Version

Variables

- version

4.1.1. version `version`

The package version of Catppuccin.

Example:

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

This package's version is 1.0.0.