



# Minimal Manuals

## Manual

Modern but sober manuals inspired by the manpages of old.

---

typst

min-manual\*

0.2.0

MIT

Maycon F. Melo<sup>†</sup>

## Contents

Quick Start .....	2
Description .....	2
Options .....	2
Command Arguments .....	4
Command Extract .....	4
Command URL .....	5
Commands for Package URLs .....	5
Comment Documentation .....	6
Markdown Documentation .....	7
Copyright .....	8

---

\*[typst.app/universe/package/min-manual/](https://typst.app/universe/package/min-manual/)

<sup>†</sup><https://github.com/mayconfmelo>

# Quick Start

```
#import "@preview/min-manual:0.2.0": manual
#show: manual.with(
  title: "Package Name",
  description: "Short description, no longer than two lines.",
  package: "pkg-name:0.4.2",
  authors: "Author <mailto:author@email.com>",
  license: "MIT",
  logo: image("assets/logo.png")
)
```

## Description

Generate modern manuals without losing the simplicity of old manpages. This package draws inspiration from old manuals while adopting the facilities of modern tools, like Markdown and documentation embedded in comments. The design aims to be sober: a minimal informative header, technical text in comfortable fonts and well-formatted code examples.

The package was created with Typst in mind, but also targeting the potential to universally document code from other languages: all *min-book* features are supported when documenting any type of program or code.

## Options

```
#show: manual.with(
  title: none,
  description: none,
  by: none,
  package: none,
  authors: none,
  license: none,
  url: none,
  logo: none,
  use-defaults: false,
  from-comments: none,
  from-markdown: none,
  comment-delim: auto,
  body,
)
```

<b>title:</b>	string	content	(required)
Descriptive name of the package (what is being documented).			
<b>description:</b>	string	content	
Short package description, generally two lines long or less.			
<b>by:</b>	string	content	
Manual author (fallback to <code>authors.at(0)</code> if not set).			
<b>package:</b>	string		(required)
<p>"pkg:type/namespace/name@version"</p> <p>Package identification,<sup>1</sup> where <code>pkg:type/namespace/</code> is optional (fallback to <code>pkg:typst/</code>) and <code>name@version</code> can also be written <code>name:version</code>.</p>			
<b>authors:</b>	string	array of strings	(required)
<p>"name &lt;url&gt;"</p> <p>Package author or authors — the <code>&lt;url&gt;</code> are optional.</p>			
<b>license:</b>	string	content	(required)
Package license.			
<b>url:</b>	string	content	
Package URL.			
<b>logo:</b>	image	content	
Manual logo.			
<b>use-defaults:</b>	boolean		
Use Typst defaults instead of min-manual defaults.			
<b>from-comments:</b>	string	read	
Retrieve documentation from comments in source files.			
<b>from-markdown:</b>	string	read	
Retrieve documentation from markdown files (experimental).			
<b>comment-delim:</b>	array of strings		
<p>("/// ", "/* ", "*/ ")</p> <p>Set documentation comment delimiters.</p>			

---

<sup>1</sup>Inspired by <https://github.com/package-url/purl-spec/>

# Command Arguments

```
#arg(
  title,
  body
)
```

Defines and explains possible arguments/parameters (see `/tests/commands/arg/`).

**title** `string` *(required)*

"name <- type | type -> type <required>"

Title data: A mandatory name identifier, followed by optional ASCII arrows indicating input/output types, and a final <required> to define required arguments.

# Command Extract

```
#extract(
  name,
  from: auto,
  rule: none,
  lang: "typ",
  model: auto,
  display: none,
)
```

Extract code from another file or location (see `/tests/commands/extract/`).

**name** `string`

Name of the code structure to retrieve (the last match is used).

**from:** `string` `read` *(required)*

File from where the code will be retrieved (required in the first use).

**rule:** `string` `none`

Render Typst code in different ways.

**lang:** `string`

Programming language of the code.

**model:** `string`

Custom regex pattern to retrieve code.

**display:** `string`

Custom way to render retrieved code. Replaces `<name>` and `<capt>` markers by the name and retrieved code, respectively.

When extracting Typst code, the `#extract(rule)` supply almost all use cases of a Typst code; otherwise, the `#extract(model, display)` options can be used to achieve any other desired result.

## Command URL

`#url(url, id, text)`

Creates a paper-friendly link, attached to a footnote containing the URL itself for readability when printed.

**url** `string` **label** *(required)*

URL set to link and shown in footnote.

**id** `label`

Label set to the footnote for future reference.

**text** `string` **content**

Text to be shown in-place as the link itself.

## Commands for Package URLs

`#pkg(url)`

`#univ(name)`

`#pip(name)`

`#crate(name)`

`#gh(slug)`

Generates paper-friendly links to packages from different sources/platforms using only essential data like its name (see `/tests/commands/links/`).

**url** `string`

Package URL (used by `#pkg`). The package name is extracted if enclosed in `{}` or fallback to the last `/slug` of the URL.

**name** `string`

Package name as it is in the source repository/platform (used by `#pip`, `#univ`, and `#crate`).

**slug** string

A user/name path, as it appears in GitHub repository paths (used by #gh).

## Comment Documentation

```
/// = Feature
/// The `#feature` command does something.
#feature(title)
```

The documentation can be embedded into the source code itself through special comments, sometimes called *doc-comments*. These comments contains Typst code retrieved by *min-manual* to generate a complete manual, while at the same time they are useful as in-code documentation.

By default, documentation comments are an extension of Typst comments, both one-line and block comments:

Normal	Documentation
//	///
/* */	/** **/

Custom comment delimiters can be set in manual initialization with an array of strings containing the one-line and opening/closing block comments used:

```
#manual(
  comment-delim: ("///", "/*", "**/")
)
```

In addition to Typst code, documentation comments also support all *min-book* features both as commands and through special syntax:

```
// #extract (from documentation files itself)
:rule name: lang "model" => display

// #arg (optional arrows)
name <- type | type -> type <required>
body |
```

The *min-manual* itself is documented through comments in source code, check it out to see how it looks like in practice.

# Markdown Documentation

## # Feature

The ``#feature`` command does something.

```
```typst
#feature(title)
```
```

The documentation can be written in Markdown and *min-manual* will manage to generate a manual from it. The conversion between Markdown and Typst code is done using the *cmarker*<sup>2</sup> package, with some little tweaks — therefore some *cmarker* features are available.

In addition to CommonMark Markdown, *min-manual* also retrieves a special HTML structure as `#arg` commands:

```
<dl id="arg">
  <dt>
    <code class="typ">name</code>
    &larr; types &rarr; types [required]
  </dt>
  <dd>body<dd>
</dl>
```

This syntax is perfectly rendered in Markdown, and in *min-manual* it is equivalent to:

```
#arg("`typ name`" <- types -> types <required>)[body]
```

From *cmarker* is inherited the support for Markdown-only and Typst-only code through special comments:

```
<!--raw-typst [This code appears only in Typst]-->

<!--typst-begin-exclude-->
This code appears only in Markdown
<!--typst-end-exclude-->
```

As it is a recent implementation, the Markdown documentation is still experimental and may or may not present errors, bugs, or unexpected behaviors. Used it with caution for now.

<sup>2</sup><https://typst.app/universe/package/cmarker>

# Copyright

Copyright © 2025 Maycon F. Melo.

This manual is licensed under MIT.

The manual source code is free software: you are free to change and redistribute it.

There is NO WARRANTY, to the extent permitted by law.

The logo was obtained from Flaticon website.