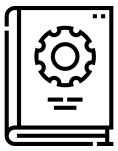

Minimal Manuals

Manual

Simple and sober manuals inspired by the OG Linux manpages.

---

typst	min-manual*	0.1.2	MIT

Maycon F. Melo†

Contents

*typst.app/universe/package/min-manual/
†https://github.com/mayconfmelo

Quick Start

```
#import "@preview/min-manual:0.1.1": manual
 title: "Package Name",
 description: "Short description, no longer than two lines.",
 authors: "Author <mailto:author@email.com>",
 cmd: "pkg-name",
 version: "0.4.2",
 license: "MIT",
 logo: image("assets/logo.png")
)
```

Description

Generate modern manuals, without loosing the simplicity and looks of old manuals. This package draws inspiration from the Linux manpages, as they look in terminal emulators until today, and adapts it to the contemporary formatting possibilities.

The package is designed to universally document any type of program or code, including Typst packages and templates. It allows to create documentation separated in dedicated files or extract it from the source code itself through doc-comments.

This manual will be updated only when new versions break or modify something; otherwise, it will be valid to all newer versions starting by the one documented here.

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,
)
```

`title:` string content (*required*)

Descriptive name of the package (what is being documented).

`description:` string content

Short package description, generally two lines long or less.

`by:` string content

Manual author (fallback to `authors.at(0)` if not set).

`package:` string (*required*)

`"pkg:type/namespace/name@version"`

Package identification,[1] where `pkg:type/namespace/` is optional (fallback to `pkg:typst/`) and `name@version` can also be written `name:version`.

`authors:` string array of strings (*required*)

Package author or authors.

`license:` string content

Package license.

`url:` string content

Package URL.

`logo:` image content

Manual logo.

`use-defaults:` boolean

Use Typst defaults instead of min-manual defaults.

`from-comments:` string read

Retrieve documentation from comments in file.

`from-markdown:` string read

Retrieve documentation from markdown file (not implemented yet).

`comment-delim:` array of strings

Set comment delimiters.

[1] Inspired by *https://github.com/package-url/purl-spec/*

Command Arguments

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

Defines and explain 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.

`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, respectivelly.

The `#extract` command was created with Typst code in mind, but without excluding the support for other languages. That's why it has a handy `rule` option that gets the work done for Typst code, but in other languages requires to manually use `model` and maybe `display` options to retrieve and show code.

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 essencial 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.
#let feature() = { }
```

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-mannual* to generate a complete manual, while at the same time they are usefull as in-code documentation.

By default, documentation comments are a 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.

Copyright

The logo was obtained from Flaticon website.