just Command runner

justfile found upwards from cwd → `just <recipe>` → **Dependencies** run first, once each → `then` → **Recipe lines** each in its own shell → `cwd = justfile dir` → **Result** stops at first failure

CONCEPTS

just is a command runner, not a build system: recipes always run, no file timestamps are checked. Indent recipe bodies consistently with spaces or tabs.

Every recipe line runs in a fresh shell, so cd and shell variables do not carry to the next line. Use a shebang recipe or && for multi-step state.

RUNNING

`just`
Run the default (first) recipe

`just <recipe> [<args>...]`
Run one or more recipes in order

`just --list / just -l`
List recipes with doc comments

`just --dry-run / just -n`
Print recipe lines without running them

`just --show <recipe> / just -s <recipe>`
Print a recipe's source

RUNNING (cont.)

`just --init`
Write a starter justfile in the project root

`just --choose`
Pick recipes with a chooser (fzf by default)

`just <var>=<value> <recipe>`
Override a variable for this run

`just --justfile <file> / just -f <file>`
Use a specific justfile

RECIPE SYNTAX

A recipe

```
# doc comment shown by --
list
build:
    cargo build --release
```

`<name>: <dep> <dep>`
Run dependencies before the recipe

`<name>: (<dep> "<arg>")`
Pass an argument to a dependency

`@<name>:`
Don't echo this recipe's lines

`@<command>`
Don't echo one line

RECIPE SYNTAX (cont.)

`-<command>`
Keep going if this line fails

`{{<var>}}`
Insert a variable or expression into a line

`<name> <param>:`
Required parameter

`<name> <param>=<default>:`
Parameter with a default

`<name> +<files>:`
Variadic: one or more values

`<name> *<flags>:`
Variadic: zero or more values

List recipes when run bare

```
default:
    @just --list
```

VARIABLES

`<name> := "<value>"`
Define a variable

`export <NAME> := "<value>"`
Export to recipe environments

`<name> := \`<command>\``
Capture a command's output

VARIABLES (cont.)

`if <a> == <b> { <x> } else { <y> }`
Conditional expression

`set dotenv-load`
Load .env into the environment

`env("<NAME>", "<default>")`
Env var with fallback

`justfile_directory()`
Directory of the justfile

`invocation_directory()`
Directory just was started from

SETTINGS

`set shell := ["bash", "-euo", "pipefail", "-c"]`
Choose the shell for lines and backticks

ATTRIBUTES

`[private]`
Hide from --list (also names starting with _)

`[confirm]`
Ask before running

`[group("<name>")]`
Group recipes in --list

`[no-cd]`
Run in the caller's directory

ATTRIBUTES (cont.)

`[linux] / [macos] / [windows] / [unix]`
Define the recipe only on that OS

SCRIPTS & MODULES

Shebang recipe

```
stats:
    #!/usr/bin/env python3
    import sys
    print(sys.version)
```

Script recipe

```
[script("python3")]
stats:
    import sys
    print(sys.version)
```

`import "<file>"`
Include another justfile's recipes

`mod <name>`
Load <name>.just or <name>/justfile as a module

`just <mod> <recipe> / just <mod>::<recipe>`
Run a module's recipe

RECIPES

Guard a dangerous recipe

```
[confirm("Deploy to
production?")]
deploy env="staging": test
    ./deploy.sh {{env}}
```