

### Simple justfile

```
#!/usr/bin/env just --justfile
# hello is recipe's name
hello:
    echo " Hello World! "
```

**Attention:** don't use keywords as recipe name, such as "import", "export", "alias" etc.

### default Recipe

```
default: lint build test
# default recipe to display help
information
default:
    @just --list
# if no default recipe, first
recipe will be default
```

### Aliases

```
alias t := test
alias c := check
```

### Settings

```
set shell := ["zsh", "-cu"]
#set shell := ["bu n", " exe c"]
set dotenv -required
set dotenv -load := true
serv:
    echo " $DA TAB ASE _AD -
DRESS from .env"
set positiona l-arguments :=
true
foo:
    echo $0
    echo $1
```

### Strings - escape with Double-quoted

```
string-with-tab := "\t"
string -with- newline := " \n"
escapes := '\t\n \r"\`\'
shell- exp and ed-path :=
x'~/ $F OO/ ${BAR}'
# this string will evaluate to
`foo\n bar\n`
x := '''
    foo
    bar
'''
```

### just command line

```
# run recipe
$ just hello param1
# list recipes in alphabetical order
$ just --list
$ just --summary
# Show full information about recipe
just --show test
# select recipes to run interactively
$ just --choose
# shell completion
just --completions zsh
```

### GitHub Actions

```
- uses: extractions/setup-
just@v1
with:
    jus t-v ersion: 1.36.0
```

### IDE integration

VS Code: <https://marketplace.visualstudio.com/items?itemName=skellock.just>

JetBrains: <https://plugins.jetbrains.com/plugin/18658-just>

### Just module/import

```
# load bar/justfile,
bar/.justfile, bar.just
mod bar
# include the contents of
another justfile
import 'foo/b ar.j ust'
```

```
just --unstable bar::hello
```

### Recipe with parameters

```
filter PATTERN:
    echo {{PATTERN}}
# param with default value
email address=' mas ter @ex -
amp le.c om':
    echo {{address}}
# param with expression
test triple=(a rch() + " -un -
kno wn- unk now n"):
    ./test {{triple}}
# variadic param: '+' accept one
or more values
[doc(' Backup files')]
backup +FILES:
    scp {{FILES}}me@exa mpl -
e.com
# variadic param with *: zero or
more values
commit MESSAGE *FLAGS:
    git commit {{FLAGS}} -m "
{{MESSAGE}}"
```



### Recipe with env variable for command

```
# recipe param as env variable
with $ sign
hello $name:
    echo $name
```

### Recipe Dependencies - Before, After & Around

```
# execution sequence: a -> b ->
c -> d
b: a && c d
# execute recipe 'a' around
b:
    echo 'B start!'
    just a
    echo 'B end!'
# depend with params by
expression
default: (build "main")
build target:
    @echo 'Building
{{target}}...'
```

### Command annotate: quiet(@), suppress(-), invert(!)

```
hello:
    @ echo " command will not be
echoed "
    - echo " ignore none-zero
exit status and continue"
@hello2:
    echo " command will not be
echoed "
# Invert command exit status by
! - shell feature
hello3:
    # if command succeeds(exit
status is 0), exit just
    ! git branch | grep '*
master'
```

### Recipe with other Languages

```
bash-test:
    #!/usr/bin/env bash
    set -euxo pipefail
    hello='Yo'
    echo "$hello from bash!"
[script("bash")]
bash-test2:
    set -euxo pipefail
    hello='Yo'
    echo "$hello from bash!"
```

### Private Recipes - name starts with \_

```
test: _test-helper
    ./bin/test
# omitted from 'just --list'
_test-helper:
    ./bin/super-secret-test-helper-stuff
```

### Recipes as shell alias

```
for recipe in `just -f
~/justfile --summary`; do
    alias $recipe="just -f
~/justfile -d. $recipe"
done
```

### Recipe with Python venv

```
venv:
    [-d .venv] || uv venv
run: venv
    ./venv/bin/python3
main.py
```

### Variable and Substitution

```
version := "0.2.7"
tardir := "awesome" +
version
tarball := tardir + ".tar.gz"
test:
    echo {{version}}
# set/override variables from
just command line
$ just --set version 1.1.0
```

### Environment variable for commands

```
export RUST_BACKTRACE := "1"
test:
    # will print a stack
trace if it crashes
    cargo test
```

### backtick - capture output from evaluation

```
JAVA_HOME := `jbang jdk home 11`
# backtick code block
stuff := ``
    foo="hello"
    echo $foo "world"
``
done BRANCH=`git rev-parse --
abbrev-ref HEAD`
git checkout master
sloc:
    @echo "`wc -l *.c` lines
of code"
# backtick works anywhere:
string /variable /params
```

### Just functions

```
hello name:
    echo {{os()}}
    echo {{uppercase(name)}}

# function categories
* System Information
* Environment Variables
* Justfile and Justfile
Directory
* String Manipulation
* Path Manipulation
# String contact: (key + " :" +
value)
```

### Attention (cont)

```
> if true; then \
    echo 'True!'; \
fi

# justfile is case insensitive: Justfile,
JUSTFILE etc
# justfile could be hidden: '.justfile'
# Call recipe from sub dir: `~/app1/target>$
just build`
```

### Conditional expressions: if, loop and while

```
# regular expression match
fo := if " hi" =~ 'h.+' { " -
mat ch" } else { " mis mat ch" }
test:
    if true; then echo 'True!';
fi
    for file in `ls .`; do echo
$file; done
    while `serve r-i s-d ead`;
do ping -c 1 server; done
foo bar:
    echo {{ if bar == " bar " {
" hel lo" } else { " bye " } }}
```

### Attention

```
# Each command line is executed
by a new shell.
# If a command line failed, just
will exit, \
# and subsequent command lines
will not be executed.
change -wo rki ng-dir:
    cd bar && pwd
    # multi-line construct -
escape newline with slash
```

