Justfile Cheat Sheet

by linux_china via cheatography.com/141366/cs/30282/

Simple justfile

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

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 dotenv-load := true
serv:
   echo "$DATABASE_ADDRESS from .env"
set positional-arguments := true
foo:
  echo $0
  echo $1
```

Strings - escape with Double-quoted

```
string-with-tab := "\t"
string-with-newline := "\n"
escapes := '\t\n\r\"\\'
# this string will evaluate to `foo\nbar\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:
    just-version: 0.10.5
```

IDE integration

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

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

Recipe with parameters

```
filter PATTERN:
   echo {{PATTERN}}
# param with default value
email address='master@example.com':
   echo {{address}}
# param with expression
test triple=(arch() + "-unknown-unknown"):
  ./test {{triple}}
# variadic param: '+' aceept one or more values
backup +FILES:
  scp {{FILES}} me@example.com
# variadic param with *: zero or more values
```

Recipe with parameters (cont)

```
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'
```

Cheatography

Justfile Cheat Sheet

by linux_china via cheatography.com/141366/cs/30282/

Recipe with other Languages

```
bash-test:
    #!/usr/bin/env bash
    set -euxo pipefail
    hello='Yo'
    echo "$hello from bash!"
```

Private Recipes - name starts with _

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

Recipes as shell alias

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

Variable and Substitution

```
version := "0.2.7"
tardir := "awesomesauce-" +
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)
```

Conditional expressions: if, loop and while

```
# regular expression match
fo := if "hi" =~ 'h.+' { "match"
} else { "mismatch" }
test:
   if true; then echo 'True!';
fi
   for file in `ls .`; do echo
$file; done
   while `server-is-dead`; do
ping -c 1 server; done
foo bar:
  echo {{ if bar == "bar" { "-
hello" } 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-working-dir:
   cd bar && pwd
   # multi-line construct -
escape newline with slash
   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`
```
```