Bashmatic Usage Docs (v3.2.2)

Table of Contents

NOTICE: shdoc documentation is auto-extracted from the Bashmatic Sources.

File lib/yarn.sh

- yarn_sha()

yarn_install()

Installs YARN via npm if not found; then runs yarn install Note that yarn install is skipped if package.json and yarn.lock haven't changed since the last run of yarn install.

yarn_sha()

Prints to STDOUT the SHA based on package.json and yarn.lock

File lib/dropbox.sh

- function dropbox.ignore {
- dropbox.unignore()

function dropbox.ignore {

Set file to be ignored by Dropbox

See also

- https://help.dropbox.com/files-folders/restore-delete/ignored-files

dropbox.unignore()

Set a file or directorhy to be ignored by Dropbox

See also

- https://help.dropbox.com/files-folders/restore-delete/ignored-files

---

File lib/bashit.sh

- bashit-prompt-terraform()
- bashit-install()

bashit-prompt-terraform()

Possible Bash It Powerline Prompt Modules

aws_profile battery clock command_number cwd dirstack gcloud go history_number hostname in_toolbox in_vim k8s_context last_status node python_venv ruby scm shlvl terraform user_info wd

bashit-install()

Installs Bash-It Framework

File lib/asciidoc.sh

Provides helper functions for dealing with asciidoc format.

- asciidoc.rouge-themes()

asciidoc.rouge-themes()

Installs gem "rouge" and prints all available themes

File lib/output-utils.sh

- is-dbg()
- dbg()

is-dbg()

Checks if we have debug mode enabled

dbg()

Local debugging helper, activate it with `export BASHMATIC_DEBUG=1`

File lib/brew.sh

- package.is-installed()

package.is-installed()

For each passed argument checks if it's installed.

File lib/output.sh

- output.screen-width.actual()
- output.screen-height.actual()

- section()

output.screen-width.actual()

OS-independent way to determine screen width.

output.screen-height.actual()

OS-independent way to determine screen height.

section()

Prints a "arrow-like" line using powerline characters

Arguments

- @arg1 Width (optional) — only intepretered as width if the first argument is a number.
- @args Text to print

File lib/file-helpers.sh

- .file.make_executable()

.file.make_executable()

Makes a file executable but only if it already contains a "bang" line at the top.

File lib/path.sh

Utilities for managing the $PATH variable

- path.strip-slash()
- path.dirs()
- path.dirs.size()
- path.dirs.uniq()
- path.dirs.delete()
- path.uniq()
- PATH.uniqify()
- path.append()
- path.prepend()

- path.mutate.uniq()
- path.mutate.delete()
- path.mutate.append()
- path.mutate.prepend()
- path.absolute()

path.strip-slash()

Removes a trailing slash from an argument path

path.dirs()

Prints a new-line separated list of paths in PATH

Arguments

- @arg1 A path to split, defaults to $PATH

path.dirs.size()

Prints the tatal number of paths in the path argument, which defaults to $PATH

path.dirs.uniq()

Prints all folders in $PATH, one per line, removing any duplicates, Does not mutate the $PATH

path.dirs.delete()

Deletes any number of folders from the PATH passed as the first string argument (defaults to $PATH). Does not mutate the $PATH, just prints the result to STDOUT

Arguments

- @arg1 String representation of a PATH, eg "/bin:/usr/bin:/usr/local/bin"
- @arg2 An array of paths to be removed from the PATH

path.uniq()

Removes duplicates from the $PATH (or argument) and prints the results in the PATH format (column-joined). DOES NOT mutate the actual $PATH

PATH.uniqify()

Using sed and tr uniq the PATH without re-sorting it.

path.append()

Appends a new directory to the $PATH and prints the result to STDOUT, Does NOT mutate the actual $PATH

path.prepend()

Prepends a new directory to the $PATH and prints to STDOUT, If one of the arguments already in the PATH its moved to the front. DOES NOT mutate the actual $PATH

path.mutate.uniq()

Removes any duplicates from $PATH and exports it.

path.mutate.delete()

Deletes paths from the PATH provided on the command line

path.mutate.append()

Appends valid directories to those in the PATH, and exports the new value of the PATH

path.mutate.prepend()

Prepends valid directories to those in the PATH, and exports the new value of the PATH

path.absolute()

Returns an absolute version of a given path

File lib/bashmatic.sh

bashmatic.is-developer()

True if .envrc.local file is present. We take it as a sign you may be developing bashmatic.

bashmatic.is-installed()

This function returns 1 if bashmatic is installed in the location pointed to by ${BASHMATIC_HOME} or the first argument.

Arguments

- $1 The location to check for bashmatic instead of ${BASHMATIC_HOME}

File `lib/shdoc.sh`

lib/shdoc.sh

Helpers to install gawk and shdoc properly.0

see `${BASHMATIC_HOME}/lib/shdoc.md` for an example of how to use SHDOC. and also project's github page.

`gawk.install()`

Installs gawk into /usr/local/bin/gawk

File `lib/memory.sh`

`memory.size-to-bytes()`

Pass in a value eg. 32GB or 16M and it returns back the number of bytes

`memory.bytes-to-units()`

This function receives up to three arguments:

Arguments

- @arg1 A number of bytes to convert into a more human-friendly format
- @arg2 An optional printf format string, defaults to '%.1f'
- @arg3 An optional suffix ('b' or "B" or none at all)

File `lib/color.sh`

- color.current-background()

`color.current-background()`

Prints the background color of the terminal, assuming terminal responds to the escape sequence. More info: https://stackoverflow.com/questions/2507337/how-to-determine-a-terminals-background-color

---

File `lib/pg.sh`

- pg.is-running()
- pg.running.server-binaries()
- pg.running.data-dirs()
- pg.server-in-path.version()

`pg.is-running()`

Returns true if PostgreSQL is running locally

`pg.running.server-binaries()`

if one or more PostgreSQL instances is running locally, prints each server's binary postgres file path

`pg.running.data-dirs()`

For each running server prints the data directory

`pg.server-in-path.version()`

Grab the version from `postgres` binary in the PATH and remove fractional sub-version

---

File `lib/7z.sh`

lib/7z.sh

p7zip conversions routines.

File `lib/dir.sh`

This file contains many useful functions for handling directories, sync'ing and copying from and to directories, and so on.

`dir.with-file()`

Returns the first folder above the given that contains a file.

Arguments

- @arg1 file without the path to search for, eg ".evnrc"
- @arg2 Starting file path to seartch

`dir.short-home()`

Replaces the first part of the directory that matches ${HOME} with '~/'

`dir.rsync-to()`

Rsyncs the files from a "from" directory specified by the first argument, to the to directory specified by the second.

Arguments

- @arg1 The source locl directory
- @arg2 The destination locl directory
- @arg3 optional `--sudo`: runs rsync in sudo mode. Careful!
- @arg4 Any additional arguments to rsync such as --verbose

`dir.rsync-from-mac()`

This is a variation on the above that preserves extended attributes of the source files, such as icons for direcories. When copying a folder from the Mac OS-X this is recommended.

`dir.rsync-from-mac-verbose()`

This is a variation on the above that preserves extended attributes of the source files, such as icons for direcories, and add --verbose to rsync flags so that you can the files being synced.

File `lib/config.sh`

`config.get-format()`

Get current format

`config.set-file()`

Set the default config file

`config.get-file()`

Get the file name

`config.dig()`

Reads the value from a two-level configuration hash

Arguments

- @arg1 hash key
- @arg2 hash sub-key

`config.dig.pretty()`

Uses `jq` utility to format JSON with color, supports partial

File `lib/nvm.sh`

nvm.is-valid-dir()

Returns true if NVM_DIR is correctly set, OR if a directory passed as an argument contains nvm.sh

nvm.detect()

Returns success and exports NVM_DIR whenver nvm.sh is found underneath any of the possible locations tried.

nvm.install()

Installs NVM via Curl if not already installed.

nvm.load()

Loadd

File lib/net.sh

net.is-host-port-protocol-open()

Uses pingless connection to check if a remote port is open Requires sudo for UDP

Arguments

- @arg1 host
- @arg2 port
- @arg3 [optional] protocol (defaults to "tcp", supports also "udp")

File lib/files-normalize.sh

files.normalize-tree()

Renames files matching the input parameters to `find` by replacing spaces with dashes and lower casing the file.

File `lib/util.sh`

Miscellaneous utilities.

`system.uname()`

Finds the exact absolute path of the `uname` utility on a unix file system.

`util.random-number()`

Generates a random number up to 1000000

`util.generate-password()`

Generates a password of a given length

`util.random-string.of-length()`

Generates a random string of a given length

File `lib/runtime.sh`

`run.print-variables()`

Adds a variable to the list of the variables to be obfuscated

`run.inspect-vars()`

Prints values of all variables starting with prefixes in args

---

File `bin/jemalloc-check`

- js.jemalloc.detect-or-exit()
- jm.jemalloc.stats()
- jm.jemalloc.detect-quiet()
- jm.jemalloc.detect-loud()
- usage()

jm.ruby.report()

prints the info about current version of ruby

jm.ruby.describe()

Prints ruby version under test

js.jemalloc.detect-or-exit()

detects jemalloc or exits

jm.jemalloc.stats()

prints jemalloc statistics if jemalloc is available

jm.jemalloc.detect-quiet()

returns 0 if jemalloc was detected or 1 otherwise

jm.jemalloc.detect-loud()

detects if jemalloc is linked and if so prints the info to output

usage()

Prints the help screen and exits

File bin/install-direnv

Add direnv hook to shell RC files

- direnv.register()

direnv.register()

Add direnv hook to shell RC files

File `bin/regen-usage-docs`

Regenerates USAGE.adoc && USAGE.pdf

File `bin/pdf-reduce`

- pdf.do.shrink()

`pdf.do.shrink()`

shrinkgs PDF

File `bin/scheck`

- manual-install()

`manual-install()`

Manually Download and Install ShellCheck

Copyright & License

- Copyright © 2017-2024 Konstantin Gredeskoul, All rights reserved.
- Distributed under the MIT License.