

# Bashmatic Usage Docs (v19.4)

## Table of Contents

File <code>lib/array.sh</code> .....	4
<code>array.has-element()</code> .....	4
Example .....	4
<code>array.includes()</code> .....	4
<code>array.join()</code> .....	4
Example .....	4
<code>array.sort()</code> .....	4
Example .....	5
<code>array.sort-numeric()</code> .....	5
Example .....	5
<code>array.min()</code> .....	5
Example .....	5
<code>array.max()</code> .....	5
Example .....	5
<code>array.uniq()</code> .....	5
Example .....	5
File <code>lib/asciidoc.sh</code> .....	6
<code>asciidoc.rouge-themes()</code> .....	6
File <code>lib/output-utils.sh</code> .....	6
<code>is-dbg()</code> .....	6
<code>dbg()</code> .....	6
File <code>lib/brew.sh</code> .....	6
<code>package.is-installed()</code> .....	6
File <code>lib/output.sh</code> .....	7
<code>section()</code> .....	7
Arguments .....	7
File <code>lib/video.sh</code> .....	7
is.sh .....	7
File <code>lib/path.sh</code> .....	7
<code>path.strip-slash()</code> .....	8
<code>path.dirs()</code> .....	8
Arguments .....	8
<code>path.dirs.size()</code> .....	8
<code>path.dirs.uniq()</code> .....	8
<code>path.uniq()</code> .....	8
<code>path.append()</code> .....	8
<code>path.prepend()</code> .....	8

path.mutate.uniq()	8
path.mutate.append()	8
path.mutate.prepend()	8
PATH_add()	9
File lib/osx.sh	9
osx.app.is-installed()	9
Example	9
Arguments	9
Exit codes	9
File lib/db.sh	9
db.config.parse()	10
Example	10
db.psql.connect()	10
Example	10
db.psql.connect.just-data()	10
Example	10
db.psql.connect.table-settings-set()	10
Example	10
db.psql.db-settings()	11
Example	11
db.psql.connect.db-settings-pretty()	11
Example	11
Arguments	11
db.psql.connect.db-settings-toml()	11
Example	11
Arguments	11
File lib/shdoc.sh	11
lib/shdoc.sh	11
File lib/git.sh	12
git.cfgu()	12
Example	12
git.open()	12
Example	12
Arguments	12
File lib/package.sh	12
package.ensure.is-installed()	13
package.ensure.command-available()	13
Example	13
File lib/shasum.sh	13
shasum.set-command()	13
shasum.set-algo()	13
Example	13
shasum.sha()	14

shasum.sha-only()	14
shasum.sha-only-stdin()	14
shasum.to-hash()	14
Example	14
shasum.all-files()	14
Example	14
shasum.all-files-in-dir()	14
Example	14
File lib/pg.sh	15
pg.is-running()	15
pg.running.server-binaries()	15
pg.running.data-dirs()	15
pg.server-in-path.version()	15
File lib/dir.sh	15
dir.short-home()	15
File lib/is.sh	15
__is.validation.error()	16
Arguments	16
Exit codes	16
is-validations()	16
__is.validation.ignore-error()	16
__is.validation.report-error()	16
whenever()	16
Example	16
File lib/util.sh	16
util.rot13-stdin()	17
Example	17
File lib/pdf.sh	17
Bashmatic Utilities for PDF file handling	17
File bin/install-direnv	17
direnv.register()	17
File bin/regen-usage-docs	18
File bin/specs	18
specs.init()	18
specs.determine-test-filename()	18
File bin/pdf-reduce	18
pdf.do.shrink()	18
Copyright & License	18

NOTICE: [shdoc](#) documentation is auto-extracted from the Bashmatic Sources.

# File lib/array.sh

- `array.has-element()`
- `array.includes()`
- `array.join()`
- `array.sort()`
- `array.sort-numeric()`
- `array.min()`
- `array.max()`
- `array.uniq()`

## `array.has-element()`

Returns "true" if the first argument is a member of the array passed as the second argument:

### Example

```
$ declare -a array=("a string" test2000 moo)
if [[ $(array.has-element "a string" "${array[@]}") == "true" ]]; then
    ...
fi
```

## `array.includes()`

Similar to `array.has-elements`, but does not print anything, just returns 0 if includes, 1 if not.

## `array.join()`

Joins a given array with a custom character

### Example

```
$ declare -a array=(one two three)
$ array.join "," "${array[@]}"
one,two,three
```

## `array.sort()`

Sorts the array alphanumerically and prints it to STDOUT

## Example

```
declare -a unsorted=(hello begin again again)
local sorted="$(array.sort "${unsorted[@]}")"
```

## array.sort-numeric()

Sorts the array numerically and prints it to STDOUT

## Example

```
declare -a unsorted=(1 2 34 45 6)
local sorted="$(array.sort-numeric "${unsorted[@]}")"
```

## array.min()

Returns a minimum integer from an array. Non-numeric elements are ignored and skipped over. Negative numbers are supported, but non-integers are not.

## Example

```
$ declare -a array=(10 20 30 -5 5)
$ array.min "," "${array[@]}"
-5
```

## array.max()

Returns a maximum integer from an array. Non-numeric elements are ignored and skipped over. Negative numbers are supported, but non-integers are not.

## Example

```
$ declare -a array=(10 20 30 -5 5)
$ array.min "," "${array[@]}"
30
```

## array.uniq()

Sorts and uniqs the array and prints it to STDOUT

## Example

```
declare -a unsorted=(hello hello hello goodbye)
local uniqed="$(array.sort-numeric "${unsorted[@]}")"
```

---

## 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 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`

- `section()`

### `section()`

Prints a "arrow-like" line using powerline characters

#### Arguments

- `@arg1` Width (optional) ~~2-2~~ Only interpreted as width if the first argument is a number.
  - `@args` Text to print
- 

## File `lib/video.sh`

### `is.sh`

video conversions

---

## File `lib/path.sh`

Utilities for managing the `$PATH` variable

- `path.strip-slash()`
  - `path.dirs()`
  - `path.dirs.size()`
  - `path.dirs.uniq()`
  - `path.uniq()`
  - `path.append()`
  - `path.prepend()`
  - `path.mutate.uniq()`
  - `path.mutate.append()`
  - `path.mutate.prepend()`
  - `PATH_add()`
-

## 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 total 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.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.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.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\_add()

This function exists within direnv, but since we are sourcing in .envrc we need to have this defined to avoid errors.

---

## File lib/osx.sh

OSX Specific Helpers and Utilities

- [osx.app.is-installed\(\)](#)

## osx.app.is-installed()

Checks if a given parameter matches any of the installed applications under /Applications and ~/Applications

By the default prints the matched application. Pass **-q** as a second argument to disable output.

### Example

```
> osx.app.is-installed safari
Safari.app
> osx.app.is-installed safari -q && echo installed
installed
> osx.app.is-installed microsoft -c
6
```

### Arguments

- **\$1(a)**: string value to match (case insentively) for an app name
- **\$2..**: additional arguments to the last invocation of **grep**

### Exit codes

- **0**: if match was found
- **1**: if not

---

## File lib/db.sh

- [db.config.parse\(\)](#)
- [db.psql.connect\(\)](#)
- [db.psql.connect.just-data\(\)](#)
- [db.psql.connect.table-settings-set\(\)](#)

- `db.psql.db-settings()`
- `db.psql.connect.db-settings-pretty()`
- `db.psql.connect.db-settings-toml()`

## db.config.parse()

Returns a space-separated values of db host, db name, username and password

### Example

```
db.config.set-file ~/.db/database.yml
db.config.parse development
#=> hostname dbname dbuser dbpass
declare -a params=($(db.config.parse development))
echo ${params[0]} # host
```

## db.psql.connect()

Connect to one of the databases named in the YAML file, and optionally pass additional arguments to psql. Informational messages are sent to STDERR.

### Example

```
db.psql.connect production
db.psql.connect production -c 'show all'
```

## db.psql.connect.just-data()

Similar to the `db.psql.connect`, but outputs just the raw data with no headers.

### Example

```
db.psql.connect.just-data production -c 'select datname from pg_database;'
```

## db.psql.connect.table-settings-set()

Set per-table settings, such as `autovacuum`, eg:

### Example

```
db.psql.connect.table-settings-set prod users autovacuum_analyze_threshold 1000000
db.psql.connect.table-settings-set prod users autovacuum_analyze_scale_factor 0
```

## db.psql.db-settings()

Print out PostgreSQL settings for a connection specified by args

### Example

```
db.psql.db-settings -h localhost -U postgres appdb
```

## db.psql.connect.db-settings-pretty()

Print out PostgreSQL settings for a named connection

### Example

```
db.psql.connect.db-settings-pretty primary
```

### Arguments

- @arg1 dbname database entry name in ~/.db/database.yml

## db.psql.connect.db-settings-toml()

Print out PostgreSQL settings for a named connection using TOML/ini format.

### Example

```
db.psql.connect.db-settings-toml primary > primary.ini
```

### Arguments

- @arg1 dbname database entry name in ~/.db/database.yml

---

## 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()`

## `gawk.install()`

Installs gawk into `/usr/local/bin/gawk`

---

## File `lib/git.sh`

- `git.cfgu()`
- `git.open()`

## `git.cfgu()`

Sets or gets user values from global gitconfig.

### Example

```
git.cfgu email
git.cfgu email kigster@gmail.com
git.cfgu
```

## `git.open()`

Reads the remote of a repo by name provided as an argument (or defaults to "origin") and opens it in the browser.

### Example

```
git clone git@github.com:kigster/bashmatic.git
cd bashmatic
source init.sh
git.open
git.open origin # same thing
```

### Arguments

- `$1` (optional): name of the remote to open, defaults to "origin"
- 

## File `lib/package.sh`

- `package.ensure.is-installed()`

- `package.ensure.command-available()`

## `package.ensure.is-installed()`

fr

## `package.ensure.command-available()`

### Example

```
In this example we skip installation if 'gem' exists and in the PATH.  
Otherwise we install the package and retry, and return if not found
```

## File `lib/shasum.sh`

SHA Functions

SHASUM related functions, that compute SHA for a single file, collection of files, or entire directories.

- `shasum.set-command()`
- `shasum.set-algo()`
- `shasum.sha()`
- `shasum.sha-only()`
- `shasum.sha-only-stdin()`
- `shasum.to-hash()`
- `shasum.all-files()`
- `shasum.all-files-in-dir()`

## `shasum.set-command()`

Override the default SHA command and algorithm Default is `shasum -a 256`

## `shasum.set-algo()`

Override the default SHA algorithm

### Example

```
$ shasum.set-algo 256
```

## shasum.sha()

Compute SHA for all given files, ignore STDERR NOTE: first few arguments will be passed to the shasum command, or whatever you set via shasum.set-command.

## shasum.sha-only()

Print SHA ONLY removing the file components

## shasum.sha-only-stdin()

Print SHA ONLY removing the file components

## shasum.to-hash()

This function populates a pre-declare associative array with filenames mapped to their SHAs, but only in the current directory Call **dbg-on** to enable additional debugging info.

### Example

```
$ declare -A file_shas
$ shasum.to-hash file_shas $(find . -type f -maxdepth 2)
$ echo "Total of ${#file_shas[@]} files in the hash"
```

## shasum.all-files()

For a given array of files, sort them, take a SHA of each file, and return a single SHA finger-printing this set of files. # NOTE: the files are sorted prior to hashing, so the return SHA should ONLY change when files are either changed, or added/removed. Only computes SHA of the files provided, does not recurse into folders

### Example

```
$ shasum.all-files *.cpp
```

## shasum.all-files-in-dir()

For a given directory and an optional file pattern, use **find** to grab every single file (that matches optional pattern) and return a single SHA

### Example

```
$ shasum.all-files-in-dir . '*.pdf'
cc35aad389e61942c75e111f1eddb634d74b4b1
```

---

## 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/dir.sh

- `dir.short-home()`

### `dir.short-home()`

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

---

## File lib/is.sh

Various validations and asserts that can be chained and be explicit in a DSL-like way.

- `<<isvalidationerror,is.validation.error(>>`
  - `is-validations()`
  - `<<isvalidationignore-error,is.validation.ignore-error(>>`
  - `<<isvalidationreport-error,is.validation.report-error(>>`
-

- [whenever\(\)](#)

## `__is.validation.error()`

Invoke a validation on the value, and process the invalid case using a customizable error handler.

### Arguments

- @arg1 func Validation function name to invoke
- @arg2 var Value under the test
- @arg4 error\_func Error function to call when validation fails

### Exit codes

- 0: if validation passes

## `is-validations()`

Returns the list of validation functions available

## `__is.validation.ignore-error()`

Private function that ignores errors

## `__is.validation.report-error()`

Private function that ignores errors

## `whenever()`

a convenient DSL for validating things

### Example

```
whenever /var/log/postgresql.log is.an-empty-file && {  
    touch /var/log/postgresql.log  
}
```

---

## File `lib/util.sh`

Miscellaneous utilities.

- [util.rot13-stdin\(\)](#)



## util.rot13-stdin()

Convert STDIN using rot13

### Example

```
echo "test" | util.rot13-stdin
```

---

## File lib/pdf.sh

# Bashmatic Utilities for PDF file handling

Install and uses GhostScript to manipulate PDFs.

- [pdf.combine\(\)](#)

## pdf.combine()

Combine multiple PDFs into a single one using ghostscript.

### Example

```
pdf.combine ~/merged.pdf 'my-book-chapter*'
```

### Arguments

- \$1 (pathname): to the merged file
- ... (the): rest of the PDF files to combine

---

## 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/specs`

- `specs.init()`
- `specs.determine-test-filename()`

### `specs.init()`

Initialize specs

### `specs.determine-test-filename()`

Based on a shortname attempt to determine the actual test file names

---

## File `bin/pdf-reduce`

- `pdf.do.shrink()`

### `pdf.do.shrink()`

shrinks PDF

## Copyright & License

- Copyright © 2017-2021 Konstantin Gredeskoul, All rights reserved.
- Distributed under the MIT License.