Bashmatic Usage Docs (v1.8.0)

Table of Contents

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

---

1. `array.includes()`

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

2. `array.join()`

Joins a given array with a custom character

2.1. Example

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

3. `array.sort()`

Sorts the array alphanumerically and prints it to STDOUT

3.1. Example

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

4. `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.

4.1. Example

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

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.

5.1. Example

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

6. `array.uniq()`

Sorts and uniqs the array and prints it to STDOUT

6.1. Example

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

---

7. File `lib/output-utils.sh`

7.1. `dbg()`

Local debugging helper, activate it with DEBUG=1

---

8. File `lib/output.sh`

8.1. `section()`

Prints a "arrow-like" line using powerline characters

8.1.1. Arguments

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

---

9. File `lib/path.sh`

9.1. function path.add() {

Adds valid directories to those in the PATH and prints to the output. DOES NOT MODIFY $PATH

9.2. function path.append() {

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

9.3. `PATH_add()`

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

---

10. File `lib/osx.sh`

11. osx.sh

11.1. Overview

OSX Specific Helpers and Utilities

---

12. File `lib/db.sh`

12.1. `db.config.parse()`

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

12.1.1. 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
```

12.2. 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.

12.2.1. Example

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

---

13. File lib/shdoc.sh

14. lib/shdoc.sh

Helpers to install gawk and shdoc properly.0

14.1. Overview

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

- function gawk.install() {
- function shdoc.reinstall() {
- function shdoc.install() {

14.2. function gawk.install() {

Installs gawk into /usr/local/bin/gawk

14.3. function shdoc.reinstall() {

Reinstall shdoc completely

14.4. function shdoc.install() {

Installs shdoc unless already exists

---

15. File `lib/git.sh`

15.1. function git.cfgu() {

Sets or gets user values from global gitconfig.

15.1.1. Example

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

15.2. function git.open() {

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

15.2.1. Example

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

15.2.2. Arguments

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

---

16. File `lib/is.sh`

17. is.sh

17.1. Overview

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

- whenever()

17.2. function __is.validation.error() {

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

17.2.1. Arguments

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

17.2.2. Exit codes

- **0**: if validation passes

17.3. `whenever()`

a convenient DSL for validating things

17.3.1. Example

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

---

18. File `lib/util.sh`

19. util.sh

19.1. Overview

Miscellaneous utilities.

---

20. File `lib/pdf.sh`

21. Bashmatic Utilities for PDF file handling

21.1. Overview

Install and uses GhostScript to manipulate PDFs.

- function pdf.combine() {

21.2. function pdf.combine() {

Combine multiple PDFs into a single one using ghostscript.

21.2.1. Example

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

21.2.2. Arguments

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

---

22. File `bin/specs`

- function specs.init() {

22.1. function specs.init() {

Initialize specs

---

23. File `bin/pdf-reduce`

- function pdf.do.shrink() {

23.1. function pdf.do.shrink() {

shrinkgs PDF

\n\n---\n\n## Copyright & License\n\n * Copyright © 2017-2021 Konstantin Gredeskoul, All rights reserved.\n * Distributed under the MIT License.\n\n