# Bashmatic Usage Docs (v1.9.1)

# Table of Contents

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

# Chapter 1. File `lib/array.sh`

## 1.1. array.includes()

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

## 1.2. array.join()

Joins a given array with a custom character

### 1.2.1. Example

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

## 1.3. array.sort()

Sorts the array alphanumerically and prints it to STDOUT

### 1.3.1. Example

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

## 1.4. 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.

### 1.4.1. Example

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

## 1.5. 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.

### 1.5.1. Example

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

## 1.6. array.uniq()

Sorts and uniqs the array and prints it to STDOUT

### 1.6.1. Example

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

# Chapter 2. File `lib/asciidoc.sh`

# asciidoc

# Chapter 3. Overview

Provides helper functions for dealing with asciidoc format.

# Chapter 4. File `lib/output-utils.sh`

## 4.1. dbg()

Local debugging helper, activate it with DEBUG=1

# Chapter 5. File `lib/output.sh`

## 5.1. section()

Prints a "arrow-like" line using powerline characters

### 5.1.1. Arguments

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

# Chapter 6. File `lib/path.sh`

## 6.1. path.add()

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

## 6.2. path.append()

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

## 6.3. PATH_add()

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

# Chapter 7. File `lib/osx.sh`

## osx.sh

# Chapter 8. Overview

OSX Specific Helpers and Utilities

# Chapter 9. File `lib/db.sh`

## 9.1. db.config.parse()

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

### 9.1.1. 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
```

## 9.2. 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.

### 9.2.1. Example

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

# Chapter 10. File `lib/shdoc.sh`

# lib/shdoc.sh

Helpers to install gawk and shdoc properly.0

# Chapter 11. Overview

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

- gawk.install()

## 11.1. gawk.install()

Installs gawk into /usr/local/bin/gawk

# Chapter 12. File `lib/git.sh`

## 12.1. git.cfgu()

Sets or gets user values from global gitconfig.

### 12.1.1. Example

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

## 12.2. git.open()

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

### 12.2.1. Example

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

### 12.2.2. Arguments

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

# Chapter 13. File `lib/is.sh`

## is.sh

# Chapter 14. Overview

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

- __is.validation.error()
- whenever()

## 14.1. __is.validation.error()

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

### 14.1.1. Arguments

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

### 14.1.2. Exit codes

- **0**: if validation passes

## 14.2. whenever()

a convenient DSL for validating things

### 14.2.1. Example

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

# Chapter 15. File `lib/util.sh`

# util.sh

# Chapter 16. Overview

Miscellaneous utilities.

# Chapter 17. File `lib/pdf.sh`

# Bashmatic Utilities for PDF file handling

# Chapter 18. Overview

Install and uses GhostScript to manipulate PDFs.

- pdf.combine()

## 18.1. pdf.combine()

Combine multiple PDFs into a single one using ghostscript.

### 18.1.1. Example

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

### 18.1.2. Arguments

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

# Chapter 19. File `bin/install-direnv`

## install-direnv

# Chapter 20. Overview

Add direnv hook to shell RC files

- direnv.register()

## 20.1. direnv.register()

Add direnv hook to shell RC files

# Chapter 21. File `bin/regen-usage-docs`

# regen-usage-docs

# Chapter 22. Overview

Regenerates USAGE.adoc && USAGE.pdf

# Chapter 23. File `bin/specs`

## 23.1. specs.init()

Initialize specs

# Chapter 24. File `bin/pdf-reduce`

## 24.1. pdf.do.shrink()

shrinkgs PDF

# Chapter 25. Copyright & License