

File lib/array.sh

- [array.includes\(\)](#)
- [array.join\(\)](#)
- [array.sort\(\)](#)
- [array.min\(\)](#)
- [array.max\(\)](#)
- [array.uniq\(\)](#)

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.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 unique="$(array.sort-numeric "${unsorted[@]}")"
```

File `lib/output-utils.sh`

- `dbg()`

dbg()

Local debugging helper, activate it with `DEBUG=1`

File `lib/output.sh`

- `section()`

section()

Prints a "arrow-like" line using powerline characters

Arguments

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

File `lib/path.sh`

- `path.add()`
- `path.append()`
- `PATH_add()`

`path.add()`

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

`path.append()`

Appends 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.sh`

Overview

OSX Specific Helpers and Utilities

File `lib/db.sh`

- `db.config.parse()`
- `db.psql.connect()`

`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'
```

File `lib/shdoc.sh`

`lib/shdoc.sh`

Helpers to install gawk and shdoc properly.0

Overview

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/is.sh`

is.sh

Overview

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

- `__is.validation.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

`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`

`util.sh`

Overview

Miscellaneous utilities.

File `lib/pdf.sh`

Bashmatic Utilities for PDF file handling

Overview

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/regen-usage-docs`

regen-usage-docs

Overview

Regenerates USAGE.adoc && USAGE.pdf

File `bin/specs`

- `specs.init()`

`specs.init()`

Initialize specs

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.