

10 tips for making Bash a sane programming language

Yaroslav Tkachenko | @sap1ens | sap1ens.com

Senior Data Engineer at Activision

```
#!/usr/bin/env bash LANG="${LANG:-en}" locale=$(echo $LANG | cut -c1-2) unset configuredClient ## This function determines which http get tool the system has installed and
returns an error if there isnt one getConfiguredClient() { if command -v curl &>/dev/null; then configuredClient="curl" elif command -v wget &>/dev/null; then
configuredClient="wget" elif command -v http &>/dev/null; then configuredClient="httpie" elif command -v fetch &>/dev/null; then configuredClient="fetch" else echo "Error:
This tool requires either curl, wget, httpie or fetch to be installed\." >&2 return 1 fi } ## Allows to call the users configured client without if statements everywhere httpGet() { case
"$configuredClient" in curl) curl -A curl -s "$@" ;; wget) wget -qO- "$@" ;; httpie) http -b GET "$@" ;; fetch) fetch -q "$@" ;; esac } getIPWeather() { country=$(httpGet
ipinfo.io/country) > /dev/null ## grab the country if [[ $country == "US" ]]; then ## if were in the us id rather not use longitude and latitude so the output is nicer city=$(httpGet
ipinfo.io/city) > /dev/null region=$(httpGet ipinfo.io/region) > /dev/null if [[ $(echo $region | wc -w) == 2 ]];then region=$(echo $region | grep -Eo "[A-Z]*" | tr -d "[:space:]") fi
httpGet $locale.wttr.in $city,$region$1 else ## otherwise we are going to use longitude and latitude location=$(httpGet ipinfo.io/ip) > /dev/null httpGet
$locale.wttr.in $location$1 fi } getLocationWeather() { args=$(echo "$@" | tr " " + ) httpGet $locale.wttr.in ${args} } checkInternet() { httpGet github.com > /dev/null 2>&1 || { echo
"Error: no active internet connection" >&2; return 1; } # query github with a get request } update() { # To test the tool enter in the default values that are in the examples for each
variable repositoryName="Bash-Snippets" #Name of repostiory to be updated ex. Sandman-Lite username that hosts the reposti nameOfInstallFile="install.sh" # change this if the
installer file has a different name be sure to include file extension if there is one latestVersion=$(httpGet https://api.github.com/repos/$githubUserName/$repositoryName/tags |
grep -Eo '"name":.*?[^\\",|' head -1 | grep -Eo "[0-9.]+") #always grabs the tag without the v option if [[ $currentVersion == "" || $repositoryName == "" || $githubUserName ==
"" || $nameOfInstallFile == "" ]]; then echo "Error: update utility has not been configured correctly." >&2 exit 1 elif [[ $latestVersion == "" ]]; then echo "Error: no active internet
connection" >&2 exit 1 else if [[ "$latestVersion" != "$currentVersion" ]]; then echo "Version $latestVersion available" echo -n "Do you wish to update $repositoryName [Y/n]: "
read -r answer if [[ "$answer" == [Yy] ]]; then cd ~ || { echo 'Update Failed'; exit 1; } if [[ -d ~/$repositoryName ]]; then rm -r -f $repositoryName || { echo "Permissions Error: try
running the update as sudo"; exit 1; }; fi echo -n "Downloading latest version of: $repositoryName." git clone -q "https://github.com/$githubUserName/$repositoryName" &&
touch .BSnippetsHiddenFile || { echo "Failure!"; exit 1; } & while [ ! -f .BSnippetsHiddenFile ]; do { echo -n "."; sleep 2; };done rm -f .BSnippetsHiddenFile echo "Success!" cd
$repositoryName || { echo 'Update Failed'; exit 1; } git checkout "v$latestVersion" 2> /dev/null || git checkout "$latestVersion" 2> /dev/null || echo "Couldn't git checkout to stable
release, updating to latest commit." chmod a+x install.sh #this might be necessary in your case but wasnt in mine. . $nameOfInstallFile "update" || exit 1 cd .. rm -r -f
$repositoryName || { echo "Permissions Error: update succesfull but cannot delete temp files located at ~ $repositoryName delete this directory with sudo"; exit 1; } else exit 1 fi
else echo "$repositoryName is already the latest version" fi fi } usage() { cat <<EOF Weather Description: Provides a 3 day forecast on your current location or a specified
location. With no flags Weather will default to your current location. Usage: weather or weather [flag] or weather [country] or weather [city] [state] weather [i][M] get weather in
imperial units, optional M means windspeed in m/s weather [m][M] get weather in metric units, optional M means windspeed in m/s weather [Moon] grabs the phase of the
moon -u Update Bash-Snippet Tools -h Show the help -v Get the tool version Examples: weather weather Paris m weather Tokyo weather Moon weather mM EOF }
getConfiguredClient || exit 1 while getopts "uvh" opt; do case "$opt" in \?) echo "Invalid option: -$OPTARG" >&2 exit 1 ;; h) usage exit 0 ;; v) echo "Version $currentVersion" exit 0
;; u) checkInternet || exit 1 # check if we have a valid internet connection if this isnt true the rest of the script will not work so stop here update || exit 1 exit 0 ;; :) echo "Option -
$OPTARG requires an argument." >&2 exit 1 ;; esac done if [[ $# == "0" ]]; then checkInternet || exit 1 getIPWeather || exit 1 exit 0 elif [[ $1 == "help" || $1 == ":help" ]]; then
usage exit 0 elif [[ $1 == "update" ]]; then checkInternet || exit 1 update || exit 1 exit 0 fi checkInternet || exit 1 if [[ $1 == "m" ]]; then getIPWeather "?m" || exit 1 elif [[ "${@: -1}"
== "m" ]];then args=$( echo "${@:1:(( $# - 1))}" ?m | sed s " " /g) getLocationWeather $args || exit 1 elif [[ $1 == "M" ]]; then getIPWeather "?M" || exit 1 elif [[ "${@: -1}" == "M"
]];then args=$( echo "${@:1:(( $# - 1))}" ?M | sed s " " /g) getLocationWeather $args || exit 1 elif [[ $1 == "mM" || $1 == "Mm" ]]; then getIPWeather "?m?M" || exit 1 elif [[ "${@: -
1}" == "mM" || "${@: -1}" == "Mm" ]];then args=$( echo "${@:1:(( $# - 1))}" ?m?M | sed s " " /g) getLocationWeather $args || exit 1 elif [[ $1 == "iM" || $1 == "Mi" ]]; then
getIPWeather "?u?M" || exit 1 elif [[ "${@: -1}" == "iM" || "${@: -1}" == "Mi" ]];then args=$( echo "${@:1:(( $# - 1))}" ?u?M | sed s " " /g) getLocationWeather $args || exit 1 elif [[ $1
== "i" ]]; then getIPWeather "?u" || exit 1 elif [[ "${@: -1}" == "i" ]];then args=$( echo "${@:1:(( $# - 1))}" ?u | sed s " " /g) getLocationWeather $args || exit 1 else
getLocationWeather "$@" || exit 1 fi
```

1: Header, part 1

```
#!/usr/bin/env bash
```

```
set -o errexit
```

```
set -o pipefail
```

1: Header, part 2

```
set -o nounset  
[[ "${DEBUG}" == 'true' ]] && set -o xtrace
```

2: Constants

`readonly SOMETHING='immutable value'`

3: Variables, part 1

"\${variable}"

4: Variables, part 2

string interpolation: "\${variable}.yaml"

default / fallback value: "\${variable:-blah}"

string replacement: "\${variable//from/to}"

5: Functions, part 1

```
_my_function () {  
    ...  
}
```

6: Functions, part 2

```
_my_function () {  
  local one_param="$1"  
  local another_param="$2"  
  ...  
}
```

7: Conditionals

```
if [[ -f $file1 && ( -d $dir1 || -d $dir2 ) ]]; then  
    ...  
fi
```

8: Includes

```
readonly BINPATH="$(dirname "$0")"
```

```
# ...
```

```
source "${BINPATH}/../shared/some_functions"
```

9: Pipelines

Short

```
command1 | command2
```

Long

```
command1 \  
| command2 \  
| command3 \  
| command4
```

10: Structure, part 1

```
_main () {}  
_usage () {}  
_setup () {}  
_cleanup () {}
```

10: Structure, part 2

```
_cleanup () {  
    ...  
}
```

```
trap _cleanup TERM INT QUIT
```

Bonus: Linting

<https://www.shellcheck.net>

```
apt-get install shellcheck  
brew install shellcheck
```

...

ShellCheck Output

```
$ shellcheck myscript
```

Line 3:

```
for f in $(ls *.m3u)
```

^-- [SC2045](#): Iterating over ls output is fragile. Use globs.

^-- [SC2035](#): Use `./*glob*` or `-- *glob*` so names with dashes won't become options.

Line 5:

```
grep -qi hq.*mp3 $f \
```

^-- [SC2062](#): Quote the grep pattern so the shell won't interpret it.

^-- [SC2086](#): Double quote to prevent globbing and word splitting.

Line 6:

```
&& echo -e 'Playlist $f contains a HQ file in mp3 format'
```

^-- [SC2039](#): In POSIX sh, echo flags are undefined.

^-- [SC2016](#): Expressions don't expand in single quotes, use double quotes for that.

```
$
```

Bonus: Learn Bash the Hard Way



Bonus: use Python when needed

- “Oh, I didn’t know it’s that hard to implement in bash” → use Python
- “Oh, I wish bash had introduced a package manager” → use Python
- “Oh, we could use that unit testing framework for bash” → use Python

Instead of a Summary

Learn basic building blocks && be consistent!

Thanks!