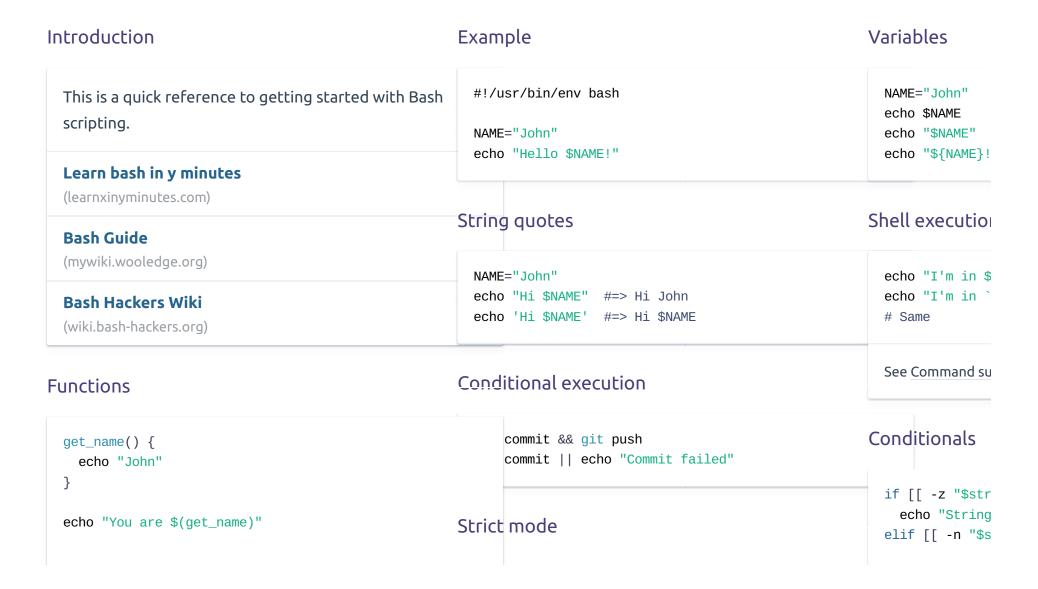
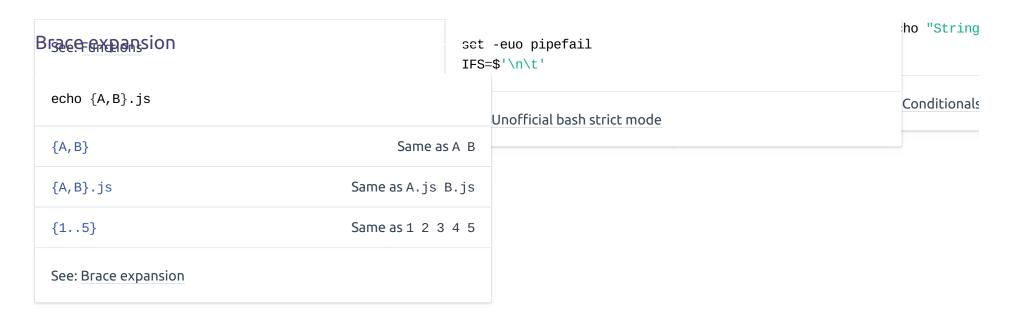
DEVHINTS.IO

Edit

Bash scripting cheatsheet





‡ Parameter expansions

Substitution	Com	iments
\${F00%suffix}	Remove s # 5	Single line
\${F00#prefix}	Remove p	
\${F00%suffix}	Remove long 3	is is a Lti line
\${F00##prefix}		comment
	\${F00%suffix} \${F00#prefix} \${F00%suffix}	\${F00%suffix} Remove s \${F00#prefix} Remove p : \${F00%suffix} Remove long s

```
echo ${food:-Cake} #=> $food or "Cake"
                                                                                                  Replace first metabolitation
                                                       Deficient values
length=2
                                                                                          $F00, or val if unset (or null)
                                                        ${F00:-val}
                                                                                                                     "HELLO WOR
echo ${name:0:length} #=> "Jo"
                                                                                                                       ${STR,}
                                                                                       Set $F00 to val if unset (or null)
                                                        ${F00:=val}
                                                                                                                       ${STR,,}
See: Parameter expansion
                                                                                        val if $F00 is set (and not null)
                                                         ${F00:+val}
                                                                                                                      "hello wor
                                                                                                                       ${STR^}
STR="/path/to/foo.cpp"
                                                                                  Show error message and exit if $F00 is
                                                        ${F00:?message}
                                                                                                                       ${STR^^}
echo ${STR%.cpp}
                     # /path/to/foo
                                                                                                       unset (or null)
echo ${STR%.cpp}.o # /path/to/foo.o
echo ${STR%/*}
                     # /path/to
                                                         Omitting the : removes the (non)nullity checks, e.g. ${F00-val}
echo ${STR##*.}
                                                        expands to val if unset otherwise $F00.
                     # cpp (extension)
echo ${STR##*/}
                     # foo.cpp (basepath)
echo ${STR#*/}
                     # path/to/foo.cpp
echo ${STR##*/}
                     # foo.cpp
echo ${STR/foo/bar} # /path/to/bar.cpp
STR="Hello world"
echo ${STR:6:5} # "world"
echo ${STR: -5:5} # "world"
SRC="/path/to/foo.cpp"
BASE=${SRC##*/} #=> "foo.cpp" (basepath)
DIR=${SRC%$BASE} #=> "/path/to/" (dirpath)
```

‡ Loops

Basic for loop	C-like for loop	Ranges
<pre>for i in /etc/rc.*; do echo \$i done</pre>	<pre>for ((i = 0 ; i < 100 ; i++)); do echo \$i done</pre>	for i in {15 echo "Welc done
		With step size
Reading lines	Forever	for i in {55
<pre>cat file.txt while read line; do echo \$line done</pre>	while true; do done	echo "Welc

Functions

Defining functions Returning values Raising errors myfunc() { myfunc() { myfunc() { echo "hello \$1" local myresult='some value' return 1 echo \$myresult } # Same as above (alternate syntax) if myfunc; the result="\$(myfunc)" function myfunc() { echo "succes echo "hello \$1" else echo "failur fi Arguments

myfunc "John"	\$# Number of arguments
	\$* All positional arguments (as a single word)
	\$@ All positional arguments (as separate strings)
	\$1 First argument
	\$_ Last argument of the previous command
	Note: \$@ and \$* must be quoted in order to perform as described. Otherwise, they do exactly the same thing (arguments as separate strings). See Special parameters.

Conditionals

Conditions	File condition	ons	Example
Note that [[is actually a command/program that	returns eithe [[-e FILE]]	xists
(true) or 1 (false). Any program that obeys the same logic (lik base utils, such as grep(1) or ping(1)) can be used as condit see examples.	- File]] Read	lable
	[[-h FILE]] Syr	mlink
[[-z STRING]]	Empty s [[-d FILE]] Direct	itory

```
Not empty s [[ -w FILE ]]
[[ -n STRING ]]
                                                                                                         Writ
                                                                                                              # String
                                                                                                              if [[ -z "$str
                                                                                                   Size is > 0 t
                                                    E [[ -s FILE ]]
[[ STRING == STRING ]]
                                                                                                                echo "String
                                                                                                              elif [[ -n "$s
                                                Not E [[ -f FILE ]]
[[ STRING != STRING ]]
                                                                                                                echo "String
                                                                                                               else
[[ NUM -eq NUM ]]
                                                    E [[ -x FILE ]]
                                                                                                       Execut
                                                                                                                echo "This n
[[ NUM -ne NUM ]]
                                                Note [[ FILE1 -nt FILE2 ]]
                                                                                            1 is more recent th
                                                                                                              # Combinations
                                                       [[ FILE1 -ot FILE2 ]]
                                                                                            2 is more recent th
[[ NUM -lt NUM ]]
                                                 Less
                                                                                                              if [[ X && Y ]
                                         Less than or e [[ FILE1 -ef FILE2 ]]
                                                                                                       Same
[[ NUM -le NUM ]]
                                                                                                               fi
                                              Greater than
[[ NUM -gt NUM ]]
                                                                                                              # Equal
[[ NUM -ge NUM ]]
                                      Greater than or equal
                                                                                                              if [[ "$A" ==
[[ STRING =~ STRING ]]
                                                  Regexp
                                                                                                              # Regex
                                                                                                              if [[ "A" =~ .
((NUM < NUM))
                                        Numeric conditions
More conditions
                                                                                                              if (( $a < $b
[[ -o noclobber ]]
                                 If OPTIONNAME is enabled
                                                                                                                  echo "$a is
                                                                                                              fi
[[ ! EXPR ]]
                                                      Not
                                                                                                              if [[ -e "file
[[ X && Y ]]
                                                      And
                                                                                                                echo "file e
                                                                                                              fi
                                                       Ог
[[ X || Y ]]
```

‡ Arrays

Defining arrays

```
Fruits=('Apple' 'Banana' 'Orange')

Fruits[0]="Apple"
Fruits[1]="Banana"
Fruits[2]="Orange"
```

Operations

```
Fruits=("${Fruits[@]}" "Watermelon") # Push
Fruits+=('Watermelon') # Also Push
Fruits=( ${Fruits[@]/Ap*/} ) # Remove by regex
unset Fruits[2] # Remove one item
Fruits=("${Fruits[@]}") # Duplicate
Fruits=("${Fruits[@]}" "${Veggies[@]}") # Concatenate
lines=(`cat "logfile"`) # Read from file
```

Working with arrays

```
echo ${Fruits[0]}
                             # Element #0
echo ${Fruits[-1]}
                             # Last elemer
echo ${Fruits[@]}
                             # All element
echo ${#Fruits[@]}
                             # Number of \epsilon
echo ${#Fruits}
                             # String leng
echo ${#Fruits[3]}
                             # String leng
echo ${Fruits[@]:3:2}
                             # Range (from
echo ${!Fruits[@]}
                             # Keys of all
```

Iteration

```
for i in "${arrayName[@]}"; do
  echo $i
done
```

‡ Dictionaries

Defining Working with dictionaries Iteration

```
Iterate over values
                                                         echo ${sounds[dog]} # Dog's sound
declare -A sounds
                                                         echo ${sounds[@]}
                                                                              # All values
                                                                                                                   for val in "${
                                                         echo ${!sounds[@]} # All keys
                                                                                                                     echo $val
sounds[dog]="bark"
                                                                              # Number of elements
                                                         echo ${#sounds[@]}
                                                                                                                   done
sounds[cow]="moo"
                                                         unset sounds[dog]
                                                                              # Delete dog
sounds[bird]="tweet"
                                                                                                                   Iterate over keys
sounds[wolf]="howl"
                                                                                                                   for key in "${
Declares sound as a Dictionary object (aka associative array).
                                                                                                                     echo $key
                                                                                                                   done
```

Options

Options

```
set -o noclobber # Avoid overlay files (echo "hi" > foo)
set -o errexit # Used to exit upon error, avoiding case
set -o pipefail # Unveils hidden failures
set -o nounset # Exposes unset variables
```

Glob options

```
shopt -s nullglob  # Non-matching globs
shopt -s failglob  # Non-matching globs
shopt -s nocaseglob  # Case insensitive g
shopt -s dotglob  # Wildcards match dc
shopt -s globstar  # Allow ** for recur
```

Set GLOBIGNORE as a colon-separated list of patte removed from glob matches.

Bash scripting cheatsheet https://devhints.io/bash

History

Commands

history	Show history
shopt -s histverify	Don't execute expanded result immediately

Operations

11	Execute last command again
!!:s/ <from>/<to>/</to></from>	Replace first occurrence of <from> to <t0> in most recent command</t0></from>
!!:gs/ <from>/<t0>/</t0></from>	Replace all occurrences of <from> to <t0> in most recent command</t0></from>
!\$:t	Expand only basename from last parameter of most recent command
!\$:h	Expand only directory from last parameter of most recent command
!! and !\$ can be replaced with any valid expansion.	

Expansions

!\$	Expand last parameter of most
!*	Expand all parameters of most
! -n	Expand nth most
!n	Expand nth co
! <command/>	Expand most recent invoca

Slices

!!:n	Expand only nth token from most (command is 0; fi
iv	Expand first argument from most
!\$	Expand last token from most
!!:n-m	Expand range of tokens from most
!!:n-\$	Expand nth token to last from most
!! can be r	eplaced with any valid expansion i.e. !c

‡ Miscellaneous

Numeric calculations

```
$((a + 200))  # Add 200 to $a
$(($RANDOM%200))  # Random number 0..199
```

Inspecting commands

```
command -V cd
#=> "cd is a function/alias/whatever"
```

Trap errors

```
trap 'echo Error at about $LINENO' ERR

or

traperr() {
   echo "ERROR: ${BASH_SOURCE[1]} at about ${BASH_LINENO[0]}}

set -o errtrace
trap traperr ERR
```

Subshells

```
(cd somedir; echo "I'm now in $PWD")
pwd # still in first directory
```

Redirection

```
python hello.py > output.txt # stdout t
python hello.py >> output.txt # stdout t
python hello.py 2> error.log # stderr t
python hello.py 2>&1 # stderr t
python hello.py 2>/dev/null # stderr t
python hello.py &>/dev/null # stdout a

python hello.py < foo.txt # feed foc
diff <(ls -r) <(ls) # Compare</pre>
```

Case/switch

```
case "$1" in
  start | up)
  vagrant up
  ;;

*)
  echo "Usage: $0 {start|stop|ssh}"
```

Bash scripting cheatsheet

Elementa Doministi Planse nd's result

```
if ping -c 1 google.com; then
  echo "It appears you have a working internet connection"
fi
                                                  PID of shell
$$
                                    Filename of the shell script
$0
                       Last argument of the previous command
$_
                        return value of piped commands (array)
${PIPESTATUS[n]}
See Special parameters.
                                               All whitespace
[:space:]
                                                   All letters
[:alpha:]
                                          All letters and digits
[:alnum:]
Example
echo "Welcome To Devhints" | tr [:lower:] [:upper:]
WELCOME TO DEVHINTS
```

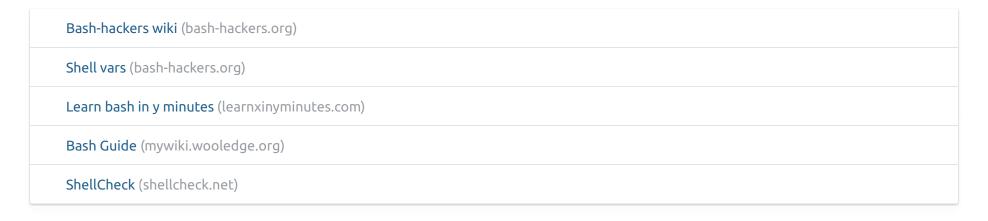
```
if grep -q 'foo' ~/.bash_history; then
echo "You appear to have typed 'foo' ir
fi

pwd # /home/user/foo

SHIFT, STIFFFPFF

-f | --flag )
flag=1
;;
esac; shift; done
if [[ "$1" == '--' ]]; then shift; fi
```

‡ Also see



devhints.io / Search 356+ cheatsheets



Bash scripting cheatsheet https://devhints.io/bash

cheatsheet cheatsheet cheatsheet developers for Debug Bridge) cheatsheet developers. Vim scripting Vim Devhints home cheatsheet cheatsheet Fish shell composer cheatsheet cheatsheet