DEVHINTS.IO

Edit

Bash scripting cheatsheet

Example

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

NAME="John"
echo "Hello $NAME!"
```

Variables

```
NAME="John"
echo $NAME
echo "$NAME"
echo "${NAME}!"
```

String quotes

```
NAME="John"
echo "Hi $NAME"  #=> Hi J
echo 'Hi $NAME'  #=> Hi $
```

Shell execution

```
        in $(pwd)"
        in `pwd`"
```

nd substitution

Conditional execution

```
git commit && git push
git commit || echo "Commit failed"
```

Functions

```
get_name() {
  echo "John"
}

echo "You are $(get_name)"
```

ns

Conditionals

```
if [ -z "$string" ]; then
  echo "String is empty"
```

Strict mode

```
elif [ -n "$string" ]; then
  echo "String is not empty"
fi
```

See: Conditionals

Brace expansion

```
echo {A,B}.js
```

`{A,B}`	Same as `A B`
`{A,B}.js`	Same as `A.js B.js`
`{1..5}`	Same as `1 2 3 4 5`

See: Brace expansion

```
set -euo pipefail
IFS=$'\n\t'
```

ial bash strict mode

Parameter expansions

Basics

```
name="John"
echo ${name}
echo ${name/J/j}    #=> "john" (substitution)
echo ${name:0:2}    #=> "jo" (slicing)
echo ${name::2}     #=> "jo" (slicing)
echo ${name::-1}    #=> "joh" (slicing)
echo ${food:-Cake}  #=> $food or "Cake"
```

```
length=2
echo ${name:0:length}  #=> "jo"
```

See: Parameter expansion

Substitution

`${FOO%suffix}`	Re
`${FOO#prefix}`	Re
`${FOO%%suffix}`	Remove
`${FOO##prefix}`	Remove
`${FOO/from/to}`	Replace
`${FOO//from/to}`	Replace all
`${FOO/%from/to}`	Re

Length

Comments

```
# Single line comment
```

```
: '
This is a
multi line
comment
'
```

Substrings

```
${FOO:0:3}
```

```
STR="/path/to/foo.cpp"
echo ${STR%.cpp}    # /path/to/foo
echo ${STR%.cpp}.o  # /path/to/foo.o

echo ${STR##*.}     # 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=${STR##*/}   #=> "foo.cpp" (basepath)
DIR=${SRC%$BASE}  #=> "/path/to" (dirpath)
```

`${#FOO}`	Length of $FOO

3}

Default values

`${FOO:-val}`	
`${FOO:=val}`	
`${FOO:+val}`	
`${FOO:?message}`	Sh

The `:` is optional (eg, `${FOO=wo`

Loops

Basic for loop

```
for i in /etc/rc.*; do
  echo $i
done
```

Ranges

```
for i in {1..5}; do
    echo "Welcome $i"
done
```

Reading lines

```
cat file.txt | while read
  echo $line
done
```

With step size

```
for i in {5..50..5}; do
        "Welcome $i"
```

Forever

```
while true; do
  ···
done
```

Functions

Defining functions

```
myfunc() {
    echo "hello $1"
}
```

```
# Same as above (alternate syntax)
function myfunc() {
    echo "hello $1"
}
```

```
myfunc "John"
```

Returning values

```
myfunc() {
    local myresult='some value'
    echo $myresult
}
```

```
result=$(myfunc)
```

Raising errors

```
myfunc() {
  return 1
}
```

```
if myfunc; then
  echo "success"
else
  echo "failure"
fi
```

Arguments

`$#`	Number of arguments
`$*`	All arguments
`$@`	All arguments, starting from first

`$1`	First argument

See Special parameters.

Conditionals

Conditions

`[ -z STRING ]`	E
`[ -n STRING ]`	Not e
`[ NUM -eq NUM ]`	
`[ NUM -ne NUM ]`	
`[ NUM -lt NUM ]`	
`[ NUM -le NUM ]`	Less th
`[ NUM -gt NUM ]`	(
`[ NUM -ge NUM ]`	Greater th
`[[ STRING =~ STRING ]]`	
`(( NUM < NUM ))`	Numeri
`[ -o noclobber ]`	If OPTIONNAM
`[ ! EXPR ]`	Not

File conditions

`[ -e FILE ]`	
`[ -r FILE ]`	
`[ -h FILE ]`	
`[ -d FILE ]`	
`[ -w FILE ]`	
`[ -s FILE ]`	Size
`[ -f FILE ]`	
`[ -x FILE ]`	
`[ FILE1 -nt FILE2 ]`	1 is more re
`[ FILE1 -ot FILE2 ]`	2 is more re
`[ FILE1 -ef FILE2 ]`	

Example

```
# String
if [ -z "$string" ]; then
  echo "String is empty"
elif [ -n "$string" ]; th
  echo "String is not emp
fi
```

```
# Combinations
if [ X ] && [ Y ]; then
  ...
fi
```

```
# Regex
if [[ "A" =~ "." ]]
```

```
if (( $a < $b ))
```

```
if [ -e "file.txt" ]; the
  echo "file exists"
fi
```

`[ X ] && [ Y ]`	And
`[ X ] || [ Y ]`	Or

Arrays

Defining arrays

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

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

Working with arrays

```
echo ${Fruits[0]}           # Element #0
echo ${Fruits[@]}           # All elements, space-se
echo ${#Fruits[@]}          # Number of elements
echo ${#Fruits}             # String length of the 1
echo ${#Fruits[3]}          # String length of the N
echo ${Fruits[@]:3:2}       # Range (from position 3
```

Operations

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

Iteration

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

Options

Options

Glob options

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

```
set -o nullglob    # Non-matching globs are removed
set -o failglob    # Non-matching globs throw errors
set -o nocaseglob  # Case insensitive globs
set -o globdots    # Wildcards match dotfiles ("*.sh
set -o globstar    # Allow ** for recursive matches
```

Set `GLOBIGNORE` as a colon-separated list of patterns to be remo
glob matches.

History

Commands

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

Operations

`!!:s/<FROM>/<TO>/`	Replace first occurrence of <FROM> to <TO> in most recent command
`!!:gs/<FROM>/<TO>/`	Replace all occurrences of <FROM> to <TO> in most recent command
`!$:t`	Expand only basename from last parameter of most recent command

Expansions

`!$`	Expand last parameter of most recen
`!*`	Expand all parameters of most recen
`!-n`	Expand nth most recen
`!n`	Expand nth comman
`!<command>`	Expand most recent invocation of command

Slices

`!!:n`	Expand only nth token from most recent command is `0`; first
`!!:n-m`	Expand range of tokens from most recen

`!$:h`	Expand only directory from last parameter of most recent command

`!!` and `!$` can be replaced with any valid expansion.

`!!:n-$`	Expand nth token to last from most recent

`!!` can be replaced with any valid expansion i.e. `!cat`, `!-2`, `!42`, e

Miscellaneous

Numeric calculations

```
$((a + 200))      # Add 200 to $a
```

```
$((RANDOM%=200))  # Random number 0..200
```

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]}"
}
```

Subshells

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

Redirection

```
python hello.py > output.txt   # stdout to (file)
python hello.py >> output.txt  # stdout to (file), a
python hello.py 2> error.log   # stderr to (file)
python hello.py 2>&1           # stderr to stdout
python hello.py 2>/dev/null    # stderr to (null)
python hello.py &>/dev/null    # stdout and stderr t
```

```
python hello.py < foo.txt
```

Case/switch

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

```
set -o errtrace
trap traperr ERR
```

Source relative

```
source "${0%/*}/../share/foo.sh"
```

Directory of script

```
DIR="${0%/*}"
```

Heredoc

```
cat <<END
hello world
END
```

Reading input

```
echo -n "Proceed? [y/n]: "
read ans
echo $ans
```

```
read -n 1 ans    # Just one character
```

```
    ;;

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

printf

```
printf "Hello %s, I'm %s" Sven Olga
#=> "Hello Sven, I'm Olga
```

Getting options

```
while [[ "$1" =~ ^- && ! "$1" == "--" ]]; do case $1
  -V | --version )
    echo $version
    exit
    ;;
  -s | --string )
    shift; string=$1
    ;;
  -f | --flag )
    flag=1
    ;;
esac; shift; done
if [[ "$1" == '--' ]]; then shift; fi
```

Special variables

`$?`	Exit status
`$!`	PID of last backg
`$$`	

See Special parameters.

Also see

- Bash-hackers wiki (bash-hackers.org)
- Shell vars (bash-hackers.org)
- Learn bash in y minutes (learnxinyminutes.com)

▶ **4 Comments** for this cheatsheet. Write yours!

devhints.io / Search 368+ cheatsheets

Over 368 curated cheatsheets, by developers for developers.

Devhints home

Other CLI cheatsheets

Cron
cheatsheet

httpie
cheatsheet

adb (Android Debug Bridge)
cheatsheet

composer
cheatsheet

Fish shell
cheatsheet

Rsync
cheatsheet

Top cheatsheets

Elixir
cheatsheet

ES2015+
cheatsheet

React.js
cheatsheet

Vim
cheatsheet

Vim scripting
cheatsheet

Capybara
cheatsheet