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

Shell execution

```
echo "I'm in $(pwd)"
echo "I'm in `pwd`"
# Same
```

Command substitution

See Command substitution

```
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

Parameter expansions

Basics

```
name="John"
echo ${name}
echo ${name/J/j}    #=> "john" (subst
echo ${name:0:2}    #=> "Jo" (slicing
echo ${name::2}     #=> "Jo" (slicing
echo ${name::-1}    #=> "Joh" (slicin
echo ${name:(-1)}   #=> "n" (slicing
echo ${name:(-2):1} #=> "h" (slicing
echo ${food:-Cake}  #=> $food or "Cak
```

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

See: Parameter expansion

```
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 (basepa

echo ${STR#*/}      # path/to/foo.cpp
echo ${STR##*/}     # foo.cpp

echo ${STR/foo/bar} # /path/to/bar.cp
```

```
STR="Hello world"
echo ${STR:6:5}   # "world"
```

`${FOO:-val}`	$FOO, or val if not set
`${FOO:=val}`	Set $FOO to val if not set
`${FOO:+val}`	val if $FOO is set
`${FOO:?message}`	Show error message and exit if $FOO is not set

The `:` is optional (eg, `${FOO=word}` works)

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

♯ Loops

Basic for loop

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

C-like for loop

```
for ((i = 0 ; i < 100 ; i++)); do
```

```
  echo $i
done
```

Ranges

Reading lines

```
while true; do
  ...
done
```

With step size

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

Functions

Defining functions

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

```
# Same as above (alternate syntax)
function myfunc() {
```

```
echo "hello $1"
}
```

$#	Number of arguments
$*	All arguments
$@	All arguments, starting from first
$1	First argument

See Special parameters.

Conditionals

Conditions

Note that `[[` is actually a command/program that returns either `0` (true) or `1` (false). Any program that obeys the same logic (like all base utils, such as `grep(1)` or `ping(1)`) can be used as condition, see examples.

`[[ -z STRING ]]`	Empty string
`[[ -n STRING ]]`	Not empty string
`[[ STRING == STRING ]]`	Equal
`[[ STRING != STRING ]]`	Not Equal
`[[ NUM -eq NUM ]]`	Equal
`[[ NUM -ne NUM ]]`	Not equal
`[[ NUM -lt NUM ]]`	Less than
`[[ NUM -le NUM ]]`	Less than or equal
`[[ NUM -gt NUM ]]`	Greater than
`[[ NUM -ge NUM ]]`	Greater than or equal
`[[ STRING =~ STRING ]]`	Regexp

File conditions

`[[ -e FILE ]]`	Exists
`[[ -r FILE ]]`	Readable

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

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

```
# Equal
if [[ "$A" == "$B" ]]
```

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

```
if (( $a < $b )); then
   echo "$a is smaller than $b"
fi
```

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

♯ Arrays

Defining arrays

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

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

Working with arrays

```
echo ${Fruits[0]}           # Element
echo ${Fruits[@]}           # All ele
echo ${#Fruits[@]}          # Number
echo ${#Fruits}             # String
echo ${#Fruits[3]}          # String
echo ${Fruits[@]:3:2}       # Range (
```

Operations

```
Fruits=("${Fruits[@]}" "Watermelon")
Fruits+=('Watermelon')
Fruits=( ${Fruits[@]/Ap*/} )
unset Fruits[2]
Fruits=("${Fruits[@]}")
Fruits=("${Fruits[@]}" "${Veggies[@]}
lines=(`cat "logfile"`)
```

Iteration

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

Dictionaries

Defining

```
declare -A sounds
```

```
sounds[dog]="bark"
sounds[cow]="moo"
sounds[bird]="tweet"
sounds[wolf]="howl"
```

Declares sound as a Dictionary object (aka associative array).

Working with dictionaries

```
echo ${sounds[dog]} # Dog's sound
echo ${sounds[@]}   # All values
echo ${!sounds[@]}  # All keys
echo ${#sounds[@]}  # Number of eleme
unset sounds[dog]   # Delete dog
```

Iteration

Iterate over values

```
for val in "${sounds[@]}"; do
  echo $val
done
```

Iterate over keys

```
for key in "${!sounds[@]}"; do
  echo $key
done
```

Options

Globbing options

```
set -o nullglob    # Non-matching glo
set -o failglob    # Non-matching glo
set -o nocaseglob  # Case insensitive
set -o globdots    # Wildcards match
set -o globstar    # Allow ** for rec
```

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

History

Commands

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

Expansions

`!$`	Expand last parameter of most recent command
`!*`	Expand all parameters of most recent command
`!-n`	Expand nth most recent command

Slices

`!!`	Execute last command
`!!:n`	Expand only nth token from most recent command (command is `0`; first argument is `1`)
`!^`	Expand first argument from most recent command
`!$`	Expand last token from most recent command
`!!:n-m`	Expand range of tokens from most recent command
`!!:n-$`	Expand nth token to last from most recent command

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

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

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

Miscellaneous

Numeric calculations

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

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

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

```
pwd # /home/user/foo
```

`$0`	Filename of the shell script

See Special parameters.

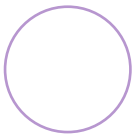
```
    ;;
esac; shift; done
if [[ "$1" == '--' ]]; then shift; fi
```

♯ Also see

- Bash-hackers wiki (bash-hackers.org)
- Shell vars (bash-hackers.org)
- Learn bash in y minutes (learnxinyminutes.com)
- Bash Guide (mywiki.wooledge.org)
- ShellCheck (shellcheck.net)

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

devhints.io / Search 381+ cheatsheets



Over 381 curated cheatsheets, by developers for developers.

Devhints home

Other CLI cheatsheets

- Cron cheatsheet
- Homebrew cheatsheet
- httpie cheatsheet
- adb (Android Debug Bridge) cheatsheet
- composer cheatsheet
- Fish shell cheatsheet

Top cheatsheets

- Elixir cheatsheet
- ES2015+ cheatsheet
- React.js cheatsheet
- Vimdiff cheatsheet
- Vim cheatsheet
- Vim scripting cheatsheet