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}!"
```

## Conditional execution

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

## Functions

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

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

See: Functions

## Conditionals

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

See: Conditionals

## Brace expansion

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

| | |
|---|---|
| {A,B} | Sa |
| {A,B}.js | Same as A |
| {1..5} | Same as 1 |

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"
echo ${STR:-5:5}  # "world"
```

## Substitution

| | |
|---|---|
| `${FOO%suffix}` | Rem |
| `${FOO#prefix}` | Rem |
| `${FOO%%suffix}` | Rer |
| `${FOO##prefix}` | Rer |
| `${FOO/from/to}` | Re |
| `${FOO//from/to}` | R |
| `${FOO/%from/to}` | Repl |
| `${FOO/#from/to}` | Repl |

## Length

| | |
|---|---|
| `${#FOO}` | Lengt |

## Default values

| | |
|---|---|
| `${FOO:-val}` | $FOO, or |
| `${FOO:=val}` | Set $FO |
| `${FOO:+val}` | val if $ |

```
SRC="/path/to/foo.cpp"
BASE=${SRC##*/}   #=> "foo.cpp" (base
DIR=${SRC%$BASE}  #=> "/path/to/" (di
```

```
${FOO:?message}          S
                    messag
                    if $FOO
```

The : is optional (eg, ${FOO=wor

# Loops

## Basic for loop

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

## C-like for loop

```
for ((i = 0 ; i < 100 ; i++
  echo $i
done
```

## Reading lines

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

## Forever

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

# Functions

## Defining functions

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

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

## Returning values

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

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

## Arguments

```
myfunc "John"
```

| | |
|---|---|
| $# | Number of a |
| $* | All a |
| $@ | All arguments, starting |
| $1 | First |
| $_ | Last argument of the |

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 |

## File conditions

| | |
|---|---|
| `[[ -e FILE ]]` | |
| `[[ -r FILE ]]` | |
| `[[ -h FILE ]]` | |
| `[[ -d FILE ]]` | |
| `[[ -w FILE ]]` | |
| `[[ -s FILE ]]` | |
| `[[ -f FILE ]]` | |
| `[[ -x FILE ]]` | E |
| `[[ FILE1 -nt FILE2 ]]` | re |
| `[[ FILE1 -ot FILE2 ]]` | re |

| | |
|---|---|
| `[[ 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 |
| `(( NUM < NUM ))` | Numeric conditions |
| `[[ -o noclobber ]]` | If OPTIONNAME is enabled |
| `[[ ! EXPR ]]` | Not |

| | |
|---|---|
| `[[ FILE1 -ef FILE2 ]]` | S |

# Arrays

## Defining arrays

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

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

## W

## Operations

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

## Ite

# 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
echo ${sounds[@]}   # All v
echo ${!sounds[@]}  # All k
echo ${#sounds[@]}  # Numbe
unset sounds[dog]   # Delet
```

# Options

## Options

```
set -o noclobber  # Avoid overlay fil
set -o errexit    # Used to exit upon
set -o pipefail   # Unveils hidden fa
set -o nounset    # Exposes unset var
```

## Gl

# History

## Commands

## Ex

### Operations

| | |
|---|---|
| `history` | Show history |
| `!!` | Execute last command again |
| `!!: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 |
| `!$:t` | Expand only basename from last parameter of most recent command |
| `!$: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..
```

### Inspecting commands

### Sli

### Su

### Re

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

```
if ping -c 1 google.com; then
  echo "It appears you have a working
fi
```

| | |
|---|---|
| $$ | PID of shell |
| $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)

devhints.io / Search 383+ cheatsheets

Over 383 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