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)"
```

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}
```

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

```
{1..5}
```

See: Brace expansion

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

```
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)
```

Substitution

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

Length

```
${#FOO}
```

Loops

Basic for loop

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

Ranges

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

```
done
```

Forever

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

```
done
```

With step size

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)
```

Arguments

`$#`
`$*`
`$@`
`$1`

See Special parameters.

Conditionals

Conditions

`[ -z STRING ]`
`[ -n STRING ]`
`[ NUM -eq NUM ]`

File conditions

`[ -e FILE ]`
`[ -r FILE ]`
`[ -h FILE ]`

`[ NUM -ne NUM ]`	`[ -d FILE ]`
`[ NUM -lt NUM ]`	`[ -w FILE ]`
`[ NUM -le NUM ]`	`[ -s FILE ]`
`[ NUM -gt NUM ]`	`[ -f FILE ]`
`[ NUM -ge NUM ]`	`[ -x FILE ]`
`[[ STRING =~ STRING ]]`	`[ FILE1 -nt FILE2 ]`
`(( NUM < NUM ))`	`[ FILE1 -ot FILE2 ]`
`[ -o noclobber ]`	`[ FILE1 -ef FILE2 ]`

`[ ! EXPR ]`	Not
`[ X ] && [ Y ]`	And
`[ X ] || [ Y ]`	Or

Arrays

Defining arrays

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

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

Working with array

```
echo ${Fruits[0]}
echo ${Fruits[@]}
echo ${#Fruits[@]}
echo ${#Fruits}
echo ${#Fruits[3]}
echo ${Fruits[@]: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 "${arrayNa
  echo $i
done
```

Options

Options

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

Glob options

```
set -o nullglob
set -o failglob
set -o nocaseglob
set -o globdots
set -o globstar
```

Set GLOBIGNORE as a co

History

Commands

```
history
```

```
shopt -s histverify
```

Don’t execute expande

Expansions

```
!$
```

```
!*
```

```
!-n
```

```
!n
```

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

```
!$:h
```

Expand only directory from last parameter of m

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

Slices

```
!!:n
```

```
!!:n-m
```

```
!!:n-$
```

!! can be replaced wit

Miscellaneous

Numeric calculations

Subshells

```
$((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]}"
}

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

```
(cd somedir; echo "
pwd # still in firs
```

Redirection

```
python hello.py > o
                >>
                2>
                >&
python hello.py 2>/
python hello.py &>/
```

Case/switch

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

  *)
    echo "Usage: $0
    ;;
esac
```

printf

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

Getting options

```
while [[ "$1" =~ ^-
  -V | --version )
                  )
    shift; string=$
    ;;
  -f | --flag )
    flag=1
    ;;
```

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

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

Special variables

$?
$!
$$
See Special parameters

Also see

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

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