Bash Functions Cheat Sheet

Edit Cheat Sheet

See also Bash

Declaring Functions

Note: the () after the function name is optional.

```
my_func() {
    printf "Hello!\n"
}

# Call it with
my_func
```

Passing Parameters

Note: Bash doesn't support prototyping, parameter types or references.

```
my_func() {
    printf "Hello %s %s\n" "$1" "$2"
}

my_func "literal 1" "$var2"
```

It is good style to "shift" parameters like this

```
my_func() {
    param1=$1; shift
    param2=$1; shift

    printf "Hello %s %s\n" "$param1" "$param2"
}
```

Return Values

You can return **numbers only**! Similar to program exit codes, function have return codes. Check below to see how to return data from a function.

CHEAT SHEETS

Agile
Containers
DevOps Automation
DevOps Linux
DevOps Services
Languages
Network
Scripting
Security
Solutions
Virtualization
Windows
Glib Examples
Go Examples
Javascript Examples
PHP Examples

APIs
Bash Associative Array
Bash Functions
Bash Regex
Bash *
Color Distance
Shell Problems
Shell-Scripting
awk
packages.json
sed

```
my_func() {
    return 1
}

my_func

# Check return code
if [ $? -ne 0 ]; then
    printf "Return code: %d\n" $?
fi
```

Returning data from a function

The best way to return data is catching the functions STDOUT using \$()

```
my_func() {
    printf "Some output lines\nLine2\nLine3\n"
}

output=$(my_func)
```