Categories

Blogs

Cheatsheet

Recent Articles

TypeScript Cheat Sheet

PowerShell Cheat Sheet

Shell Scripting Cheat Sheet

Bash Cheat Sheet

JavaScript Cheat Sheet

Shell Scripting Cheat Sheet

22 January 2024 by Husaif Sayyed

Shell scripting is a powerful skill that allows users to automate tasks, streamline workflows, and perform system administration tasks efficiently. Whether you're a beginner or an experienced developer, having a cheatsheet at your fingertips can be invaluable. In this blog post, we'll provide a comprehensive shell scripting cheat sheet to help you navigate the world of shell scripting effortlessly.

Links

- Shell Scripting Documentation
- Bash Cheatsheet
- Download Shell Cheat Sheet PDF
- ★ Want More Cheatsheet

Hello World! Shell Script Program

Create a `hello.sh`

```
#!/bin/bash
echo "Hello World"
```

Run the bash file using `./hello.sh` or `sh hello.sh` or `bash hello.sh`

Basics of Shell Scripting

Shebang Line

```
#!/bin/bash
```

Specify the shell to be used for running the script.

Comments

```
# This is a comment
```

Variables

```
variable_name="Hello, World!"
```

Printing

```
echo "This is a message"
```

Input and Output: Handling Data

Read User Input

```
read -p "Enter your name: " name
```

Output Redirection

```
command > output.txt
```

Input Redirection

```
command < input.txt
```

Conditional Statements: Making Decisions

If-Else Statement

```
if [ condition ]; then
    # code to execute if condition is true
else
    # code to execute if condition is false
fi
```

Case Statement

```
case "$variable" in
    pattern1) echo "Pattern 1 matched";;
    pattern2) echo "Pattern 2 matched";;
    *) echo "No pattern matched";;
esac
```

Loops: Repeating Tasks

For Loop

```
for item in list; do
    # code to execute for each item
done
```

While Loop

```
while [ condition ]; do
    # code to execute while condition is true
done
```

Until Loop

```
until [ condition ]; do
    # code to execute until condition is true
done
```

Functions: Reusable Code Blocks

Defining a Function

```
function_name() {
  # code for the function
}
```

Calling a Function

```
function_name
```

File Operations: Manipulating Files and Directories

Creating a Directory

```
mkdir directory_name
```

Removing a File

```
rm file_name
```

Copying Files

```
cp source_file destination
```

Error Handling: Dealing with Issues

Exit Status

```
command
if [ $? -eq 0 ]; then
    echo "Command executed successfully"
else
    echo "Command failed"
fi
```

Try-Catch

```
{ command1 && command2; } || { echo "Error occurred"; }
```

Regular Expressions and Pipelines

Grep

```
grep "pattern" file
```

Awk

```
awk '{ print $1 }' file
```

Pipelines

```
command1 | command2
```

Environment Variables: Managing System Variables

Viewing Environment Variables

```
echo $HOME
```

Setting Environment Variables

```
export MY_VARIABLE="some_value"
```

PATH Variable

```
echo $PATH
```

Arrays: Handling Lists of Data

Declaring an Array

```
my_array=("item1" "item2" "item3")
```

Accessing Array Elements

```
echo ${my_array[1]}
```

Looping through an Array

```
for item in "${my_array[@]}"; do
    echo $item
done
```

Math Operations: Performing Arithmetic

Basic Arithmetic

```
result=$((5 + 3))
echo $result
```

Using expr

```
result=$(expr 5 + 3)
echo $result
```

String Manipulation: Working with Text

Concatenation

```
str1="Hello"
str2="World"
result=$str1$str2
echo $result
```

Substring Extraction

```
string="abcdef"
echo ${string:1:3}  # Output: bcd
```

Command Line Arguments: Handling Inputs

Accessing Arguments

```
echo "Script Name: $0"
echo "First Argument: $1"
```

$0 is the script name, $1, $2, ... are the arguments

Number of Arguments

```
echo "Number of Arguments: $#"
```

Sleep and Delay: Pausing Execution

Sleep Command

```
sleep 5  # sleeps for 5 seconds
```

Pause Script Execution

```
read -p "Press Enter to continue..."
```

Regular Expressions: Pattern Matching

Matching Patterns with [[and =~:

```
if [[ $string =~ ^[0-9]+$ ]]; then
    echo "String is a number"
fi
```

Wildcard Matching

```
if [[ $file_name == *.txt ]]; then
    echo "File is a text file"
fi
```

Networking Commands: Checking Connectivity

Ping

```
ping -c 4 google.com
```

Check Open Ports

```
netstat -lntu
```

Debugging Scripts: Troubleshooting

Enable Debug Mode

```
set -x
```

Print Messages for Debugging

```
echo "Debug message here"
```

Creating Aliases: Shortcuts for Commands

Alias Creation

```
alias ll='ls -al'
```

List All Aliases

```
alias
```

Congratulations on reaching the end of this Shell Scripting Programming Language Cheatsheet! This resource is designed to make your coding experience easy and efficient. Feel free to bookmark this page or download the PDF for future reference. Happy Shell Scripting!

Cheatsheet

Bash Programming Language, Programming Language Cheatsheet, Shell Scripting

Bash Cheat Sheet

PowerShell Cheat Sheet

Type cheatsheet, code

Search

Related Posts

Python Cheat Sheet

Zig Cheat Sheet

Java Cheat Sheet

Bash Cheat Sheet

JavaScript Cheat Sheet

TypeScript Cheat Sheet

PowerShell Cheat Sheet

More Cheatsheet Tutorials