**Bash syntax**

**Read input from console**

read var1

**Variable declaration**

no need to declare variable just assign

var1=”foo”

var2=$var1

echo $var2

any space will occurred an error

**Constant declaration**

readonly cnst=”constant value”

**String Concatenation**

var1=”foo”

var2=”bar”

var3=$var1$var2”extra\_portion”

**Prepare multi line string:**

echo "\

.state('app.$1', {\n\

url: '\/$2',\n\

views: {\n\

'uiView': {\n\

templateUrl: 'app\/main\/module\/$var2\/$var3\/$var4.html',\n\

controller: '$ctrlName as vm'\n\

}\n\

}\n\

})"

to print $ use \$

to print “ use “\””

to print / use \/

to use new line use escape char \

to print new line use \n

to use var use $varname

**Write to a file**

echo $var1>file name

echo $var1>>file name to append

**Bash conditional statement:**

*if TEST-COMMAND;then*

*STATEMENTS1*

*else*

*STATEMENTS2*

*fi*

use double third bracket to use conditional operator and separate using space

*if [[ $isJsonAvailable == 'Y' || $isJsonAvailable == 'N' ]]; then*

*finished=true*

*fi*

* -z VAR - True if the VAR is empty.
* -n VAR - True if the length of VAR is greater than zero.
* STRING1 = STRING2 - True if STRING1 and STRING2 are equal.
* STRING1 != STRING2 - True if STRING1 and STRING2 are not equal.
* INTEGER1 -eq INTEGER2 - True if INTEGER1 and INTEGER2 are equal.
* INTEGER1 -gt INTEGER2 - True if INTEGER1 is greater than INTEGER2.
* INTEGER1 -lt INTEGER2 - True if INTEGER1 is less than INTEGER2.
* INTEGER1 -ge INTEGER2 - True if INTEGER1 is equal or greater than INTEGER2.
* INTEGER1 -le INTEGER2 - True if INTEGER1 is equal or less than INTEGER2.
* -h FILE - True if the FILE exists and is a symbolic link.
* -r FILE - True if the FILE exists and is readable.
* -w FILE - True if the FILE exists and is writable.
* -x FILE - True if the FILE exists and is executable.
* -d FILE - True if the FILE exists and is a directory.
* -e FILE - True if the FILE exists and is a file, regardless of type (node, directory, socket, etc.).
* -f FILE - True if the FILE exists and is a regular file (not a directory or device)

**Bash functions**

**syntax:**

function\_name () {

commands

}

example:

hello\_world () {

echo 'hello, world'

}

hello\_world

In bash all variables are by default global. That means any variable can be accessed anywhere

So for function return normally use global function

*var1='A'*

*var2='B'*

*my\_function () {*

*local var1='C'*

*var2='D'*

*}*

*echo "Before executing function: var1: $var1, var2: $var2"*

*my\_function*

*echo "After executing function: var1: $var1, var2: $var2"*

**If we want to use local variable then use local keyword**

*my\_function () {*

*local func\_result="some result"*

*echo "$func\_result"*

*}*

local keyword only can be used inside functions

**To pass parameter in function**

*my\_function () {*

*local func\_result="$1 : $2"*

*echo "$func\_result"*

*}*

*my\_function “name” “zahid”*

**Return with local keyword**

*my\_function () {*

*local func\_result="$1 : $2"*

*echo "$func\_result"*

*}*

*func\_result="$(my\_function “name” “zahid”)"*

*echo $func\_result*

Here if we want to pass value with variable then using double quote is safer because

if

*$var1=”name”*

*$var2=”Zahid Hassan”*

my\_function $var1 $var2

then

$1=name

$2=Zahid

$3=Hassan

so use my\_function “$var1” “$var2”