

# Bash Variable – Syntax & Examples

## Bash Variable

---

**Bash Variable** in bash shell scripting is a memory location that is used to contain a number, a character, a string, an array of strings, etc.

### Some important points to remember about variables in bash scripting

---

- There are no data types for a variable. It can contain a number, a character, a string, an array of strings, etc. and be overridden with any other value.
- There is no need to declare a variable explicitly. When you assign a value to the reference, variable declaration happens implicitly.

We shall go through the following topics in this tutorial

- [Syntax](#)
- [Example Script](#)
- [Local Variable](#)

## Syntax

---

Following is the syntax to initialize a variable

```
variableReference=value
```

**Note :** No space should be given before and after = , failing which produces error “**syntax error near unexpected token**”.

## Examples for Bash Variable

---

Following example demonstrates simple initialization of bash variables of types : number, character, string and array.

### Bash Script File

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

```
# number variable
num=10
echo $num

# character variable
ch='c'
echo $ch

# string variable
str="Hello Bob!"
echo $str

# array variable
arr=( "bash" "shell" "script" )
echo "${arr[0]}"
echo "${arr[1]}"
echo "${arr[2]}"
```

When the above **bash variable example** is run in Terminal, we will get the following output.

### Output

```
$ ./bash-variable-example
10
c
Hello Bob!
bash
shell
script
```

## Bash Local Variable

---

Bash Local Variable is used to override a global bash variable, in local scope, if already present with the same name.

### Syntax

---

Following is the syntax of a bash local variable

```
local variableReference=value
```

### Example

---

Following is an example bash script to demonstrate the usage of local variable.

### Bash Script File

```
#!/bin/bash

# bash variable
SHELL="Unix"

function bashShell {
    # bash local variable
    local SHELL="Bash"
    echo $SHELL
}

echo $SHELL
bashShell
echo $SHELL
```

When above **bash local variable example** is run in Terminal, we will get the following output.

### Output

```
arjun@arjun-VPCEH26EN:~/workspace/bash$ ./bash-local-variable-example
Unix
Bash
Unix
```

The first echo statement is in global scope and SHELL has value of UNIX, but when bashShell function is called, the local variable SHELL overrides the global variable and hence the echo \$SHELL echoed Bash.

## Conclusion

In this [Bash Tutorial – Bash Variable](#), we have learnt that there are no data types in bash, and the syntax to initialize a variable, and also about local bash local variables with example scripts.

### Bash Shell Scripting

- ◆ [Bash Tutorial](#)
- ◆ [Bash Script Example](#)
- ◆ [Bash File Extension](#)
- ◆ [Bash Echo](#)
- ◆ [Bash Comments](#)
- ⇒ **[Bash Variable](#)**
- ◆ [Bash Command Line Arguments](#)
- ◆ [Bash Read User Input](#)

- ◆ [Bash Read Password](#)

- ◆ [Bash Date Format](#)

- ◆ [Bash Sleep](#)

## Operators

- ◆ [Bash Arithmetic Operators](#)

## Conditional Statements

- ◆ [Bash If](#)

- ◆ [Bash If Else](#)

- ◆ [Bash Else If](#)

- ◆ [Bash Case](#)

## Loops

- ◆ [Bash For Loop](#)

- ◆ [Bash While Loop](#)

- ◆ [Bash Until Loop](#)

## Strings

- ◆ [Bash String Manipulation Examples](#)

- ◆ [Bash String Length](#)

- ◆ [Bash If String Equals](#)

- ◆ [Bash Split String](#)

- ◆ [Bash SubString](#)

- ◆ [Bash Concatenate String](#)

- ◆ [Bash Concatenate Variables to Strings](#)

## Functions

- ◆ [Bash Function](#)

- ◆ [Bash Override Built-in Commands](#)

## Arrays

- ◆ [Bash Array](#)

## Files

◆ Bash Write to File

◆ Bash Read File

◆ Bash Read File line by line

◆ Bash If File Exists

◆ Bash If File is Directory

◆ Bash If File is Readable

### **Bash Others**

◆ Bash Check if variable is set