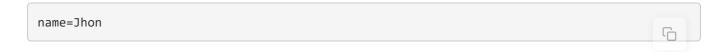# Variable Assignment & Expansion

Get yourself accustomed to variable assignment and expansion.

How to set a variable?

Variables usually come in handy when you need to store or retieve something during program execution. To assign a value to a variable we use **"="** operator. In Bash, you do not need to declare the type of a variable unlike C++ and JAVA. Which means you can assign any value to a variable whether it is a number, a character or a string of characters.

Some variables are preset by the system but we can set our own variables as well. The syntax to assign value to a variable is given below:

```
name=Jhon
```

Variable Expansion

We use `"$"` sign to give a reference to a variable. This technique is often referred as *Parameter Expansion* in Bash. `$` sign basically tells Bash that there are variables present in the script which needs to be substituted before interpretation of any line so that it can replace the name of that variable with its value.

For example if `my_variable` is a name of the variable, then `$my_variable` is a reference to that variable and returns the value stored in that memory location. `echo` command is used to check the value of any variable. Here's how we will code this in Bash:

```
name=Jhon
echo "Hello $name, Welcome to Educative!"
```

Parameter Expansion Operators

Sometimes, we might need to modify the value to make the output look more presentable. We use three operator for this purpose:

Operator	Use
`//`	Replace characters, special characters etc.
`%`	Trim the value from end based on any character or delimeter
`#`	Trim the value from start based on any character or delimeter

Important Points

- It is a good practice to follow variable naming conventions to keep the code simple and readable for other programmers
- Variables are a good way to store information which is reused many times in the program, for example you can store the directory path in a variable and then call it wherever you want instead of writing it again and again
- While assigning a value, there should not be any space between the assignment operator. Bash produces an error if you do so
- Sometimes the `$` sign is not enough, so we add curly brackets with `$` sign to indicate Bash the beginning and ending of variable name, like this: `${name}'s`
- It is a common practice to write Shell variables in lowercase and Environment variables in uppercase format, for example, `$PATH`
- If a variable is uninitialized, then a `null` value is stored in its memory location
- If there's a space between two string, Bash would consider it as two

seperate items, so always use Quotations (single or double) in order to

avoid this error. So, the correct syntax to initialize a string with spaces would be: `message="Hello John!"`