Batch file Functions

Packaging up code into a discrete functions, each with a clear purpose is a very common programming technique. Re-using known, tested code, means you can solve problems very quickly by just bolting together a few functions.

The CMD shell does not have any documented support for functions, but you can fake it by passing arguments/parameters to a subroutine and you can use SETLOCAL to control the visibility of variables.

At first glance building a function may look as simple as this:

```
:myfunct
SETLOCAL
SET _var1=%1
SET _var2="%_var1%--%_var1%--%_var1%"
SET _result=%_var2%
ENDLOCAL
```

but there is a problem, the ENDLOCAL command will throw away the `_result` variable and so the function returns nothing.

```
:myfunct2
SETLOCAL
SET _var1=%1
SET _var2="%_var1%--%_var1%--%_var1%"
ENDLOCAL
SET _result=%_var2%
```

This version is getting close, but it still fails to return a value, this time because ENDLOCAL will throw away the `_var2` variable

The solution to this is to take advantage of the fact that the CMD shell evaluates variables on a line-by-line basis - so placing ENDLOCAL **on the same line** as the SET statement(s) gives the result we want. This technique is known as 'tunneling' and works for both functions and entire batch scripts:

```
:myfunct3
SETLOCAL
SET _var1=%1
SET _var2="%_var1%--%_var1%--%_var1%"
ENDLOCAL & SET _result=%_var2%
```

In examples above there are just 2 local variables (_var1 and _var2) but in practice there could be far more, by turning the script into a function with SETLOCAL and ENDLOCAL we don't have to worry if any variable names will clash.

In other words you can do this:

```
@ECHO OFF
SET _var1=64
SET _var2=123
CALL :myfunct3 Testing
echo _var1 is %_var1%
echo Final result %_result%
goto :eof

:myfunct3
SETLOCAL
SET _var1=%1
SET _var2="%_var1%--%_var1%--%_var1%"
ENDLOCAL & SET _result=%_var2%
```

When working with functions it can be useful to use Filename Parameter Extensions against the function name, `%0` will contain the call label, `%nx0` the file name, see Rob Hubbards blog for an example. Note that if you have two scripts one calling another, this will not reveal the location of the 'calling' script.

"Cats are intended to teach us that not everything in nature has a function" ~ Garrison Keillor

Related:

CALL - Call one batch program from another
SETLOCAL - Control the visibility of environment variables
SHIFT - Shift the position of replaceable parameters in a batch file