Anfänger Tutorial - Funktionen

Vorwort:

Herzlich willkommen zu meinen Tutorial. Hier wirst du folgendes lernen:
- Wie man Funktionen deklariert und verwendet
- Wie man Parameter definiert
- Wofür die Schlüsselwörter Return, ByRef und Const gut sind

1. Funktionen deklarieren und aufrufen:

Damit wir auch Funktionen auch definieren können, benötigen wir 2 Befehle.
- Func | Damit wird angegeben, dass eine Funktion beginnt.
- EndFunc | Damit wird angegeben, dass eine Funktion beendet wird.

Zudem benötigen wir auch einen Funktionsnamen. Diesen können wir uns aussuchen. Allerdings gibt es ein paar Sachen zu beachten:
- Ein Funktionsname darf nicht genau so heißen, wie ein Befehl.
- Nach dem Funktionsname müssen Klammern gesetzt werden.
- Ein Funktionsname darf nur folgende Zeichen beinhalten:
  | A-Z | a-z | 0-9 | _ | <<< Der Trennstrich ist ebenfalls nicht zugelassen.

Hier mal ein paar Beispiele für mögliche Funktionsnamen:

Falsch: Exit()	Richtig: _Exit()	Der Funktionsname darf ja nicht gleich mit einem Befehl sein.
Falsch: Äpfel()	Richtig: Aepfel()	Das Zeichen „Ä“ ist nicht zugelassen.
Falsch: _Funk	Richtig: _Funk()	Hier wurden die Klammern vergessen.

Nun wollen wir auch eine Funktion deklarieren. Dazu müssen wir erst einmal den Befehl Func aufrufen. Dahinter kommt dann der Funktionsname, dann das Script was ausgeführt werden soll und zu guter Letzt der Befehl EndFunc damit wir alles abschließen können.

```
Func _Funktionsname()
   ;~ Script was ausgeführt werden soll.
EndFunc
```

Nun haben wir eine Funktion deklariert. Natürlich fehlt uns jetzt ein Script was gestartet wird, sobald die Funktion aufgerufen wird. Um dies ein wenig abzukürzen nehmen wir eine einfache MsgBox.

```
Func _Funktionsname()
   MsgBox(0, 'Funktion:', 'Hallo aus der Funktion!')
EndFunc
```

Jetzt wird es an der Zeit, dass wir unsere Funktion aufrufen. Diese können wir wie ein Befehl nach Belieben aufrufen. Damit das ganze besser zu verstehen ist, habe ich noch eine zweite MsgBox eingebaut.

```
MsgBox(0, 'Script:', 'Hallo aus dem Script!')
_Funktionsname()    ;~ Hier wird die Funktion aufgerufen.

Func _Funktionsname()
   MsgBox(0, 'Funktion:', 'Hallo aus der Funktion!')
EndFunc
```

2. Funktionen und deren Parameter:

Damit wir auch alles verstehen, werden wir in diesem Abschnitt den Befehl MsgBox nachbauen. Dies werden wir aber einfach halten und nur das nötigste schreiben. Wenn du nicht weißt was ein Parameter ist, wird es dir gleich klar werden.

Als erstes benötigen wir eine Funktion. Wir benutzen mal als Funktionsname _MsgBox. Als Script nehmen wir MsgBox(). Die Klammern lässt du vorerst leer.

```
Func _MsgBox()
   MsgBox()
EndFunc
```

Nun überlegen wir mal was wir für unsere MsgBox brauchen. 3 Pflichtangaben und 2 Optionale. Diese Angaben sind übrigens die Parameter. Wir benötigen also:
- Die Flag Eingabe
- Den Titel
- Und die Text Eingabe

Um die beiden anderen Optionalen Parameter kümmern wir uns später. Nun müssen wir erst einmal Parameter erstellen. Diese können wir einfach in die Klammern schreiben. Wichtig ist aber das diese Variable sind.

```
Func _MsgBox($Flag, $Titel, $Text)
   MsgBox()
EndFunc
```

Wichtig ist es auch zu wissen, dass die Parameter gleichzeitig Local in der Funktion definiert werden. Das bedeutet dass diese Variablen nur in der Funktion zur Verfügung stehen. Mit diesem Wissen können wir nun auch die entsprechenden Werte in unsere MsgBox einbauen.

```
Func _MsgBox($Flag, $Titel, $Text)
   MsgBox($Flag, $Titel, $Text)
EndFunc
```

Nun können wir wieder unsere Funktion aufrufen, nur dass wir Parameter angeben müssen.

```
_MsgBox(0, 'Funktion:', 'Hallo aus der Funktion!')

Func _MsgBox($Flag, $Titel, $Text)
   MsgBox($Flag, $Titel, $Text)
EndFunc
```

Wir haben eine Funktion geschrieben, die 3 Pflichteingaben enthält. Aber wie können wir noch 2 Optionen hinein bauen? Dafür geben wir der Variable direkt beim Deklarieren einen bestimmten Wert. Dieser wird dann immer verwendet, wenn wir den Parameter nicht angeben. Bevor wir aber überhaupt etwas machen, müssen wir und überlegen welche 2 Optionalen Angaben wir noch brauchen und welchen Wert diese zugeteilt bekommen.
- Wir benötigen noch den Parameter Timeout. Der Standard ist 0, wenn dieser Parameter nicht angegeben wird. Also bekommt unser Script ebenfalls für diesen Parameter den Wert 0.
- Dann fehlt uns noch das Fensterhandle, kurz: Hwnd. Dieser Parameter hat keinen Standartwert. Zu mindestens wird davon nichts in der Hilfedatei erwähnt. Daher bekommt dieser Parameter einen Leerstring, sprich: ''.

```
Func _MsgBox($Flag, $Titel, $Text, $Timeout = 0, $Hwnd = '')
   MsgBox($Flag, $Titel, $Text, $Timeout, $Hwnd)
EndFunc
```

Wir rufen die Funktion 2x auf. Einmal mit dem Parameter $Timeout und einmal ohne. Schließlich wollen wir ja auch sehen ob die Parameter so funktionieren wie wir es wollen.

```
_MsgBox(0, 'Funktion:', 'Hallo aus der Funktion! #1', 2)
_MsgBox(0, 'Funktion:', 'Hallo aus der Funktion! #2')
   ;~ Die erste MsgBox schließt automatisch nach 2 Sekunden.

Func _MsgBox($Flag, $Titel, $Text, $Timeout = 0, $Hwnd = '')
   MsgBox($Flag, $Titel, $Text, $Timeout, $Hwnd)
EndFunc
```

3. Return, ByRef und Const:

Return:

Normalerweise ist es so, dass eine Funktion 0 zurückgibt. Aber mit dem Befehl Return kann dieser Wert manuell gesetzt werden. Zudem ist es auch möglich, Zeichenketten als Return Wert zu nehmen.

Hier ein einfaches Beispiel von einer Funktion, die keinen Return Befehl besitzt:

```
$Ausgabe  =  _Funktion()
MsgBox(0, 'Ausgabe:', $Ausgabe)

Func _Funktion()
   $x = 5
EndFunc
```

Jetzt könnte man meinen, dass 5 ausgegeben wird. Aber leider ist dies falsch. Damit 5 wirklich ausgegeben werden kann, gibt es 2 Möglichkeiten.

1. Man gibt $x zurück, also: Return $x
2. Oder man gibt direkt die 5 zurück, und löscht $x = 5, also: Return 5

Hier mal das Beispiel für einen Return:

```
$Ausgabe  =  _Funktion()
MsgBox(0, 'Ausgabe:', $Ausgabe)

Func _Funktion()
   Return 5
EndFunc
```

Statt dem Wert 5 kann auch eine Zeichenkette verwendet werden. Z.B. 'Ausgabe erfolgt!'

```
$Ausgabe  =  _Funktion()
MsgBox(0, 'Ausgabe:', $Ausgabe)

Func _Funktion()
   Return 'Ausgabe erfolgt!'
EndFunc
```

Und hier noch ein Beispiel, um eine Variable als Return Wert zu nehmen:

```
$Ausgabe  =  _Funktion()
MsgBox(0, 'Ausgabe:', $Ausgabe)

Func _Funktion()
   $x = 10
   Return $x
EndFunc
```

ByRef:

Weil es in der Hilfe schon so schön steht, habe ich mir die Beschreibung einfach mal heraus kopiert:

```
Das Schlüsselwort ByRef zeigt an, dass der Parameter als Referenz zu dem
Originalobjekt behandelt werden soll. Standardmäßig wird der Parameter in
eine neue Variable kopiert. Jedoch verknüpft ByRef die neue Variable auf den
Originalparameter. [...] Ein Buchstabe kann jedoch nicht als ByRef Parameter
übergeben werden.
```

Um das zu verstehen, habe ich auch schon ein Beispiel für dich geschrieben. Viel kann ich zu ByRef nicht sagen, allerhöchstens erklären wie es verwendet wird.

Hier mal ein einfaches Beispiel einer Addition:

```
$a = 2
$b = 7
_Summe($a, $b)
MsgBox(0, 'Ausgabe:', $a)

Func _Summe(ByRef $x, $y)
   $x = $x + $y
EndFunc
```

Hier die Erklärung:
Sobald wir die Funktion _Summe aufrufen, Wird die Variable $x mit $x + $y umgeschrieben. Da wir als Parameter die Variable $a und $b verwendet haben, sieht das ganze so aus:
$x = 2 ($x also) + 7 ($y also)

Das bedeutet das $x den Wert 9 enthält. Da wir als Parameter ByRef $x angegeben haben, und unser Parameter Wert $a ist, wird die Variable $x in Variable $a geschrieben. Das bedeutet das die Variable $a ab sofort 9 beinhaltet.

Nun wird das ganze ausgegeben und fertig ist unsere Addition. Dies funktioniert aber auch mit einer Subtraktion. Ersetz doch mal das Plus durch ein Minus.

Const:
Das ist eigentlich schnell erklärt. Mit dem Schlüsselwort Const, wird angegeben dass der Parameter nicht veränderbar ist. Das bedeutet das der Parameter nicht verändert werden kann/darf.

Hier ein Beispiel Script, dieses gibt allerdings einen Fehler aus:

```
$Ausgabe = _Funktion(10)
MsgBox(0, 'Ausgabe:', $Ausgabe)

Func _Funktion(Const $Wert)
   $Wert += 1
   Return $Wert
EndFunc
```

Schlusswort:
Wenn es Fragen dazu gibt, oder Rechtschreibfehler / Scriptfehler gefunden wurden, so schreibe mir eine E-Mail:
cederik.althaus@googlemail.com

Mfg. Cederik