Kostenloses eBook

LERNEN Visual Basic 6

Free unaffiliated eBook created from
Stack Overflow contributors.

#vb6

Inhaltsverzeichnis

Über

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: visual-basic-6

It is an unofficial and free Visual Basic 6 ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official Visual Basic 6.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Kapitel 1: Erste Schritte mit Visual Basic 6

Bemerkungen

In diesem Abschnitt erhalten Sie einen Überblick darüber, was vb6 ist und warum ein Entwickler es verwenden möchte.

Es sollte auch alle großen Themen in vb6 erwähnen und auf die verwandten Themen verweisen. Da die Dokumentation für vb6 neu ist, müssen Sie möglicherweise erste Versionen dieser verwandten Themen erstellen.

Examples

Installation oder Setup

Detaillierte Anweisungen zum Einrichten oder Installieren von vb6.

Hallo Welt

```
' A "Hello, World!" program in Visual Basic.
Module Hello
  Sub Main()
      MsgBox("Hello, World!") ' Display message on computer screen.
  End Sub
End Module
```

Erste Schritte mit Visual Basic 6 online lesen: https://riptutorial.com/de/vb6/topic/3439/erste-schritte-mit-visual-basic-6

Kapitel 2: Funktionsverfahren

Einführung

Funktion ist eine Reihe von Anweisungen, die von den Anweisungen "Function" und "End Function" eingeschlossen werden.

Die Funktion führt eine Aktivität aus und gibt die Kontrolle an den Anrufer zurück. Wenn es die Kontrolle zurückgibt, gibt es auch einen Wert an den aufrufenden Code zurück.

Sie können eine Funktion in einer Klasse, Struktur und einem Modul definieren. Standardmäßig ist es öffentlich. Das bedeutet, Sie können es von überall in Ihrer Anwendung aufrufen, die Zugriff auf die Klasse, Struktur oder das Modul hat, in dem Sie es definiert haben.

Syntax

- [Modifiers] Funktionsname_Of_The_Function [(Arg_List)] As Return_Type
- [Statements]
- Funktion beenden

Bemerkungen

- Die zwei in diesen Beispielen verwendeten Funktionsmodifizierer sind Public und Private. Diese Modifikatoren definieren den Umfang der Funktion.
- Funktionen mit einem privaten Bereich können nur von der Quelldatei aus aufgerufen werden, von der sie definiert wurden. In unserem Fall kann mit im Modul aufgerufen werden. Und kann nicht außerhalb des Moduls aufgerufen werden.
- Funktionen mit öffentlichem Umfang können sowohl außerhalb als auch innerhalb des Moduls aufgerufen werden. Wir können einfach sagen: "Wir können es überall im Programm anrufen".
- Der Standard-Modifizierer der Funktion ist Public.
- Standardmäßig werden die Funktionsargumente als Referenz übergeben (In einem separaten Thema wird dies ausführlich erläutert).

Examples

Funktion erstellen & aufrufen

Dieses Beispiel mit Standard-EXE-Projekt Mit dem Hinzufügen einer Moduldatei.

- Erstellen Sie ein neues "Standard EXE" -Projekt. Hier wird ein Formular standardmäßig zum Projekt hinzugefügt.
- Fügen Sie dem Projekt eine Moduldatei hinzu
- Platzieren Sie eine Befehlsschaltfläche in dem Formular

- Befehlsschaltfläche erstellen Klicken Sie auf Ereignis.

  Modulcode

Zwei Funktionen im Modul erstellt. Eine ist eine öffentliche Funktion (FnAdd). Es sind zwei Integer-Argumente val_1 und val_2 erforderlich. Es wird eine Ganzzahl zurückgegeben. Diese Funktion fügt die beiden Argumente hinzu und gibt den Wert an den Aufrufer zurück. Vor dem Hinzufügen werden die beiden Argumente in einer anderen Funktion einem Prozess unterzogen. Welches ist eine private Funktion. Merkmale / Regeln der Funktion Public & Private, die im Abschnitt "Bemerkungen" erläutert werden.

```
Public Function FnAdd(val_1 As Integer, val_2 As Integer) As Integer

'Calling private function
val_1 = FnMultiplyBy5(val_1)

'Calling private function
val_2 = FnMultiplyBy5(val_2)

'Function return statement
FnAdd = val_1 + val_2

End Function
```

Unten ist die Private-Funktion im Modul. Es dauert ein ganzzahliges Argument val. Es wird eine ganze Zahl zurückgegeben. Diese Funktion multipliziert einen Wert 5 mit dem Argument und gibt das Ergebnis an den Aufrufer zurück.

```
Private Function FnMultiplyBy5(Val As Integer) As Integer

'Function return statement
FnMultiplyBy5 = Val * 5

End Function
```

  Formularcode

Klicken Sie in der Befehlsschaltfläche auf Ereignis. Hier nennen wir die Module Public-Funktion "FnAdd"

```
Private Sub Command1_Click()
Debug.Print FnAdd(3, 7)
End Sub
```

  Ergebnis im Direktfenster

```
50
```

Funktionsverfahren online lesen: https://riptutorial.com/de/vb6/topic/9227/funktionsverfahren

Kapitel 3: Grundlegende Syntax

Examples

if / else-Anweisung

```
If condition Then
    code to execute if true
ElseIf condition Then
    code
Else
    code to execute if conditions are both false
End If
```

für Schleife

```
For I as Integer = 1 To 10 Step 1
    code to execute
Next
```

Schritt ist optional und Schritt 1 ist die Standardeinstellung. In Schritt wird angegeben, wie gezählt werden soll. Mit -1 wird also jedes Mal 1 abgezogen, in Schritt 5 wird jedes Mal 5 durch die Schleife addiert.

Für den Fall, dass die Schleife gestoppt werden muss, kann die `Exit For` Anweisung wie im folgenden Beispiel verwendet werden.

```
Dim iIndex as integer

For I as Integer = 1 To 10 Step 1

Debug.Print  I
iIndex = I * 10

If iIndex > 90 Then
    Exit For
End If

Loop
```

Statt 1 bis 10 zu drucken, wird hier bei 9 angehalten, da die Bedingung dem Prozess mitteilt, zu stoppen, wenn iIndex 90 erreicht.

Machen Sie eine Schleife

Eine andere häufige Art von Schleife in Visual Basic ist die `DO loop` , die ein Stück Code kontinuierlich ausführen würde, bis es aufgefordert wird, anzuhalten. Im Gegensatz zu einigen anderen Schleifen, bei denen Indizes verwendet werden, um den Prozess zu stoppen, sollte in dieser speziellen Schleife angehalten werden, anzuhalten.

Ein einfaches Beispiel zur Veranschaulichung der Schleife ist wie folgt

```
Dim iIndex1 As Integer
iIndex1 = 1

Do
   Debug.Print iIndex1
   iIndex1 = iIndex1 + 1

   If iIndex1 = 10 Then
      Exit Do
   End If
Loop
```

Der obige Code übernimmt einen Index, der auf 1 initialisiert wird, und erhöht ihn. Ein `Debug.Print` hilft dabei, den Index zu drucken, um die Schleife `Debug.Print` . In jeder Schleife überprüft der Code, ob der Index 10 erreicht hat, und wenn und nur wenn die Bedingung erfüllt ist, wird `Exit Do` ausgeführt, wodurch die Schleife gestoppt wird.

Wählen Sie Case Statement aus

```
    Dim number As Integer = 8
    Select Case number
        Case 1 To 5
            Debug.WriteLine("Between 1 and 5, inclusive")
            ' The following is the only Case clause that evaluates to True.
        Case 6, 7, 8
            Debug.WriteLine("Between 6 and 8, inclusive")
        Case 9 To 10
            Debug.WriteLine("Equal to 9 or 10")
        Case Else
            Debug.WriteLine("Not between 1 and 10, inclusive")
    End Select
```

Grundlegende Syntax online lesen: https://riptutorial.com/de/vb6/topic/9389/grundlegende-syntax

Kapitel 4: Variablen

Examples

Variablentypen

Es gibt verschiedene Variablentypen für verschiedene Zwecke. In Visual Basic 6 stehen folgende Variablentypen zur Verfügung:

- Array
- Boolean
- Byte
- Währung
- Datum
- Doppelt
- Ganze Zahl
- Lange
- Single
- String
- Variante

Sie deklarieren eine Variable mit dem Schlüsselwort `Dim` :

```
Dim RandomNumber As Integer
```

Wenn Sie keinen Variablentyp angeben, wird die Variable standardmäßig auf `Variant` :

```
Dim Foo
```

ist äquivalent zu

```
Dim Foo As Variant
```

Boolean

Boolean ist der einfachste Variablentyp, da er nur einen von zwei Werten enthalten kann: True oder False.

```
Foo = True
Bar = False
```

Booleans können verwendet werden, um den Code-Fluss zu steuern:

```
Dim Foo as Boolean
Foo = True
```

```
If Foo Then
  MsgBox "True"
Else
  MsgBox "False"
End If
```

Ganze Zahl

Eine Ganzzahl ist ein numerischer Datentyp und kann einen vorzeichenbehafteten 16-Bit-Wert (-32768 bis +32767) enthalten. Wenn Sie wissen, dass eine Variable nur ganze Zahlen (z. B. 9) und keine gebrochenen Zahlen (z. B. 5,43) enthält, müssen Sie sie als Ganzzahl- (oder Lang-) Datentyp definieren.

```
Dim RandomNumber As Integer
RandomNumber = 9
```

Ganzzahlen werden normalerweise als Zähler in `For...Next` Schleifen verwendet:

```
Dim Counter As Integer

For Counter = 0 to 2
  MsgBox Counter
Next Counter
```

Wenn Sie versuchen, einer Ganzzahl einen Wert unter -32768 oder größer als 32767 zuzuweisen, führt dies zu einem Laufzeitfehler:

```
Dim MyNumber As Integer
MyNumber = 40000  'Run-time error '6': Overflow
```

String

Eine Zeichenfolgenvariable kann einen leeren Text, ein Zeichen, ein Wort oder einen Text mit variabler Länge enthalten. Der Zeichenfolgewert muss in Anführungszeichen (`"`) stehen.

```
Dim Fruit as String
Fruit = "Banana"
```

Wenn Sie Anführungszeichen in einem String-Literal benötigen, verwenden Sie zwei aufeinanderfolgende Anführungszeichen (`""`).

```
Dim Quote as String
Quote = "Bill says: ""Learn VB!"""
```

Variablen online lesen: https://riptutorial.com/de/vb6/topic/7511/variablen

Kapitel 5: VB6 unter Windows 10 installieren

Examples

Installationsassistent

Installationsassistent für Visual Studio 6.0

Standardmäßig werden die folgenden Pakete unter Windows 10 nicht ordnungsgemäß installiert:

- Visual Studio 6 Enterprise
- Visual Studio 6 Professional
- Visual Basic 6 Enterprise
- Visual Basic 6 Professional

Um die obigen Pakete zu installieren, müssen Sie entweder zahlreiche Anpassungen und Registrierungs-Hacks vornehmen oder den fantastischen Visual Studio 6.0-Installationsassistenten von Giorgio Brausi verwenden.

Sie benötigen die folgenden Elemente, bevor Sie beginnen:

- Ihre ursprünglichen Visual Studio / Basic-Programm-CDs und -Tasten
- Ihre ursprünglichen MSDN-CDs
- Visual Studio Service Pack 6
- Installationsassistent für Visual Studio 6.0
- Unter Windows 10 Build 1511 oder höher benötigen Sie Administratorrechte.

Der Assistent führt Sie durch die notwendigen Schritte für eine erfolgreiche Installation von Visual Basic 6.

Beachten Sie, dass die Installation der Serveranwendungen derzeit nicht möglich ist.

VB6 unter Windows 10 installieren online lesen: https://riptutorial.com/de/vb6/topic/4230/vb6-unter-windows-10-installieren

Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit Visual Basic 6	Community, Scheffer
2	Funktionsverfahren	Jeet
3	Grundlegende Syntax	Nadeem_MK, Talal Abdoh, user7491506
4	Variablen	BlueEel
5	VB6 unter Windows 10 installieren	ThunderFrame