Basic For Qt® Syntax

Contents

First Edition

This edition applies to release to the latest release of Basic For Qt® and to all subsequent released and modfications until otherwise indicated in new editions. Make sure you are using the correct edtion for the level of the product. The term „Basic For Qt®“ as used in this publication, refers to the Basic For Qt® product set.

Conventions Used In This Book / Way Of Writing

normal text appears in writing Arial. Here is an example here: This is normal text

Syntax and source code code appear in writing Courier New. Here the example:

```
Dim i As Integer
```

Important references and keywords are italically deposited: *Arguments*

Basic For Qt®-Syntax

The syntax of sub, function or statement in the Basic For Qt® help entry shows all elements, which are needed to correctly use the sub, function or statement. How you can understand those information shows the following lines.

Example: One Syntax of the MsgBox-Function

MsgBox(prompt[, buttons] [, title])

Arguments, which are inside of [ ], are optional. (Do not write these [ ] in your Basic For Qt® code). The only argument, what you have give the MsgBox-Function is the one for the showing the text: ‘prompt’.

Arguments for functions or subs can be used with the help of their position or their name. In order to use the arguments defined with their position, you do not have to ignore the position written in the syntax. You must write them exactly in the same order they occur in the syntax. All arguments must be separated by a comma. Example:

```
MsgBox("The answer is right!", 0, "window with answer")
```

If you would like to use a argument with its name, use the name of the argument and colon and equals sign (:= and the value of the argument. You can write these named arguments in any order you wish. Example:

```
MsgBox(title:="window with answer", prompt:="The answer is right!")
```

Some arguments are written inside of {} in the syntax of functions or subs.

MessageBox(Icon As Integer, Title As String, Text As String, InformativeText As String,

DetailedText As String, {StandardButton As Integer | List(Text As String, Role As Integer), ...})

In the syntax of the MessageBox command: { } together with | means that one of the elements must be written.(Do not write these { } in your Basic For Qt® code).

Syntax of the 'Dim'-Statement

Dim VarName[ARRAY] [As DataType] [, VarName[ARRAY] [As DataType]] …

'Dim' is a keyword in the syntax of the 'Dim'-Statement. The only needed element is VarName (the name of the variable). The following statement creates three variables: myVar, nextVar and thirdVar. These variables are declared as 'Object'-variables automatically.

```
Dim myVar, nextVar, thirdVar
```

The following example declares a variable of type 'String'.

```
Dim myAns As String
```

If you want to declare many variables in one line, you should declare every datatype of each variable explicitly. Variables without declared datatype get the default datatype, which is 'Object'.

```
Dim x As Integer, y As Integer, z As Integer
```

X and y get in the datatype 'Object' in the following statement. Only z has the 'Integer' datatype.

```
Dim x, y, z As Integer
```

You have to put [ ] (new style), if you want to declare an array variable. The indexes of the array are optional. The following statement declares a dynamic array named myArray.

```
Dim myArray[]
```

Variable

Declaration

Dim

Public

Private

Static

As

```
Dim sName As String

Public sName As String

Private sName As String



Dim Name[ARRAY] [As Type] [, Name[ARRAY] [As Type]] ...
```

```
Dim Name [= Expression]

Dim Name [As Type] [= Expression]

[Public | Private | Dim | Static] Name [= Expression] [As Type]
```

Assignment

```
Dim yourName As String
yourName = InputBox("What is your name?")
MsgBox "Your Name is " & yourName
```

User Defined Type

Type

```
Type Name
 Name [ARRAY] As Type
   ...
End Type
```

Comment

'

```
' this is a comment
```

Literal

Byte, Short, Integer

```
1, 2, -44, 4453, +78
```

Hex

```
&HAA43
```

Binary

```
&B11110001
```

Single/Double/Float

```
21.32, 0.344, -435.235421.21, +67.8
```

Decimal

```
45.3@
```

DateTime

```
There is no literal. Use the proper date function instead.
```

String

```
"hello"
```

Boolean

```
True, False
```

Constant

Const

As

```
Const Border As Integer = 377
```

```
Const Name = Expression
```

```
Const Name [As Type] = Expression [, Name [As Type] = Expression] ...
```

```
[Public | Private] Const Name [As Type] = Expression
```

Working With Objects

Create Class

```
Variables / Constants / Properties / Types / Enumerations
Functions
Subs
...
```

Access Class Variable And Instance Variable

```
classname.classVariable
objectname.instanceVariable
```

Access Class Method Or Intance Method (Function Or Sub)

```
objectname.instanceVariable = 99
```

Class Method Or Intance Method

```
Sub myInstanceMethod
...
End Sub
```

Access Class Type

```
objectname.typefield
```

Access Class Enum

```
objectname.enumfield
```

Access Class Property

```
objectname.classproperty
```

Call Method

```
objectname.myMethod()
```

Current Instance Of Object

Me

Scope modifier

Private

Public

Array

Dim

```
Dim variableName[Index] As Type

Dim variableName[Index, Index, ...] As Type

Dim variableName[Index To Index] As Type

Dim variableName[Index To Index, Index To Index, ...] As Type
```

Access Array

```
i[3] = 10
o[3, 88] = 10
```

Lower And Upper Bound Of Array

UBound

LBound

```
UBound (arrayVariable[,(Dimension])

LBound (arrayVariable[,(Dimension])
```

Multi-Dimension

```
Dim i(100, 50, 400)

Dim sngMulti(1 To 5, 1 To 10) As Single
```

Dynamic Array

```
Dim a[] As Integer

Redim

Redim variableName[Index]

Redim variableName[Index, Index, ...]

Redim variableName[Index To Index]

Redim variableName[Index To Index, Index To Index, ...]
```

Flow Control - Decision

Single Decision

If

Then

Else

End If

```
If Expression Then Statement

If Expression Then Statement : Else Statement

If Expression Then LineNo

If Expression Then LabelName:

If Expression Then
  [Statements]
```

```
End If

If Expression Then
  [Statements]
Else
  [Statements]
End If

If Expression Then
  [Statements]
Else If Expression
  [Statements]
Else
  [Statements]
End If

If Expression Then
  [Statements]
Else If Expression
  [Statements]
Else If Expression
  [Statements]
Else
  [Statements]
End If

If Expression Then
  [Statements]
Else If Expression
  [Statements]
End If
```

IIf – Short If

```
IIf(Expression, ThenReturnExpression, ElseReturnExpression)
```

Multi Decision

Select

Select Case

Case

End Select

```
Select Expression ' modern style
Case Expression
  [Statements]
Case Expression
  [Statements]
End Select

Select Case Expression
Case Expression
  [Statements]
Case Expression
```

```
  [Statements]
End Select

Select Case Expression
Case Expression
  [Statements]
Case Expression To Expression
  [Statements]
Case Is Expression
  [Statements]
Case Else
  [Statements]
End Select
```

Uncoditional Jump

GoTo

```
GoTo label:

GoTo myExit:
GoTo nextStep:
```

Flow Control - Loop

For Next

To

Step

```
For variable = beginExpr To endExpr [Step Expression]
  [Statements]
Next [variable]
```

Do While ... Loop

```
Do While Expression
  [Statements]
Loop
```

Do ... Loop Until

```
Do
  [Statements]
Loop Until Expression
```

Do ... Loop While

```
Do
  [Statements]
Loop While Expression
```

Do Until ... Loop

```
Do Until Expression
  [Statements]
Loop
```

While ... End While

```
While Expression
  [Statements]
End While
```

Explicit Leave Of Loop

```
Exit For
Exit Do
Break (new style)
```

Explicit Test of Loop Condition

```
Iterate For
Iterate Do
Continue (new style)
```

Subs / Procedures

Sub-Procedure

Sub

End Sub

```
Sub Name([Argumente])
  [Statements]
End Sub

Sub Name([Argumente])
  [Statements]
End Sub
```

Function-Procedure

Function

End Function

```
Function Name([Argumente]) [As Type]
  [Statements]
End Function

Function Name([Argumente]) [As Type]
  [Statements]
End Function
```

Argument

```
Name As Type

[ByVal | ByRef] Name As Type

[ByVal | ByRef] Name [As Type]

[ByVal | ByRef] Name [[]][As Type]
```

Call Of Sub or Function

```
Sub Main()
   MultiBeep(56)
   Meldung()
End Sub

Sub MultiBeep(Anzahl)
        For n As Integer = 1 To Anzahl
                Beep
        Next n
End Sub

Sub Meldung()
        MsgBox "Zeit für eine Pause!"
End Sub
```

Explicit Leave Of Procedures

```
Exit Sub
Exit Function
Return (new style)
```

Functions

Function

End Function

```
Function Name([Argumente])[As Type]
  [Statements]
End Function

Function Name([Argumente])[As Type]
  [Statements]
End Function
```

Return Function Value

```
Return Expression
```

Return Expression

```
FunctionName = Expression
```

Property

Access Property

```
varname.classproperty = 99
Print varname.classproperty
```

Property

```
Property Function Get_Name(Argument)
  [Statements]
End  Function

Property Sub Set_Name(Argument)
  [Statements]
End  Sub
```

User defined Type

Access Type

```
Type
  varname.typefield = 99
End Type

Type Name
 Name [ARRAY] As Type
  ...
End Type
```

Enumeration

Access Enum

```
varname.enumfield = 99


Enum Name
  Name [= Expression]
  ...
End Enum
```

Module

Access Module Variable

```
modulename.moduleVariable
moduleVariable
```

Access Module Sub Or Function

```
modulename.moduleSub(99)
```

Module Sub Or Function

```
Sub myModuleSub
...
End Sub

Function myModuleFunction
...
End Function
```

Call Module Sub Or Function

```
modulename.myModuleSub()
```

Access Module Type

```
modulename.typefield
```

Access Module Enum

```
modulename.enumfield
```