

Objective-Basic
Syntax

First Edition

This edition applies to release 1.0 of Objective-Basic and to all subsequent releases and modifications until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product. The term „Objective-Basic“ as used in this publication, refers to the Objective-Basic product set (September 2006).

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 and grey background.
Here the example:

```
Dim I As Integer
```

Important references and keywords are italicly deposited:
Arguments

Objective-Basic Syntax

Table of Contents

First Edition.....	2
Conventions Used In This Book / Way Of Writing.....	2
Objective-Basic-Syntax.....	6
Variable.....	8
Declaration.....	8
Dim.....	8
Public.....	8
Private.....	8
Protected.....	8
Static.....	8
AsRef.....	8
AsConstRef.....	8
As	
.....	8
Assignment.....	9
User Defined Type.....	9
Type.....	9
Comment.....	9
//.....	9
'.....	9
/*.....	9
*/.....	9
/**.....	9
Literal.....	10
Byte, Short, Integer, Long.....	10
Hex.....	10
Binary.....	10
Single (Decimal).....	10
Double (Decimal).....	10
String.....	10
Boolean.....	10
Constant.....	11
Const.....	11
Working With Objects.....	12
Create Object.....	12
Create Class.....	12
Access Class Variable And Instance Variable.....	12
Access Class Method Or Intance Method (Function Or Sub).....	12
Class Method Or Intance Method.....	12
Access Class Type.....	12

Objective-Basic Syntax

Access Class Enum.....	13
Call Method.....	13
Current Instance Of Object.....	13
Me.....	13
Parent.....	13
Hidden Variable.....	13
Hidden Method (Sub Or Function).....	13
Scope modifier.....	13
Private.....	13
Protected.....	13
Public.....	13
Array.....	14
Dim.....	14
Access Array	14
Flow Control - Decision.....	15
Single Decision.....	15
If.....	15
Then.....	15
Else.....	15
End If.....	15
Multi Decision.....	16
Select Case.....	16
Case.....	16
Like.....	16
Case Else.....	16
End Select.....	16
Uncoditional Jump.....	16
GoTo.....	16
With.....	16
Flow Control - Loop.....	18
For Next	
For Each.....	18
To.....	18
Step.....	18
Do While ... Loop.....	18
Do ... Loop While.....	18
Explicit Leave Of Loop.....	19
Explicit Test of Loop Condition.....	19
Subs / Procedures.....	20
Sub-Procedure	20
Sub.....	20
End Sub.....	20
Function-Procedure	20
Function.....	20
End Function.....	20

Objective-Basic Syntax

Argument.....	20
Named Argument.....	21
Call Of Sub or Function.....	21
Explicit Leave Of Procedures.....	21
Function.....	21
Function.....	21
End Function.....	21
Return.....	22
User defined Type.....	22
Access Type.....	22
Type.....	22
End Type.....	22
Enumeration.....	22
Access Enum.....	22
Enum.....	22
End Enum.....	22
Class.....	23
Class.....	23
Inherits.....	23
Sub.....	23
Function.....	23
End Class.....	23
Error Handling.....	24
New Exception.....	24
Throw.....	24
Exception.....	24
Try.....	24
Catch.....	24
End Catch.....	24
(C)Copyright KBasic Software 2007.....	25

Objective-Basic-Syntax

The syntax of sub, function or statement 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: Syntax of myMsgBox-Function

```
myMsgBox(prompt[, buttons] [, title] [, helpfile])
```

Arguments, which are inside of [], are optional. (Do not write these [] in your Objective-Basic 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:

```
myMsgBox("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:

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

Syntax of the 'Dim'-Statement

```
Dim VarName[([Indexes])] [As Type] [, VarName[([Indexes])] [As Type]] ...
```

'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 'Variant'-variables automatically (or 'Double' in 'VeryOldBasic Mode').

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

The following example declares a variable of type 'String'. If you declared the datatype of the variable explicitly, it will help KBasic to optimize the RAM-usage and will help you to find errors in your code.

```
Dim myAns As String
```

Objective-Basic Syntax

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 'Variant'.

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

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

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

You have to put [], 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

Protected

Static

AsRef

AsConstRef

As

```
Dim sName As String
```

```
Public sName As String
```

```
Private sName As String
```

```
Protected sName As String
```

```
Dim sName AsRef String
```

```
Public sName AsRef String
```

```
Private sName AsConstRef String
```

```
Protected sName AsConstRef String
```

```
Dim Name([Index]) [As Type] [, Name([Index]) [As Type]] ...
```

```
Dim Name [As Type] [= Expression]
```

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

Assignment

```
Dim yourName As String
yourName = myInputBox("What is your name?")
```

User Defined Type

Type

```
Type Name
  Name [(Index)] As Type
  ...
End Type
```

Comment

```
//
```

```
,
```

```
/*
```

```
*/
```

```
/**
```

```
// this is a comment
```

```
` this is a comment as well
```

```
/* start comment and stop comment */
```

```
/** start documentation comment and stop documentation comment */
```

Literal

Byte, Short, Integer, Long

1, 2, -44, 4453, +78

Hex

&HAA43

Binary

&B11110001

Single (Decimal)

21.32, 0.344, -435.235421.21, +67.8

Double (Decimal)

212.23

String

„hello“ (is Unicode or C-String, depending on context)

@„hello“ (is always Unicode)

Boolean

True, False

Constant

Const

```
Const Border = 377
```

```
Const Name = Expression
```

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

```
[Public | Protected | Private] Const Name = Expression
```

Working With Objects

Create Object

```
objectVariable = ClassName.alloc().init()
```

Please read the Cocoa documentation and the Objective-Basic Manual about creating objects.

Create Class

```
Class oak Inherits tree
```

- Variables / Constants / Types / Enumerations
- Functions
- Subs

```
End Class
```

Access Class Variable And Instance Variable

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

Access Class Method Or Instance Method (Function Or Sub)

```
objectname.instanceVariable = 99
```

Class Method Or Instance Method

```
Static Sub myClassMethod()  
...  
End Sub  
  
Sub myInstanceMethod  
...  
End Sub
```

Access Class Type

```
objectname.typefield
```

Access Class Enum

```
objectname.enumfield
```

Call Method

```
objectname.myMethod()
```

Current Instance Of Object

Me

Parent

Hidden Variable

```
Parent.myVariable ' access parent class  
myVariable       ' access current class (me)
```

Hidden Method (Sub Or Function)

```
Parent.myMethod() ' access parent class  
myMethod          ' access current class (me)
```

Scope modifier

Private

Protected

Public

```
Class plane  
  
    Private wings As Integer  
    Protected wings2 As Integer  
  
    Private Function countWings()  
        ...  
    End Function  
  
End Class
```

Array

Dim

```
Dim variableName[Index] As Type  
Dim variableName[Index, Index, ...] As Type ' VB6 style  
Dim variableName[Index][Index][...] As Type ' the recommend style
```

Access Array

```
i[3] = 10  
O[3, 88] = 10 ' VB6 style  
O[3][88] = 10 ' the recommend style
```

Flow Control - Decision

Single Decision

If

Then

Else

End If

```
If Expression Then Statement

If Expression Then Statement : Else Statement

If Expression Then LabelName:

If Expression Then
  [Statements]
End If

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

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

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

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

Multi Decision

Select Case

Case

Like

Case Else

End Select

```
Select Case Expression
Case Expression
    [Statements]
Case Expression
    [Statements]
Like Expression
    [Statements]
Like Expression
    [Statements]
Case Else
    [Statements]
End Select
```

Unconditional Jump

GoTo

```
GoTo {label:}

GoTo myExit:
GoTo nextStep:
```

With

```
Sub FormatOrder ()
    With myclass.
        .Value = 30
        .Font.Bold = True
    End With
End Sub
```

```
Sub setValue ()
    With j(3)
        .e.bkname = "Frankfurter Zoo"
    End With
End Sub
```

Objective-Basic Syntax

```
With .e
    .isbn ( 99 ) = 333
End With
End With
End Sub
```

Flow Control - Loop

For Next

For Each

To

Step

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

Dim i As id
Dim a As Array

For Each i In a
  [Statements]
Next

For Each variable As Type In a
  [Statements]
Next

For Each variable AsRef Type In a
  [Statements]
Next
```

Do While ... Loop

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

Do ... Loop While

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

Explicit Leave Of Loop

`Exit`

Explicit Test of Loop Condition

`Iterate`

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 [Alias 2ndName]
```

```
Name AsRef Type [Alias 2ndName]
```

```
Name AsConstRef Type [Alias 2ndName]
```

Named Argument

```
Sub PassArg(strName As String [Alias strName2], intAlter As Integer  
[Alias intAlter2])  
    Print strName, intAlter  
End Sub
```

```
PassArg(Frank", 26)
```

```
PassArg(intAlter:=26, strName:="Frank")
```

Call Of Sub or Function

```
Sub Main()  
    myMultiBeep(56)  
    mydo()  
End Sub  
  
Sub myMultiBeep(Anzahl)  
    For n As Integer = 1 To Anzahl  
        Next n  
End Sub  
  
Sub mydo()  
    myMsgBox("Zeit für eine Pause!")  
End Sub
```

Explicit Leave Of Procedures

```
Exit Sub
```

```
Exit Function
```

Function

Function

End Function

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

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

Objective-Basic Syntax

```
[Statements]  
End Function
```

Return

```
Return Expression
```

User defined Type

Access Type

```
varname.typefield = 99
```

Type

End Type

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

Enumeration

Access Enum

```
varname.enumfield = 99
```

Enum

End Enum

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

Class

Class

Inherits

Sub

Function

End Class

```
Class Name Inherits ParentClassName

  [Static] Dim Name As Type
  [Static] Public Name As Type
  [Static] Protected Name As Type
  [Static] Private Name As Type
  Const Name = Expression
  Public Const Name As Type
  Protected Const Name As Type
  Private Const Name As Type
  ...

  [Public | Protected | Private]
  Enum Name
    Name As Type
    ...
  End Enum
  ...

  [Public | Protected | Private]
  Type Name
    Name As Type
    ...
  End Type
  ...

  [Static] [Public | Protected | Private]
  Function Name([Arguments]) [As Type]
    [Statements]
  End Function
  ...

  [Static] [Public | Protected | Private]
  Sub Name([Arguments])
    [Statements]
  End Sub
```

```
...
```

```
End Class
```

Error Handling

New Exception

Throw

```
Throw ExceptionObject
```

Exception

Try

Catch

End Catch

```
Try  
  [Statements]  
Catch (Name As Exception)  
  [Statements]  
End Catch
```

```
Try  
  [Statements]  
Catch (Name As Exception)  
  [Statements]  
Catch (Name As Exception)  
  [Statements]  
End Catch
```

```
Try  
  [Statements]  
Catch (Name As Exception)  
  [Statements]  
Finally  
  [Statements]  
End Finally
```

(C)Copyright KBasic Software 2007

By Bernd Noetscher
www.objective-basic.com