# DroidBasic Syntax

## Contents

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DroidBasic Syntax</td>
<td>1</td>
</tr>
<tr>
<td>First Edition</td>
<td>3</td>
</tr>
<tr>
<td>Conventions Used In This Book / Way Of Writing</td>
<td>3</td>
</tr>
<tr>
<td>DroidBasic-Syntax</td>
<td>3</td>
</tr>
<tr>
<td>Variable</td>
<td>4</td>
</tr>
<tr>
<td>Declaration</td>
<td>4</td>
</tr>
<tr>
<td>Dim</td>
<td>4</td>
</tr>
<tr>
<td>Public</td>
<td>4</td>
</tr>
<tr>
<td>Private</td>
<td>4</td>
</tr>
<tr>
<td>Static</td>
<td>4</td>
</tr>
<tr>
<td>As</td>
<td>4</td>
</tr>
<tr>
<td>Assignment</td>
<td>5</td>
</tr>
<tr>
<td>User Defined Type</td>
<td>5</td>
</tr>
<tr>
<td>Type</td>
<td>5</td>
</tr>
<tr>
<td>Comment</td>
<td>5</td>
</tr>
<tr>
<td>Literal</td>
<td>5</td>
</tr>
<tr>
<td>Byte, Short, Integer</td>
<td>5</td>
</tr>
<tr>
<td>Hex</td>
<td>5</td>
</tr>
<tr>
<td>Binary</td>
<td>5</td>
</tr>
<tr>
<td>Single/Double/Float</td>
<td>5</td>
</tr>
<tr>
<td>Decimal (under construction)</td>
<td>5</td>
</tr>
<tr>
<td>DateTime</td>
<td>6</td>
</tr>
<tr>
<td>String</td>
<td>6</td>
</tr>
<tr>
<td>Boolean</td>
<td>6</td>
</tr>
<tr>
<td>Constant</td>
<td>6</td>
</tr>
<tr>
<td>Const</td>
<td>6</td>
</tr>
<tr>
<td>As</td>
<td>6</td>
</tr>
<tr>
<td>Working With Objects</td>
<td>6</td>
</tr>
<tr>
<td>Create Class</td>
<td>6</td>
</tr>
<tr>
<td>Access Class Variable And Instance Variable</td>
<td>6</td>
</tr>
<tr>
<td>Access Class Method Or Instance Method (Function Or Sub)</td>
<td>6</td>
</tr>
<tr>
<td>Class Method Or Instance Method</td>
<td>6</td>
</tr>
<tr>
<td>Access Class Type</td>
<td>7</td>
</tr>
<tr>
<td>Access Class Enum</td>
<td>7</td>
</tr>
<tr>
<td>Access Class Property</td>
<td>7</td>
</tr>
<tr>
<td>Call Method</td>
<td>7</td>
</tr>
<tr>
<td>Current Instance Of Object</td>
<td>7</td>
</tr>
<tr>
<td>Me</td>
<td>7</td>
</tr>
<tr>
<td>Scope modifier</td>
<td>7</td>
</tr>
<tr>
<td>Private</td>
<td>7</td>
</tr>
<tr>
<td>Public</td>
<td>7</td>
</tr>
<tr>
<td>Array</td>
<td>7</td>
</tr>
<tr>
<td>Dim</td>
<td>7</td>
</tr>
<tr>
<td>Access Array</td>
<td>7</td>
</tr>
<tr>
<td>Lower And Upper Bound Of Array</td>
<td>8</td>
</tr>
</tbody>
</table>
Module
Enumeration
Property
Functions
Flow Control - Loop
 Flow Control - Decision
 Flow Control - Loop
 Subs / Procedures
 Functions
 Property
User defined Type
First Edition

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

DroidBasic-Syntax

The syntax of sub, function or statement in the DroidBasic 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 DroidBasic 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 DroidBasic 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 (not supported yet)

As

Dim sName As String

Public Dim sName As String

Private Dim sName As String

Dim Name[ARRAY] [As Type] [, Name[ARRAY] [As Type]] …
Dim Name [= Expression]
Dim Name [As Type] [= Expression]
[Public | Private | Static] Dim Name [= Expression] [As Type]

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

User Defined Type

Type (not supported yet)
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 (not supported yet)
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 Instance Method (Function Or Sub)
objectname.instanceVariable = 99

Class Method Or Instance 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 (not supported yet)**

**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 (not supported yet)**

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

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

User defined Type (not supported yet)

**Access Type**

```
Type
  varname.typefield = 99
End Type

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

Enumeration (not supported yet)

**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

(C)Copyright KBasic Software 2012

Android is a trademark of Google Inc.