B4X 手册

B4X 语言

版本 2.3

最后更新: 2022.08.15

1 B4X 平台 9
2 变量与对象 9
2.1 变量类型 10
2.2 变量名称 13
2.3 声明变量 13
2.3.1 简单变量 13
2.3.2 数组变量 14
2.3.3 常量变量 Const 关键字 15
2.3.4 视图/节点（对象）数组 16
2.3.5 类型变量 只限 B4A、B4i 和 B4J 19
2.4 铸件 20
2.5 作为方法 21
2.6 范围 22
2.6.1 过程变量 22
2.6.2 活动变量 只限 B4A 23
2.6.3 局部变量 23
2.7 提示 23
3 程序流程 / 流程生命周期 24
3.1 B4A 程序流程 24
3.1.1 程序启动 25
3.1.2 过程全局变量 26
3.1.3 活动变量 26
3.1.4 启动服务 27
3.1.5 程序流程 28
3.1.6 Sub Process_Globals / Sub Globals 29
3.1.7 Sub Activity_Create (FirstTime As Boolean) 29
3.1.8 变量声明总结 30
3.1.9 Sub Activity_Resume Sub Activity_Pause (UserClosed As Boolean) 31
3.1.10 Activity.Finish / ExitApplication 32
3.2 B4i 程序流程 33
3.3 B4J 程序流程 34
3.4 B4R 程序流程 35
3.5 B4A / B4i / B4J 程序流程比较 36
3.5.1 程序启动 B4A / B4i / B4J 36
3.5.2 旋转装置 B4A / B4i 36
3.6 B4XPages 程序流程 37
4 B4X 语言 38
4.1 表达式 38
4.1.1 数学表达式 38
4.1.2 关系表达式 39
4.1.3 布尔表达式 39
4.2 标准关键字 40
Abs (Number As Double) As Double 43
ACos (Value As Double) As Double 43
ACosD (Value As Double) As Double 43
Array 43
Asc (Char As Char) As Int 43
ASin (Value As Double) As Double 43
ASinD (Value As Double) As Double 43

ATan (Value As Double) As Double 43
ATan2 (Y As Double, X As Double) As Double 43
ATan2D (Y As Double, X As Double) As Double 43
ATanD (Value As Double) As Double 43
BytesToString (Data() As Byte, StartOffset As Int, Length As Int, CharSet As String) As String 44
CallSub (Component As Object, Sub As String) As Object 44
CallSub2 (Component As Object, Sub As String, Argument As Object) As Object 44
CallSub3 (Component As Object, Sub As String, Argument1 As Object, Argument2 As Object) As Object 44
CallSubDelayed (Component As Object, Sub As String) 44
CallSubDelayed2 (Component As Object, Sub As String, Argument As Object) 44
CallSubDelayed3 (Component As Object, Sub As String, Argument1 As Object, Argument2 As Object) 44
Catch 45
cE As Double 45
Ceil (Number As Double) As Double 45
Chr (UnicodeValue As Int) As Char 45
Continue 45
Cos (Radians As Double) As Double 45
CosD (Degrees As Double) As Double 45
cPI As Double 45
CreateMap 45
CRLF As String 46
Dim 46
Exit 46
Floor (Number As Double) As Double 46
For 46
GetType (object As Object) As String 47
If 47
IIf 47
Is 47
IsNumber (Text As String) As Boolean 47
LoadBitmapSample (Dir As String, FileName As String, MaxWidth As Int, MaxHeight As Int) As Bitmap 48
Log (Message As String) 48
Logarithm (Number As Double, Base As Double) As Double 48
LogColor (Message As String, Color As Int) 48
Max (Number1 As Double, Number2 As Double) As Double 48
Me As Object 48
Min (Number1 As Double, Number2 As Double) As Double 48
Not (Value As Boolean) As Boolean 49
Null As Object 49
NumberFormat (Number As Double, MinimumIntegers As Int, MaximumFractions As Int) As String 49

NumberFormat2 (Number As Double, MinimumIntegers As Int, MaximumFractions As Int, MinimumFractions As Int, GroupingUsed As Boolean) As String 49
Power (Base As Double, Exponent As Double) As Double 49
QUOTE As String 49
Regex As Regex 49
Return 49
Rnd (Min As Int, Max As Int) As Int 49
RndSeed (Seed As Long) 49
Round (Number As Double) As Long 49
Select 50
Sender As Object 50
Sin (Radians As Double) As Double Calculates the trigonometric sine function. Angle measured in radians. 50
SinD (Degrees As Double) As Double 50
Calculates the trigonometric sine function. Angle measured in degrees. 50
Sleep (Value As Double) As Double 50
Sqrt (Value As Double) As Double 50
Sub 51
SubExists (Object As Object, Sub As String) As Boolean 51
TAB As String 51
Tan (Radians As Double) As Double 51
TanD (Degrees As Double) As Double 51
True As Boolean 51
Try 51
Type 52
Until 52
While 52
4.3 条件语句 53
4.3.1 If - Then - Else 53
4.3.1.1 布尔求值顺序 54
4.3.2 IIf 内联如果 55
4.3.3 Select - Case 56
4.4 循环结构 58
4.4.1 For - Next 58
4.4.2 For - Each 59
4.4.3 Do - Loop 60
4.5 内联转换 As 62
4.6 Subs 63
4.6.1 Declaring 63
4.6.2 Calling a Sub 63
4.6.3 Calling a Sub from another module 63
4.6.4 Naming 64
4.6.5 Parameters 64
4.6.6 Returned value 65
4.7 Resumable Subs 66
4.7.1 Sleep 66
4.7.2 Wait For 67
4.7.3 Code Flow 69
4.7.4 Waiting for a resumable sub to complete 70
4.7.5 Resumable Sub return value 71

4.7.6 B4A only KeyPress and Wait For MsgBox2Async 73
4.7.7 DoEvents deprecated ! 74
4.7.8 Dialogs 75
4.7.9 SQL with Wait For 76
4.7.9.1 Queries 76
4.7.9.2 B4J 77
4.7.10 Notes & Tips 77
4.8 Events 78
4.8.1 B4A 78
4.8.2 B4i 81
4.8.3 B4J 83
4.8.4 B4R 87
4.8.5 User interface summary 88
4.9 Libraries 89
4.9.1 Standard libraries 90
4.9.2 Additional libraries folder 90
4.9.2.1 Paths configuration B4A 92
4.9.2.2 Paths configuration B4i 92
4.9.2.3 Paths configuration B4J 93
4.9.2.4 Paths configuration B4R 93
4.9.3 B4X Libraries *.b4xlib 94
4.9.4 Load and update a Library 95
4.9.5 Error message "Are you missing a library reference?" 95
4.9.6 Where do I find libraries? 96
4.9.6.1 Online libraries index 96
4.10 String manipulation 97
4.10.1 B4A, B4i, B4J String 97
4.10.2 String concatenation 99
4.10.3 B4A, B4i, B4J StringBuilder 100
4.10.3.1 StringBuilder Methods 101
4.10.4 Smart String Literal 102
4.10.4.1 String Interpolation 102
4.10.4.2 Number Formatter 102
4.10.4.3 Other Formatters 103
4.10.5 B4A, B4i CharSequence CSBuilder 104
4.10.5.1 Text 104
4.10.5.2 With FontAwesome or MaterialIcons 106
4.10.5.3 Images 106
4.10.5.4 Clickable text 107
4.10.5.5 Highlight text 107
4.10.5.6 Center aligned text 108
4.10.5.7 CSBuilder Methods 109
4.10.5.7.1B4A / B4i 109
4.10.5.7.2B4A only 110
4.10.5.7.3B4i only 112
4.10.6 B4J TextFlow class 113
4.10.7 B4R 114
4.11 Number formatting 117
4.11.1 B4A, B4i, B4J 117
4.11.2 B4X NumberFormatter 117
4.11.3 B4R 117
4.12 Timers 118

4.13 Files B4A, B4i, B4J 120
4.13.1 File object 120
4.13.1.1 File locations 120
4.13.1.1.1B4X 120
4.13.1.1.2B4A only 121
4.13.1.1.3B4i only 124
4.13.1.1.4B4J only 124
4.13.1.2 File exists ? B4A, B4i, B4J 125
4.13.1.3 Common methods B4A, B4i, B4J 125
4.13.2 Filenames 128
4.13.3 Subfolders 128
4.13.4 B4A, B4J TextWriter 129
4.13.5 B4A, B4J TextReader 130
4.13.6 Text encoding 132
4.14 Lists B4A, B4i and B4J only 134
4.14.1 Non-dynamic Lists 136
4.15 Maps B4A, B4i and B4J only 137
4.16 Class modules 139
4.16.1 Getting started 139
4.16.1.1 Adding a Class module 141
4.16.1.2 Polymorphism 142
4.16.1.3 Self-reference 144
4.16.1.4 Activity object B4A only 144
4.16.2 Standard Class module 145
4.16.2.1 Structure 145
5 Find object methods, properties, events 147
5.1 B4X Help Viewer 147
5.2 Hovering over an object 148
5.3 Define an event routine 150
6 "Code smells" code to be avoided 151
6.1 Initializing an object and then assigning a different object to the same variable 151
6.2 Deprecated methods - DoEvents, Msgbox 151
6.3 Deprecated methods - Map.GetKeyAt / GetValueAt 151
6.4 File.DirDefaultExternal - This is always a mistake 152
6.5 Not using parameterized queries 152
6.6 Using Cursor instead of ResultSet - Cursor 152
6.7 Building the complete layout programmatically 152
6.8 Repeating the code 153
6.9 Long strings without using smart strings 153
6.10 Using global variables when not needed 153
6.11 Not using Wait For when possible 153
6.12 Using code modules instead of classes 155
6.13 Understanding booleans 155
6.14 Converting "random" bytes to strings 155
6.15 Generating or parsing XML / JSON by hand 155
7 Features that Erel recommends to avoid 156
8 Tips 159
8.1 Separate data from code 159
8.2 Don't Repeat Yourself (DRY principle) 159
8.3 Map collection 159
8.4 New technologies and features 159

8.5 Logs..... 159
8.6 B4A Avoid calling DoEvents..... 161
8.7 Strings are made of characters not bytes..... 161
8.8 B4A Use services, especially the Starter service..... 161
8.9 UI Layouts..... 161
8.10 B4J as a backend solution..... 161
8.11 Search..... 163
8.12 Notepad++..... 163
8.12.1 Encoding..... 163

主要贡献者: Klaus Christl (klaus), Erel Uziel (Erel)

要搜索给定的单词或句子，请使用“编辑”菜单中的“搜索”功能。

针对以下版本进行了更新:
B4A 版本 11.8
B4i 版本 8.00
B4J 版本 9.80
B4R 版本 3.90

B4X 手册:
B4X Getting Started
B4X 语言
B4X IDE Integrated Development Environment
B4X Visual Designer
B4X Help tools

B4XPages Cross-platform projects
B4X CustomViews
B4X Graphics
B4X XUI B4X User Interface
B4X SQLite Database
B4X JavaObject NativeObject

B4R 示例项目

您可以在此链接 [B4X] 文档手册中在线查阅这些手册。
请注意，外部链接在在线显示中不起作用。

1 B4X 平台

B4X 是一套适用于不同平台的编程语言。

B4X 套件支持比任何其他工具更多的平台
ANDROID | IOS | WINDOWS | MAC | LINUX | ARDUINO | RASPBERRY PI | ESP8266 |
和更多...

B4A **Android**

B4A 是一款 **100% 免费**的安卓应用程序开发工具，它包括快速开发任何类型的安卓应用程序所需的所有功能。

B4i **iOS**

B4i 是原生 iOS 应用程序的开发工具。
B4i 遵循与 B4A 相同的概念，允许您重用大部分代码并为安卓和 iOS 构建应用程序。

B4J **Java / Windows / Mac / Linux / Raspberry PI**

B4J 是一款 **100% 免费**的桌面、服务器和物联网解决方案开发工具。
使用 B4J，您可以轻松创建桌面应用程序 (UI)、控制台程序（非 UI）和服务器解决方案。

- 编译后的应用程序可以在 Windows、Mac、Linux 和 ARM 板（如树莓派）。

B4R **Arduino / ESP8266**

B4R 是 **100% 免费**的原生 Arduino 和 ESP8266 程序开发工具。 B4R 遵循其他 B4X 工具的相同概念，提供简单而强大的开发工具。
B4R、B4A、B4J 和 B4i 共同构成物联网 (IoT) 的最佳开发解决方案。

B4XPages

B4XPage 是 B4A、B4i 和 B4J 的内部库，允许轻松开发跨平台程序。
B4XPages 在 B4XPages 跨平台项目手册中有详细的解释。
即使您只想在一个平台上进行开发，使用 B4XPage 库也很有趣，它使程序流程更简单，尤其是对于 B4A。

2 变量与对象

变量是赋予某些已知或未知数量或信息的符号名称，目的是允许名称独立于它所代表的信息使用。计算机源代码中的变量名通常与数据存储位置相关联，因此也与它的内容相关联，这些变量名可能会在程序执行过程中发生变化（来源 Wikipedia）。

有两种类型的变量：原始类型和非原始类型。
原语包括数字类型：Byte、Short、Int、Long、Float 和 Double。
原语还包括：布尔值和字符。

2.1 变量类型

B4A、B4i、B4J

类型列表及其范围：

B4X	类型	最小值	最大值
Boolean	布尔值	False	True
Byte	整数 8 bits	-2^{7} -128	$2^{7}-1$ 127
Short	整数 16 bits	-2^{15} -32768	$2^{15}-1$ 32767
Int	整数 32 bits	-2^{31} -2147483648	$2^{31}-1$ 2147483647
Long	长整数 64 bits	-2^{63} -9223372036854775808	$2^{63}-1$ 9223372036854775807
Float	浮点数 32 bits	-2^{-149} 1.4E-45	$(2-2^{-23}) * 2^{127}$ 3.4028235 E 38
Double	双精度数 64 bits	-2^{-1074} 2.2250738585072014 E -308	$(2-2^{-52}) * 2^{1023}$ 1.7976931348623157 E 308
Char	字符		
String	字符数组		

B4R

类型列表及其范围：
数字类型：

Byte	0 - 255	
Int (2 bytes)	-32,768 - 32,768	类似于其他 B4X 工具中的 Short 类型。
UInt (2 bytes)	0 - 65,535	B4R 专用。
Long (4 bytes)	-2,147,483,648 - 2,147,483,647	类似于其他 B4X 工具中的 Int 类型。
ULong (4 bytes)	0 - 4,294,967,295	B4R 专用。
Double (4 bytes)	4 字节浮点	类似于其他 B4X 工具中的 Float 类型。
Float 与 Double 相同。Short 与 Int 相同。		

以上在所有板上都是正确的，包括 Arduino Duo。

其他类型:
Boolean True 或 False。 实际上，它被保存为一个值为 1 或 0 的字节。
String 字符串由以空字节结尾的字节数组组成（值为 0 的字节).
Object 对象可以保存其他类型的值。

原始类型总是按值传递给其他子程序或分配给其他变量。例如:

```
Sub S1
   Private A As Int
   A = 12                  变量 A = 12
   S2(A)                   它按值传递给例程 S2
   Log(A)  ' 打印 12       变量 A 仍然等于 12，即使 B 在例程 S2 中改变了。
End Sub

Sub S2(B As Int)           变量 B = 12
   B = 45                  其值更改为 B = 45
End Sub
```

所有其他类型，包括原始类型数组和字符串，都归类为非原始类型。
当您将非原始类型传递给子程序或将其分配给不同的变量时，将传递引用的副本。
这意味着数据本身不会重复。
它与通过引用传递略有不同，因为您无法更改原始变量的引用。

所有类型都可以视为对象。
Lists 和 Maps 之类的 Collections 与 Objects 一起使用，因此可以存储任何值。
下面是一个常见错误的示例，其中开发人员试图将多个数组添加到列表中:

```
Private arr(3) As Int
Private List1 As List
List1.Initialize
For I = 1 To 5
   arr(0) = I * 2
   arr(1) = I * 2
   arr(2) = I * 2
   List1.Add(arr)     '将整个数组添加为单个项目
Next
arr = List1.Get(0)    '获取列表中的第一项
Log(arr(0))           '这里会打印什么？？？
```

您可能预计它打印 2。但是，它会打印 10。
我们创建了一个数组并将该数组的 5 个引用添加到列表中。
单个数组中的值是上次迭代中设置的值。
为了解决这个问题，我们需要在每次迭代时创建一个新数组。
这是通过每次迭代调用 Private 来完成的:

```
Private arr(3) As Int '在这种情况下，这个称呼是多余的。
Private List1 As List
List1.Initialize
For i = 1 To 5
   Private arr(3) As Int
   arr(0) = i * 2
   arr(1) = i * 2
   arr(2) = i * 2
   List1.Add(arr) '将整个数组添加为单个物品
Next
arr = List1.Get(0) '从列表中获取第一个物品
Log(arr(0)) '将打印 2
```

2.2 变量名称

除了保留字外，您可以为变量指定任何名称。
变量名必须以字母开头，并且必须由以下字符 A-Z、a-z、0-9 和下划线“_”组成，不能有空格，不能有括号等。
变量名不区分大小写，这意味着 Index 和 index 指的是同一个变量。

但是给它们起有意义的名字是一种很好的做法。
例子:
Interest = Capital * Rate / 100 是有意义
n1 = n2 * n3 / 100 没有意义

对于视图(B4A, B4i)，节点(B4J)，在名称中添加一个定义其类型的三个字符的前缀很有用。
例子:

lblCapital	lbl > Label	Capital >目的
edtInterest	edt > EditText	Interest >目的
btnNext	btn > Button	Next >目的

2.3 声明变量

2.3.1 简单变量

变量声明为 **Private** 或者 **Public** 关键词后跟变量名和 **As** 关键词然后是变量类型。详情请看范围章节.
存在着 **Dim** 关键词，这是为了兼容性而维护的。

例子:

```
Private Capital As Double          将三个变量声明为 Double,
Private Interest As Double         双精度数。
Private Rate As Double

Private i As Int                   声明三个变量为Int, 整数。
Private j As Int
Private k As Int

Private lblCapital As Label
Private lblInterest As Label       将三个变量声明为标签视图。
Private lblRate As Label

Private btnNext As Button          将两个变量声明为按钮视图。
Private btnPrev As Button
```

也可以用简短的方式声明相同的变量。

```
Private Capital, Interest, Rate As Double
Private i, j, k As Int
Private lblCapital, lblInterest, lblRate As Label
Private btnNext, btnPrev As Button
```

变量名用逗号分隔，后跟类型声明。

以下变量声明有效:

```
Private i = 0, j = 2, k = 5 As Int

Private txt = "test" As String, value = 1.05 As Double, flag = False As Boolean
```

如果我们想在代码中使用它们，就必须声明视图名称。
例如，如果我们要在代码中更改 Label 视图中的文本，例如

```
lblCapital.Text = "1200",
```

我们需要通过它的名字 lblCapital 来引用这个 Label view，这是通过 Private 声明完成的。
如果我们从未在代码中的任何地方引用此 Label 视图，则不需要声明。
对该视图使用事件例程也不需要声明。

要将值分配给变量，请写入其名称后跟等号再后跟值，例如:

```
Capital = 1200
LastName = "SMITH"
```

请注意，对于 Capital，我们只写了 1200，因为 Capital 是一个数字。
但是对于 LastName，我们写了"SMITH"，因为 LastName 是一个字符串。
字符串必须始终写在双引号之间。

2.3.2 数组变量

数组是可以通过索引选择的数据或对象的集合。 数组可以有多个维度。
声明包含 Private 或 Public 关键字，后跟变量名 LastName、方括号(50)之间的项目数、关键字 As 和变量类型 String。
有关详细信息，请参阅范围章节。 存在 Dim 关键字，这是为了兼容性而维护的。

注意：B4R 只支持一维数组！

例子:

```
Public LastName(50) As String     一维字符串数组，物品总数 50。
Public Matrix(3, 3) As Double     二维数组 Doubles，物品总数 9。
Public Data(3, 5, 10) As Int      三维整数数组，物品总数 150。
```

数组中每个维度的第一个索引是 0。

```
LastName(0), Matrix(0,0), Data(0,0,0)
```

最后一个索引等于每个维度中的项目数减 1。

```
LastName(49), Matrix(2,2), Data(2,4,9)
```

```
Public LastName(10) As String
Public FirstName(10) As String
Public Address(10) As String
Public City(10) As String
```

或者

```
Public LastName(10), FirstName(10), Address(10), City(10) As String
```

此示例显示如何访问三维数组中的所有项目。

```
Public Data(3, 5, 10) As Int

For i = 0 To 2
   For j = 0 To 4
      For k = 0 To 9
         Data(i, j, k) = ...
      Next
   Next
Next
```

声明数组的一种更通用的方法是使用变量。

```
Public NbPers = 10 As Int
Public LastName(NbPers) As String
Public FirstName(NbPers) As String
Public Address(NbPers) As String
Public City(NbPers) As String
```

我们将变量声明为 `Public NbPers = 10 As Int` 并将其值设置为 10。
然后我们用这个变量来声明数组，而不是像以前那样用数字 10 来声明。
最大的优点是如果在某个时候我们需要更改项目的数量，我们只更改『一个』值。

对于 Data 数组，我们可以使用以下代码。

```
Public NbX = 2 As Int
Public NbY = 5 As Int
Public NbZ = 10 As Int
Public Data(NbX, NbY, NbZ) As Int
```

和访问例程。

```
For i = 0 To NbX - 1
   For j = 0 To NbY - 1
      For k = 0 To NbZ - 1
         Data(i, j, k) = ...
      Next
   Next
Next
```

使用 Array 关键字填充数组：

```
Public Name() As String
Name = Array As String("Miller", "Smith", "Johnson", "Jordan")
```

2.3.3 常量变量 Const 关键字

Const 变量是不能在代码中的任何地方更改的常量变量。
为此，我们在 `Private` 或 `Public` 之后使用 `Const` 关键字，如下所示，

```
Private Const Size As Int = 10
Public Const ItemNumber As Int = 100
```

2.3.4 视图/节点（对象）数组

视图/节点或对象也可以在一个『数组』中。 以下代码显示了一个示例：
在 B4A 和 B4i 中用户界面对象称为*视图 (views)* 而在 B4J 中称为*节点 (nodes)*。

在下面的示例中，『按钮』(Button) 通过代码添加到父视图 / 节点。

B4A

```
Sub Globals
   Private Buttons(6) As Button
End Sub

Sub Activity_Create(FirstTime As Boolean)
   Private i As Int

   For i = 0 To 5
      Buttons(i).Initialize("Buttons")
      Activity.AddView(Buttons(i), 10dip, 10dip + i * 60dip, 150dip, 50dip)
      Buttons(i).Tag = i + 1
      Buttons(i).Text = "Test " & (i + 1)
   Next
End Sub

Sub Buttons_Click
   Private btn As Button
   btn = Sender
   Log("Button " & btn.Tag & " clicked")
End Sub
```

B4i

```
Sub Process_Globals

   Private Buttons(6) As Button
End Sub

Private Sub Application_Start (Nav As NavigationController)

   Private i As Int
   For i = 0 To 5
      Buttons(i).Initialize("Buttons")
      Page1.RootPanel.AddView(Buttons(i), 10dip, 10dip + i * 60dip, 150dip, 50dip)
      Buttons(i).Tag = i + 1
      Buttons(i).Text = "Test " & (i + 1)
   Next
End Sub

Sub Buttons_Click
   Private btn As Button
   btn = Sender
   Log("Button " & btn.Tag & " clicked")
End Sub
```

B4J

```
Sub Process_Globals

   Private Buttons(6) As Button
End Sub

Sub AppStart (Form1 As Form, Args() As String)

   Private i As Int
   For i = 0 To 5
      Buttons(i).Initialize("Buttons")
      MainForm.RootPane.AddNode(Buttons(i), 10, 10 + i * 60, 150, 50)
      Buttons(i).Tag = i + 1
      Buttons(i).Text = "Test " & (i + 1)
   Next
End Sub

Sub Buttons_MouseClicked (EventData As MouseEvent)
   Private btn As Button
   btn = Sender
   Log("Button " & btn.Tag & " clicked")
End Sub
```

『按钮』也可以添加到布局文件中，在这种情况下，它们既不能被初始化，也不能被添加到父视图 / 节点，并且文本和标签属性也应该在设计器中设置。
在这种情况下，代码将如下所示：

B4A

```
Sub Globals
   Private b1, b2, b3, b4, b5, b6, b7 As Button
   Private Buttons() As Button
End Sub

Sub Activity_Create(FirstTime As Boolean)

   Buttons = Array As Button(b1, b2, b3, b4, b5, b6, b7)
End Sub

Sub Buttons_Click
   Private btn As Button
   btn = Sender
   Log("Button " & btn.Tag & " clicked")
End Sub
```

B4i

```
Sub Process_Globals

   Private b1, b2, b3, b4, b5, b6, b7 As Button
   Private Buttons(6) As Button
End Sub

Private Sub Application_Start (Nav As NavigationController)

   Buttons = Array As Button(b1, b2, b3, b4, b5, b6, b7)
End Sub

Sub Buttons_Click
   Private btn As Button
   btn = Sender
   Log("Button " & btn.Tag & " clicked")
End Sub
```

B4J

```
Sub Process_Globals

   Private b1, b2, b3, b4, b5, b6, b7 As Button
   Private Buttons(6) As Button
End Sub

Sub AppStart (Form1 As Form, Args() As String)

   Buttons = Array As Button(b1, b2, b3, b4, b5, b6, b7)
End Sub

Sub Buttons_MouseClicked (EventData As MouseEvent)
   Private btn As Button
   btn = Sender
   Log("Button " & btn.Tag & " clicked")
End Sub
```

2.3.5 类型变量 只限 B4A、B4i 和 B4J

类型不能是私有的。 一旦声明它在任何地方都可用（类似于 Class 模块）。
声明它们的最佳位置是在 Main 模块的 Process_Globals 例程中。

让我们用一个人的数据重用这个例子。
我们可以使用 Type 关键字定义一个个人类型变量，而不是单独声明每个参数:

```
Public NbUsers = 10 As Int
Type Person(LastName As String, FirstName As String. Address As String, City As String)
Public User(NbUsers) As Person
Public CurrentUser As Person
```

新的个人类型是 Person，然后我们声明此个人类型的单个变量或数组。
要访问特定项目，请使用以下代码。

```
CurrentUser.FirstName
CurrentUser.LastName

User(1).LastName
User(1).FirstName
```

变量名称，后跟一个点和所需的参数。
如果变量是一个数组，则名称后跟括号之间的所需索引。

可以将一个类型化变量分配给另一个相同类型的变量，如下所示。

```
CurrentUser = User(1)
```

2.4 铸件

B4X 根据需要自动铸件类型。 它还自动将数字转换为字符串，反之亦然。
在许多情况下，您需要将 Object 显式铸件为特定类型。
这可以通过将 Object 分配给所需类型的变量来完成。
例如，Sender 关键字引用一个对象，它是引发事件的对象。
以下代码更改按下按钮的颜色。
请注意，有多个按钮共享相同的事件子程序。

```
Sub Globals
   Private  Btn1, Btn2, Btn3 As Button
End Sub

Sub Activity_Create(FirstTime As Boolean)
   Btn1.Initialize("Btn")
   Btn2.Initialize("Btn")
   Btn3.Initialize("Btn")
   Activity.AddView(Btn1, 10dip, 10dip, 200dip, 50dip)
   Activity.AddView(Btn2, 10dip, 70dip, 200dip, 50dip)
   Activity.AddView(Btn3, 10dip, 130dip, 200dip, 50dip)
End Sub

Sub Btn_Click
   Private  btn As Button
   btn = Sender     ' 将对象铸件成按钮
   btn.Color = Colors.RGB(Rnd(0, 255), Rnd(0, 255), Rnd(0, 255))
End Sub
```

以上的代码也可以写得更优雅:

```
Sub Globals

End Sub

Sub Activity_Create(FirstTime As Boolean)
   Private  i As Int
   For i = 0 To 9   ' 创建 10 个按钮
      Private  Btn As Button
      Btn.Initialize("Btn")
      Activity.AddView(Btn, 10dip, 10dip + 60dip * i, 200dip, 50dip)
   Next
End Sub

Sub Btn_Click
   Private  btn As Button
   btn = Sender     ' 将对象投射到按钮
   btn.Color = Colors.RGB(Rnd(0, 255), Rnd(0, 255), Rnd(0, 255))
End Sub
```

2.5 作为方法

您可以使用“As”方法轻松地将一个对象转换为另一个对象。

当您想要将特定于平台的对象转换为跨平台对象时，这可能很有用，反之亦然。

例如，B4XView 确实存在 Rotation 属性，但对于“标准”标签就不存在。

```
Label1.As(B4XView).Rotation = 90
```

以上的行是短途，以下的三行做同样的事情，但很长：

```
Private xLabel1 As B4XView
xLabel1 = Label1
xLabel1.Rotation = 90
```

您还可以返回以设置特定于平台的属性：

```
xLabel1.As(Label).Padding(Array As Int(10dip, 0, 10dip, 0))
```

2.6 范围

2.6.1 过程变量

只要过程存在，这些变量就会存在。
您应该在 Sub Process_Globals 中声明这些变量。
这个 sub 在进程启动时被调用一次（这对所有模块都是如此，而不仅仅是主模块）。
这些变量是唯一的“公开”变量。 这意味着它们也可以从其他模块访问。

但是，在 B4A 中，并非所有类型的对象都可以声明为流程变量。
例如，视图 / 节点不能声明为流程变量。
原因是我们不想持有对应该与活动一起销毁的对象的引用。
换句话说，一旦 Activity 被销毁，该 Activity 中包含的所有视图也将被销毁。
如果我们持有对视图的引用，垃圾收集器将无法释放资源，并且我们将发生内存泄漏。 编译器强制执行此要求。

要访问其他模块中的进程全局变量，而不是声明它们的模块，它们的名称必须具有它们被声明为前缀的模块名称。
例子：
在名为 *MyModule* 的模块中定义的变量

```
Sub Process_Globals
   Public MyVar As String
End Sub
```

访问 *MyModule* 模块中的变量：

```
   MyVar = "Text"
```

访问任何其他模块中的变量：

```
   MyModule.MyVar = "Text"
```

变量可以声明为：

```
Dim MyVar As String
```

在这种情况下，变量是公开的，与 Public 相同。

像这样声明变量是一种很好的做法：

```
Public MyVar As String
```

这个变量是公开的。

可以像这样在 Sub Process_Globals 中声明私有变量：

```
Private MyVar As String
```

该变量对于声明它的 activity 或模块是私有的。
对于 activity，最好在 Sub Globals 中声明它们。

对于在 Sub Class_Globals 中的 Class 模块中声明的变量，与上述相同的规则是有效的。

```
   Public MyVarPublic As String          ' 公共
   Private MyVarPublic As String         ' 私有
   Dim MyVar As String                   ' 像公共一样公开 public like Public
```

不推荐在 Sub Class_Globals 中使用 Dim ！

2.6.2 活动变量 只限 B4A

这些变量包含在活动中。
您应该在 Sub Globals 中声明这些变量。
这些变量是“私有的”，只能从当前活动模块访问。
所有对象类型都可以声明为活动变量。
每次创建活动时，都会调用 Sub Globals（在 Activity_Create 之前）。
只要活动存在，这些变量就存在。

2.6.3 局部变量

在子程序中声明的变量是该子程序的局部变量。
它们是“私有的”，只能从声明它们的子例程中访问。
所有对象类型都可以声明为局部变量。
在子程序的每次调用中，局部变量都被初始化为其默认值或您在代码中定义的任何其他值，并在退出子程序时被“销毁”。

2.7 提示

可以将视图 / 节点分配给变量，以便您可以轻松更改视图的通用属性。

例如，以下代码禁用作为面板 / 窗格的直接子级的所有视图：

```
For i = 0 To MyPanel.NumberOfViews - 1
   Private  v As View
   v = MyPanel.GetView(i)
   v.Enabled = False
Next
```

如果我们只想禁用按钮：

```
For i = 0 To MyPanel.NumberOfViews - 1
   Private  v As View
   v = MyPanel.GetView(i)
   If v Is Button Then ' 检查它是否是一个按钮
      v.Enabled = False
   End If
Next
```

注意：MyPanel 是 B4A 和 B4i 中的一个面板 *(Pane1)*，但它是 B4J 中的一个窗格 *(Pane)*。

3 程序流程 / 流程生命周期

每个平台都有自己的程序流程。

为了制作跨平台项目，现在使用 B4XPages 更容易。
B4XPages 跨平台项目手册中详细解释了 B4XPages。

3.1 B4A 程序流程

让我们从简单的开始：
每个 B4A 程序都在自己的进程中运行。
一个进程有一个主线程，它也被称为 UI 线程，只要进程存在，它就会存在。 一个进程也可以有更多的线程，这对后台任务很有用。

一个进程在用户启动您的应用程序时启动，假设它尚未在后台运行。

过程结束的决定性较小。 它会在用户或系统关闭所有活动后的某个时间发生。
例如，如果您有一个活动并且用户按下后退键，则活动将关闭。 稍后当手机内存不足（最终会发生）时，该过程将退出。
如果用户再次启动您的程序并且该进程没有被杀死，那么相同的进程将被重用。

B4A 应用由一项或多项活动组成。

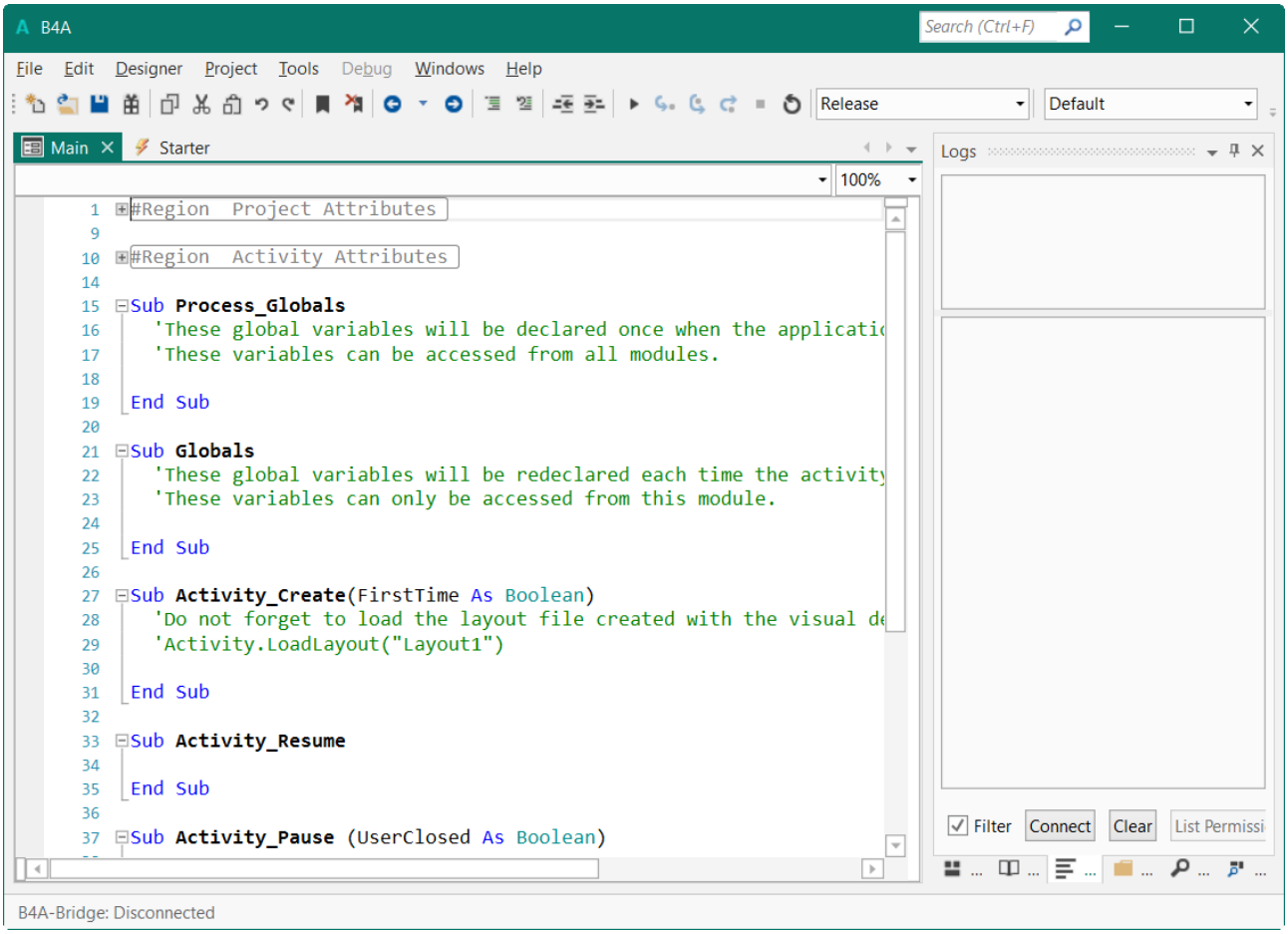
活动有点类似于 Windows 窗体。

一个主要区别是，当一个活动不在前台时，它可以被杀死以保留内存。 通常你会希望在活动丢失之前保存它的状态。 在与进程关联的持久存储或内存中。
稍后将在需要时重新创建此活动。

当设备发生重大配置更改时，会发生另一个微妙的问题。 最常见的是方向改变（用户旋转设备）。
当发生这样的变化时，当前的活动将被销毁，然后重新创建。 现在可以根据新配置创建活动（例如，我们现在知道新的屏幕尺寸）。

3.1.1 程序启动

当我们启动一个新程序时，我们得到以下模板：



在左上角，我们看到两个模块选项卡 

:
Main Activity
Starter Service

Starter Service 用于声明所有 ProcessGlobal 变量，并且可以从项目中的任何模块访问这些变量。
Main Activity 是起始 Activity，它不能被移除。

变量可以是全局的或局部的。 局部变量是在 Process_Globals 或 Globals 之外的 sub 中声明的变量。
局部变量是包含子或模块的局部变量。 一旦 sub 结束，这些变量就不再存在。
可以从包含模块中的所有子访问全局变量。

有两种类型的全局变量。
流程变量（可从所有模块访问）和活动变量（可从单个模块访问）。

3.1.2 过程全局变量

只要过程存在，这些变量就会存在。
您应该在 Starter Service 的 Sub Process_Globals 中将这些变量声明为 Public 似

```
Sub Process_Globals
   '这些全局变量将在应用程序启动时声明一次。
   '可以从所有模块访问这些变量。
   Public MyVariable  = "Test" As String
```

该子程序在进程启动时被调用一次。
这些变量是唯一的“公共”变量。这意味着它们也可以从其他模块访问。

每个 Activity 模块中还有一个 Process_Globals 例程。
如果您需要仅在 Activity 中有效的变量，它们仅在程序启动时初始化一次，您应该将它们放在 Activity 的 Process_Globals 例程中（这适用于所有活动，而不仅仅是第一个活动）。

但是，并非所有类型的对象都可以声明为过程变量。
例如，所有视图都不能声明为过程变量。
原因是我们不想持有对应该与活动一起销毁的对象的引用。
换句话说，当 Activity 被销毁时，该 Activity 中包含的所有视图也会被销毁。如果我们不这样做，并且在 Activity 被销毁后保留对视图的引用，那么垃圾收集器将无法释放资源，并且会发生内存泄漏。
编译器强制执行此要求。

3.1.3 活动变量

这些变量归活动所有。
您应该在 Sub Globals 中声明这些变量。
这些变量是“私有的”，只能从当前活动模块访问。
所有对象类型都可以声明为活动变量。
每次创建活动时，都会调用 Sub Globals（在 Activity_Create 之前）。
只要活动存在，这些变量就存在。

3.1.4 启动服务

任何非小型 Android 应用程序的开发人员都需要应对的挑战之一是多个可能的入口点。

在几乎所有情况下的开发过程中，应用程序都将从 Main 活动开始。
许多程序以类似于以下的代码开头：

```
Sub Activity_Create (FirstTime As Boolean)
   If FirstTime Then
      SQL.Initialize(...)
      SomeBitmap = LoadBitmap(...)
      '加载应用程序范围资源的附加代码
   End If
End Sub
```

在开发过程中，一切似乎都运行良好。然而，该应用程序“奇怪地”不时在最终用户设备上崩溃。
这些崩溃的原因是操作系统可以从不同的活动或服务启动进程。例如，如果您使用 StartServiceAt
并且操作系统在后台终止该进程。
现在 SQL 对象和其他资源将不会被初始化。

从 B4A v5.20 开始，有一个名为 Starter 服务的新功能，它提供了一个单一且一致的入口点。如果存在 Starter 服务，则该进程将始终从该服务启动。

将创建并启动 Starter 服务，然后才会启动应该启动的活动或服务。
这意味着 Starter 服务是初始化所有应用程序范围资源的最佳位置。
其他模块可以安全地访问这些资源。
Starter 服务应该是所有公共流程全局变量的默认位置。 SQL 对象、从文件中读取的数据和多个活动使用的位图都应该在 Starter 服务的 Service_Create 子程序中进行初始化。

笔记

- Starter 服务由其名称标识。 您可以将名为 Starter 的新服务添加到现有项目中，它将成为程序入口点。
  这是通过选择项目 > 添加新模块 > 服务模块来完成的。
- 这是一项可选功能。 您可以删除 Starter 服务。
- 如果您不希望服务继续运行，您可以在 Service_Start 中调用 StopService(Me)。 但是，这意味着该服务将无法处理事件（例如，您将无法使用异步 SQL 方法）。
- 启动服务应该从编译的库中排除。 默认情况下，它的 #ExcludeFromLibrary 属性在服务属性区域中设置为 True。

3.1.5 程序流程

程序流程如下：

- **Main Process_Globals** 主要模块的 Process_Globals 例程
  在这里，我们为 Main 模块声明所有私有变量和对象。

- **Starter Sevice Process_Globals** 如果服务存在，它就会运行。
  在这里，我们声明所有公共进程全局变量和对象，如 SQL、位图等。

- **其他 Activity Main Process_Globals** 其他模块的 Process_Globals 例程
  在这里，我们为给定模块声明所有私有变量和对象。

- **Starter Service Service_Create** 如果服务存在，它就会运行。
  在这里，我们初始化所有公共进程全局变量和对象，如 SQL、位图等。

- **Starter Sevice Service_Start** 如果服务存在，它就会运行。
  我们可以将这个例程留空。

- Globals
  在这里，我们为给定的 Activity 声明所有私有变量。

- Sub Activity_Create
  这里我们加载布局并初始化代码添加的活动对象

- Activity_Resume
  每次活动更改其状态时都会运行此例程。

- Activity_Pause
  此例程在 Activity 暂停时运行，例如方向更改、启动另一个 Activity 等。

3.1.6 Sub Process_Globals / Sub Globals

在任何 Activity 中，都应该使用 Process_Globals 和 Globals 来声明变量。
您还可以设置“简单”变量（数字、字符串和布尔值）的值。

你不应该在那里放任何其他代码。
您应该将代码放在 Activity_Create 中。

3.1.7 Sub Activity_Create (FirstTime As Boolean)

创建活动时调用此子程序。
活动已创建
- 当用户首次启动应用程序时
- 设备配置已更改（用户旋转设备）并且活动被破坏
- 当活动在后台并且操作系统决定销毁它以释放内存时。

这个子程序的主要目的是加载或创建布局（以及其他用途）。
FirstTime 参数告诉我们这是否是第一次创建此活动。第一次涉及到当前的进程。
您可以使用 FirstTime 运行与流程变量相关的各种初始化。
例如，如果您有一个包含需要读取的值列表的文件，则可以在 FirstTime 为 True 时读取它，并通过在 Sub Process_Globals 中声明该列表将列表存储为流程变量
现在我们知道，只要进程存在，这个列表就可用，即使重新创建活动，也不需要重新加载它。

总而言之，你可以测试 FirstTime 是否为 True，然后初始化 Activity 的 Sub Process_Globals 中声明的流程变量。

3.1.8 变量声明总结

我们应该声明哪个变量在哪里以及在哪里初始化我们的变量：

- 您想从多个模块访问的变量和无用户界面对象。
  像 SQL、地图、列表、位图等。
  这些必须在 Starter Process_Globals 中声明为 Public，例如：

```
Sub Process_Globals
	Public SQL1 As SQL
	Public Origin = 0 As Int
	Public MyBitmap As Bitmap
End Sub
```

  并在 Starter Service_Create 中初始化，如：

```
Sub Service_Create
	SQL1.Initialize(...)
	MyBitmap.Initialize(...)
End Sub
```

- 一个活动中的所有子程序都可以访问的变量，应该只初始化一次。
  这些必须在 Activity Process_Globals 中声明为 Private，例如：

```
Sub Process_Globals
	Private MyList As List
	Private MyMap As Map
End Sub
```

  并在 Activty_Create 中初始化，如：

```
Sub Activity_Create
	MyList.Initialize
	MyMap.Initialize
End Sub
```

  - Class 或 Code module 中的变量
    这些大多被声明为 Private，如果您希望它们可以从 Class 或 Code 模块外部访问，您可以将它们声明为 Public。
    B4X Booklet CustomViews Booklet 中详细解释了类模块。

  - 用户界面对象
    这些必须在 Activity 模块中声明，它们在 Globals 中使用，例如：

```
Sub Globals
    Private btnGoToAct2, btnChangeValues As Button
    Private lblCapital, lblInterest, lblRate  As Label
End Sub
```

像 Int、Double String 和 Boolean 这样的简单变量可以直接在声明行中初始化，即使在 Process_Globals 例程中也是如此。
例子：

```
Public Origin = 0 as Int
```

不应在 Process_Globals 例程中编写任何代码！

3.1.9 Sub Activity_Resume
Sub Activity_Pause (UserClosed As Boolean)

Activity_Resume 在 Activity_Create 完成后或恢复暂停的活动后立即调用（活动移到后台，现在它返回到前台）。
请注意，当您打开不同的活动（通过调用 StartActivity）时，当前活动首先暂停，然后如果需要，将创建另一个活动并（始终）恢复。

每次活动从前台移动到后台时，都会调用 Activity_Pause。
当 Activity 处于前台并且发生配置更改（这导致 Activity 暂停然后销毁）时，也会调用 Activity_Pause。
Activity_Pause 是保存重要信息的最后一个位置。
通常有两种机制可以让您保存活动状态。
仅与当前应用程序实例相关的信息可以存储在一个或多个流程变量中。
其他信息应存储在持久存储（文件或数据库）中。
例如，如果用户更改了某些设置，此时您应该将更改保存到持久存储中。否则更改可能会丢失。

每次活动从前台移动到后台时都会调用 Activity_Pause。这可能是因为：
1. 开始了不同的活动。
2. 按下主页按钮。
3. 引发了配置更改事件（例如方向更改）。
4. 后退按钮被按下。

在场景 1 和 2 中，活动将被暂停并暂时保存在内存中，因为它预计稍后会被重用。

在场景 3 中，活动将被暂停、销毁，然后再次创建（并恢复）。

在场景 4 中，活动将被暂停并销毁。**按下返回按钮类似于关闭活动。**在这种情况下，您**不**需要保存任何特定于实例的信息（例如 pacman 在 PacMan 游戏中的位置）。

UserClosed 参数在这种情况下为真，在所有其他情况下为假。请注意，当您调用 Activity.Finish 时也是如此。该方法暂停并销毁当前活动，类似于后退按钮。

您可以使用 UserClosed 参数来决定要保存哪些数据以及是否将任何相关的流程变量重置为其初始状态（如果位置是流程变量，则将 pacman 位置移动到中心）。

3.1.10 Activity.Finish / ExitApplication

关于如何以及何时使用 Activity.Finish 和 ExitApplication 的一些解释。

可以在这里找到一篇关于 Android 功能的有趣文章：
Android 方式的多任务处理。

大多数应用程序不应该使用 ExitApplication 而是更喜欢 Activity.Finish 让操作系统决定何时终止进程。
仅当您确实需要完全终止该进程时才应使用它。

我们什么时候应该使用 Activity.Finish 什么时候不应该?
让我们考虑以下没有任何 Activity.Finish 的示例：

- **Main 活动**
  - StartActivity(SecondActivity)
- **SecondActivity 活动**
  - StartActivity(ThirdActivity)
- **ThirdActivity 活动**
  - 单击返回按钮
  - 操作系统返回上一个活动，SecondActivity
- **SecondActivity 活动**
  - 单击返回按钮
  - 操作系统返回之前的活动，Main
- **Main 活动**
  - 单击返回按钮
  - 操作系统退出程序

现在让我们考虑以下在每个 StartActivity 之前使用 Activity.Finish 的示例：

- **Main 活动**
  - Activity.Finish
  - StartActivity(SecondActivity)
- **SecondActivity 活动**
  - Activity.Finish
  - StartActivity(ThirdActivity)
- **ThirdActivity 活动**
  - 单击返回按钮
  - 操作系统退出程序

仅当我们不想使用 Back 按钮返回此活动时，我们才应该在开始另一个活动之前使用 Activity.Finish。

3.2 B4i 程序流程

B4i 中的程序流程比 B4A 程序流程简单得多。

当我们运行一个新项目时，我们会得到以下模板：

```
Sub Process_Globals
   'These global variables will be declared once when the application starts.
   'Public variables can be accessed from all modules.
   Public App As Application
   Public NavControl As NavigationController
   Private Page1 As Page

End Sub

Private Sub Application_Start (Nav As NavigationController)
   'SetDebugAutoFlushLogs(True) 'Uncomment if program crashes before all logs are
printed.
   NavControl = Nav
   Page1.Initialize("Page1")
   Page1.Title = "Page 1"
   Page1.RootPanel.Color = Colors.White
   NavControl.ShowPage(Page1)
End Sub

Private Sub Page1_Resize(Width As Int, Height As Int)

End Sub

Private Sub Application_Background

End Sub
```

当您启动程序时，例程将按上述顺序执行。

请注意，Page1 的尺寸在 Application_Start 中是未知的，它们仅在 Page1_Resize 例程的 Width 和 Height 参数中已知。
如果您想调整视图，您必须在此处进行。

3.3 B4J 程序流程

B4J 中的程序流程比 B4A 程序流程简单得多，类似于 B4i。

当我们运行一个新项目时，我们会得到以下模板：

```
Sub Process_Globals
   Private fx As JFX
   Private MainForm As Form
End Sub

Sub AppStart (Form1 As Form, Args() As String)
   MainForm = Form1
   'MainForm.RootPane.LoadLayout("Layout1") '加载布局文件.
   MainForm.Show
End Sub

'返回 true 以允许默认异常处理程序处理未捕获的异常.
Sub Application_Error (Error As Exception, StackTrace As String) As Boolean
   Return True
End Sub
```

当您启动程序时，例程将按上述顺序执行。

如果要在用户调整表单大小时调整节点，则必须为此表单添加 Resize 例程，例如：

```
Private Sub MainForm_Resize (Width As Double, Height As Double)
   '你的代码
End Sub
```

如果您在设计器中使用锚点，则在大多数情况下不需要调整大小事件。

3.4 B4R 程序流程

B4R 中的程序流程是直截了当的。

当我们运行一个新项目时，我们会发现这个代码模板：

```
Sub Process_Globals
   '这些全局变量将在应用程序启动时声明一次。
   '可以从所有模块访问公共变量。
   Public Serial1 As Serial
End Sub

Private Sub AppStart
   Serial1.Initialize(115200)
   Log("AppStart")
End Sub
```

运行程序时，会执行 Process_Globals，然后执行 AppStart。

```
Serial1.Initialize(115200)   初始化比特率。
Log("AppStart")              在日志中写入“AppStart”。
```

3.5 B4A / B4i / B4J 程序流程比较

3.5.1 程序启动 B4A / B4i / B4J

B4A	B4i	B4J
Main Process_Globals	Main Process_Globals	Main Process_Globals
Starter Process_Globals		
其他 modules Process_Globals	其他 modules Process_Globals	其他 modules Process_Globals
Starter Service_Create	Main Application_Start	Main AppStart
Starter Service_Start	Main Page1_Resize	Main MainForm_Resize
Main Globals		
Main Activity_Create FirstTime = True		
Main Activity_Resume		

3.5.2 旋转装置 B4A / B4i

B4A	B4i
Main Activity_Pause	
Main Globals	Main Page1_Resize
Main Activity_Create FirstTime = False	
Main Activity_Resume	

3.6 B4XPages 程序流程

对于具有 B4XPages 库的跨平台项目，所有三个平台的程序流程都是相同的。 所有平台特定的代码都隐藏在 B4XPages 库中，对程序员是透明的。

B4XPages 跨平台项目手册中的 B4XPagesThreePages 项目显示了在页面之间导航时的程序流程。

例子：
项目启动，执行如下例程：

- MainPage Create 主页创建
- MainPage Foreground 主页前景
- MainPage Appear 主页出现
- ainPage Resize 主页调整大小

打开一个页面，示例中为 Page2：

- Page2 Create Page2 创建
- Page2 Foreground Page2 前景
- Page2 Appear Page2 出现

关闭页面，示例中为 Page2：

- Page2 Disappear Page2 消失

4 B4X 语言

4.1 表达式

编程语言中的表达式是根据特定编程语言的特定优先级和关联规则解释的显式值、常量、变量、运算符和函数的组合，它计算然后产生（返回）另一个值。 这个过程，就像数学表达式一样，被称为求值。 该值可以是各种类型，例如数字、字符串和逻辑（来源 Wikipedia）。

例如，2 + 3 是算术和编程表达式，其计算结果为 5。变量是表达式，因为它是指向内存中值的指针，因此 y + 6 是表达式。 关系表达式的一个示例是 4 = 4，其计算结果为 True（来源 Wikipedia）。

4.1.1 数学表达式

运算符	例子	优先级	运算
+	x + y	3	添加
-	x - y	3	减法
*	x * y	2	乘法
/	x / y	2	分配
Mod	x Mod y	2	模数
Power	Power(x,y) x^y	1	次方

优先级：在表达式中，级别 1 的操作在级别 2 的操作之前进行评估，而级别 2 的操作在级别 3 的操作之前进行评估。

例子：

```
4 + 5 * 3 + 2 = 21              >   4 + 15 + 2

(4 + 5) * (3 + 2)  =  45        >   9 * 5

(4 + 5)² * (3 + 2)  =  405      >  9² * 5    >  81 * 5
Power(4 + 5, 2) * (3 + 2)

11 Mod 4 = 3                    >   Mod 是 11 / 4 的余数

23³   Power(23, 3)              >   23 的 3 次方

  -2²  = -4
(-2)²  = 4
```

4.1.2 关系表达式

在关系表达式的计算机科学中，运算符测试两个实体之间的某种关系。 这些包括数值相等（例如，5 = 5）和不等式
（例如，4 >= 3）。
在 B4X 中，这些运算符返回 **True** 或 **False**，这取决于两个操作数之间的条件关系是否成立。

运算符	例子	用于测试
=	x = y	两个值的等价
<>	x <> y	两个值的否定等价
>	x > y	如果左边表达式的值大于右边的值
<	x < y	如果左侧表达式的值小于右侧表达式的值
>=	x >= y	如果左表达式的值大于或等于右表达式的值
<=	x <= y	如果左侧表达式的值小于或等于右侧表达式的值

4.1.3 布尔表达式

在计算机科学中，Boolean expression 布尔表达式是在计算时产生 Boolean 布尔值的表达式，即 **True** 或 **False** 之一。 布尔表达式可以由布尔常量 **True** 或 **False**、布尔类型变量、布尔值运算符和布尔值函数（来源 Wikipedia）的组合组成。

布尔运算符用于条件语句，例如 IF-Then 和 Select-Case。

运算符	评论
Or 或	布尔 或 Z = X Or Y Z = True 如果 X 或 Y 等于 True 或 布两者都是 True
And 与	布尔 与 Z = X And Y Z = True 如果 X 和 Y 都等于 True
Not 非( )	布尔 非 X = True Y = Not(X) > Y = False

		Or 或	And 与
X	Y	Z	Z
False	False	False	False
True	False	True	False
False	True	True	False
True	True	True	True

4.2 标准关键字

并非所有关键字都在 B4R 中可用。

- Abs (Number As Double) As Double
- ACos (Value As Double) As Double
- ACosD (Value As Double) As Double
- Array
- Asc (Char As Char) As Int
- ASin (Value As Double) As Double
- ASinD (Value As Double) As Double
- ATan (Value As Double) As Double
- ATan2 (Y As Double, X As Double) As Double
- ATan2D (Y As Double, X As Double) As Double
- ATanD (Value As Double) As Double
- BytesToString (Data() As Byte, StartOffset As Int, Length As Int, CharSet As String) As String
- CallSub (Component As Object, Sub As String) As Object
- CallSub2 (Component As Object, Sub As String, Argument As Object) As Object
- CallSub3 (Component As Object, Sub As String, Argument1 As Object, Argument2 As Object) As Object
- CallSubDelayed (Component As Object, Sub As String)
- CallSubDelayed 2 (Component As Object, Sub As String, Argument As Object)
- CallSubDelayed 3 (Component As Object, Sub As String, Argument1 As Object, Argument2 As Object)
- Catch
- cE As Double
- Ceil (Number As Double) As Double
- CharsToString (Chars() As Char, StartOffset As Int, Length As Int) As String
- Chr (UnicodeValue As Int) As Char
- Continue
- Cos (Radians As Double) As Double
- CosD (Degrees As Double) As Double
- cPI As Double
- CreateMap
- CRLF As String
- Dim
- Exit
- False As Boolean
- Floor (Number As Double) As Double

- For
- GetType (object As Object) As String
- If
- Is
- IsNumber (Text As String) As Boolean
- LoadBitmap (Dir As String, FileName As String) As Bitmap
- LoadBitmapResize (Dir As String, FileName As String, Width As Int, Height As Int, KeepAspectRatio As Boolean) As Bitmap

- LoadBitmapSample (Dir As String, FileName As String, MaxWidth As Int, MaxHeight As Int) As Bitmap
- Log (Message As String)
- Logarithm (Number As Double, Base As Double) As Double
- LogColor (Message As String, Color As Int)
- Max (Number1 As Double, Number2 As Double) As Double
- Me As Object
- Min (Number1 As Double, Number2 As Double) As Double
- Not (Value As Boolean) As Boolean
- Null As Object
- NumberFormat (Number As Double, MinimumIntegers As Int, MaximumFractions As Int) As String
- NumberFormat2 (Number As Double, MinimumIntegers As Int, MaximumFractions As Int, MinimumFractions As Int, GroupingUsed As Boolean) As String
- Power (Base As Double, Exponent As Double) As Double
- QUOTE As String
- Regex As Regex
- Return
- Rnd (Min As Int, Max As Int) As Int
- RndSeed (Seed As Long)
- Round (Number As Double) As Long
- Round2 (Number As Double, DecimalPlaces As Int) As Double
- Select
- Sender As Object
- Sin (Radians As Double) As Double
- SinD (Degrees As Double) As Double
- Sleep (Milliseconds As Int)
- SmartStringFormatter (Format As String, Value As Object) As String
- Sqrt (Value As Double) As Double
- Sub
- SubExists (Object As Object, Sub As String) As Boolean
- TAB As String
- Tan (Radians As Double) As Double
- TanD (Degrees As Double) As Double
- True As Boolean
- Try
- Type
- Until
- While

Abs (Number As Double) As Double
返回绝对值。

ACos (Value As Double) As Double
计算三角反余弦函数。 返回用弧度测量的角度。

ACosD (Value As Double) As Double
计算三角反余弦函数。 返回用度数测量的角度。

Array
创建指定类型的一维数组。
语法是：Array [As type]（值列表）。
如果省略类型，则将创建一个对象数组。
例子：

```
Dim Days() As String
Days = Array As String("Sunday", "Monday", ...)
```

Asc (Char As Char) As Int
返回给定字符或字符串中第一个字符的 unicode 代码点。

ASin (Value As Double) As Double
计算三角反正弦函数。 返回用弧度测量的角度。

ASinD (Value As Double) As Double
计算三角反正弦函数。 返回用度数测量的角度。

ATan (Value As Double) As Double
计算三角反正切函数。 返回用弧度测量的角度。

ATan2 (Y As Double, X As Double) As Double
计算三角反正切函数。 返回用弧度测量的角度。

ATan2D (Y As Double, X As Double) As Double
计算三角反正切函数。 返回用度数测量的角度。

ATanD (Value As Double) As Double
计算三角反正切函数。 返回用度数测量的角度。

BytesToString (Data() As Byte, StartOffset As Int, Length As Int, CharSet As String) As String

将给定的字节数组解码为字符串。
数据 - 字节数组。
StartOffset - 要读取的第一个字节。
长度 - 要读取的字节数。
CharSet - 字符集的名称。
例子：

```
Dim s As String
s = BytesToString(Buffer, 0, Buffer.Length, "UTF-8")
```

CallSub (Component As Object, Sub As String) As Object

调用给定的子程序。 CallSub 可用于调用属于不同模块的子程序。
但是，只有在其他模块没有暂停时才会调用子程序。 在这种情况下，将返回一个空字符串。
您可以使用 IsPaused 来测试模块是否已暂停。
这意味着一个活动不能调用不同活动的子程序。 因为其他活动肯定会暂停。
CallSub 允许活动调用子程序或服务调用活动子程序。
请注意，不能调用代码模块的子程序。
CallSub 也可以用于调用当前模块中的子程序。 在这种情况下，将 Me 作为组件传递。
例子：

```
CallSub(Main, "RefreshData")
```

CallSub2 (Component As Object, Sub As String, Argument As Object) As Object

类似于 CallSub。 使用单个参数调用 sub。

CallSub3 (Component As Object, Sub As String, Argument1 As Object, Argument2 As Object) As Object

CallSubDelayed (Component As Object, Sub As String)

CallSubDelayed 是 StartActivity、StartService 和 CallSub 的组合。
与仅适用于当前正在运行的组件的 CallSub 不同，CallSubDelayed 将在需要时首先启动目标组件。
CallSubDelayed 也可以用于调用当前模块中的子程序。 不是直接调用这些子程序，而是将一条消息发送到消息队列。
处理消息时将调用子程序。 这在您想要在当前子程序“之后”执行某些操作（通常与 UI 事件相关）的情况下很有用。
请注意，如果您在整个应用程序处于后台（没有可见活动）时调用 Activity，则一旦目标活动恢复，子程序将被执行。

CallSubDelayed2 (Component As Object, Sub As String, Argument As Object)

类似于 CallSubDelayed。 使用单个参数调用子程序。

CallSubDelayed3 (Component As Object, Sub As String, Argument1 As Object, Argument2 As Object)

类似于 CallSubDelayed。 使用两个参数调用子程序。

Catch

Any exception thrown inside a try block will be caught in the catch block.
Call LastException to get the caught exception.
Syntax:

```
Try
  ...
Catch
  ...
End Try
```

cE As Double

e (natural logarithm base) constant.

Ceil (Number As Double) As Double

Returns the smallest double that is greater or equal to the specified number and is equal to an integer.

CharsToString (Chars() As Char, StartOffset As Int, Length As Int) As String

Creates a new String by copying the characters from the array.
Copying starts from StartOffset and the number of characters copied equals to Length.

Chr (UnicodeValue As Int) As Char

Returns the character that is represented by the given unicode value.

Continue

Stops executing the current iteration and continues with the next one.

Cos (Radians As Double) As Double

Calculates the trigonometric cosine function. Angle measured in radians.

CosD (Degrees As Double) As Double

Calculates the trigonometric cosine function. Angle measured in degrees.

cPI As Double

PI constant.

CreateMap

Creates a Map with the given key / value pairs.
The syntax is: CreateMap (key1: value1, key2: value2, ...)
Example:

```
Dim m As Map = CreateMap("January": 1, "February": 2)
```

CRLF As String
New line character. The value of Chr(10).

Dim
Declares a variable.
Syntax:
Declare a single variable:
Dim variable name [As type] [= expression]
The default type is String.

Declare multiple variables. All variables will be of the specified type.
Dim [Const] variable1 [= expression], variable2 [= expression], ..., [As type]
Note that the shorthand syntax only applies to Dim keyword.
Example: `Dim a = 1, b = 2, c = 3 As Int`

Declare an array:
Dim variable(Rank1, Rank2, ...) [As type]
Example: `Dim Days(7) As String`
The actual rank can be omitted for zero length arrays.

Exit

Exits the most inner loop.
Note that Exit inside a Select block will exit the Select block.

False As Boolean

Floor (Number As Double) As Double

返回小于或等于指定数字且等于整数的最大双精度数。

For

Syntax:
```
For variable = value1 To value2 [Step interval]
  ...
Next
```
If the iterator variable was not declared before it will be of type Int.

Or:
```
For Each variable As type In collection
  ...
Next
```
Examples:
```
For i = 1 To 10
  Log(i) 'Will print 1 to 10 (inclusive).
Next
For Each n As Int In Numbers 'an array
  Sum = Sum + n
Next
```

Note that the loop limits will only be calculated once before the first iteration.

GetType (object As Object) As String

Returns a string representing the object's java type.

If

Single line:
```
If condition Then true-statement [Else false-statement]
```
Multiline:
```
If condition Then
  statement
Else If condition Then
  statement
  ...
Else
  statement
End If
```

IIf

Inline If - returns TrueValue if Condition is True and False otherwise. Only the relevant expression is evaluated.

```
IIf (Condition As BOOL, TrueValue As Object, FalseValue As Object)
```

Is

Tests whether the object is of the given type.
Note that when a number is converted to object it might change its type to a different type of number
(for example a Byte might be converted to an Int).
Example:
```
For Each v As View in Page1.RootPanel.GetAllViewsRecursive
   If v Is Button Then
      Dim b As Button = v
      b.Color = Colors.Blue
   End If
Next
```

IsNumber (Text As String) As Boolean

Tests whether the specified string can be safely parsed as a number.

LoadBitmap (Dir As String, FileName As String) As Bitmap

Loads the bitmap.
Note that the Android file system is case sensitive.
You should consider using LoadBitmapSample if the image size is large.
The actual file size is not relevant as images are usually stored compressed.
Example:
```
Activity.SetBackgroundImage(LoadBitmap(File.DirAssets, "SomeFile.jpg"))
```



```
LoadBitmapResize (Dir As String, FileName As String, Width As Int, Height As Int,
KeepAspectRatio As Boolean) As Bitmap
```

Loads the bitmap and sets its size.
The bitmap scale will be the same as the device scale.
Unlike LoadBitmapSample which requires the container Gravity to be set to FILL,
LoadBitmapResize provides better results when the Gravity is set to CENTER.
Example:

```
Dim bd As BitmapDrawable = Activity.SetBackgroundImage(LoadBitmapResize(File.DirAssets,
"SomeFile.jpg", 100%x, 100%y, True))
bd.Gravity = Gravity.CENTER
```

Or:

```
Activity.SetBackgroundImage(LoadBitmapResize(File.DirAssets, "SomeFile.jpg", 100%x,
100%y, True)).Gravity = Gravity.CENTER
```

```
LoadBitmapSample (Dir As String, FileName As String, MaxWidth As Int, MaxHeight As Int) As
Bitmap
```

Loads the bitmap.
The decoder will subsample the bitmap if MaxWidth or MaxHeight are smaller than the bitmap dimensions.
This can save a lot of memory when loading large images.
Example:

```
Panel1.SetBackgroundImage(LoadBitmapSample(File.DirAssets, "SomeFile.jpg",
Panel1.Width, Panel1.Height))
```

```
Log (Message As String)
```

Logs a message. The log can be viewed in the Logs tab.

```
Logarithm (Number As Double, Base As Double) As Double
```

```
LogColor (Message As String, Color As Int)
```

Logs a message. The message will be displayed in the IDE with the specified color.

```
Max (Number1 As Double, Number2 As Double) As Double
```

Returns the larger number between the two numbers.

```
Me As Object
```

For classes: returns a reference to the current instance.
For activities and services: returns a reference to an object that can be used with CallSub, CallSubDelayed and SubExists keywords.
Cannot be used in code modules.

```
Min (Number1 As Double, Number2 As Double) As Double
```

Returns the smaller number between the two numbers.

Not (Value As Boolean) As Boolean

Inverts the value of the given boolean.

Null As Object

NumberFormat (Number As Double, MinimumIntegers As Int, MaximumFractions As Int) As String

Converts the specified number to a string.
The string will include at least Minimum Integers and at most Maximum Fractions digits.
Example:
```
Log(NumberFormat(12345.6789, 0, 2)) '"12,345.68"
Log(NumberFormat(1, 3 ,0)) '"001"
```

NumberFormat2 (Number As Double, MinimumIntegers As Int, MaximumFractions As Int, MinimumFractions As Int, GroupingUsed As Boolean) As String

Converts the specified number to a string.
The string will include at least Minimum Integers, at most Maximum Fractions digits and at least Minimum Fractions digits.
GroupingUsed - Determines whether to group every three integers.
Example:
```
Log(NumberFormat2(12345.67, 0, 3, 3, false)) '"12345.670"
```

Power (Base As Double, Exponent As Double) As Double

Returns the Base value raised to the Exponent power.

QUOTE As String

Quote character ". The value of Chr(34).

Regex As Regex

Regular expressions related methods.

Return

Returns from the current sub and optionally returns the given value.
Syntax: `Return [value]`

Rnd (Min As Int, Max As Int) As Int
Returns a random integer between Min (inclusive) and Max (exclusive).

RndSeed (Seed As Long)
Sets the random seed value.
This method can be used for debugging as it allows you to get the same results each time.

Round (Number As Double) As Long
Returns the closest long number to the given number.

Round2 (Number As Double, DecimalPlaces As Int) As Double
Rounds the given number and leaves up to the specified number of fractional digits.

Select

Compares a single value to multiple values.
Example:
```
Dim value As Int
value = 7
Select value
  Case 1
    Log("One")
  Case 2, 4, 6, 8
    Log("Even")
  Case 3, 5, 7, 9
    Log("Odd larger than one")
  Case Else
    Log("Larger than 9")
End Select
```

Sender As Object

Returns the object that raised the event.
Only valid while inside the event sub.
Example:
```
Sub Button_Click
  Dim b As Button
  b = Sender
  b.Text = "I've been clicked"
End Sub
```

Sin (Radians As Double) As Double

Calculates the trigonometric sine function. Angle measured in radians.

SinD (Degrees As Double) As Double

Calculates the trigonometric sine function. Angle measured in degrees.

Sleep (Value As Double) As Double

Pauses the current sub execution and resumes it after the specified time.

SmartStringFormatter (Format As String, Value As Object) As String
Internal keyword used by the Smart String literal.

Sqrt (Value As Double) As Double

Returns the positive square root.

Sub

Declares a sub with the parameters and return type.
Syntax: Sub name [(list of parameters)] [As return-type]
Parameters include name and type.
The lengths of arrays dimensions should not be included.
Example:

```
Sub MySub (FirstName As String, LastName As String, Age As Int, OtherValues() As
Double) As Boolean
  ...
End Sub
```

In this example OtherValues is a single dimension array.
The return type declaration is different than other declarations as the array parenthesis follow the type and not
the name (which does not exist in this case).

SubExists (Object As Object, Sub As String) As Boolean

Tests whether the object includes the specified method.
Returns false if the object was not initialized or not an instance of a user class.

TAB As String

Tab character.

Tan (Radians As Double) As Double

Calculates the trigonometric tangent function. Angle measured in radians.

TanD (Degrees As Double) As Double

Calculates the trigonometric tangent function. Angle measured in degrees.

True As Boolean

Try

Any exception thrown inside a try block will be caught in the catch block.
Call LastException to get the caught exception.
Syntax:

```
Try
...
Catch
...
End Try
```

Type

Declares a structure.
Can only be used inside sub Globals or sub Process_Globals.
Syntax:
Type type-name (field1, field2, ...)
Fields include name and type.
Example:

```
Type MyType (Name As String, Items(10) As Int)
Dim a, b As MyType
a.Initialize
a.Items(2) = 123
```

Until

Loops until the condition is true.
Syntax:

```
Do Until condition
  ...
Loop
```

While

Loops while the condition is true.
Syntax:

```
Do While condition
...
Loop
```

4.3 条件语句

B4X 中提供了不同的条件语句。

4.3.1 If - Then - Else

If-Then-Else 结构允许操作条件测试并根据测试结果执行不同的代码段。
一般情况：

```
If test1 Then
   '代码1
Else If test2 Then
   '代码2
Else
   '代码3
End If
```

If-Then-Else 结构的工作原理如下：
1. 到达带有 **If** 关键字的行时，执行 test1。
2、如果测试结果为 True，则执行 **代码 1**，直到带有 **Else If** 关键字的那一行。并跳转到 **End If** 关键字之后的行并继续。
3. 如果结果为 False，则执行 **test2**。
4. 如果测试结果为 True，则执行 **代码 2**，直到包含 **Else** 关键字的那一行。
并跳转到 **End If** 关键字之后的行并继续。
5. 如果结果为 False，则执行 **代码 3** 并在 **End If** 关键字后面的行继续。

测试可以是任何类型的条件测试，有两种可能性 True 或 False。
一些例子：

```
If b = 0 Then
   a = 0                                  最简单的 If-Then 结构。
End If

If b = 0 Then a = 0                       相同，但在一行中。

If b = 0 Then
   a = 0                                  最简单的 If-Then-Else 结构。
Else
   a = 1
End If

If b = 0 Then a = 0 Else a = 1            相同，但在一行中。
```

就个人而言，我更喜欢多行的结构，更好的可读性。
几十年前 HP Basic 的一个旧习惯，这个 Basic 每行只接受一条指令。

笔记： 以下两个区别：

B4X	VB
Else If	**ElseIf**

在 B4X 中，**Else** 和 **If** 之间有一个空白字符。

一些用户尝试使用这种表示法：

```
If b = 0 Then a = 0 : c = 1
```

B4X 和 VB 之间有很大的不同会导致错误：
上述语句等价于：

B4X

```
If b = 0 Then
   a = 0
End If
c = 1
```

VB

```
If b = 0 Then
   a = 0
   c = 1
End If
```

以上一行中的冒号字符 ‘ ： ’ 在 B4X 中被视为 CarriageReturn CR 字符。

此结构会引发错误。

```
Sub Plus1 : x = x + 1 : End Sub
```

您不能在同一行有一个 Sub 声明和 End Sub。

4.3.1.1 布尔求值顺序

在这个例子中：

```
If InitVar2(Var1) and Var1 > Var2 then ....
```

如果 InitVar2(Var1) 返回 false 是停止评估还是没有规则？

它从左到右运行，并在确定结果后立即停止（短路评估）。

这个非常重要。
它允许编写代码，例如：

```
If i < List.Size And List.Get(i) = "abc" Then
```

4.3.2 IIf 内联如果

IIf - 内联如果 Inline If，也称为三元如果 *ternary if*，因为它是具有三个参数的运算符。

```
Label1.Text = IIf(EditText1.Text <> "", EditText1.Text, "Please enter value")
```

IIf 基本上相当于这个子：

```
Sub PseudoIIf (Condition As Boolean, TrueValue As Object, FalseValue As Object) As
Object
   If Condition = True Then Return TrueValue Else Return FalseValue
End Sub
```

与这个 sub 不同，IIf 关键字将只计算相关表达式。 这意味着此代码将正常工作：

```
Return IIf(List1.Size > 0, List1.Get(0), "List is empty")
```

（还有一个与返回类型有关的细微差别。如果使用新的 As 方法显式设置它，编译器将避免将值转换为 Object 并返回到目标类型。这仅在非常紧凑和长循环中有意义）。

4.3.3 Select - Case

Select - Case 结构允许将 TestExpression 与其他**表达式**进行比较，并根据 TestExpression 和**表达式**之间的匹配执行不同的代码段。

一般情况：

```
Select TestExpression
    Case ExpressionList1
        ' code1
    Case ExpressionList2
        ' code2
    Case Else
        ' code3
End Select
```

TestExpression 是要测试的表达式。
ExpressionList1 是要与 **TestExpression** 进行比较的表达式列表
ExpressionList2 是要与 **TestExpression** 进行比较的另一个表达式列表

Select - Case 结构的工作原理如下：

1. 评估 **TestExpression**。
2. 如果 **ExpressionList1** 中的一个元素与 **TestExpression** 匹配，则执行 **code1** 并在 **End Select** 关键字之后的行继续。
3. 如果 **ExpressionList2** 中的一个元素与 **TestExpression** 匹配，则执行 **code2** 并继续 **End Select** 关键字之后的行。
4. 对于没有匹配的表达式，**TestExpression** 执行 **code3** 并在 **End Select** 关键字之后的行继续。

TestExpression 可以是任何表达式或值。
ExpressionList1 是任何表达式或值的列表。

例子：

```
Select Value
    Case 1, 2, 3, 4
```

Value 变量是一个数值。

```
Select a + b
    Case 12, 24
```

TestExpression 是 a + b 的总和

```
Select Txt.CharAt(Index)
    Case "A", "B", "C"
```

TestExpression 是给定索引处的字符。

```
Sub Activity_Touch (Action As Int, X As Float, Y As Float)
    Select Action
        Case Activity.ACTION_DOWN

        Case Activity.ACTION_MOVE

        Case Activity.ACTION_UP

    End Select
End Sub
```

笔记。 以下两者之间的差异：

B4X

```
Select Value
    Case 1, 2, 3, 4, 8, 9, 10
```

VB

```
Select Case Value
    Case 1 To 4, 8 To 9
```

在 VB 中，关键字 `Case` 被添加在 `Select` 关键字之后。
VB 接受 `Case 1 To 4` ，这在 B4X 中没有实现。

4.4 循环结构

B4X 中提供了不同的循环结构。

4.4.1 For - Next

在 For-Next 循环中，代码块将被执行一定次数。
例子：

```
For i = n1 To n2 Step n3
   ' 代码块
Next
```

i 增量变量
n1 初始值
n2 终值
n3 步

For - Next 循环的工作原理如下：

1. 一开始，增量变量 i 等于初始值 n1。
   i = n1
2. 执行 **For** 和 **Next** 关键字之间的具体代码。
3. 到达 Next 时，增量变量 i 增加步长值 n3。
   i = i + n3。
4. 程序跳转回 **For**，比较增量变量 i 是否小于或等于最终值 n2。
   测试 i <= n2
5. 如果**是**，程序继续执行第 2 步，即 **For** 关键字之后的行。
6. 如果**否**，程序在 **Next** 关键字后面的行继续。

如果步长值等于“+1”，则不需要步长关键字。

```
For i = 0 To 10
Next
```

是相同

```
For i = 0 To 10 Step 1
Next
```

步长变量可以是负数。

```
For i = n3 To 0 Step -1

Next
```

可以使用 Exit 关键字退出 For - Next 循环。

```
For i = 0 To 10
   ' code
   If A = 0 Then Exit
   ' code
Next
```

10 在本例中，如果变量 a 等于 0
然后退出循环。

注意：以下两者之间的差异

B4X	VB
`Next`	`Next i`
`Exit`	`Exit For`

在 VB 中：

- 在 **Next** 关键字之后添加增量变量。
- 在 **Exit** 关键字之后指定循环类型。

4.4.2 For - Each

它是 For - Next 循环的变体。

例子：

```
For Each n As Type In Array
   ' 具体代码
Next
```

n	变量任何类型或对象
Type	变量 n 的类型
Array	值或对象数组

For - Each 循环的工作原理如下：

1. 一开始，n 获取 Array 中第一个元素的值。
   n = Array(0)
2. 执行 **For** 和 **Next** 关键字之间的具体代码。
3. 当到达 **Next** 时，程序检查 n 是否是数组中的最后一个元素。
4. 如果**否**，变量 n 获取数组中的下一个值并继续执行步骤 2，即 **For** 关键字之后的行。
   n = Array(next)
5. 如果**是**，程序在 **Next** 关键字后面的行继续。

For - Each 例子:

```
Private Numbers() As Int
Private Sum As Int

Numbers = Array As Int(1, 3, 5 , 2, 9)

Sum = 0
For Each n As Int In Numbers
   Sum = Sum + n
Next
```

相同的示例，但带有 For - Next 循环:

```
Private Numbers() As Int
Private Sum As Int
Private i As Int

Numbers = Array As Int(1, 3, 5 , 2, 9)

Sum = 0
For i = 0 To Numbers.Length - 1
   Sum = Sum + Numbers(i)
Next
```

此示例显示了 For - Each 循环的强大功能：

```
For Each lbl As Label In Activity
   lbl.TextSize = 20
Next
```

For - Next 循环的相同示例：

```
For i = 0 To Activity.NumberOfViews - 1
   Private v As View
   v = Activity.GetView(i)
   If v Is Label Then
      Private lbl As Label
      lbl = v
      lbl.TextSize = 20
   End If
Next
```

4.4.3 Do - Loop

存在几种配置：

```
Do While test          test 是任何表达式
   ' 代码              在 test 为 True 时执行代码
Loop

Do Until test          test 是任何表达式
   ' 代码              执行代码直到 test 为 True
Loop
```

Do While-Loop 循环的工作原理如下：

1. 一开始，对 **test** 进行评估。
2. 如果为 True，则执行代码
3. 如果 False 在 **Loop** 关键字之后的行继续。

Do Until-Loop 循环的工作原理如下：

1. 一开始，对 **test** 进行评估。
2. 如果为 False，则执行**代码**
3. 如果 True 在 **Loop** 关键字之后的行继续。

可以使用 Exit 关键字退出 Do-Loop 结构。

```
Do While test
   ' 代码
   If a = 0 Then Exit          如果 a = 0 则退出循环
   ' 代码
Loop
```

例子：

Do Until Loop:
```
Private i, n As Int

i = 0
Do Until i = 10
   ' 代码
   i = i + 1
Loop
```

Do While Loop:
```
Private i, n As Int

i = 0
Do While i < 10
   ' 代码
   i = i + 1
Loop
```

读取一个文本文件并填写一个列表：
```
Private lstText As List
Private line As String
Private tr As TextReader

tr.Initialize(File.OpenInput(File.DirInternal, "test.txt"))
lstText.Initialize
line = tr.ReadLine
Do While line <> Null
   lstText.Add(line)
   line = tr.ReadLine
Loop

tr.Close
```

注意：以下两者之间的差异

B4X	VB
`Exit`	`Exit Loop`

在 VB 中，循环类型在 **Exit** 关键字之后指定。

VB 还接受以下循环，这些循环在 B4X 中不受支持。
```
Do                          Do
  ' 代码                      ' 代码
Loop While test             Loop Until test
```

4.5 内联转换 As

As - Inline casting. Allows inline casting from one type to another. Some examples:

```
   Dim Buttons As List = Array(Button1, Button2, Button3, Button4, Button5)
   Dim s As String = Buttons.Get(2).As(B4XView).Text
   Buttons.Get(2).As(B4XView).Text = "abc"
   Dim j As String = $"{
data: {
key1: value1,
complex_key2: {key: value2}
},
items: [0, 1, 2]
}"$

   Dim parser As JSONParser
   parser.Initialize(j)
   Dim m As Map = parser.NextObject
   Dim value1 As String = m.Get("data").As(Map).Get("key1")
   Dim value2 As String = m.Get("data").As(Map).Get("complex_key2").As(Map).Get("key")
```

And, for B4J:

```
Button1.As(JavaObject).RunMethod("setMouseTransparent", Array(True))
```

It can also be used with numbers, which is especially useful when calling external APIs with JavaObject, as the types need to be exact (for B4J):

```
   Log(Me.As(JavaObject).RunMethod("sum", Array((10).As(Float), (20).As(Double))))
   'equivalent to:
   Dim jme As JavaObject = Me
   Dim f As Float = 10
   Dim d As Double = 20
   Log(jme.RunMethod("sum", Array(f, d)))

#if Java
public double sum(float n1, double n2) {
return n1 + n2;
}
#End If
```

4.6 Subs

A Subroutine ("Sub") is a piece of code. It can be any length, and it has a distinctive name and a defined scope (in the means of variables scope discussed earlier). In B4X code, a subroutine is called "Sub", and is equivalent to procedures, functions, methods and subs in other programming languages. The lines of code inside a Sub are executed from first to last, as described in the program flow chapter.
It is not recommended to have Subs with a large amount of code, they get less readable.

4.6.1 Declaring

A Sub is declared in the following way:

```
Sub CalcInterest(Capital As Double, Rate As Double) As Double
   Return Capital * Rate / 100
End Sub
```

It starts with the keyword **Sub**, followed by the Sub's name, followed by a parameter list, followed by the return type and ends with the keywords **End Sub**.
Subs are always declared at the top level of the module, you cannot nest two Subs one inside the other.

4.6.2 Calling a Sub

When you want to execute the lines of code in a Sub, you simply write the Sub's name.

For example:
```
   Interest = CalcInterest(1234, 5.2)
```

`Interest` Value returned by the Sub.
`CalcInterest` Sub name.
`1235` Capital value transmitted to the Sub.
`5.25` Rate value transmitted to the Sub.

4.6.3 Calling a Sub from another module

A subroutine declared in a code module can be accessed from any other module but the name of the routine must have the name of the module where it was declared as a prefix.

Example: If the `CalcInterest` routine is declared in module `MyModule` then calling the routine must be:
```
   Interest = MyModule.CalcInterest(1234, 5.2)
```

instead of:
```
   Interest = CalcInterest(1234, 5.2)
```

4.6.4 Naming

Basically, you can name a Sub any name that's legal for a variable. It is recommended to name the Sub with a significant name, like **CalcInterest** in the example, so you can tell what it does from reading the code.
There is no limit on the number of Subs you can add to your program, but it is not allowed to have two Subs with the same name in the same module.

4.6.5 Parameters

Parameters can be transmitted to the Sub. The list follows the sub name. The parameter list is put in brackets.
The parameter types should be declared directly in the list.

```
Sub CalcInterest(Capital As Double, Rate As Double) As Double
   Return Capital * Rate / 100
End Sub
```

In B4X, parameters are transmitted by value and not by reference.

4.6.6 Returned value

A sub can return a value, this can be any object.
Returning a value is done with the Return keyword.
The type of the return value is added after the parameter list.

```
Sub CalcInterest(Capital As Double, Rate As Double) As Double
   Return Capital * Rate / 100
End Sub
```

You can return any object.

```
Sub InitList As List
   Private MyList As List
   MyList.Initialize

   For i = 0 To 10
       MyList.Add("Test" & i)
   Next
   Return MyList
End Sub
```

If you want to return an array then you need to add a parenthesis at the end of the object type.

```
Sub StringArray As String ()
   Public strArr(2) As String
   strArr(0) = "Hello"
   strArr(1) = "world!"
   Return strArr
End Sub
```

If you want to return a multidimentional array you need to add comma per supplementary dimension.
One comma for a two-dimensional array.

```
Sub StringMatrix As String (,)
   Public strMatrix(2,2) As String
   strMatrix(1,1) = "Hello world!"
   Return strMatrix
End Sub
```

4.7 Resumable Subs

Resumable subs is a new feature added in B4A v7.00 / B4i v4.00 / B4J v5.50. It dramatically simplifies the handling of asynchronous tasks.
(This feature is a variant of stackless coroutines.)

You find more examples in the forum.

The special feature of resumable subs is that they can be paused, without pausing the executing thread, and later be resumed.
The program doesn't wait for the resumable sub to be continued. Other events will be raised as usual.

Any sub with one or more calls to Sleep or Wait For is a resumable sub.
The IDE shows this indicator  next to the sub declaration:

```
Private Sub CountDown(Start As Int)
   For i = Start To 0 Step -1
      Label1.Text = i
      Sleep(1000)
   Next
End Sub
```

4.7.1 Sleep

Pauses the current sub execution and resumes it after the specified time.

Sleep (Milliseconds As Int) Milliseconds, time delay in milliseconds.
Example:
Sleep(1000)

Using Sleep is simple:

```
Log(1)
Sleep(1000)
Log(2)
```

The sub will be paused for 1000 milliseconds and then be resumed.

You can call Sleep(0) for the shortest pause. This can be used to allow the UI to be refreshed. It is a good alternative to DoEvents (which doesn't exist in B4J and B4i and should be avoided in B4A).

```
Sub VeryBusySub
   For i = 1 To 10000000
      'do something
      If i Mod 1000 = 0 Then Sleep(0) 'allow the UI to refresh every 1000 iterations.
   Next
   Log("finished!")
End Sub
```

4.7.2 Wait For

B4X programming languages are event driven. Asynchronous tasks run in the background and raise an event when the task completes.
With the new Wait For keyword you can handle the event inside the current sub.

For example, this code will wait for the GoogleMap Ready event (B4J example):

```
Sub AppStart (Form1 As Form, Args() As String)
   MainForm = Form1
   MainForm.RootPane.LoadLayout("1") 'Load the layout file.

   gmap.Initialize("gmap")
   Pane1.AddNode(gmap.AsPane, 0, 0, Pane1.Width, Pane1.Height)
   MainForm.Show
   Wait For gmap_Ready '<----------------
   gmap.AddMarker(10, 10, "Marker")
End Sub
```

A bit more complicated example with FTP:
Listing all files in a remote folder and then downloading all the files:

```
Sub DownloadFolder (ServerFolder As String)
  FTP.List(ServerFolder)
  Wait For FTP_ListCompleted (ServerPath As String, Success As Boolean, Folders() As
    FTPEntry, Files() As FTPEntry) '<----
  If Success Then
    For Each f As FTPEntry In Files
      FTP.DownloadFile(ServerPath & f.Name, False, File.DirApp, f.Name)
      Wait For FTP_DownloadCompleted (ServerPath2 As String, Success As Boolean) '<----
      Log($"File ${ServerPath2} downloaded. Success = ${Success}"$)
     Next
  End If
  Log("Finish")
End Sub
```

When the Wait For keyword is called, the sub is paused and the internal events dispatcher takes care to resume it when the event is raised. If the event is never raised then the sub will never be resumed. The program will still be completely responsive.
If Wait For is later called with the same event then the new sub instance will replace the previous one.

Lets say that we want to create a sub that downloads an image and sets it to an ImageView:

```
'Bad example. Don't use.
Sub DownloadImage(Link As String, iv As ImageView)
   Dim job As HttpJob
   job.Initialize("", Me) 'note that the name parameter is no longer needed.
   job.Download(Link)
   Wait For JobDone(job As HttpJob)
   If job.Success Then
      iv.SetImage (job.GetBitmap) 'replace with iv.Bitmap = job.GetBitmap in B4A / B4i
   End If
   job.Release
End Sub
```

It will work properly if we call it once (more correctly, if we don't call it again before the previous call completes).
If we call it like this:

```
   DownloadImage("https://www.b4x.com/images3/android.png", ImageView1)
   DownloadImage("https://www.b4x.com/images3/apple.png", ImageView2)
```

Then only the second image will show because the second call to Wait For JobDone will overwrite the previous one.
This brings us to the second variant of Wait For.
To solve this issue, Wait For can distinguish between events based on the event sender.
This is done with an optional parameter:

Wait For (<sender>) <event signature>

Example:

```
'Good example. Use.
Sub DownloadImage(Link As String, iv As ImageView)
   Dim job As HttpJob
   job.Initialize("", Me) 'note that the name parameter is no longer needed.
   job.Download(Link)
   Wait For (job) JobDone(job As HttpJob)
   If job.Success Then
      iv.SetImage (job.GetBitmap) 'replace with iv.Bitmap = job.GetBitmap in B4A / B4i
   End If
   job.Release
End Sub
```

With the above code, each resumable sub instance will wait for a different event and will not be affected by other calls.

The difference is in the Wait For lines:
Bad: `Wait For JobDone(job As HttpJob)`
Good: `Wait For (job) JobDone(job As HttpJob)`

4.7.3 Code Flow

```
Sub S1
   Log("S1: A")
   S2
   Log("S1: B")
End Sub

Sub S2
   Log("S2: A")
   Sleep(0)
   Log("S2: B")
End Sub
```

The output is:
S1: A
S2: A
S1: B
S2: B

Whenever Sleep or Wait For are called, the current sub is paused. This is equivalent to calling Return.

4.7.4 Waiting for a resumable sub to complete

When one sub calls a second resumable sub, the code in the first sub will continue after the first Sleep or Wait For call (in the second sub).

If you want to wait for the second sub to complete then you can raise an event from the second sub and wait for it in the first:

```
Sub FirstSub
   Log("FirstSub started")
   SecondSub
   Wait For SecondSub_Complete
   Log("FirstSub completed")
End Sub

Sub SecondSub
   Log("SecondSub started")
   Sleep(1000)
   Log("SecondSub completed")
   CallSubDelayed(Me, "SecondSub_Complete")
End Sub
```

Logs:
FirstSub started
SecondSub started
SecondSub completed
FirstSub completed

Notes:
- It is safer to use CallSubDelayed than CallSub. CallSub will fail if the second sub is never paused (for example if the sleep is only called based on some condition).
- There is an assumption here that FirstSub will not be called again until it is completed.

4.7.5 Resumable Sub return value

Resumable subs can return a *ResumableSub* value.

Example:
```
Sub Button1_Click
   Sum(1, 2)
   Log("after sum")
End Sub

Sub Sum(a As Int, b As Int)
   Sleep(100) 'this will cause the code flow to return to the parent
   Log(a + b)
End Sub
```

```
Output:
after sum
3
```

This is the reason why it is not possible to simply return a value.

Solution.

Resumable subs can return a new type named ResumableSub. Other subs can use this value to wait for the sub to complete and get the desired return value.

```
Sub Button1_Click
   Wait For(Sum(1, 2)) Complete (Result As Int)
   Log("result: " & Result)
   Log("after sum")
End Sub

Sub Sum(a As Int, b As Int) As ResumableSub
   Sleep(100)
   Log(a + b)
   Return a + b
End Sub
```

```
Output:
3
result: 3
after sum
```

The above Button1_Click code is equivalent to:

```
Sub Button1_Click
   Dim rs As ResumableSub = Sum(1, 2)
   Wait For(rs) Complete (Result As Int)
   Log("result: " & Result)
   Log("after sum")
End Sub
```

The steps required are:

1. Add *As ResumableSub* to the resumable sub signature.
2. Call Return with the value you like to return.
3. In the calling sub, call the resumable sub with Wait For (<sub here>) Complete (Result As <matching type>)

Notes & Tips:

- If you don't need to return a value but still want to wait for the resumable sub to complete then return Null from the resumable sub and set the type in the calling sub to Object.
- Multiple subs can safely call the resumable sub. The complete event will reach the correct parent.
- You can wait for resumable subs in other modules (in B4A it is relevant for classes only).
- The Result parameter name can be changed.

4.7.6 B4A only KeyPress and Wait For MsgBox2Async

In B4A, the Back key is often checked to prevent the user to exit the program inadvertently.

You can use this code:

```
Sub Activity_KeyPress (KeyCode As Int) As Boolean 'Return True to consume the event
   Select KeyCode
      Case KeyCodes.KEYCODE_BACK
         OpenMsgBox
         Return True
      Case Else
         Return False
   End Select
End Sub

Sub OpenMsgBox
   Private Answ As Int

   Msgbox2Async("Do you want to exit?", "E x i t", "Yes", "", "No", Null, False)
   Wait For Msgbox_Result (Answ As Int)
   If Answ = DialogResponse.POSITIVE Then
      Activity.Finish
   End If
End Sub
```

4.7.7 DoEvents deprecated !

Starting from B4A v7.0 the following warning will appear for DoEvents calls:
DoEvents is deprecated. It can lead to stability issues. Use Sleep(0) instead (if really needed).

The purpose of DoEvents was to allow the UI to be updated while the main thread is busy. DoEvents which shares the same implementation as the modal dialogs implementation, is a low level implementation. It accesses the process message queue and runs some of the waiting messages.

As Android evolved, the handling of the message queue became more sophisticated and fragile.
The reasons for deprecating DoEvents are:

1. It is a major source for instability issues. It can lead to hard to debug crashes or ANR (application not responding) dialogs. Note that this is also true for the modal dialogs (such as Msgbox and InputList).
2. There are better ways to keep the main thread free. For example use the asynchronous SQL methods instead of the synchronous methods.
3. It doesn't do what many developers expect it to do. As it only handles UI related messages, most events could not be raised from a DoEvents call.
4. It is now possible to call Sleep to pause the current sub and resume it after the waiting messages are processed. Sleep implementation is completely different than DoEvents. It doesn't hold the thread. It instead releases it while preserving the sub state.
Unlike DoEvents which only processed UI related messages, with Sleep all messages will be processed and other events will be raised.
(Note that using Wait For to wait for an event is better than calling Sleep in a loop.)

With that said, DoEvents is still there and existing applications will work exactly as before.

4.7.8 Dialogs

Modal dialogs = dialogs that hold the main thread until the dialog is dismissed.

As written above, modal dialogs share the same implementation as DoEvents. It is therefore recommended to switch to the new async dialogs instead.
Using Wait For, is really a simple change:

Instead of:

```
Dim res As Int = Msgbox2("Delete?", "Title", "Yes", "Cancel", "No", Null)
If res = DialogResponse.POSITIVE Then
   '...
End If
```

You should use:

```
Msgbox2Async("Delete?", "Title", "Yes", "Cancel", "No", Null, False)
Wait For Msgbox_Result (Result As Int)
If Result = DialogResponse.POSITIVE Then
      '...
End If
```

Wait For doesn't hold the main thread. It instead saves the current sub state and releases it. The code will resume when the user clicks on one of the dialog buttons.
The other similar new methods are: MsgboxAsync, InputListAsync and InputMapAsync.

With the exception of MsgboxAsync, the new methods also add a new *cancelable* parameter. If it is true then the dialog can be dismissed by clicking on the back key or outside the dialog. This is the default behavior of the older methods.

As other code can run while the async dialog is visible, it is possible that multiple dialogs will appear at the same time.
If this case is relevant for your app then you should set the sender filter parameter in the Wait For call:

```
Dim sf As Object = Msgbox2Async("Delete?", "Title", "Yes", "Cancel", "No", Null, False)
Wait For (sf) Msgbox_Result (Result As Int)
If Result = DialogResponse.POSITIVE Then
   '...
End If
```

This allows multiple messages to be displayed and the result events will be handled correctly.

4.7.9 SQL with Wait For

The new resumable subs feature, makes it simpler to work with large data sets with minimum effect on the program responsiveness.

The new standard way to insert data is:

```
For i = 1 To 1000
   SQL1.AddNonQueryToBatch("INSERT INTO table1 VALUES (?)", Array(Rnd(0, 100000)))
Next
Dim SenderFilter As Object = SQL1.ExecNonQueryBatch("SQL")
Wait For (SenderFilter) SQL_NonQueryComplete (Success As Boolean)
Log("NonQuery: " & Success)
```

The steps are:
- Call AddNonQueryToBatch for each commands that should be issued.
- Execute the commands with ExecNonQueryBatch. This is an asynchronous method. The commands will be executed in the background and the NonQueryComplete event will be raised when done.
- This call returns an object that can be used as the sender filter parameter. This is important as there could be multiple background batch executions running. With the filter parameter the event will be caught by the correct Wait For call in all cases.
- Note that SQL1.ExecNonQueryBatch begins and ends a transaction internally.

4.7.9.1 Queries

In most cases the queries will be fast and should therefore be issued synchronously with SQL1.ExecQuery2. However if there is a slow query then you should switch to SQL1.ExecQueryAsync:

```
Dim SenderFilter As Object = SQL1.ExecQueryAsync("SQL", "SELECT * FROM table1", Null)
Wait For (SenderFilter) SQL_QueryComplete (Success As Boolean, rs As ResultSet)
If Success Then
   Do While rs.NextRow
      Log(rs.GetInt2(0))
   Loop
   rs.Close
Else
   Log(LastException)
End If
```

As in the previous case, the ExecQueryAsync method returns an object that is used as the sender filter parameter.

Tips:
1. ResultSet type in B4A extends the Cursor type. You can change it to Cursor if you prefer. The advantage of using ResultSet is that it is compatible with B4J and B4i.
2. If the number of rows returned from the query is large then the Do While loop will be slow in debug mode. You can make it faster by putting it in a different sub and cleaning the project (Ctrl + P):

```
   Wait For (SenderFilter) SQL_QueryComplete (Success As Boolean, rs As ResultSet)
   If Success Then
      WorkWithResultSet(rs)
   Else
      Log(LastException)
   End If
End Sub

Private Sub WorkWithResultSet(rs As ResultSet)
   Do While rs.NextRow
      Log(rs.GetInt2(0))
   Loop
   rs.Close
End Sub
```

This is related to a debugger optimization that is currently disabled in resumable subs.
The performance of both solutions will be the same in release mode.

4.7.9.2 B4J

- Requires jSQL v1.50+ (https://www.b4x.com/android/forum/threads/updates-to-internal-libaries.48274/#post-503552).
- Recommended to set the journal mode to WAL:
https://www.b4x.com/android/forum/t...ent-access-to-sqlite-databases.39904/#content

4.7.10 Notes & Tips

- The performance overhead of resumable subs in release mode should be insignificant in most cases. The overhead can be larger in debug mode. (If this becomes an issue then take the slow parts of the code and move them to other subs that are called from the resumable sub.)
- Wait For events handlers precede the regular event handlers.
- Resumable subs do not create additional threads. The code is executed by the main thread, or the handler thread in server solutions.

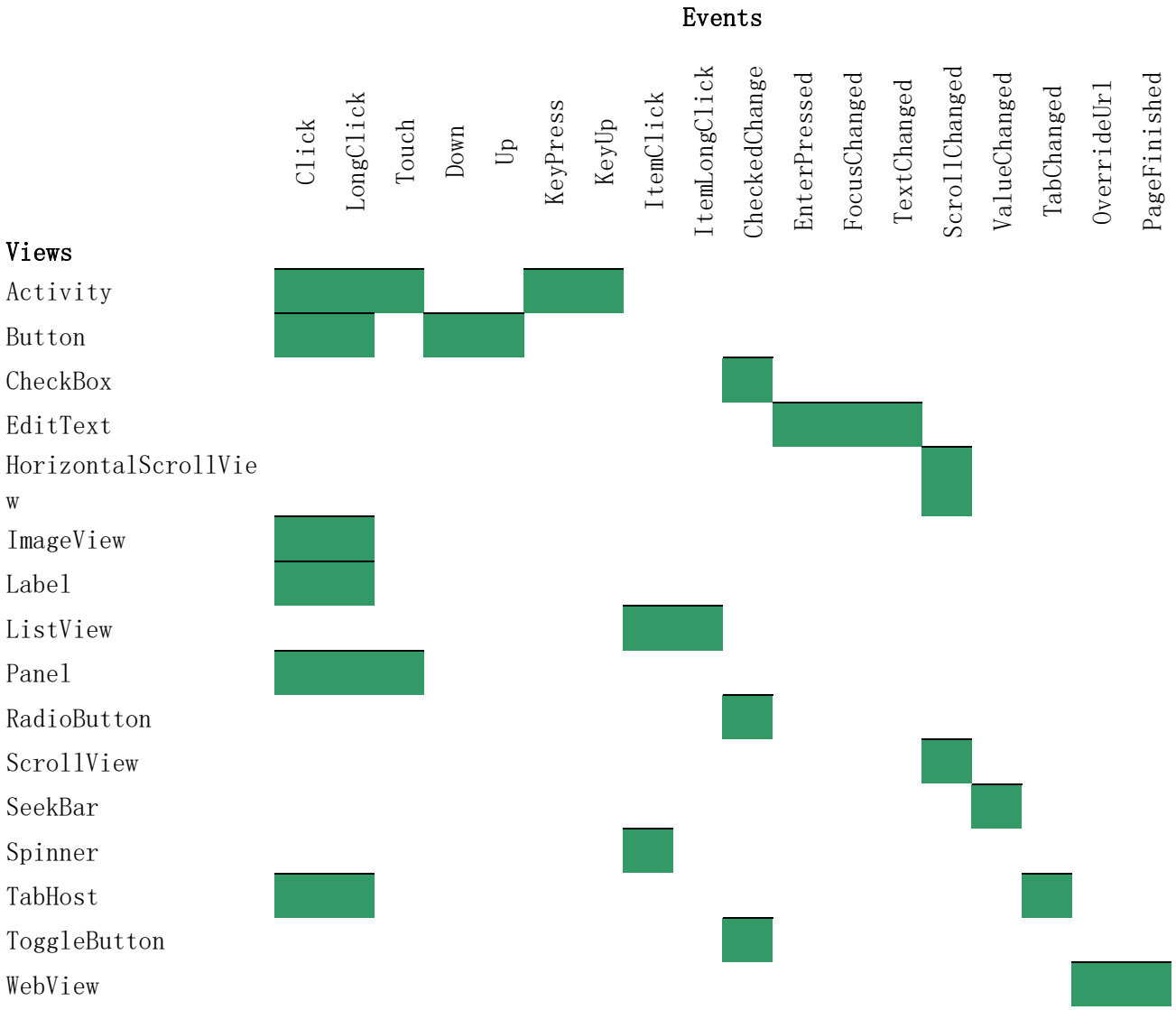
4.8 Events

In Object-oriented programming we have objects which can react on different user actions called events.
The number and the type of events an object can raise depend on the type of the object.

4.8.1 B4A

User interface objects are called 'Views' in Android.

Summary of the events for different views:



The most common events are:

- Click Event raised when the user clicks on the view.
  Example:
```
Sub Button1_Click
  ' Your code
End Sub
```

- LongClick Event raised when the user clicks on the view and holds it pressed for a while.
  Example:
```
Sub Button1_LongClick
  ' Your code
End Sub
```

- Touch (Action As Int, X As Float, Y As Float)
  Event raised when the user touches the screen.

  Three different actions are handled:
  - Activity.ACTION_DOWN, the user touches the screen.
  - Activity.ACTION_MOVE, the user moves the finger without leaving the screen.
  - Activity.ACTION_UP, the user leaves the screen.

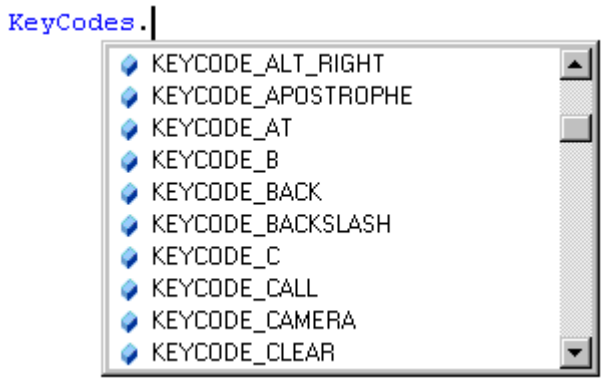
  The X an Y coordinates of the finger position are given.

  Example:
```
Sub Activity_Touch (Action As Int, X As Float, Y As Float)
  Select Action
  Case Activity.ACTION_DOWN
    ' Your code for DOWN action
  Case Activity.ACTION_MOVE
    ' Your code for MOVE action
  Case Activity.ACTION_UP
    ' Your code for UP action
  End Select
End Sub
```

- CheckChanged (Checked As Boolean)
  Event raised when the user clicks on a CheckBox or a RadioButton
  Checked is equal to True if the view is checked or False if not checked.

  Example:
```
Sub CheckBox1_CheckedChange(Checked As Boolean)
  If Checked = True Then
    ' Your code if checked
  Else
    ' Your code if not checked
  End If
End Sub
```

- **KeyPress (KeyCode As Int) As Boolean**
  Event raised when the user presses a physical or virtual key.
  KeyCode is the code of the pressed key, you can get them with the KeyCodes keyword.



The event can return either:
- True, the event is 'consumed', considered by the operating system as already executed and no further action is taken.
- False, the event is not consumed and transmitted to the system for further actions.

Example:

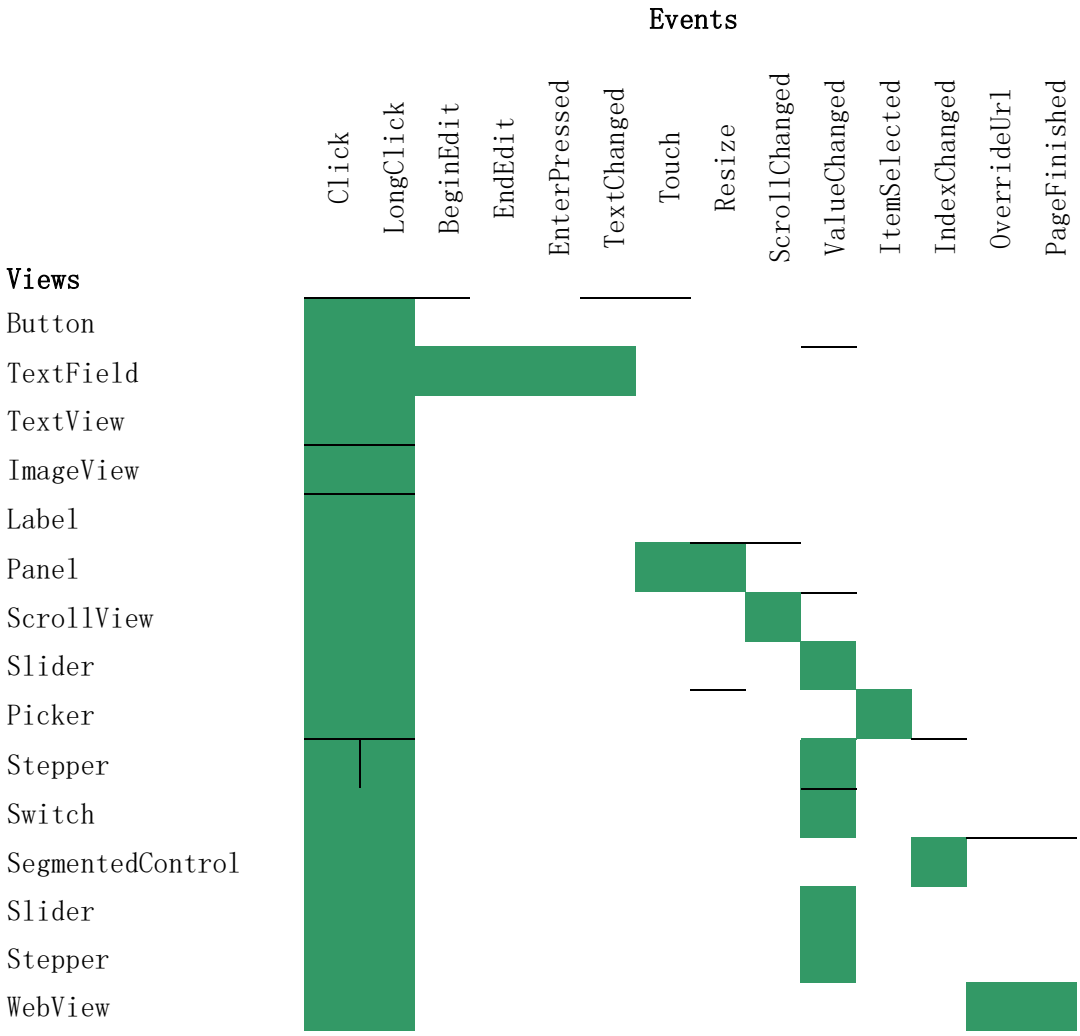
```
Sub Activity_KeyPress(KeyCode As Int) As Boolean
  Private Answ As Int
  Private Txt As String

  If KeyCode = KeyCodes.KEYCODE_BACK Then    ' Checks if KeyCode is BackKey
    Txt = "Do you really want to quit the program ?"
    Answ = Msgbox2(Txt,"A T T E N T I O N","Yes","","No",Null)' MessageBox
    If Answ = DialogResponse.POSITIVE Then  ' If return value is Yes then
      Return False     ' Return = False  the Event will not be consumed
    Else               '                   we leave the program
      Return True      ' Return = True   the Event will be consumed to avoid
    End If             '                 leaving the program
  End If
End Sub
```

4.8.2 B4i

User interface objects are called 'Views' in iOS.

Summary of the events for different views:



The most common events are:

- **Click** Event raised when the user clicks on the view.
  Example:
```
Private Sub Button1_Click
  ' Your code
End Sub
```

- **LongClick** Event raised when the user clicks on the view and holds it pressed for a while.
  Example:
```
Private Sub Button1_LongClick
  ' Your code
End Sub
```

- **Touch (Action As Int, X As Float, Y As Float)**
  Event raised when the user touches a Panel on the screen.

  Three different actions are handled:
  - Panel.ACTION_DOWN, the user touches the screen.
  - Panel.ACTION_MOVE, the user moves the finger without leaving the screen.
  - Panel.ACTION_UP, the user leaves the screen.

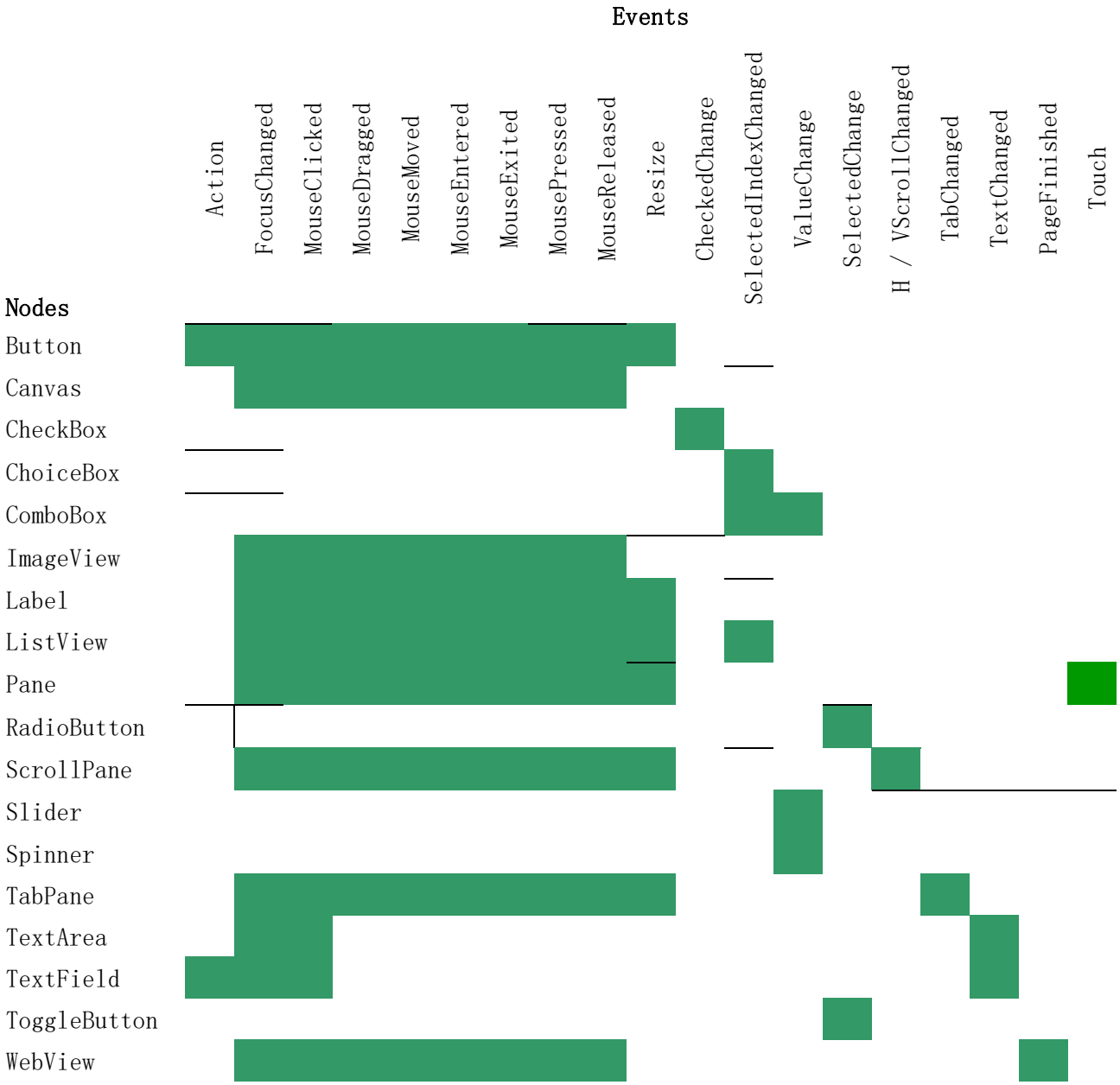
  The X and Y coordinates of the finger positions are given in Points not in Pixels.

  Example:
```
Private Sub Panel_Touch (Action As Int, X As Float, Y As Float)
  Select Action
  Case Panel.ACTION_DOWN
    ' Your code for DOWN action
  Case Panel.ACTION_MOVE
    ' Your code for MOVE action
  Case Panel.ACTION_UP
    ' Your code for UP action
  End Select
End Sub
```

4.8.3 B4J

User interface objects are called 'Nodes' in Java.

Summary of the events for different nodes:



The most common events are:

- **Action** Event raised when the user clicks on the node (Button or TextField).
  Example:
```
Private Sub Button1_Action
  ' Your code
End Sub
```

- **FocusChanged (HasFocus As Boolean)** Event raised when the node gets or looses focus.
  Example:
```
Private Sub TextField1_FocusChanged (HasFocus As Boolean)
  ' Your code
End Sub
```

- **MouseClicked (EventData As MouseEvent)**
  Event raised when the user clicks on the node.
  Example:
```
Private Sub Pane1_MouseClicked (EventData As MouseEvent)
  ' Your code
End Sub
```

- **MouseDragged (EventData As MouseEvent)**
  Event raised when the user draggs over the node (moves with a button pressed).
  Similar to ACTION_MOVE in B4A Touch events.
  Example:
```
Private Sub Pane1_MouseDragged (EventData As MouseEvent)
  ' Your code
End Sub
```

- **MouseEntered (EventData As MouseEvent)**
  Event raised when the user enters the node.
  Example:
```
Private Sub Pane1_MouseEntered (EventData As MouseEvent)
  ' Your code
End Sub
```

- **MouseExited (EventData As MouseEvent)**
  Event raised when the user exits the node.
  Example:
```
Private Sub Pane1_MouseExited (EventData As MouseEvent)
  ' Your code
End Sub
```

- **MouseMoved (EventData As MouseEvent)**
  Event raised when the user moves over the node (without a button pressed).
  Example:
```
Private Sub Pane1_MouseMoved (EventData As MouseEvent)
  ' Your code
End Sub
```

- MousePressed (EventData As MouseEvent)
  Event raised when the user presses on the node.
  Similar to ACTION_DOWN in B4A Touch events.
  Example:
```
Private Sub Pane1_MousePressed (EventData As MouseEvent)
  ' Your code
End Sub
```

- MouseReleased (EventData As MouseEvent)
  Event raised when the user releases the node.
  Similar to ACTION_UP in B4A Touch events.
  Example:
```
Private Sub Pane1_MouseReleased (EventData As MouseEvent)
  ' Your code
End Sub
```

- MouseEvent
  Data included in the MouseEvent object:

  - ClickCount Returns the number of clicks associated with this event.
  - Consume Consumes the current event and prevent it from being handled by the nodes parent.
  - MiddleButtonDown Returns true if the middle button is currently down.
  - MiddleButtonPressed Returns true if the middle button was responsible for raising the current click event.
  - PrimaryButtonDown Returns true if the primary button is currently down.
  - PrimaryButtonPressed Returns true if the primary button was responsible for raising the current click event.
  - SecondaryButtonDown Returns true if the secondary button is currently down.
  - SecondaryButtonPressed Returns true if the secondary button was responsible for raising the current click event.
  - X Returns the X coordinate related to the node bounds.
  - Y Returns the Y coordinate related to the node bounds.

  Example:

```
Private Sub pnlMain_MouseMoved (EventData As MouseEvent)
   Private x, y As Int

   If EventData.MiddleButtonPressed = True Then
      x = EventData.X
      y = EventData.Y
      ' other code
   End If
End Sub
```

- Touch (Action As Int, X As Float, Y As Float)
  Event raised when the user 'touches' the screen.
  This event is similar to the Touch events in B4A and B4i.

  Three different actions are handled:
  - Pane1.TOUCH_ACTION_DOWN, the user touches the screen.
  - Pane1.TOUCH_ACTION_MOVE, the user moves the finger without leaving the screen.
  - Pane1.TOUCH_ACTION_UP, the user leaves the screen.

  The X an Y coordinates of the mouse cursor position are given.

  Example:
```
Sub Pane1_Touch (Action As Int, X As Float, Y As Float)
  Select Action
  Case Pane1.TOUCH_ACTION_DOWN
    ' Your code for DOWN action
  Case Pane1.TOUCH_ACTION_MOVE
    ' Your code for MOVE action
  Case Pane1.TOUCH_ACTION_UP
    ' Your code for UP action
  End Select
End Sub
```

  or

```
Sub Pane1_Touch (Action As Int, X As Float, Y As Float)
  Select Action
  Case 0  'DOWN
    ' Your code for DOWN action
  Case 2  'MOVE
    ' Your code for MOVE action
  Case 1  'UP
    ' Your code for UP action
  End Select
End Sub
```

4.8.4 B4R

In B4R, the Pin and Timer objects are the only ones raising an event:

- Pin
  StateChanged (State As Boolean) Event raised when the pin changes its state.

  Example:
  ```
  Sub Pin1_StateChanged(State As Boolean)
    ' Your code
  End Sub
  ```

- Timer
  Tick Event raised at every given interval

  Example:
  ```
  Private Timer1 As Timer

  Timer1.Initialize("Timer1_Tick",1000)

  Sub Timer1_Tick
    ' Your code
  End Sub
  ```

  Be aware that in B4R the initialize method is different from the other B4X products.
  You must declare the full sub name like "Timer1_Tick", and not "Timer1" like in the other products.

4.8.5 User interface summary

The ‘standard’ user interface objects.
This shows the difference between the three operating systems.
Some views / nodes which don’t exist as standard objects can exis as CustomViews in other operating systems. You should look in the forums.

View / node	B4A	B4i	B4J
Activity			
Button			
CheckBox			
EditText			
HorizontalScrollView			
ImageView			
Label			
ListView			
Panel			
RadioButton			
ScrollView			
SeekBar			
Spinner			
TabHost			
ToggleButton			
WebView			
TextField			
TextView			
ScrollView different from B4A 2D			
Slider			
Picker			
Stepper			
Switch			
SegmentedControl			
Canvas a node on its own			
ChoiceBox			
ComboBox			
Pane similar to Panel in B4A and B4i			
ScrollPane similar to ScrollView			
TabPane			
TextArea			

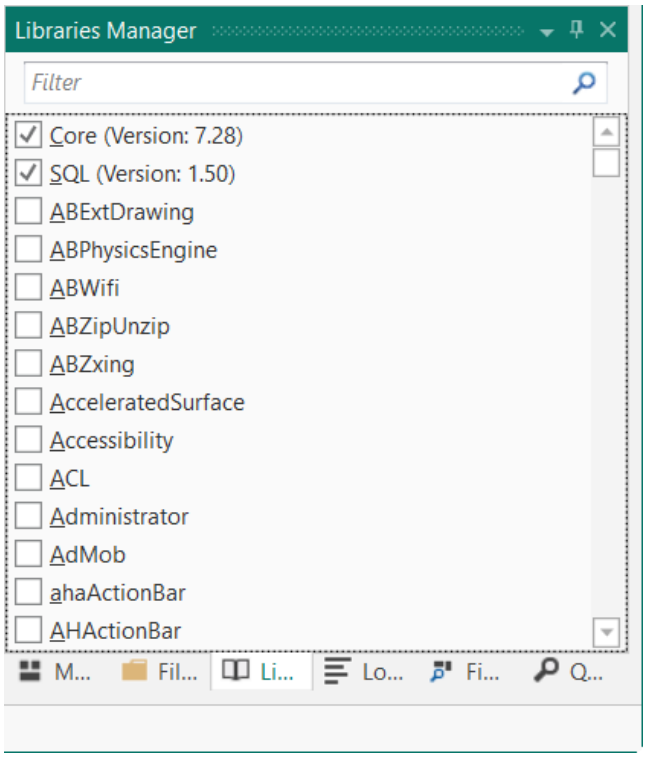
For cross-platform projects you might look at the B4X Cross-platform projects booklet and more specific chapter 4. Compatibilities B4A B4i B4J XUI.

4.9 Libraries

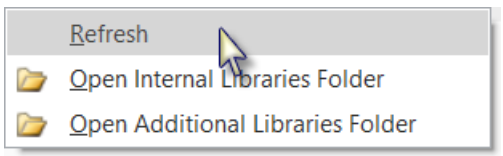
Libraries add more objects and functionalities to B4X.
Some of these libraries are shipped with the B4X products and are part of the standard development system.
Other, often developed by users, can be downloaded (by registered users only) to add supplementary functionalities to the B4X development environments.

When you need a library, you have to:

- Check it in the Libs Tab, if you already have the library.
- For additional libraries, check if it's the latest version.
  You can check the versions in the documentation page B4A, B4i, B4J, B4R
  Or in the Libraries Google sheet in the forum.
  To find the library files use a query like
  http://www.b4x.com/search?query=betterdialogs+library
  in your internet browser.
- If **yes**, then check the library in the list to select it.



- If **no**, download the library, unzip it and copy the <LibraryName>.jar and <LibraryName>.xml files to the additional libraries folder for the give product.
  If it's a B4XLibrary, copy the <LibraryName>.b4xlib file To AdditionalLibraries\B4X folder.
- Right click in the Lib area and click on Refresh and check the library in the list to select it.



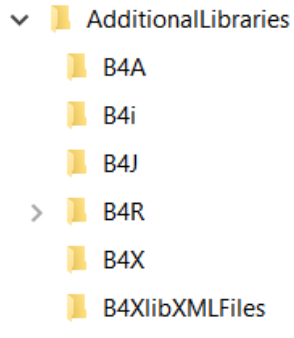
4.9.1 Standard libraries

The standard B4X libraries are saved in the Libraries folder in the B4X program folder.
Normally in:
C:\Program Files\Anywhere Software\B4A\Libraries
C:\Program Files\Anywhere Software\B4i\Libraries
C:\Program Files\Anywhere Software\B4J\Libraries
C:\Program Files\Anywhere Software\B4R\Libraries

4.9.2 Additional libraries folder

Additional Libraries are composed of two files: an *xxx.jar* and an *xxx.xml* file.
B4X libraries have only one file *xxx.b4xlib*.

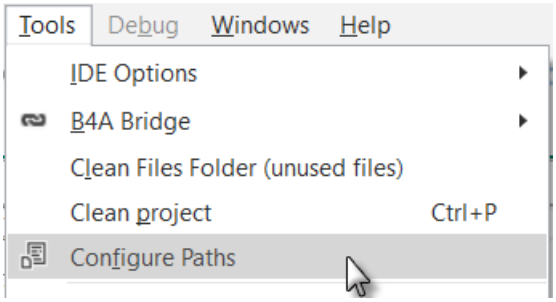
For the additional libraries it is necessary to setup a special folder to save them somewhere else.
This folder must have the following structure:



Folder for B4A additional libraries.
Folder for B4i additional libraries.
Folder for B4J additional libraries.
Folder for B4R additional libraries.
Folder for B4X libraries.
Folder for B4X libraries XML files.

One subfolder for each product: B4A, B4i, B4J, B4R and another B4X for B4X libraries.

When you install a new version of a B4X product, all standard libraries are automatically updated, but the additional libraries are not included. The advantage of the special folder is that you don't need to care about them because this folder is not affected when you install the new version of B4X.
The additional libraries are not systematically updated with new version of B4X.



When the IDE starts, it looks first for the available libraries in the Libraries folder of B4X and then in the additional libraries folders.

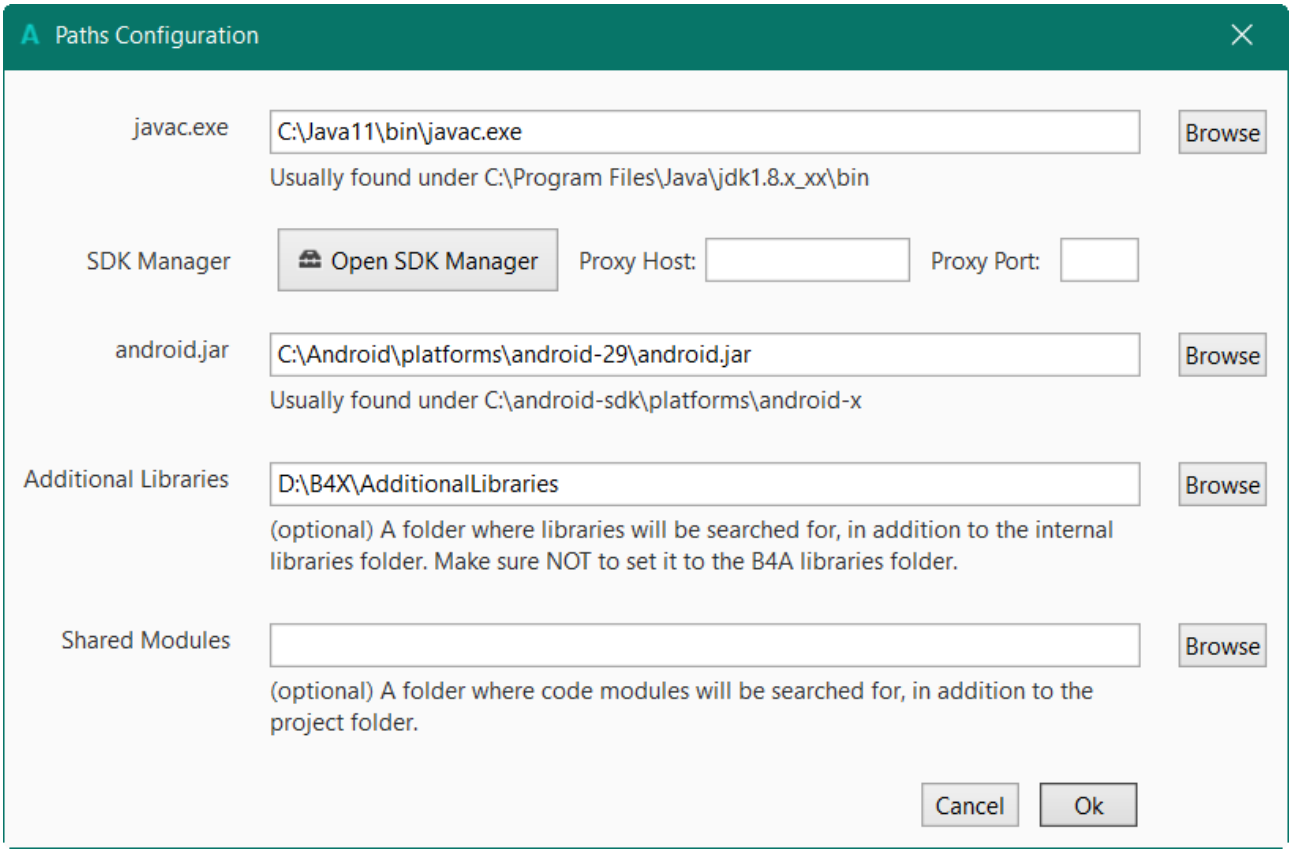
To setup the special additional libraries folder, click in the IDE menu on Tools / Configure Paths.

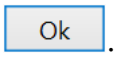
In my system, I added a B4XlibXMLFiles folder for XML help files.
The standard and additional libraries have an XML file. B4X Libraries do not.

But, if you use the B4X Help Viewer you would be interested in having these help files if they are available. The B4X Help Viewer is explained in the B4X Help tools booklet.

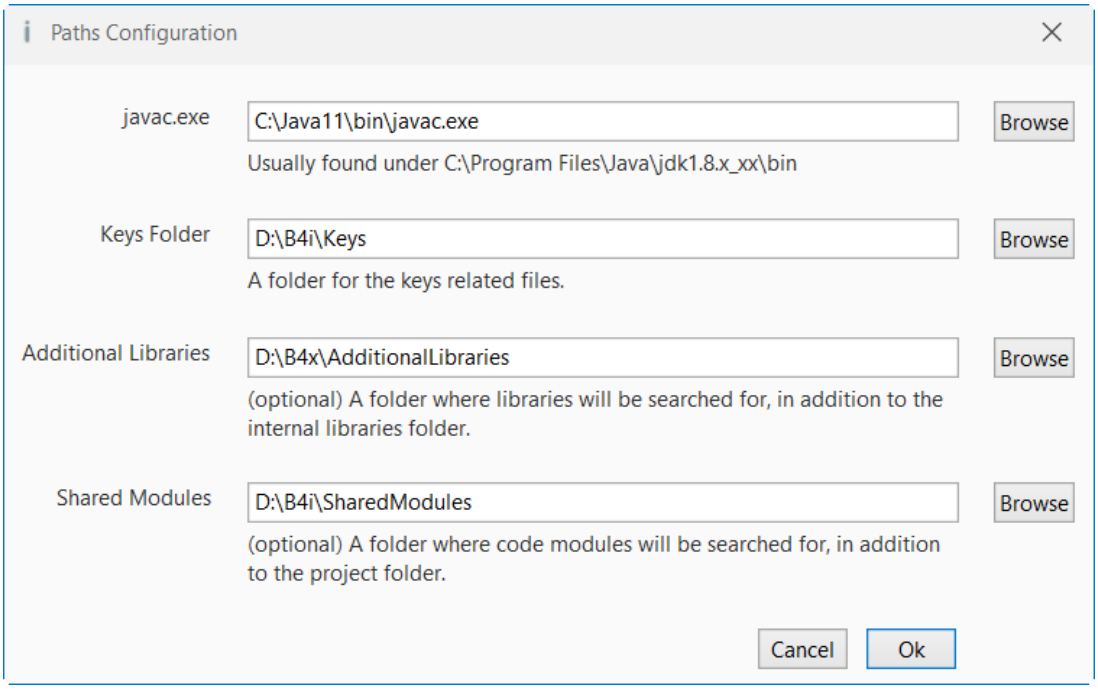
You can create xml files for b4xlib libraries with this tool: b4xlib - XML generation.

4.9.2.1 Paths configuration B4A

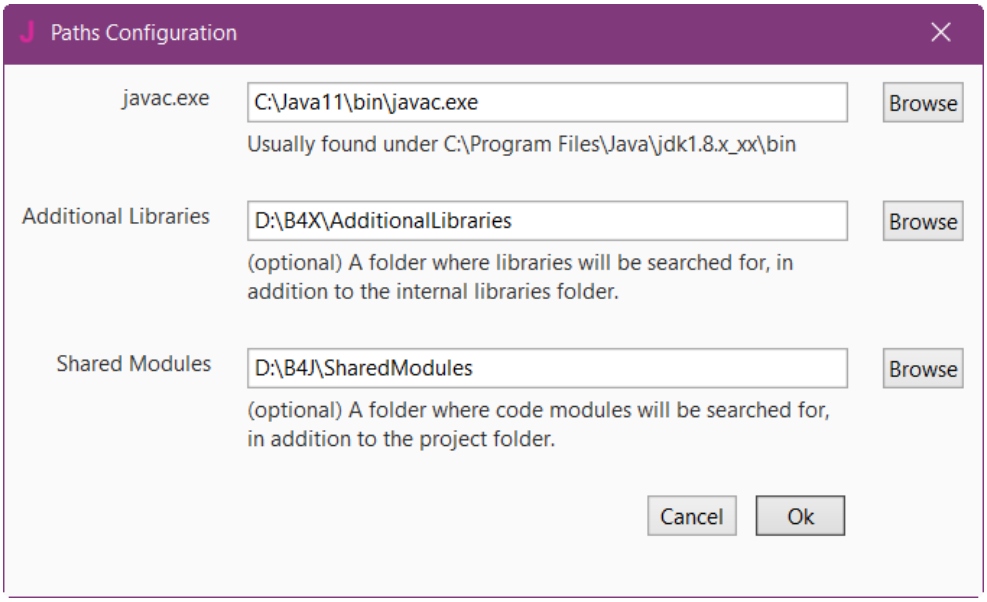


Enter the folder names and click on  .

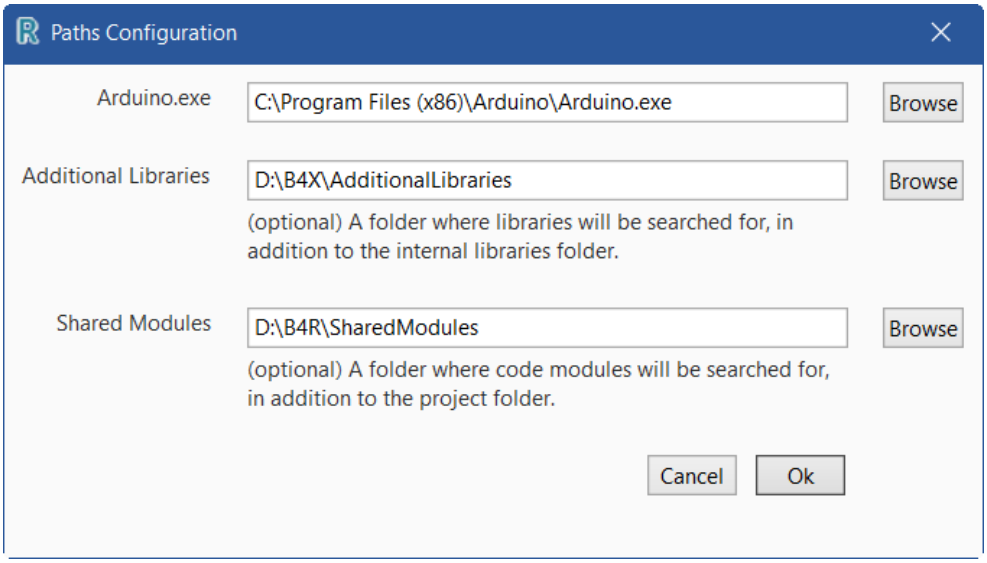
4.9.2.2 Paths configuration B4i



4.9.2.3 Paths configuration B4J



4.9.2.4 Paths configuration B4R



4.9.3 B4X Libraries *.b4xlib

B4X libraries are cross platform libraries introduced in B4A 8.80, B4i 5.50 and B4J 7.00.

These libraries contain cross platform classes which do not need to be compiled as libraries.

A B4X library is a simple zip file with the following structure:

- Code modules. All types are supported including Activities and Services.
- Files, including layout files.
- Optional manifest file with the following fields:
  - Version
  - Author
  - DependsOn (list of required libraries), Supported Platforms. Fields can be shared between the platforms or be platform specific.

Files and code modules can also be platform specific.

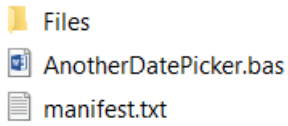
Creating a b4x library is very simple. You just need to create a zip file with these resources. The zip file extension should be b4xlib. That's all.

Note that the source code can be extracted from a b4x library.

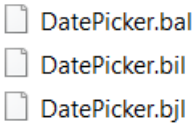
b4x libraries appear like all other libraries in the Libraries tab.

Example: the AnotherDatePicker.b4xlib

The zip file structure:



Files contains all the needed files, the three layout files in the example.



AnotherDatePicker.bas is the crossplatform Custom View file.
Manifest.txt contains:

```
Version=2.00                        version number.
B4J.DependsOn=jXUI, jDateUtils      libraries used for B4J.
B4A.DependsOn=XUI, DateUtils        libraries used for B4A.
B4i.DependsOn=iXUI, iDateUtils      libraries used for B4i.
```

Copy the xxx.b4xlib file to the AdditionalLibaries\B4X folder.
If there is an xxx.xml file, you must not save it there but in another folder.

B4XLibraries are explained in the B4X Custom Views Booklet.

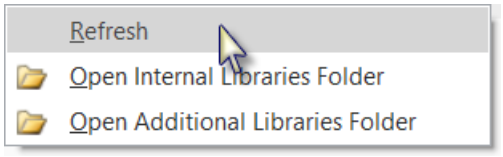
4.9.4 Load and update a Library

A list of the official and additional libraries with links to the relevant help documentation can be found on the B4X site in the:
B4A Documentation page: List of Libraries.
B4i Documentation page: List of Libraries.
B4J Documentation page: List of Libraries.
B4R Documentation page: List of Libraries.
Or in the B4X Libraries Google sheet.

To find the library files use a query like
http://www.b4x.com/search?query=betterdialogs+library
in your internet browser.

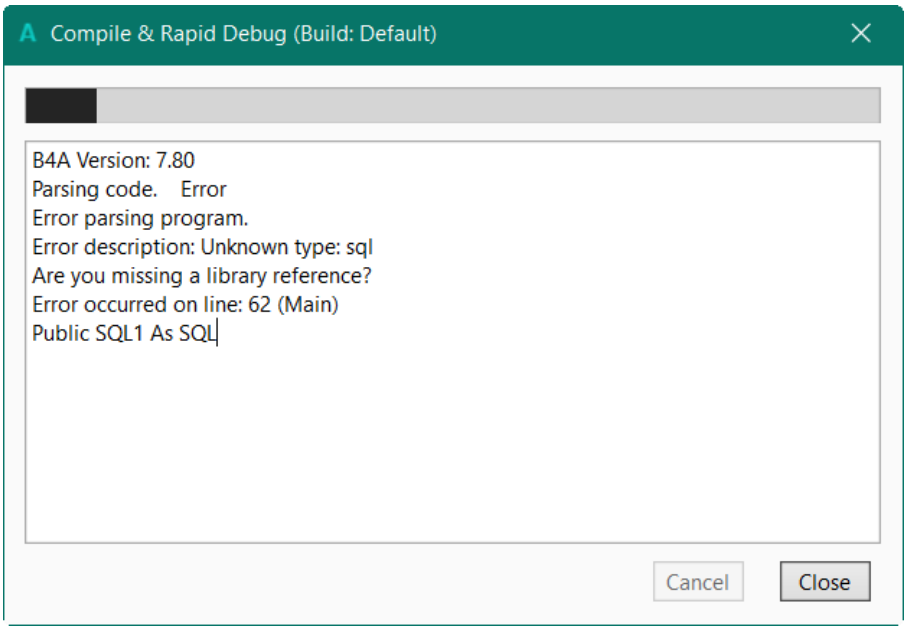
To load or update a library follow the steps below:

- Download the library zip file somewhere.
- Unzip it.
- Copy the xxx.jar and xxx.xml files to the
  - B4X Library folder for a standard B4X library
  - Additional libraries folder for an additional library.
- Right click in the libraries list in the Libraries Manager Tab and click on Refresh and select the library.



4.9.5 Error message "Are you missing a library reference?"

If you get a message similar to this, it means that you forgot to check the specified library in the Lib Tab list !



4.9.6 Where do I find libraries?

To find libraries you can either:

- Search in the forum with its name.
- Or look at the online libraries index.

4.9.6.1 Online libraries index

You can look at the online libraries index with this link:
https://docs.google.com/spreadsheets/d/1qFvc3Q70RriJS3m_ywBoJvZ47gSTVAuN_X04SIO_XBw/edit#gid=0

Screen shot:

	Library Name	Short Description	Files Names (without extension)				Last Update	
			B4A	B4i	B4J	B4R	Version	Date
3	MFRC522	RFID reader / writer				rMFRC522	1.02	15-Mar-2019
4	4 Button B4xDialog	Additional button for B4xDialog	B4xDialog4Button	B4xDialog4Button	B4xDialog4Button		1.1	21-Dec-2020
5	433MHz R/T	Support for 433MHz receiver and transmitter				rRCSwitch	1.01	5-Aug-2018
6	ABMaterial	WebApps Framework with Materialize CSS			ABMaterial		4.51	18-Nov-2018
7	ABMServer	WebApps Mini Template for ABMaterial			ABMServer		1.07	28-Feb-2021
8	ActivityRecognition	Monitor the user state	ActivityRecognition (class)				3	27-May-2020
9	AdColonyAds	AdColony Ads	AdColonyAds				4.65	13-Jan-2022
10	AdManager	AdManager Ads	AdManager				1.52	22-Feb-2021
11	Administrator	Android administrator features	Administrator				1.1	11-Sep-2017
12	AmazonAds	Amazon Ads	AmazonAds				1	24-Sep-2020

Last Update		Author	IDE Comment	Forum Link
Version	Date			Must start with https://www.b4x.com
1.02	15-Mar-2019	Erel		https://www.b4x.com/android/forum/threads/mfrc522-rfid-reader-writer.67
1.1	21-Dec-2020	Stevel05		https://www.b4x.com/android/forum/threads/b4x-b4xdialog4button.1241(
1.01	5-Aug-2018	JanDerKan		https://www.b4x.com/android/forum/threads/rcswitch-library.95830/
4.51	18-Nov-2018	Alwaysbusy		https://www.b4x.com/android/forum/threads/abmaterial-framework-for-we
1.07	28-Feb-2021	Alwaysbusy		https://www.b4x.com/android/forum/threads/abmaterial-abmserver-mini-t
3	27-May-2020	Erel		https://www.b4x.com/android/forum/threads/physical-activity-recognition
4.65	13-Jan-2022	Pendrush		https://www.b4x.com/android/forum/threads/adcolony-library.120665/
1.52	22-Feb-2021	Pendrush		https://www.b4x.com/android/forum/threads/admanager-library.122111/
1.1	11-Sep-2017	Erel		https://www.b4x.com/android/forum/threads/device-administrator-library.
1	24-Sep-2020	Pendrush		https://www.b4x.com/android/forum/threads/amazonads-library.122691/

You find:

- Library Name.
- Short Description.
- File Names (without extension) and the relevant platforms.
- Last update: With the latest Version and update date.
- Author
- IDE Comment this comment will appear in the IDE in the Libraries Manager.
- Forum Link: This link leads you to the forum thread where you find the library.

4.10 String manipulation

4.10.1 B4A, B4i, B4J String

B4A, B4i and B4J allow string manipulations like other Basic languages but with some differences.

These manipulations can be done directly on a string.
Example:

```
txt = "123,234,45,23"
txt = txt.Replace(",", ";")
```

Result: 123;234;45;23

The different functions are:

- CharAt(Index) Returns the character at the given index.
- CompareTo(Other) Lexicographically compares the string with the Other string.
- Contains(SearchFor) Tests whether the string contains the given SearchFor string.
- EndsWith(Suffix) Returns True if the string ends with the given Suffix substring.
- EqualsIgnoreCase(Other) Returns True if both strings are equal ignoring their case.
- GetBytes(Charset) Encodes the Charset string into a new array of bytes.
- IndexOf(SearchFor) Returns the index of the first occurrence of SearchFor in the string. The index is 0 based. Returns -1 if no occurrence is found.
- IndexOf2(SearchFor, Index) Returns the index of the first occurrence of SearchFor in the string. Starts searching from the given index.
  The index is 0 based. Returns -1 if no occurrence is found.
- LastIndexOf(SearchFor) Returns the index of the first occurrence of SearchFor in the string. The search starts at the end of the string and advances to the beginning.
  The index is 0 based. Returns -1 if no occurrence is found.
- LastIndexOf2(SearchFor) Returns the index of the first occurrence of SearchFor in the string. The search starts at the given index and advances to the beginning.
  The index is 0 based. Returns -1 if no occurrence is found.
- Length Returns the length, number of characters, of the string.
- Replace(Target, Replacement) Returns a new string resulting from the replacement of all the occurrences of Target with Replacement.
- StartsWith(Prefix) Returns True if this string starts with the given Prefix.
- Substring(BeginIndex) Returns a new string which is a substring of the original string.
  The new string will include the character at BeginIndex and will extend to the end of the string.
- Substring2(BeginIndex, EndIndex) Returns a new string which is a substring of the original string. The new string will include the character at BeginIndex and will extend to the character at EndIndex, not including the last character. Note that EndIndex is the end index and not the length like in other languages.

- **ToLowerCase** Returns a new string which is the result of lower casing this string.
- **ToUpperCase** Returns a new string which is the result of upper casing this string.
- **Trim** Returns a copy of the original string without any leading or trailing white spaces.

Note: The string functions are case sensitive.
If you want to use case insensitive functions you should use either ToLowerCase or ToUpperCase.

Example: `NewString = OriginalString.ToLowerCase.StartsWith("pre")`

4.10.2 String concatenation

The concatenation character to join Strings is: &

Examples:

- Strings
  ```
  Private MyString As String
  MyString = "aaa" & "bbb" & "ccc"      result: aaabbbccc
  ```

- String and number
  ```
  MyString = "$: " & 1.25               result: $: 1.25
  ```

- String and variable, it can be either another string or a number.
  ```
  Private Val As Double
  Val = 1.25
  MyString = "$: " & Val          result: $: 1.25
  ```

Don't confuse with VB syntax:

```
MyString = "aaa" + "bbb" + "ccc"
```

This doesn't work!

4.10.3 B4A, B4i, B4J StringBuilder

StringBuilder is a mutable string, unlike regular strings which are immutable.
StringBuilder is especially useful when you need to concatenate many strings.

The following code demonstrates the performance boosting of StringBuilder:

```
Dim start As Long
start = DateTime.Now
'Regular string
Dim s As String
For i = 1 To 5000
  s = s & i
Next
Log(DateTime.Now - start)
'StringBuilder
start = DateTime.Now
Dim sb As StringBuilder
sb.Initialize
For i = 1 To 5000
  sb.Append(i)
Next
Log(DateTime.Now - start)
```

Tested on a real device, the first 'for loop' took about 20 seconds and the second took less then a tenth of a second.
The reason is that the code: s = s & i creates a new string each iteration (strings are immutable).
The method StringBuilder.ToString converts the object to a string.

4.10.3.1 StringBuilder Methods

Append (Text As String) As StringBuilder
Appends the specified text at the end.
Returns the same object, so you can chain methods.
Example:
```
sb.Append("First line").Append(CRLF).Append("Second line")
```

Initialize
Initializes the object.
Example:
```
Dim sb As StringBuilder
sb.Initialize
sb.Append("The value is: ").Append(SomeOtherVariable).Append(CRLF)
```

Insert (Offset As Int, Text As String) As StringBuilder
Inserts the specified text at the specified offset.

IsInitialized As Boolean

Length As Int [read only]
Returns the number of characters.

Remove (StartOffset As Int, EndOffset As Int) As StringBuilder
Removes the specified characters.
StartOffset - The first character to remove.
EndOffset - The ending index. This character will not be removed.

ToString As String
Converts the object to a string.

4.10.4 Smart String Literal

The "smart string" literal is a more powerful version of the standard string literal.
It has three advantages:

1. Supports multi-line strings.
2. No need to escape quotes.
3. Supports string interpolation.

The smart string literal starts with $" and ends with "$.

Example:
```
Dim s As String = $"Hello world"$
Dim query As String = $"
SELECT value_id FROM table3
WHERE rowid >= random()%(SELECT max(rowid)FROM table3)
AND second_value ISNOTNULL
LIMIT 1"$
Log($"No need to escape "quotes"! "$)
```

4.10.4.1 String Interpolation

Smart strings can hold zero or more placeholders with code. The placeholders can be easily formatted.
A placeholder starts with $[optional formatter]{ and ends with }:

```
Log($"5 * 3 = ${5 * 3}"$) '5 * 3 = 15
```

You can put any code you like inside the placeholders.

```
Dim x = 1, y = 2, z = 4 As Int
Log($"x = ${x}, y = ${y}, z = ${Sin(z)}"$) 'x = 1, y = 2, z = -0.7568024953079282
```

This is a compile time feature. You cannot load the strings from a file for example.

4.10.4.2 Number Formatter

The number formatter allows you to set the minimum number of integers and the maximum number of fractions digits. It is similar to NumberFormat keyword.

The number formatter structure: MinIntegers.MaxFractions. MaxFractions component is optional.
Examples:

```
Dim h = 2, m = 15, s = 7 As Int
Log($"Remaining time $2{h}:$2{m}:$2{s}"$) 'Remaining time 02:15:07
Log($"10 / 7 = $0.3{10 / 7}"$) '10 / 7 = 1.429
Log($"$1.2{"The value is not a number!"}"$) 'NaN
```

4.10.4.3 Other Formatters

Note that the formatters are case insensitive.
Date - Equivalent to DateTime.Date:

```
Log($"Current date is $date{DateTime.Now}"$) 'Current date is 02/02/2015
```

Time - Equivalent to DateTime.Time:

```
Log($"Current time is $time{DateTime.Now}"$) 'Current time is 11:17:45
```

DateTime - Equivalent to DateTime.Date & " " & DateTime.Time:

```
Log($"Current time is $DateTime{DateTime.Now}"$) 'Current time is 02/02/2015 11:18:36
```

XML - Escapes the five XML entities (", ', <, >, &):

```
Dim UserString As String = $"will it break your parser ><'"&?"$
Log($"User input is: $xml{UserString}"$)
'User input is: will it break your parser &gt;&lt;&#39;&quot;&amp;?
```

This is also useful for html content.

4.10.5 B4A, B4i CharSequence CSBuilder

CharSequence is a native interface in Android SDK.
A String is one implementation of CharSequence.
There are other implementations of CharSequence that provide more features and allow us to format the string, add images and even make parts of the text clickable.

Starting from B4A v6.80 many methods accept CharSequence instead of String. Existing code will work properly as you can pass regular strings. However you can now also pass more interesting CharSequences.

Note to library developers, if your library makes calls to APIs that work with CharSequences then you should change your method signatures to expect CharSequence instead of String. This will allow developers to format the text.

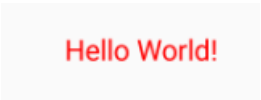
This tutorial covers the CSBuilder object.
CSBuilder is similar to StringBuilder. Instead of building strings, it builds CharSequences that include style information.

The examples are made with B4A, but the principles are the same for B4i

Using it is quite simple.

4.10.5.1 Text

```
Private cs As CSBuilder
cs = cs.Initialize.Color(Colors.Red).Append("Hello World!").PopAll
Label1.Text = cs
```

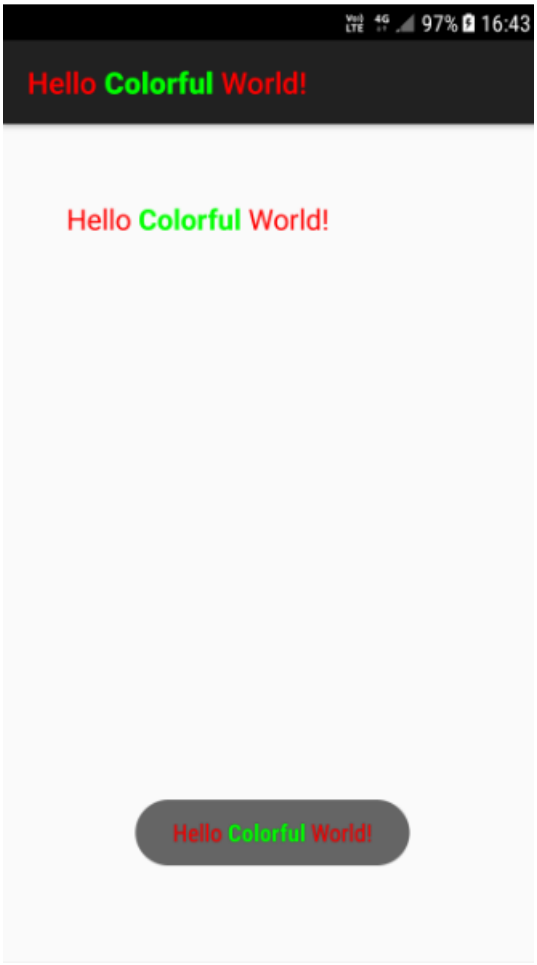


The default background color can be different depending on the Android version.

Almost all methods of CSBuilder return the object itself. This allows us to chain the method calls.
Text is always appended with the Append method.
There are various attributes that can be set. Setting an attribute marks the beginning of a style span.
Calling Pop ends the last span that was added (and not ended yet).
Calling PopAll ends all open spans. It is convenient to always call PopAll at the end to ensure that all spans are closed.

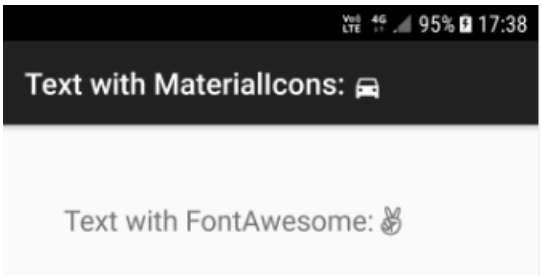
```
'example of explicitly popping an attribute:
Label1.Text = cs.Initialize.Color(Colors.Red).Append("Hello
").Pop.Append("World!").PopAll
```

```
'It doesn't matter whether the methods are chained or split into several lines:
Private cs As CSBuilder
cs.Initialize.Color(Colors.Red).Append("Hello ")
cs.Bold.Color(Colors.Green).Append("Colorful ").Pop.Pop
'two pops: the first removes the green color and the second removes the bold style
cs.Append("World!").PopAll
Label1.Text = cs
'can also be set as the activity title
Activity.Title = cs
'and Toast messages and in other places...
ToastMessageShow(cs, True)
```



4.10.5.2 With FontAwesome or MaterialIcons

```
Private  cs As CSBuilder
Label1.Text = cs.Initialize.Append("Text with FontAwesome:
").Typeface(Typeface.FONTAWESOME).Append(Chr(0xF209)).PopAll
'Using the same builder multiple times. Note that it is initialized each time.
'Note that we vertically align the material icon character.
cs.Initialize.Append("Text with MaterialIcons:
").Typeface(Typeface.MATERIALICONS).VerticalAlign(5dip).Append(Chr(0xE531)).PopAll
Activity.Title = cs
```



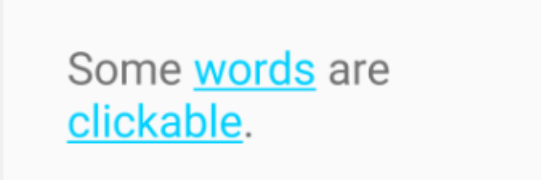
Note: The hex values of Materialicons characters begin with 0xE and FontAwesome charactes begins with 0xF

4.10.5.3 Images

```
Private  cs As CSBuilder
cs.Initialize.Size(18).Typeface(Typeface.MONOSPACE)
cs.Image(LoadBitmap(File.DirAssets, "edelweiss.jpg"), 60dip, 40dip, False).Append("
Edelweiss").Append(CRLF)
cs.Image(LoadBitmap(File.DirAssets, "gentiane.jpg"), 60dip, 40dip, False).Append("
Gentiane").Append(CRLF)
cs.Image(LoadBitmap(File.DirAssets, "lys_martagon.jpg"), 60dip, 40dip, False).Append("
Lys martagon").Append(CRLF)
cs.Image(LoadBitmap(File.DirAssets, "rose.jpg"), 60dip, 40dip, False).Append("
Rose").Append(CRLF)
cs.PopAll
Label1.Text = cs
```

4.10.5.4 Clickable text

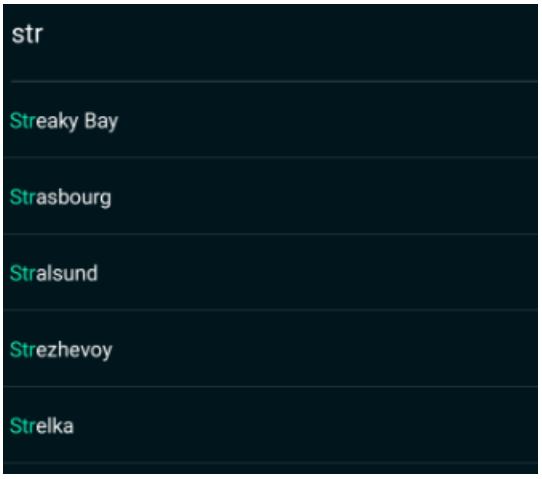
The Clickable method creates clickable text. For the event to be raised you must call cs.EnableClickEvents.
The Append method accepts a CharSequence. In the following code the CreateClickableWord sub returns a CharSequence that is then appended to the other CharSqeuence.



4.10.5.5 Highlight text

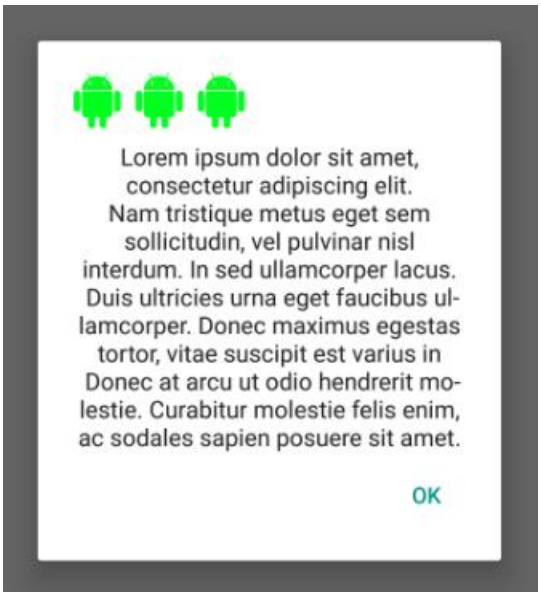
Example from the SearchView class.

```
Private Sub AddItemsToList(li As List, full As String)
   If li.IsInitialized = False Then Return
   Dim cs As CSBuilder
   For i = 0 To li.Size - 1
      Dim item As String = li.Get(i)
      Dim x As Int = item.ToLowerCase.IndexOf(full)
      If x = -1 Then
         Continue
      End If
      cs.Initialize.Append(item.SubString2(0,
x)).Color(highlightColor).Append(item.SubString2(x, x + full.Length)).Pop
      cs.Append(item.SubString(x + full.Length))
      lv.AddSingleLine(cs)
   Next
End Sub
```



4.10.5.6 Center aligned text

```
Msgbox(cs.Initialize.Alignment("ALIGN_CENTER").Append($"Lorem ipsum dolor sit am
et, consectetur adipiscing elit.
Nam tristique metus eget sem sollicitudin, vel pulvinar nisl interdum. In sed ul
lamcorper lacus.
Duis ultricies urna eget faucibus ullamcorper. Donec maximus egestas tortor, vit
ae suscipit est varius in
Donec at arcu ut odio hendrerit molestie. Curabitur molestie felis enim, ac soda
les sapien posuere sit amet."$).PopAll, _
cs.Initialize.Typeface(Typeface.FONTAWESOME).Color(0xFF01FF20).Size(40).Append(C
hr(0xF17B) & " " & Chr(0xF17B) & " "& Chr(0xF17B)).PopAll)
```



4.10.5.7 CSBuilder Methods

4.10.5.7.1 B4A / B4i

- **Alignement** (Alignment As Alignment Enum)
  Starts an alignment span.
  Alignment - One of the following strings:
  ALIGN_NORMAL, ALIGN_OPPOSITE or ALIGN_CENTER

- **Append** (Text As CharSequence)
  Appends the provided String or CharSequence.

- **BackgroundColor** (Color As Int)
  Starts a background color span.

- **Color** (Color As Int)
  Starts a foreground color span.

- **Initialize**
  Initializes the builder. You can call this method multiple times to create new CharSequences.
  Note that like most other methods it returns the current object.

- **IsInitialized**
  Tests whether this object was initialized. Returns a Boolean.

- **Pop**
  Closes the most recent span. All spans must be closed. You can call PopAll to close all open spans.

- **PopAll**
  Closes all open spans.
  It is convenient to always call PopAll at the end to ensure that all spans are closed.

- **Strikethrough**
  Starts a strikethrough span.

- **ToString**
  Returns a string with the characters.

- **Underline**
  Starts an underline span.

- **VerticalAlign** (Shift As Int)
  Starts a vertical alignment span (positive = downwards).

4.10.5.7.2 B4A only

- **Bold**
  Starts a bold span.

- **Clickable** (EventName As String, Tag As Object)
  Starts a clickable span. For the event to be raised you need to call the EnableClickEvents method.
  Example:
```
Sub Activity_Create(FirstTime As Boolean)
   Activity.LoadLayout("1")
   Dim cs As CSBuilder
   cs.Initialize.Size(30).Append("Some ").Append(CreateClickableWord("words"))
   cs.Append(" are ").Append(CreateClickableWord("clickable")).Append(".").PopAll
   Label1.Text = cs
   cs.EnableClickEvents(Label1)
End Sub

Sub CreateClickableWord(Text As String) As CSBuilder
   Dim cs As CSBuilder
   Return cs.Initialize.Underline.Color(0xFF00D0FF).Clickable("word", Text).Appen
d(Text).PopAll
End Sub

Sub Word_Click (Tag As Object)
   Log($"You have clicked on word: ${Tag}"$)
End Sub
```

- **EnableClickEvents** (Label As TextView)
  This method should be called when using clickable spans.

- **Image** (Bitmap As Bitmap, Width As Int, Height As Int, Baseline As Boolean)
  Adds an image span. This method will add a space character as a placeholder for the image.
  Unlike the other methods you do not need to call Pop to close this span as it is closed automatically.
  Bitmap - The image.
  Width / Height - Image dimensions, use 'dip' units.
  Baseline - If true then the image will be aligned based on the baseline.
  Otherwise it will be aligned based on the lowest descender in the text.

- **RelativeSize** (Proportion As Float)
  Starts a relative size span. The actual text size will be multiplied with the set Proportion.

- **ScaleX** (Proportion As Float)
  Starts a scale X span. It horizontally scales the text.

- **Size** (Size As Int)
  Starts a text size span. Note that you should not use 'dip' units with text size dimensions.

- **TypeFace** (Typeface As Typeface)
  Starts a custom typeface span.
  Similar to Font for B4i.

4.10.5.7.3 B4i only

- **Font** (Font As B4IFontWrapper)
  Starts a font span.
  Note that when AutoScaleAll is called the font is reset.
  You should change the font in the parent Resize event or remove the call to AutoScaleAll from the layout designer script.
  Similar to TypeFace for B4A.

- **KerningScale** (Scale As Float)
  Sets the kerning (horizontal spacing) scale.

- **Link** (URL As NSString)
  Creates a link. Links will be clickable in non-editable TextViews.

4.10.6 B4J TextFlow class

The TextFlow Class uses JavaObject to create a TextFlow node. With a TextFlow you can display rich text with different colors, fonts and other attributes.

Usage:
- Add the TextFlow class module to your project (Tools - Add Existing Module).
- Create a TextFlow object.
- Call AddText to add a text section and set its attributes.
- Eventually you should call CreateTextFlow to create the node that will be added to the layout.

Note that the set attributes return the class instance which allows chaining the calls.

Example code:
```
Dim tf As TextFlow
tf.Initialize
tf.AddText("1 2 3").SetColor(fx.Colors.Red).SetUnderline(True)
tf.AddText(" 4 5 6 ").SetColor(fx.Colors.Green).SetFont(fx.CreateFont("", 17, True, Tru
e))
tf.AddText("7 8 9").SetColor(fx.Colors.Blue).SetStrikethrough(True).SetFont(fx.DefaultF
ont(20))
Dim pane As Pane = tf.CreateTextFlow
MainForm.RootPane.AddNode(pane, 10, 10, 200, 100)
```

4.10.7 B4R

B4R doesn't support string manipulations like other Basic languages.

These kind of manipulations can be done with the ByteConverter object in the rRandomAccesFile library.

B4R strings are different than in other B4X tools. The reasons for these differences are:

- Very limited memory.
- Lack of Unicode encoders.

A String object in B4R is the same as a C language char* string. It is an array of bytes with an additional zero byte at the end.
The requirement of the last zero byte makes it impossible to create a substring without copying the memory to a new address.
For that reason, arrays of bytes are preferable over Strings.
The various string related methods work with arrays of bytes.

Converting a string to an array of bytes is very simple and doesn't involve any memory copying. The compiler will do it automatically when needed:

```
Private b() As Byte = "abc" 'equivalent to Private b() As Byte = "abc".GetBytes
```

Only two functions are supported:

These functions are:

- **GetBytes(Charset)** Returns the string content as an array of bytes. Note that the array and string share the same memory
- **Length** Returns the length, number of characters, of the string.

String Methods

The standard string methods are available in ByteConverter type (rRandomAccessFile library).

They are similar to the string methods in other B4X tools:

```
Private Sub AppStart
   Serial1.Initialize(115200)
   Log("AppStart")
   Dim bc As ByteConverter
   Log("IndexOf: ", bc.IndexOf("0123456", "3")) 'IndexOf: 3
   Dim b() As Byte = " abc,def,ghijkl "
   Log("Substring: ", bc.SubString(b, 3)) 'Substring: c,def,ghijkl
   Log("Trim: ", bc.Trim(b)) 'Trim: abc,def,ghijkl
   For Each s() As Byte In bc.Split(b, ",")
     Log("Split: ", s)
     'Split: abc
     'Split: def
     'Split: ghijkl
   Next
   Dim c As String = JoinStrings(Array As String("Number of millis: ", Millis, CRLF, "N
umber of micros: ", Micros))
   Log("c = ", c)
   Dim b() As Byte = bc.SubString2(c, 0, 5)
   b(0) = Asc("X")
   Log("b = ", b)
   Log("c = ", c) 'first character will be X
End Sub
```

Note how both strings and array of bytes can be used as the compiler converts strings to arrays of bytes automatically.

With the exception of JoinStrings, none of the above methods make a copy of the original string / bytes.
This means that modifying the returned array as in the last three lines will also modify the original array.

It will also happen with string literals that all share the same memory block:

```
Private Sub AppStart
   Serial1.Initialize(115200)
   Log("AppStart")
   Dim bc As ByteConverter
   Dim b() As Byte = bc.Trim("abcdef ")
   b(0) = Asc("M") 'this line will change the value of the literal string
   Dim s as String = "abcdef "
   Log(s) 'Mbcdef
End Sub
```

String manipulations in the ByteConverter object in the rRandomAccessFile library:

- **EndsWith(Source As Byte(), Suffix As Byte())**
  Returns True if the string ends with the given Suffix substring.
- **IndexOf(Source As Byte(), SearchFor As Byte())**
  Returns the index of the first occurrence of SearchFor in the string.
- **IndexOf2(Source As Byte(), SearchFor As Byte(), Index As UInt)**
  Returns the index of the first occurrence of SearchFor in the string. Starts searching from the given index.
- **LastIndexOf(Source As Byte(), SearchFor As Byte())**
  Returns the index of the first occurrence of SearchFor in the Source string. Starts searching from the end of the string.
- **LastIndexOf2(Source As Byte(), SearchFor As Byte(), Index As UInt)**
  Returns the index of the first occurrence of SearchFor in the Source string. Starts searching from the given index and advances to the beginning.
- **StartsWith(Source As Byte(), Prefix As Byte())**
  Returns True if this string starts with the given Prefix.
- **Substring(Source As Byte(), BeginIndex As UInt)**
  Returns a new string which is a substring of the original string.
  The new string will include the character at BeginIndex and will extend to the end of the string.
- **Substring2(Source As Byte(), BeginIndex As UInt, EndIndex As UInt)**
  Returns a new string which is a substring of the original string. The new string will include the character at BeginIndex and will extend to the character at EndIndex, not including the last character.
- **Trim(Source As Byte())**
  Returns a copy of the original string without any leading or trailing white spaces.

4.11 Number formatting

4.11.1 B4A, B4i, B4J

Number formatting, display numbers as strings with different formats, there are two keywords:

- NumberFormat(Number As Double, MinimumIntegers As Int, MaximumFractions As Int)
  ```
  NumberFormat(12345.6789, 0, 2)      =  12,345.68
  NumberFormat(1, 3 ,0)               =   001
  NumberFormat(Value, 3 ,0)           variables can be used.
  NumberFormat(Value + 10, 3 ,0)      arithmetic operations can be used.
  NumberFormat((lblscore.Text + 10), 0, 0) if one variable is a string add
  ```
  parentheses.

- NumberFormat2(Number As Double, MinimumIntegers As Int, MaximumFractions As Int, MinimumFractions As Int, GroupingUsed As Boolean)
  ```
  NumberFormat2(12345.67, 0, 3, 3, True)   =  12,345.670
  NumberFormat2(12345.67, 0, 3, 3, False)  =  12345.670
  ```

4.11.2 B4X NumberFormatter

B4XFormatter is an alternative to NumberFormat / NumberFormat2 keywords. It is implemented in B4X as a b4xlib and it is cross platform.

There are two types in the library:

B4XFormatter - The main class.
B4XFormatData - A type with various configurable fields.

The formatter holds a list of format data objects. A new formatter starts with a single format data which acts as the default format.

4.11.3 B4R

Number formatting, display numbers as strings with different formats:

- NumberFormat(Number As Double, MinimumIntegers As Int, MaximumFractions As Int)
  ```
  NumberFormat(12345.6789, 0, 2)      =  12,345.68
  NumberFormat(1, 3 ,0)               =   001
  NumberFormat(Value, 3 ,0)           variables can be used.
  NumberFormat(Value + 10, 3 ,0)      arithmetic operations can be used.
  NumberFormat((lblscore.Text + 10), 0, 0) if one variable is a string add
  ```
  parentheses.

4.12 Timers

A Timer object generates Tick events at specified intervals. Using a timer is a good alternative to a long loop, as it allows the UI thread to handle other events and messages.
Note that the timer events will not fire while the UI thread is busy running other code.
Timer events will not fire when the activity is paused, or if a blocking dialog (like Msgbox) is visible.
It is also important, in B4A, to disable the timer when the activity is pausing and then enable it when it resumes. This will save CPU and battery.

A timer has:

- Three parameters.
  - **Initialize** Initializes the timer with two parameters, the EventName and the interval.
    Timer1.Initialize(EventName As String, Interval As Long)
    Ex: `Timer1.Initialize("Timer1", 1000)`

  - **Interval** Sets the timer interval in milli-seconds.
    Timer1. Interval = Interval
    Ex: `Timer1.Interval = 1000`, 1 second

  - **Enabled** Enables or disables the timer. **It is False by default.**
    Ex: `Timer1.Enabled = True`

- One Event
  - **Tick** The Tick routine is called every time interval.
    Ex: `Sub Timer1_Tick`

The Timer must be declared in a Process_Global routine.

```
Sub Process_Globals
  Public Timer1 As Timer
```

But it must be initialized in one of the following routines in the module where the timer tick event routine is used.

B4A: Activity_Create routine

```
Sub Activity_Create(FirstTime As Boolean)
   If FirstTime = True Then
      Timer1.Initialize("Timer1", 1000)
   End If
```

B4i: Application_Start routine

```
Private Sub Application_Start (Nav As NavigationController)
   Timer1.Initialize("Timer1", 1000)
```

B4J: AppStart routine

```
Sub AppStart (Form1 As Form, Args() As String)
   Timer1.Initialize("Timer1_Tick", 1000)
```

B4R: AppStart routine

```
Private Sub AppStart
   Timer1.Initialize("Timer1", 1000)
```

And the Timer Tick event routine.
This routine will be called every second (1000 milli-seconds) by the operating system.

```
Private Sub Timer1_Tick
   ' Do something
End Sub
```

4.13 Files B4A, B4i, B4J

Many applications require access to a persistent storage. The two most common storage types are files and databases.

Android and iOS have their own file system. B4A nor B4i programs have access to files in the Windows system.

To add files to your project you must add those in the IDE in the Files Tab. These files will be added to the project Files folder.

4.13.1 File object

The predefined object `File` has a number of functions for working with files.

4.13.1.1 File locations

There are several important locations where you can read or write files.

File.DirAssets
The assets folder includes the files that were added with the file manager in the IDE. It's the Files folder in the project folder.
These files are read-only !
You can not create new files in this folder (which is actually located inside the apk file).
If you have a database file in the Dir.Assets folder you need to copy it to another folder before you can use it.

4.13.1.1.1 B4X

To save data generated by the application and used only by the application you might use the xui, (jxui or ixui) library get the default folder.

xui.DefaultFolder
This folder is the same as:

- B4A - Same as File.DirInternal.
- B4i - Same as File.DirDocuments.
- B4J - Same as File.DirData.
  You must first call SetDataFolder once before you can use this folder.
  xui.**SetDataFolder**(AppName As String)

4.13.1.1.2 B4A only

File.DirInternal / File.DirInternalCache
These two folders are stored in the main memory of the device and are private to your application. Other applications cannot access these files.
The cache folder may get deleted by the OS if it needs more space.

File.DirRootExternal Use this folder only if you really need it.
The storage card root folder. In most cases this is an internal storage card and not an external SD card.

File.DirDefaultExternal
The default folder for your application in the SD card.
The folder is: <storage card>/Android/data/<package>/files/
It will be created if required.

Note that calling any of the two above properties will add the EXTERNAL_STORAGE permission to your application.

Tip: You can check if there is a storage card and whether it is available with **File.ExternalReadable** and **File.ExternalWritable**.

External storage.
You should use the RuntimePermissions library to get the best folder with:
```
MyFolder = RuntimePermissions.GetSafeDirDefaultExternal(SubFolder As String)
```

Returns the path to the app's default folder on the secondary storage device.
The path to File.DirInternal will be returned if there is no secondary storage available.
It is a better alternative to File.DirDefaultExternal.
On Android 4.4+ no permission is required to access this folder.
SubFolder - A sub folder that will be created for your app. Pass an empty string if not needed.

Acces a file in external stroge devices has become cumbersome in Android.
Erel has written a Class ExternalStorage - Access SD cards and USB sticks to 'simplify' the access.

Extract from Erels thread:

Before we start:

1. External storage means a real sd card or a connected mass storage USB device.
2. It has nothing to do with File.DirRootExternal / DirDefaultExternal which actually point to an internal storage.
3. It has nothing to do with runtime permissions.
4. You can use RuntimePermissions.GetAllSafeDirsExternal to directly access a specific folder on the SD card.
5. The minimum version for this class is Android 5. It might work with Android 4.4

(change minSdkVersion if you like to try it).

Starting from Android 4.4 it is no longer possible to directly access external storages.
The only way to access these storages is through the Storage Access Framework (SAF), which is a quite complex and under-documented framework.

The ExternalStorage class makes it simpler to work with SAF.

Usage:

1. Call ExternalStorage.SelectDir. This will open a dialog that will allow the user to select the root folder. Once selected the uri of the root folder is stored and can be later used without requiring the user to select the folder again. Even after the device is booted.

2. Wait For the ExternalFolderAvailable event.
Now you can access the files under Storage.Root, including inside subfolders.

3. Files are represented as a custom type named ExternalFile.

4. The following operations are supported: ListFiles, Delete, CreateNewFile, FindFile, OpenInputStream and OpenOutputStream.

See the attached example.

Depends on: ContentResolver and JavaObject libraries.
Add:
```
#AdditionalJar: com.android.support:support-core-utils
```

4.13.1.1.3 B4i only

File.DirDocuments
The documents folder should only be used to store user generated content. It is possible to make this folder sharable through iTunes.
This folder is backed up by iTunes automatically.

File.DirLibrary
The place for any non-user generated persistent files. This folder is backed up by iTunes automatically.
You can create a subfolder named Caches. Files under that folder will not be backed up.

File.DirTemp
A temporary folder. Files in this folder are not backed up by iTunes and may be deleted from time to time.

B4i Methods to access external resources or share to external apps.

This thread in the forum shows some methods to share files:
List of methods to access external resources or share to external apps.

4.13.1.1.4 B4J only

File.DirApp
Returns the application folder.

File.DirData
Returns the path to a folder that is suitable for writing files.
On Windows, folders under Program Files are read-only. Therefore File.DirApp will be read-only as well.
This method returns the same path as File.DirApp on non-Windows computers.
On Windows it returns the path to the user data folder. For example:
C:\Users\[user name]\AppData\Roaming\[AppName]

File.DirTemp
Returns the temporary folder.

4.13.1.2 File exists ? B4A, B4i, B4J

To check if a file already exists use:
File.Exists (Dir As String, FileName As String)
Returns True if the file exists and False if not.

Note: File.Exists does not work with File.DirAssets !!!

4.13.1.3 Common methods B4A, B4i, B4J

The File object includes several methods for writing to files and reading from files.
To be able to write to a file or to read from a file, it must be opened.

File.OpenOutput (Dir As String, FileName As String, Append As Boolean)
- Opens the given file for output, the Append parameter tells whether the text will be added at the end of the existing file or not. If the file doesn't exist it will be created.

File.OpenInput (Dir As String, FileName As String)
- Opens the file for reading.

File.WriteString (Dir As String, FileName As String, Text As String)
- Writes the given text to a new file.

File.ReadString (Dir As String, FileName As String) As String
- Reads a file and returns its content as a string.

File.WriteList (Dir As String, FileName As String, List As List)
- Writes all values stored in a list to a file. All values are converted to string type if required. Each value will be stored in a separare line.
Note that if a value contains the new line character it will saved over more than one line and when you read it, it will be read as multiple items.

File.ReadList (Dir As String, FileName As String) As List
- Reads a file and stores each line as an item in a list.

File.WriteMap (Dir As String, FileName As String, Map As Map)
- Takes a map object which holds pairs of key and value elements and stores it in a text file. The file format is known as Java Properties file: .properties - Wikipedia, the free encyclopedia
The file format is not too important unless the file is supposed to be edited manually. This format makes it easy to edit it manually.
One common usage of File.WriteMap is to save a map of "settings" to a file.

File.ReadMap (Dir As String, FileName As String) As Map
- Reads a properties file and returns its key/value pairs as a Map object. Note that the order of entries returned might be different than the original order.

File.WriteBytes (Dir As String, FileName As String, Data As Byte())
- Writes the given text to a new file.

File.ReadBytes (Dir As String, FileName As String)
- Reads the data from the given file.
Returns: Byte()

File.Copy (DirSource As String, FileSource As String, DirTarget As String, FileTarget As String)
- Copies the source file from the source directory to the target file in the target directory.
Note that it is not possible to copy files to the Assets folder.

File.Copy2 (In As InputStream, Out As OutputStream)
- Copies all the available data from the input stream into the output stream.
The input stream is automatically closed at the end.

File.Delete (Dir As String, FileName As String)
- Deletes the given file from the given directory.

File.ListFiles (Dir As String) As List
- Lists the files and subdirectories in the diven directory.
Example:
```
Private List1 As List
List1 = File.ListFiles(File.DirInternal)
```
List1 can be declared in Sub Globals

File.Size (Dir As String, FileName As String)
- Returns the size in bytes of the specified file.
This method does not support files in the assets folder.

File.MakeDir (Parent As String, Dir)
- Creates the given folder (creates all folders as needed).
Example:
```
File.MakeDir(File.DirInternal, "music/90")
```

4.13.2 Filenames

B4X file names allow following characters:
a to **z**, **A** to **Z**, **0** to **9** dot **.** underscore **_** and even following characters **+ - % &**
Spaces and following characters *** ?** are not allowed.

Example: MyFile.txt

Note that B4X file names are case sensitive !
MyFile.txt is different from **myfile.txt**

4.13.3 Subfolders

You can define subfolders in B4X with.

```
File.MakeDir(File.DirInternal, "Pictures")
```

To access the subfolder you should add the subfoldername to the foldername with "/" inbetween.

```
ImageView1.Bitmap = LoadBitmap(File.DirInternal & "/Pictures", "test1.png")
```

Or add the subfoldername before the filename with "/" inbetween.

```
ImageView1.Bitmap = LoadBitmap(File.DirInternal, "Pictures/test1.png")
```

Both possibilities work.

4.13.4 B4A, B4J TextWriter

There are two other useful functions for text files: **TextWriter** and TextReader:

TextWriter.Initialize (OutputStream As OutputStream)
- Initializes a TextWriter object as an output stream.

Example:
```
Private Writer As TextWriter
Writer.Initialize(File.OpenOutput(File.DirInternal, "Test.txt" , False))
```

Writer could be declared in Sub Globals.

TextWriter.Initialize2 (OutputStream As OutputStream , Encoding As String)
- Initializes a TextWriter object as as output stream.
- Encoding indicates the CodePage (also called CharacterSet) for text encoding (see next chapter).

Example:
```
Private Writer As TextWriter
Writer.Initialize2(File.OpenOutput(File.DirInternal,"Test.txt" ,False)," ISO-8859-1")
```

Writer could be declared in Sub Globals.
See: Text encoding

TextWriter.Write (Text As String)
- Writes the given Text to the stream.

TextWriter.WriteLine (Text As String)
- Writes the given Text to the stream followed by a new line character LF Chr(10).

TextWriter.WriteList (List As List)
- Writes each item in the list as a single line.
Note that a value containing CRLF will be saved as two lines (which will return two items when reading with ReadList).
All values will be converted to strings.

TextWriter.Close
- Closes the stream.

Example:

```
Private Writer As TextWriter
Writer.Initialize(File.OpenOutput(File.DirInternal, "Text.txt", False))
Writer.WriteLine("This is the first line")
Writer.WriteLine("This is the second line")
Writer.Close
```

4.13.5 B4A, B4J TextReader

There are two other useful functions for text files: TextWriter and **TextReader**:

TextReader.Initialize (InputStream As InputStream)
- Initializes a TextReader as an input stream.

Example:
```
Private Reader  TextReader
Reader.Initialize(File.OpenInput(File.DirInternal, "Test.txt"))
```

Reader could be declared in Sub Globals.

TextReader.Initialize2 (InputStream As InputStream, Encoding As String)
- Initializes a TextReader as an input stream.
- Encoding indicates the CodePage (also called CharacterSet), the text encoding.

Example:
```
Private Reader  TextReader
Reader.Initialize2(File.OpenInput(File.DirInternal, "Test.txt", "ISO-8859-1")
```

Reader could be declared in Sub Globals.
See: Text encoding

TextReader.ReadAll As String
- Reads all of the remaining text and closes the stream.

Example:
```
txt = Reader.ReadAll
```

TextReader.ReadLine As String
- Reads the next line from the stream.
The new line characters are not returned.
Returns Null if there are no more characters to read.

Example:

```
Private Reader As TextReader
Reader.Initialize(File.OpenInput(File.DirInternal, "Text.txt"))
Private line As String
line = Reader.ReadLine
Do While line <> Null
   Log(line)
   line = Reader.ReadLine
Loop
Reader.Close
```

TextReader.ReadList As List
- Reads the remaining text and returns a List object filled with the lines.
Closes the stream when done.

Example:
```
List1 = Reader.ReadList
```

4.13.6 Text encoding

Text encoding or character encoding consists of a code that pairs each character from a given repertoire with something else. Other terms like character set (charset), and sometimes character map or code page are used almost interchangeably (source Wikipedia).

The default character set in Android is Unicode UTF-8.

In Windows the most common character sets are ASCII and ANSI.

- ASCII includes definitions for 128 characters, 33 are non-printing control characters (now mostly obsolete) that affect how text and space is processed.
- ANSI, Windows-1252 or CP-1252 is a character encoding of the Latin alphabet, used by default in the legacy components of Microsoft Windows in English and some other Western languages with 256 definitions (one byte). The first 128 characters are the same as in the ASCII encoding.

Many files generated by Windows programs are encoded with the ANSI character-set in western countries. For example: Excel csv files, Notepad files by default.
But with Notepad, files can be saved with *UTF-8* encoding.

B4X can use following character sets:

- UTF-8 default character-set
- UTF -16
- UTF - 16 BE
- UTF - LE
- US-ASCII ASCII character set
- ISO-8859-1 almost equivalent to the ANSI character-set
- Windows-1251 cyrillic characters
- Windows-1252 latin alphabet

To read Windows files encoded with ANSI you should use the *Windows-1252* character-set.
If you need to write files for use with Windows you should also use the *Windows-1252* character-set.

Another difference between Windows and B4X is the end of line character:

- B4X, only the LF (Line Feed) character Chr(10) is added at the end of a line.
- Windows, two characters CR (Carriage Return Chr(13)) and LF Chr(10) are added at the end of a line. If you need to write files for Windows you must add CR yourself.

The symbol for the end of line is:

- B4X CRLF Chr(10)
- Basic4PPC CRLF Chr(13) & Chr(10)

To read or write files with a different encoding you must use the TextReader or TextWriter objects with the Initialize2 methods. Even for reading csv files.

Tip for reading Excel csv files:
You can either:

- On the desktop, load the csv file in a text editor like *NotePad* or *Notepad++*
- Save the file with *UTF-8* encoding
  With *Notepad++* use Encode in UTF-8 without BOM, see below.

Or

- Read the whole file with TextReader.Initialize2 and "Windows-1252" encoding.
- Save it back with TextWriter.Initialize with the standard Android encoding.
- Read the file with LoadCSV or LoadCSV2 from the StringUtils library.

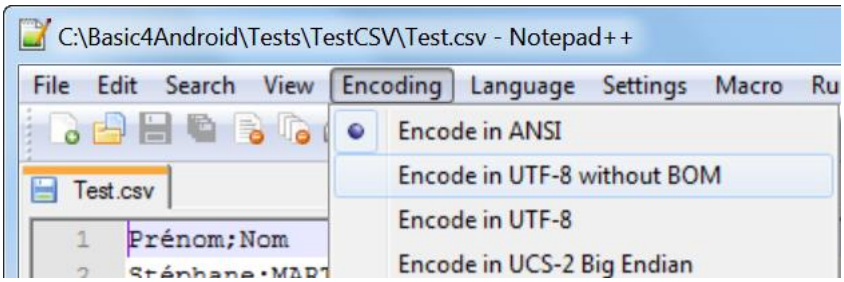
```
Private txt As String
Private tr As TextReader
tr.Initialize2(File.OpenInput(File.DirAssets, "TestCSV1_W.csv"), "Windows-1252")
txt = tr.ReadAll
tr.Close

Private tw As TextWriter
tw.Initialize(File.OpenOutput(File.DirInternal, "TestCSV1_W.csv", False))
tw.Write(txt)
tw.Close

lstTest = StrUtil.LoadCSV2(File.DirInternal, "TestCSV1_W.csv", ";", lstHead)
```

When you save a file with NotePad three additional bytes are added .
These bytes are called BOM characters (Byte Order Mark).
In *UTF-8* they are represented by this byte sequence: 0xEF,0xBB,0xBF.
A text editor or web browser interpreting the text as *Windows-1252* will display the characters ï»¿.

To avoid this you can use *Notepad++* instead of *NotePad* and use Encode in *UTF-8* without BOM.



Another possibility to change a text from *Windows-1252* to *UTF-8* is to use the code below.

```
Private var, result As String
var = "Gestió"
Private arrByte() As Byte
arrByte = var.GetBytes("Windows-1252")
result = BytesToString(arrByte, 0, arrByte.Length, "UTF8")
```

4.14 Lists B4A, B4i and B4J only

Lists are similar to dynamic arrays.

A List must be initialized before it can be used.

- Initialize Initializes an empty List.
```
Private List1 As List
List1.Initialize
List1.AddAll(Array As Int(1, 2, 3, 4, 5))
```

- Initialize2 (SomeArray)
  Initializes a list with the given values. This method should be used to convert arrays to lists. Note that if you pass a list to this method then both objects will share the same list, and if you pass an array the list will be of a fixed size.
  Meaning that you cannot later add or remove items.
  Example 1:
```
Private List1 As List
List1.Initialize2(Array As Int(1, 2, 3, 4, 5))
```
  Example 2:
```
Private List1 As List
Private SomeArray(10) As String
' Fill the array
List1.Initialize2(SomeArray)
```

You can add and remove items from a list and it will change its size accordingly.
With either:

- Add (item As Object)
  Adds a value at the end of the list.
```
List1.Add(Value)
```

- AddAll (Array As String("value1", "value2"))
  Adds all elements of an array at the end of the list.
```
List1.AddAll(List2)
List1.AddAll(Array As Int(1, 2, 3, 4, 5))
```

- AddAllAt (Index As Int, List As List)
  Inserts all elements of an array in the list starting at the given position.
```
List1.AddAll(12, List2)
List1.AddAllAt(12, Array As Int(1, 2, 3, 4, 5))
```

- InsertAt (Index As Int, Item As Object)
  Inserts the specified element in the specified index.
  As a result all items with index larger than or equal to the specified index are shifted.
```
List1.InsertAt(12, Value)
```

- RemoveAt (Index As Int)
  Removes the specified element at the given position from the list.
```
List1.RemoveAt(12)
```

A list can hold any type of object. However if a list is declared as a process global object it cannot hold activity objects (like views).
B4X automatically converts regular arrays to lists. So when a List parameter is expected you can pass an array instead.

Get the size of a List:

- `List1.Size`

Use the Get method to get an item from the list with (List indexes are 0 based):
To get the first item use Get(0).
To get the last item use Get(List1.Size - 1).

- Get(Index As Int)

```
number = List1.Get(i)
```

You can use a For loop to iterate over all the values:

```
For i = 0 To List1.Size - 1
   Private number As Int
   number = List1.Get(i)
   ...
Next
```

Lists can be saved and loaded from files with:

- File.WriteList(Dir As String, FileName As String, List As List)

```
File.WriteList(File.DirRootExternal, "Test.txt", List1)
```

- File.ReadList (Dir As String, FileName As String)

```
List1 = File.ReadList(File.DirRootExternal, "Test.txt")
```

A single item can be changed with:

- List1. Set(Index As Int, Item As Object)

```
List1.Set(12, Value)
```

A List can be sorted (the items must all be numbers or strings) with:

- Sort(Ascending As Boolean)

```
List1.Sort(True)          sort ascending
List1.Sort(False)         sort descending
```

- SortCaseInsensitive(Ascending As Boolean)

Clear a List with:

- `List1.Clear`

4.14.1 Non-dynamic Lists

The code below will not work, it will through an error:

```
List1 = Array As String("Val1", "Val2", "Val3")
List1.Add("Val4")
```

Nor will this code work:

```
List1.Initialize2(Array As String("Val1", "Val2", "Val3"))
List1.Add("Val4")
```

Because the initializations above generate non-dynamic Lists, which cannot be changed.

Be aware that if you want to duplicate a list, the code below will not work either:

```
Private List1 As List
List1.Initialize
List1.AddAll(Array As String("Val1", "Val2", "Val3"))
Private List2 As List
List2 = List1
Log(List1.Size)
Log(List2.Size)
List1.Add("Val4")
Log(List1.Size)
Log(List2.Size)
```

The Log shows:

```
3
3
4
4
```

You see that when you modify something in List1 it is also modified in List2.
This is by design, Lists are passed by reference.

To have an independent copy of a List you need to replace:

```
List2 = List1
```

by

```
List2.Initialize
List2.AddAll(List1)
```

like the code below:

```
Private List1 As List
List1.Initialize
List1.AddAll(Array As String("Val1", "Val2", "Val3"))
Private List2 As List
List2.Initialize
List2.AddAll(List1)
Log(List1.Size)
Log(List2.Size)
List1.Add("Val4")
Log(List1.Size)
Log(List2.Size)
```

The Log shows:

```
3
3
4
3
```

You see that the size of List2 has not changed.

4.15 Maps B4A, B4i and B4J only

A Map is a collection that holds pairs of keys and values.

The keys are unique. Which means that if you add a key/value pair (entry) and the collection already holds an entry with the same key, the previous entry will be removed from the map.

The key should be a string or a number. The value can be any type of object.

Similar to a list, a map can hold any object, however if it is a process global variable then it cannot hold activity objects (like views).

Maps are very useful for storing applications settings.

Maps are used in this example:

- DBUtils module
  used for database entries, keys are the column names and values the column values.

A Map must be initialized before it can be used.

- Initialize Initializes an empty Map.

```
Private Map1 As Map
Map1.Initialize
```

Add a new entry:

- Put(Key As Object, Value As Object)

```
Map1.Put("Language", "English")
```

Get an entry:

- Get(Key As Object)

```
Language = Map1.Get("Language")
```

Get a key or a value at a given index (only B4A and B4J):
Returns the value of the item at the given index.
GetKeyAt and GetValueAt should be used to iterate over all the items.
These methods are optimized for iterating over the items in ascending order.

- GetKeyAt(Index As Int)

```
Key = Map1.GetKeyAt(12)
```

Get a value at a given index (only B4A and B4J):

- GetValueAt(Index As Int)

```
Value = Map1.GetValueAt(12)
```

Check if a Map contains an entry, tests whether there is an entry with the given key:

- ContainsKey(Key As Object):

```
If Map1.ContainsKey("Language") Then
   Msgbox("There is already an entry with this key !", "ATTENTION")
   Return
End If
```

Remove an entry:

- Remove(Key As Object)

```
Map1.Remove("Language")
```

Clear, clears all items from the map:

- Clear

```
Map1.Clear
```

Maps can be saved and loaded with:

- File.WriteMap(Dir As String, FileName As String, Map As Map)

```
File.WriteMap(File.DirInternal, "settings.txt", mapSettings)
```

- ReadMap(Dir As String, FileName As String)
  Reads the file and parses each line as a key-value pair (of strings).
  Note that the order of items in the map may not be the same as the order in the file.

```
mapSettings = File.ReadMap(File.DirInternal, "settings.txt")
```

- File.ReadMap2(Dir As String, FileName As String, Map As Map)
  Similar to ReadMap. ReadMap2 adds the items to the given Map.
  By using ReadMap2 with a populated map you can force the items order as needed.

```
mapSettings = File.ReadMap2(File.DirInternal, "settings1.txt", mapSettings)
```

4.16 Class modules

In B4X, you can use three types of Class Modules:

- Standard Class modules — standard classes
- B4XPages — B4XPages
- CustomView Class Modules — specialized for custom views

In this chapter we will see only Standard Class modules.

B4XPages are explained in the B4XPages Cross-platform projects booklet.

CustomView Class Modules are explained in the B4X CustomViews booklet.

4.16.1 Getting started

Classes definition from Wikipedia:

In object-oriented programming, a class is a construct that is used to create instances of itself – referred to as class instances, class objects, instance objects or simply objects. A class defines constituent members which enable its instances to have state and behaviour. Data field members (member variables or instance variables) enable a class instance to maintain state. Other kinds of members, especially methods, enable the behaviour of a class instances. Classes define the type of their instances.

A class usually represents a noun, such as a person, place or thing, or something nominalized. For example, a "Banana" class would represent the properties and functionality of bananas in general. A single, particular banana would be an instance of the "Banana" class, an object of the type "Banana".

Let us start with an example, the source code: *SourceCode\Person* in the / Person folder.

In the Person module

```
'Class Person module
Sub Class_Globals
   Private FirstName, LastName As String
   Private BirthDate As Long
End Sub

Sub Initialize (aFirstName As String, aLastName As String, aBirthDate As Long)
   FirstName = aFirstName
   LastName = aLastName
   BirthDate = aBirthDate
End Sub

Public Sub GetName As String
   Return FirstName & " " & LastName
End Sub

Public Sub GetCurrentAge As Int
   Return GetAgeAt(DateTime.Now)
End Sub

Public Sub GetAgeAt(Date As Long) As Int
   Private diff As Long
   diff = Date - BirthDate
   Return Floor(diff / DateTime.TicksPerDay / 365)
End Sub
```

Main module.

```
Sub Activity_Create(FirstTime As Boolean)
   Private p As Person
   p.Initialize("John", "Doe", DateTime.DateParse("05/12/1970"))
   Log(p.GetCurrentAge)
End Sub
```

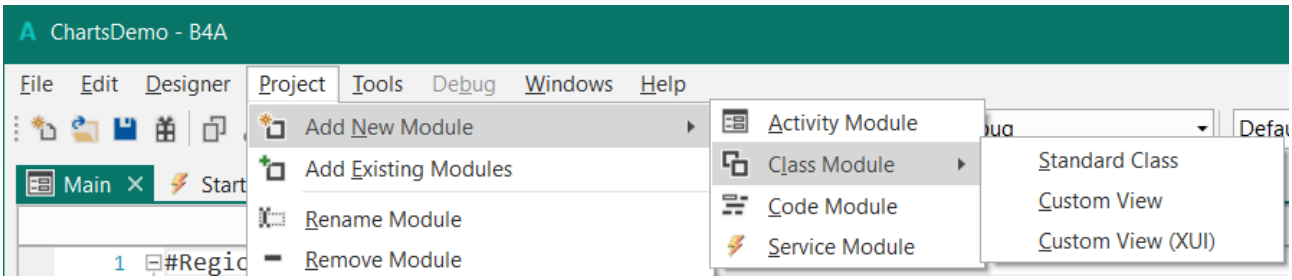
I will start by explaining the differences between classes, code modules and types.

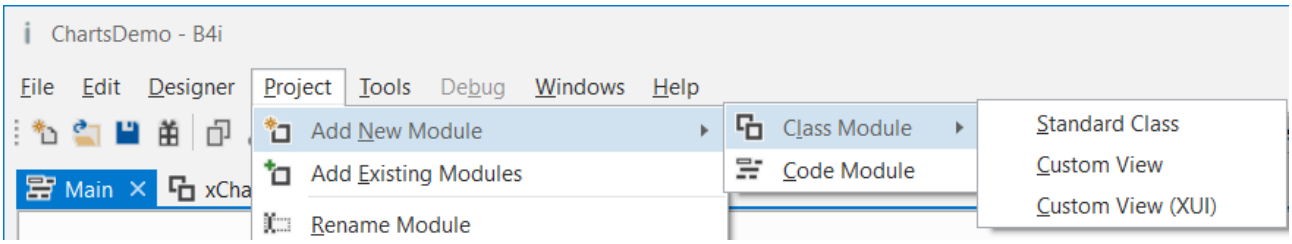
Similar to types, classes are templates. From this template, you can instantiate any number of objects.
The type fields are similar to the classes global variables. However, unlike types which only define the data structure, classes also define the behaviour. The behaviour is defined in the classes' subs.

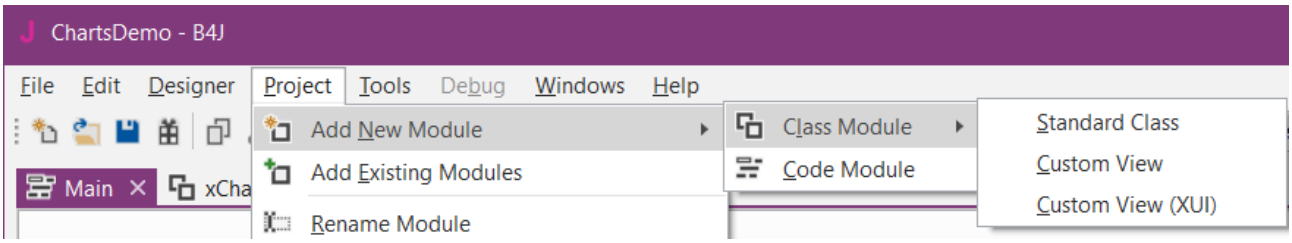
Unlike classes which are a template for objects, code modules are collections of subs. Another important difference between code modules and classes is that code modules always run in the context of the calling sub. The code module doesn't hold a reference to any context. For that reason, it is impossible to handle events or use CallSub with code modules.
Classes store a reference to the context of the module that called the Initialize sub. This means that classes objects share the same life cycle as the module that initialized them.

4.16.1.1 Adding a Class module

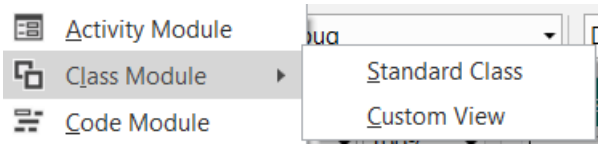
Adding a new or existing class module is done by choosing Project > Add New Module > Class module or Add Existing module.
Like other modules, classes are saved as files with *bas* extension.



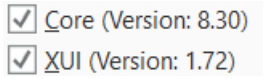




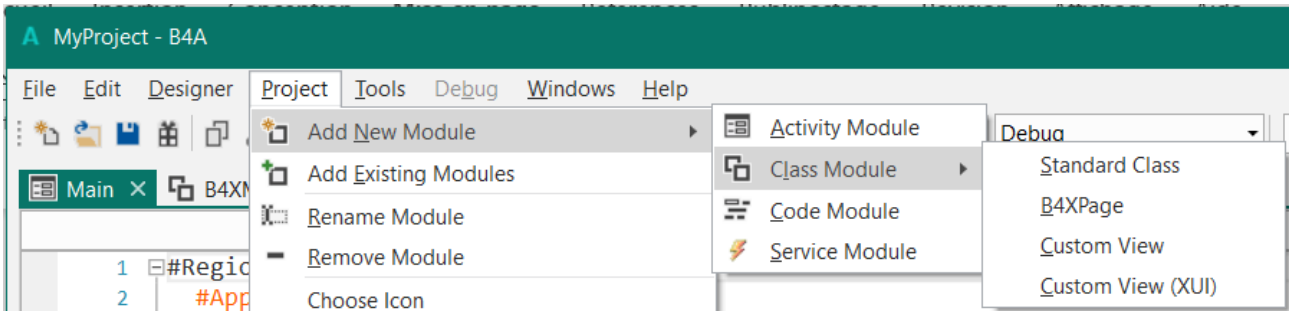
There are two class module types:
Standard Class
CustomView
CustomView (XUI)



The CustomView (XUI) is shown only when the XUI library is selected!



If you use the B4XPages template you can select B4XPage to create a B4XPage class.



4.16.1.2 Polymorphism

Polymorphism allows you to treat different types of objects that adhere to the same interface in the same way.
B4X polymorphism is similar to the Duck typing concept.

As an example we will create two classes named: Square and Circle.
Each class has a sub named Draw that draws the object to a canvas:
Source code *Draw* in the Draw folder.

The code below is the B4A code.

```
'Class Square module
Sub Class_Globals
   Private mx, my, mWidth As Int
End Sub

'Initializes the object. You can add parameters to this method if needed.
Sub Initialize (Shapes As List, x As Int, y As Int, length As Int)
   mx = x
   my = y
   mLength = length
   Shapes.Add(Me)
End Sub

Sub Draw(c As Canvas)
   Private r As Rect
   r.Initialize(mx, my, mx + mLength, my + mLength)
   c.DrawRect(r, Colors.Red, False, 1dip)
End Sub


'Class Circle module
Sub Class_Globals
   Private mx, my, mRadius As Int
End Sub

'Initializes the object. You can add parameters to this method if needed.
Sub Initialize (Shapes As List, x As Int, y As Int, radius As Int)
   mx = x
   my = y
   mRadius = radius
   Shapes.Add(Me)
End Sub

Sub Draw(cvs As Canvas)
   cvs.DrawCircle(mx, my, mRadius, Colors.Blue, False, 1dip)
End Sub
```

In the main module, we create a list Shapes with Squares and Circles. We then go over the list and draw all the objects:

```
Sub Process_Globals
   Public Shapes As List
End Sub

Sub Globals
   Private cvs As Canvas
End Sub

Sub Activity_Create(FirstTime As Boolean)
   cvs.Initialize(Activity)
   Private Square1, Square 2 As Square
   Private Circle1 As Circle
   Shapes.Initialize
   Square1.Initialize(Shapes, 110dip, 110dip, 50dip)
   Square2.Initialize(Shapes, 10dip, 10dip, 100dip)
   Circle1.Initialize(Shapes, 50%x, 50%y, 100dip)

   DrawAllShapes
End Sub

Sub DrawAllShapes
   For i = 0 To Shapes.Size - 1
      CallSub2(Shapes.Get(i), "Draw", cvs)
   Next
   Activity.Invalidate
End Sub
```

As you can see, we do not know the specific type of each object in the list. We just assume that it has a Draw method that expects a single Canvas argument. Later we can easily add more types of shapes.
You can use the `SubExists` keyword to check whether an object includes a specific sub.

You can also use the `Is` keyword to check if an object is of a specific type.

4.16.1.3 Self-reference

The Me keyword returns a reference to the current object. Me keyword can only be used inside a class module.
Consider the above example. We have passed the Shapes list to the Initialize sub and then add each object to the list from the Initialize sub:

```
Sub Initialize (Shapes As List, x As Int, y As Int, radius As Int)
   mx = x
   my = y
   mRadius = radius
   Shapes.Add(Me)
End Sub
```

4.16.1.4 Activity object B4A only

This point is related to the Android Activities special life cycle.
Make sure to first read the activities and processes life-cycle tutorial.

Android UI elements hold a reference to the parent activity. As the OS is allowed to kill background activities in order to free memory, UI elements cannot be declared as process global variables (these variables live as long as the process lives). Such elements are named Activity objects. The same is true for custom classes. If one or more of the class global variables is of a UI type (or any activity object type) then the class will be treated as an "activity object". The meaning is that instances of this class cannot be declared as process global variables.

4.16.2 Standard Class module

4.16.2.1 Structure

Default template of a standard class:

B4A and B4i

```
Sub Class_Globals

End Sub

'Initializes the object. You can add parameters to this method if needed.
Public Sub Initialize

End Sub
```

B4J

```
Sub Class_Globals
   Private fx As JFX
End Sub

'Initializes the object. You can add parameters to this method if needed.
Public Sub Initialize

End Sub
```

Only two routines are predefined:

Sub Class_Globals - This sub is similar to the Main Globals sub. These variables will be the class global variables (sometimes referred to instance variables or instance members).
In B4J, the fx library library is declared by default. You can remove it if not needed.

Sub Initialize - A class object must be initialized before you can call any other sub. Initializing an object is done by calling the Initialize sub. When you call Initialize you set the object's context (the parent object or service).
Note that you can modify this sub signature and add arguments as needed.

Example: Person class module
The source codes are in the Person folder.

The code is the same for all three B4X platforms (B4A. B4i, B4J).

```
'Class Person module
Sub Class_Globals
   Private mFirstName, mLastName As String
   Private mBirthDate As Long
End Sub

Sub Initialize (FirstName As String, LastName As String, BirthDate As Long)
   mFirstName = FirstName
   mLastName = LastName
   mBirthDate = BirthDate
End Sub

Public Sub GetName As String
    Return mFirstName & " " & mLastName
End Sub

Public Sub GetCurrentAge As Int
    Return GetAgeAt(DateTime.Now)
End Sub

Public Sub GetAgeAt(Date As Long) As Int
   Dim diff As Long
   diff = Date - mBirthDate
   Return Floor(diff / DateTime.TicksPerDay / 365)
End Sub
```

In the above code, we created a class named Person and later instantiate an object of this type in the main module:

```
   Private p As Person
   p.Initialize("John", "Doe", DateTime.DateParse("05/12/1970"))
   Log(p.GetCurrentAge)
```

Calling initialize is not required if the object itself was already initialized:

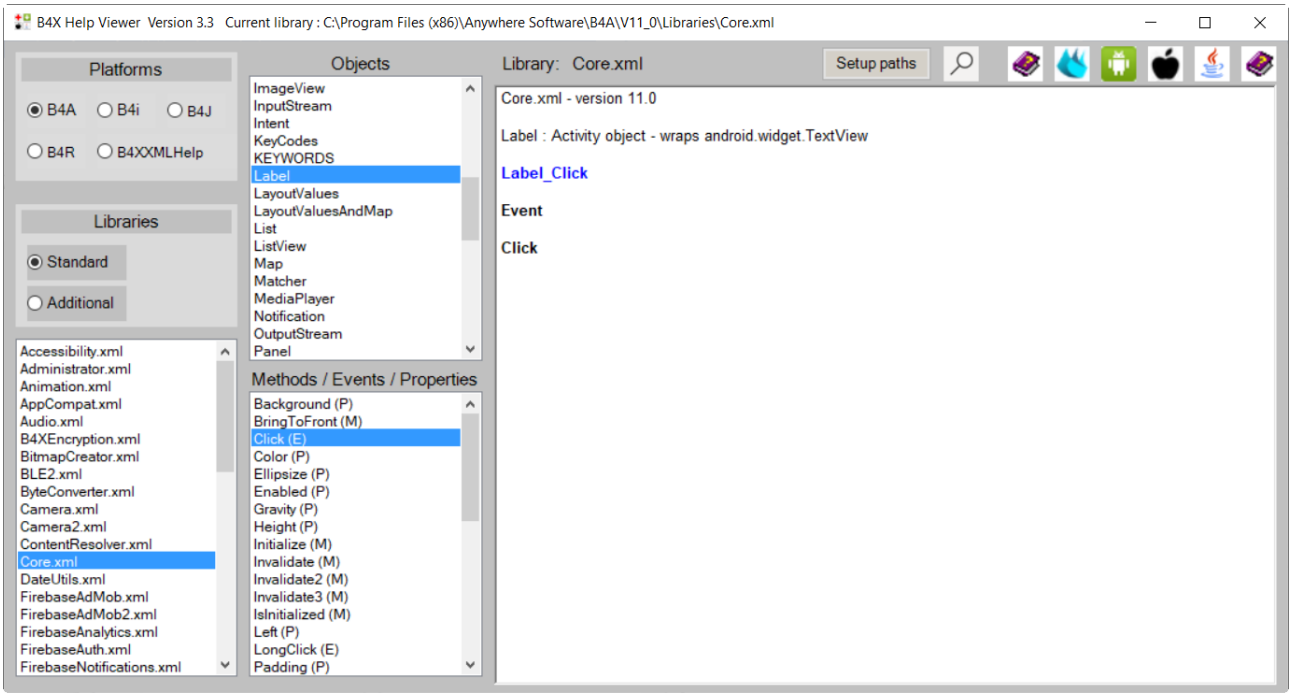
```
   Private p2 As Person
   p2 = p 'both variables now point to the same Person object.
   Log(p2.GetCurrentAge)
```

5 Find object methods, properties, events

5.1 B4X Help Viewer

The B4X Help Viewer is explained in details in the B4X Help tools booklet.

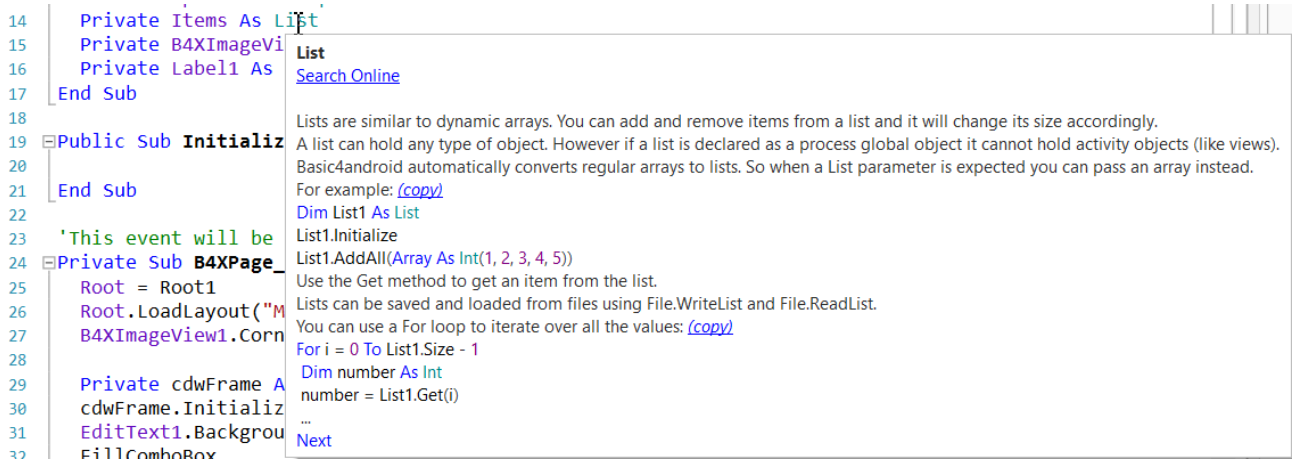
You can select a platform, a library, an object and display the subject.

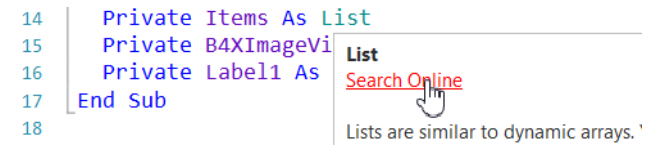


It can be downloaded from the forum with this link:
https://www.b4x.com/android/forum/threads/b4x-help-viewer.46969/.

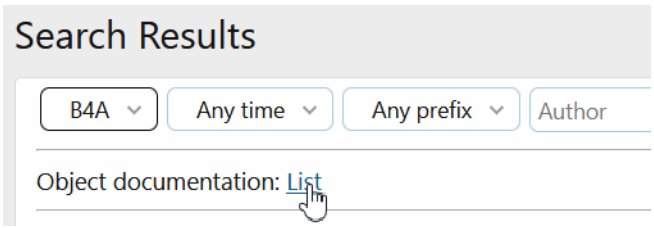
5.2 Hovering over an object

In the code, hover over an object and the in-line help will be displayed, a List in the example.





When you hover over Search Online and click:



You get this page in the forum, hover over List.

And the result.

List

Lists are similar to dynamic arrays. You can add and remove items from a list and it will change its size ac
A list can hold any type of object. However if a list is declared as a process global object it cannot hold a
Basic4android automatically converts regular arrays to lists. So when a List parameter is expected you ca
For example:

```
Dim List1 As List
List1.Initialize
List1.AddAll(Array As Int(1, 2, 3, 4, 5))
```

Use the Get method to get an item from the list.
Lists can be saved and loaded from files using File.WriteList and File.ReadList.
You can use a For loop to iterate over all the values:

```
For i = 0 To List1.Size - 1
Dim number As Int
number = List1.Get(i)
...
Next
```

Events:

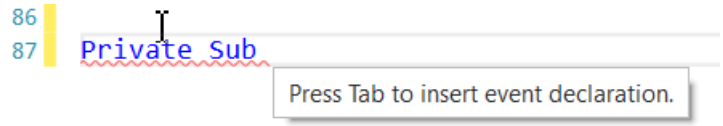
None

Members:

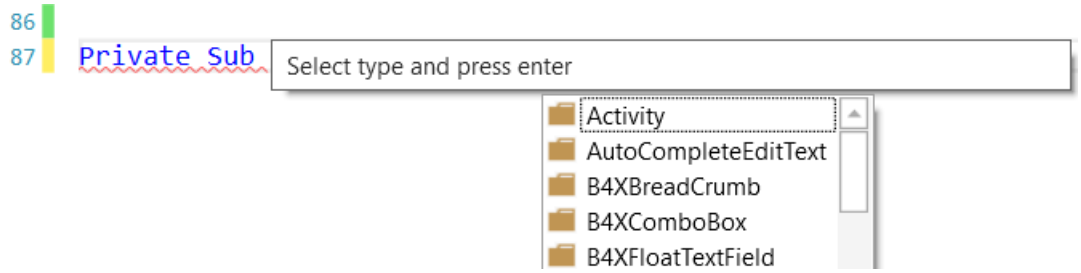
- **Add** (item As Object)
- **AddAll** (List As List)
- **AddAllAt** (Index As Int, List As List)
- **Clear**

5.3 Define an event routine.

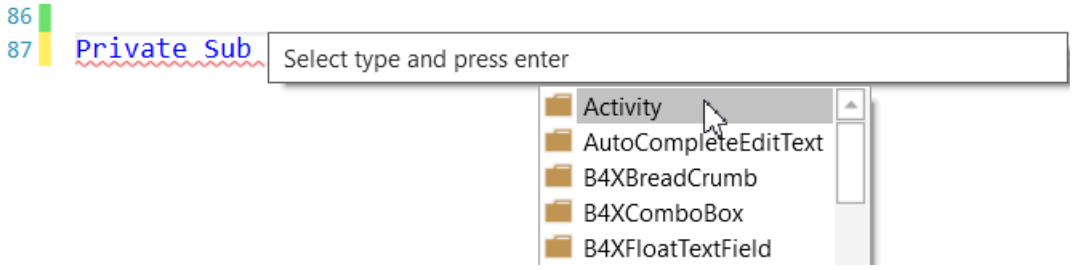
In the code type Private Sub or Sub and a space:



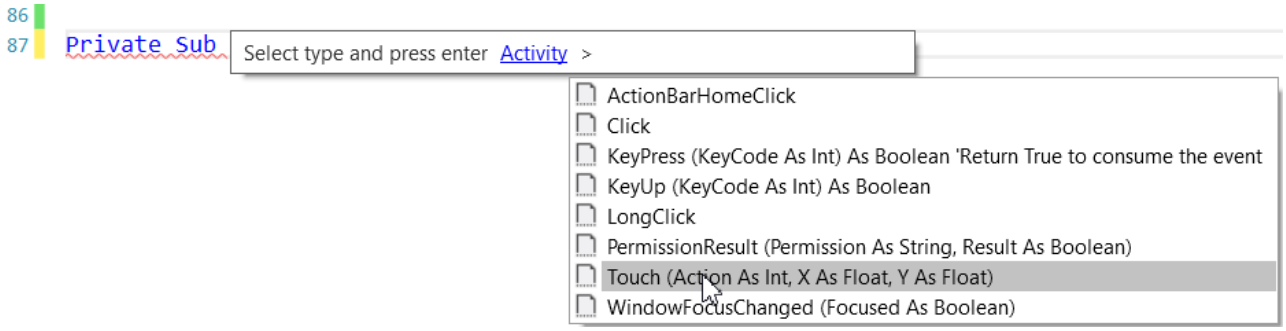
Then press Tab, you get the list of all obects possible in the project including those of the selected libraries.



Select an object, Activiy in the example:



Select the event:



Enter the object name and press Return.

```
87 Private Sub EventName_Touch (Action As Int, X As Float, Y As Float)
88
89 End Sub
```

And the result:

```
87 Private Sub Activity_Touch (Action As Int, X As Float, Y As Float)
88
89 End Sub
```

6 "Code smells" code to be avoided

"Code smells" are common patterns that can indicate that there is a problem in the code. A problem doesn't mean that the code doesn't work, it might be that it will be difficult to maintain it or that there are more elegant ways to implement the same thing.
Remember that not everything is clear cut and there are exceptions for any rule.

6.1 Initializing an object and then assigning a different object to the same variable

```
'bad
Dim List1 As List
List1.Initialize '<-- a new list was created here
List1 = SomeOtherList '<--- previous list was replaced

'good
Dim List1 As List = SomeOtherList
```

6.2 Deprecated methods - DoEvents, Msgbox

These methods are deprecated, so you should not these anymore.
More information here:
https://www.b4x.com/android/forum/t...cated-and-async-dialogs-msgbox.79578/#content

6.3 Deprecated methods - Map.GetKeyAt / GetValueAt

Deprecated methods - Map.GetKeyAt / GetValueAt - these methods were added before the For Each loop was available. They are not cross platform and are not the correct way to work with maps.

```
'bad
For i = 0 To Map1.Size - 1
   Dim key As String = Map1.GetKeyAt(i)
   Dim value As String = Map1.GetValueAt(i)
Next

'good
For Each key As String In Map1.Keys
 Dim value As String = Map1.Get(key)

Next
```

6.4 File.DirDefaultExternal - This is always a mistake.

File.DirDefaultExternal - This is always a mistake. In most cases the correct folder should be XUI.DefaultFolder (=File.DirInternal). If you do need to use the external storage then use RuntimePermissions.GetSafeDirDefaultExternal.
File.DirRootExternal - It will soon become inaccessible directly. If really needed then use ContentChooser or ExternalStorage.

6.5 Not using parameterized queries

For database queries, use parametrized queries.

```
'very bad
SQL.ExecNonQuery("INSERT INTO table1 VALUES ('" & EditText1.Text & "')") 'ugly, will
break if there is an apostrophe in the text and vulnerable to SQL injections.

'very good
SQL.ExecNonQuery2("INSERT INTO table1 VALUES (?)", Array(EditText1.Text))
```

6.6 Using Cursor instead of ResultSet - Cursor

For database queries, use ResultSet instead of Cursor.
Cursor is a B4A only object. ResultSet is a bit simpler to use and is cross platform.

```
'good
Dim rs As ResultSet = SQL.ExecQuery2(...)
Do While rs.NextRow
 ...
Loop
rs.Close
```

6.7 Building the complete layout programmatically

Building the complete layout programmatically. This is especially a mistake in B4J and B4i because of the resize event and also if you want to build a cross platform solution. Layouts can be ported very easily.

6.8 Repeating the code

There are many patterns to this one and all of them are bad.

```
'bad
If b = False Then
   Button1.Text = "disabled"
   Button2.Text = "disabled"
   Button3.Text = "disabled"
   Button1.Enabled = False
   Button2.Enabled = False
   Button3.Enabled = False
Else
   Button1.Text = "enabled"
   Button2.Text = "enabled"
   Button3.Text = "enabled"
   Button1.Enabled = True
   Button2.Enabled = True
   Button3.Enabled = True
End If

'good
For Each btn As Button In Array(Button1, Button2, Button3)
   btn.Enabled = b
   If b Then btn.Text = "enabled" Else btn.Text = "disable"
Next
```

6.9 Long strings without using smart strings

More information: https://www.b4x.com/android/forum/threads/50135/#content

```
'bad
Dim s As String = "This is the " & QUOTE & "first" & QUOTE & "line" & CRLF & _
 "and this is the second one. The time is " & DateTime.Time(DateTime.Now) & "."

'good
Dim s As String = $"This is the "first" line
and this is the second one. The time is $Time{DateTime.Now}."$
```

6.10 Using global variables when not needed

```
'bad
Job.Initialize(Me, "") 'global variable
...

'good
Dim job As HttpJob
job.Initialize(Me, "")
```

6.11 Not using Wait For when possible

Not using Wait For when possible. JobDone is a good example: [B4X] OkHttpUtils2 with Wait For

6.12 Using code modules instead of classes

Code modules are very limited in B4A. In most cases you should use classes instead of code modules. A code module is a single instance of a class.

6.13 Understanding booleans

```
'not elegant
Dim result As Boolean = DoingSomethingThatReturnTrueOrFalse
If result = True Then
   Return True
Else
   Return False
End If

' elegant
Return DoingSomethingThatReturnTrueOrFalse
```

6.14 Converting "random" bytes to strings

The only valid raw bytes that should be converted to a string, with BytesToString, are bytes that represent text. In all other cases it is a mistake to convert to string.
Even if it seems to work it will later fail in other cases.
If you think that it is more complicated to work with raw bytes then you are not familiar with the useful B4XBytesBuilder object:
https://www.b4x.com/android/forum/threads/b4x-b4xcollections-more-collections.101071/#content

6.15 Generating or parsing XML / JSON by hand.

Generating or parsing XML / JSON by hand. These formats are far from being trivial and with all kinds of edge cases that no one remembers.

```
   'bad
   Dim s As String = "{""version"":""" & Version & """,""colors"":[""red"",""green"",""blue""]}"

   'good
   Dim jg As JSONGenerator
   jg.Initialize(CreateMap("colors": Array("red", "green", "blue"), "version": Version))
   Log(jg.ToPrettyString(4))
```

7 Features that Erel recommends to avoid

Many things have changed in B4X and also in the underlying platforms. I will try to list here all kinds of (old) features that have better alternatives.
B4X is backward compatible so these features still work. The recommendations are more relevant for new projects or when implementing new features.

1. **(B4A) ListView ➤ xCustomListView.**
   ListView is difficult to work with and cannot be customized. It is also not cross platform.

2. **(B4i) TableView ➤ xCustomListView: same as above.**

3. **CustomListView module ➤ xCustomListView library.**
   Using the module will break other libraries.

4. **Sub JobDone ➤ Wait For (j) JobDone.**
   [B4X] OkHttpUtils2 with Wait For

5. **Sub Smtp_MessageSent (and others) ➤ Wait For ...**
   https://www.b4x.com/android/forum/threads/b4x-net-library-ftp-smtp-pop-with-wait-for.84821/#content

6. **DoEvents / Msgbox ➤** DoEvents deprecated and async dialogs (msgbox)

7. **All kinds of custom dialogs ➤ B4XDialogs.**
   B4XDialogs are cross platform and are fully customizable.
   [B4X] Share your B4XDialog + templates theming code

8. **File.DirDefaultExternal ➤ RuntimePermissions.GetSafeDirDefaultExternal.**
   https://www.b4x.com/android/forum/threads/67689/#content

9. **File.DirRootExternal ➤ ContentChooser / SaveAs.**
   https://www.b4x.com/android/forum/threads/132731/#content

10. **File.DirInternal / DirCache / DirLibrary / DirTemp / DirData ➤ XUI.DefaultFolder**

11. **Round2 ➤ NumberFormat, B4XFormatter**
    Most usages of Round2 are to format numbers. Modifying the number is not the correct way.

12. **TextReader / TextWriter with network streams ➤ AsyncStreams**
    Trying to implement network communication on the main thread will always result in bad results.

13. **TextReader / TextWriter ➤ File.ReadString / ReadList**
    Two exceptions - non-UTF8 files or huge files (more relevant to B4J).

14. **Activities ➤ B4XPages**
    This is a big change and it is the most important one. It is hard to explain how much simpler things are with B4XPages. The more complex the project the more important it is to use B4XPages. This is also true when building non-cross platform projects. [B4X] [B4XPages] What exactly does it solve?

15. **Platform specific API ➤ Cross platform API.**
    This is of course relevant when there is a cross platform API. Some developers have a misconception that the cross platform features have drawbacks compared to the platform specific API.
    - Node / Pane / Button / ... ➤ B4XView
    - Canvas > B4XCanvas
    - All kinds of platform specific custom views ➤ cross platform custom views (such as XUI Views).
    - EditText / TextField / TextArea / TextView ➤ B4XFloatTextField
    - fx (and others) ➤ XUI

16. **CallSubDelayed to signal a completion of a resumable sub ➤ As ResumableSub.**
    [B4X] Resumable subs that return values (ResumableSub)

17. **CallSubDelayed / CallSubPlus to do something a bit later ➤ Sleep(x).**

18. **Multiple layout variants ➤ anchors + designer script.**
    When Android was first released there were very few screen sizes. This is no longer the case. You should build flexible layouts that fill any screen size. It is easier to do with anchors + designer script. It is difficult to maintain multiple variants.

19. **Building the layout programmatically ➤ using the designer when possible.**
    If you are only developing with B4A then building the layout programmatically is a mistake but not a huge one.
    B4J and B4i handle screen resizes differently and it is much more difficult to handle the changes programmatically (there is video tutorial about it).
    Most custom views can only be added with the designer (there are workarounds that allow adding them programmatically).
    It is very simple to copy and paste designer layouts between different platforms and projects.

20. **Multiline strings with concatenation ➤ smart strings.**
    [B4X] Smart String Literal

21. **(SQL) Cursor ➤ ResultSet.**
    ResultSet is cross platform and is also a bit simpler to use.

22. **ExecQuery (non-parameterized queries) ➤ ExecQuery2.**
    Making non-parameterized queries is really unacceptable. See point #5 for more information: https://www.b4x.com/android/forum/t...ommon-mistakes-and-other-tips.116651/#content
    It is also true for ExecNonQuery

23. **ExecQuerySingleResult when it is possible that there are no results ➤ ExecQuery2.**
    This is a historic design mistake. Nulls and Strings don't go together. If there is a possibility that ExecQuerySingleResult will return no results (=Null) then don't use it and use ExecQuery2 instead.

24. **Downloading / making http requests with any other library or source other than OkHttpUtils2 (=iHttpUtils2) ➤ OkHttpUtils2.**
    OkHttpUtils2 is very powerful and can be extended in many ways, without modifying the source. It is also very simple to use.

25. **Shared modules folder ➤ referenced modules.**
    The shared modules feature was useful in the early days of B4X. With the introduction of referenced modules, there is no good reason to use it. Referenced modules cover the same use cases and more.

26. **VideoView ➤ ExoPlayer**
    ExoPlayer is much more powerful and more customizable.

8 Tips

These are Erels' tips for B4X developers ([B4X] Tips for B4X developers).

8.1 Separate data from code

Putting the data directly into the code makes your program unreadable and less maintainable.
There are many simple ways to deal with data. For example you can add a text file to the Files tab and read it to a List with:

```
Dim data As List = File.ReadList(File.DirAssets, "SomeFile.txt")
```

8.2 Don't Repeat Yourself (DRY principle).

If you find yourself copying and pasting the same code snippet multiple times and then making a small change then it is a good idea to stop and try to find a more elegant solution.
Repeated code is difficult to maintain and update. The Sender keyword can help in many cases (old and still relevant tutorial: Tick-Tack-Toe: working with arrays of views).

8.3 Map collection

All developers should know how to use a Map collection. This is by far the most useful collection. Tutorial: https://www.b4x.com/android/forum/threads/map-collection-the-most-useful-collection.60304/

8.4 New technologies and features.

Don't be afraid to learn new things. As developers we always need to learn new things. Everything is evolving whether we want it or not. I will give MQTT as a good example. I wasn't familiar with this technology. When I started learning about it I was a amazed to see how easy and powerful this solution is.
B4X specific features that all developers should be aware of:
- Smart strings literal: https://www.b4x.com/android/forum/threads/50135/#content
- For Each iterator: https://www.b4x.com/android/forum/threads/loops.57877/
- Classes: https://www.b4x.com/android/forum/threads/18626/#content

8.5 Logs

You should monitor the logs while your app is running. Especially if there is any error. If you are unable to see the logs for some reason then take the time to solve it. Specifically with B4A-Bridge the logs will only appear in Debug mode. If you

encounter an issue that only happens in release mode then you need to switch to usb debug mode.

8.6 B4A Avoid calling DoEvents.

DoEvents interferes with the internal message queue. It can cause unexpected issues. There are very few cases where it is required. This was not the case when B4A v1.0 was released. Since then the libraries have evolved and now offer better solutions. For example if the database operations are too slow (and you are correctly using transactions) then you should switch to the asynchronous methods. Or you should use Sleep or Wait For.

8.7 Strings are made of characters not bytes.

Don't try to store raw bytes as strings. It doesn't work. Use arrays of bytes instead. The proper way to convert bytes to strings is with base 64 encoding or ByteConverter.HexFromBytes.

8.8 B4A Use services, especially the Starter service

Services are simpler than Activities. They are not paused and are almost always accessible.
Three general rules about global variables:
1. All non-UI related variables should be declared in Process_Globals.
2. Public (process_global) variables should be declared and set / initialized in Service_Create of the Starter service.
3. Activity process globals should only be initialized if FirstTime is true.

This is only relevant to B4A. It is simpler in B4J and B4i as there is no special life cycle and the modules are never paused.

8.9 UI Layouts

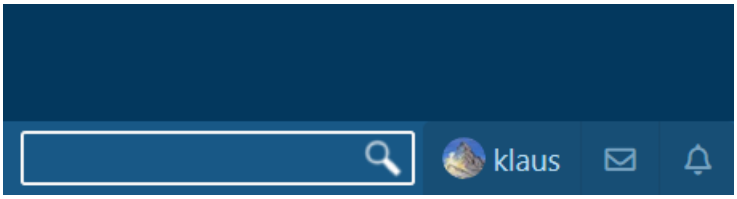
B4X provides several tools to help you implement flexible layouts that adapt to all screen sizes. The main tools are: anchors and designer script. Avoid adding multiple variants (two are fine). Variants were introduced in v1.00, before the other features. Variants are difficult to maintain and can be replaced with scripts.
Anchors are very simple and powerful.
Don't overuse percentage units (unless you are building a game).

8.10 B4J as a backend solution.

B4A, B4i, B4J share the same language, same concepts and mostly the same APIs. It is also simple to exchange data between the different platforms with B4XSerializator.
It is easy to implement powerful server solutions with B4J. Especially when the clients are implemented with B4A, B4i or B4J.

8.11 Search.

Use the forum search feature. You can filter results by adding the platform(b4a for example) to the query or by clicking on one of the filters in the results page.
Most of the questions asked in the forum can be solved with a few searches.

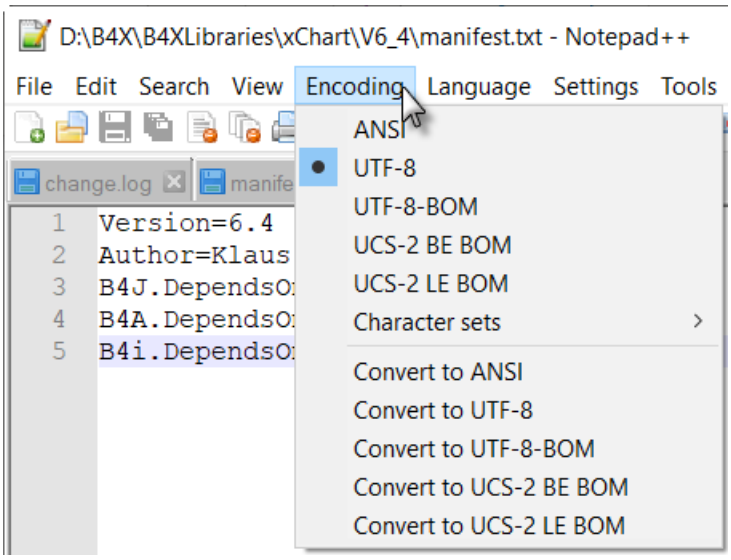


8.12 Notepad++.

At one point or another we need to work with text files. I highly recommend all developers to use a good text editor that shows the encoding, the end of line characters and other important features. https://notepad-plus-plus.org/

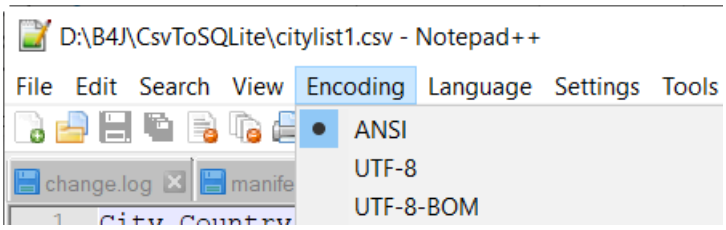
8.12.1 Encoding

To show the current encoding of a text file, you can load it and then chlick in the menu on Encoding. The current encoding is checked.
You can select another encoding and save the file.

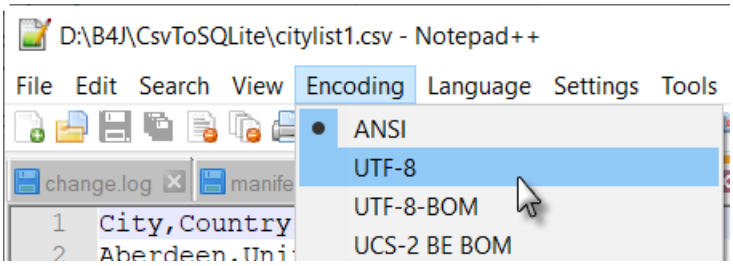


This can be useful when you have csv files generated with Excel, which are encoded with ANSI encoding, but, B4X uses UTF-8 encoding.

Original file:



Change the encoding and save the file with another file name.



When you reload this file and check the encoding, you will see this: