



The New Kid on the Block

A project by Wayne Phillips

creator of vbWatchdog and other fine products at everythingaccess.com

Presented by Mike Wolfe of nolongerset.com

Outline

- IDE (VS Code)
- Debugging
- Compatibility
- New features
- Compilation benefits
- Current limitations
- Future plans
- **My bold prediction**

IDE (Visual Studio Code)

- Semantic highlighting

```
Class oVehicle
Public Make As String
Public Model As String
Public Year As Integer

Public Sub New(Make As String, Model As String, Year As Integer)
    Me.Make = Make
    Me.Model = Model
    Me.Year = Year
End Sub
End Class
```

IDE (Visual Studio Code)

- Semantic highlighting
- Inline parameter hints

```
MsgBox "Hello world!", vbInformation, Title:="Greetings"
```

```
MsgBox "Hello world!", vbInformation, "Greetings"
```



```
MsgBox prompt:= "Hello world!", buttons:= vbInformation, title:= "Greetings"
```

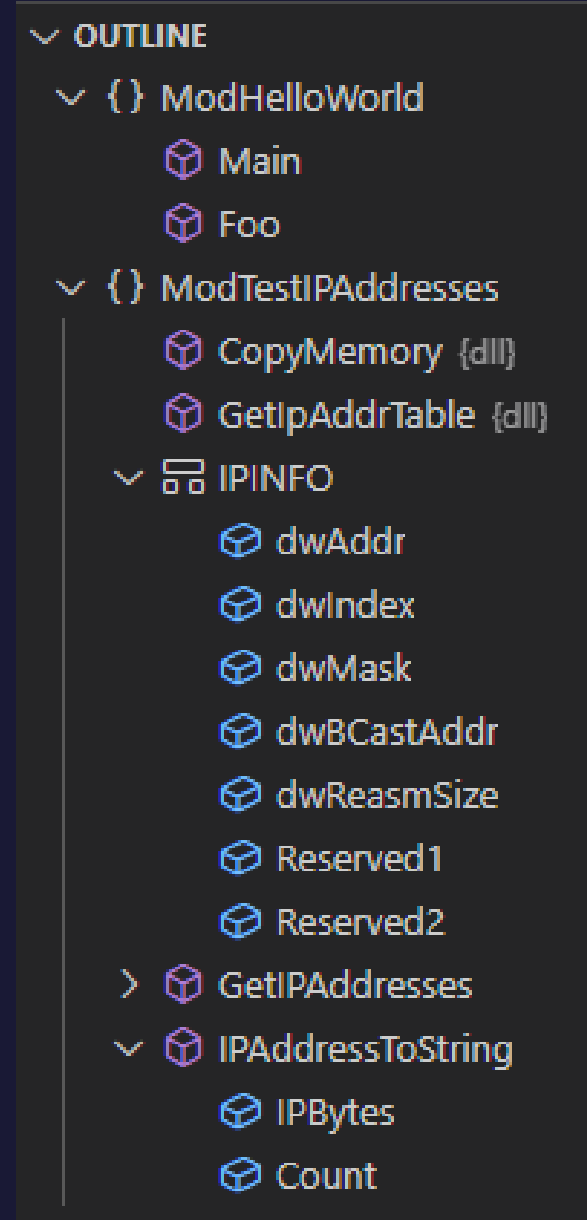
IDE (Visual Studio Code)

- Semantic highlighting
- Inline parameter hints
- Code folding

```
5 > .....Public Sub Main() ...  
27 .....End Sub  
28  
29 > → Public Function Foo() ...  
39 → End Function  
--
```

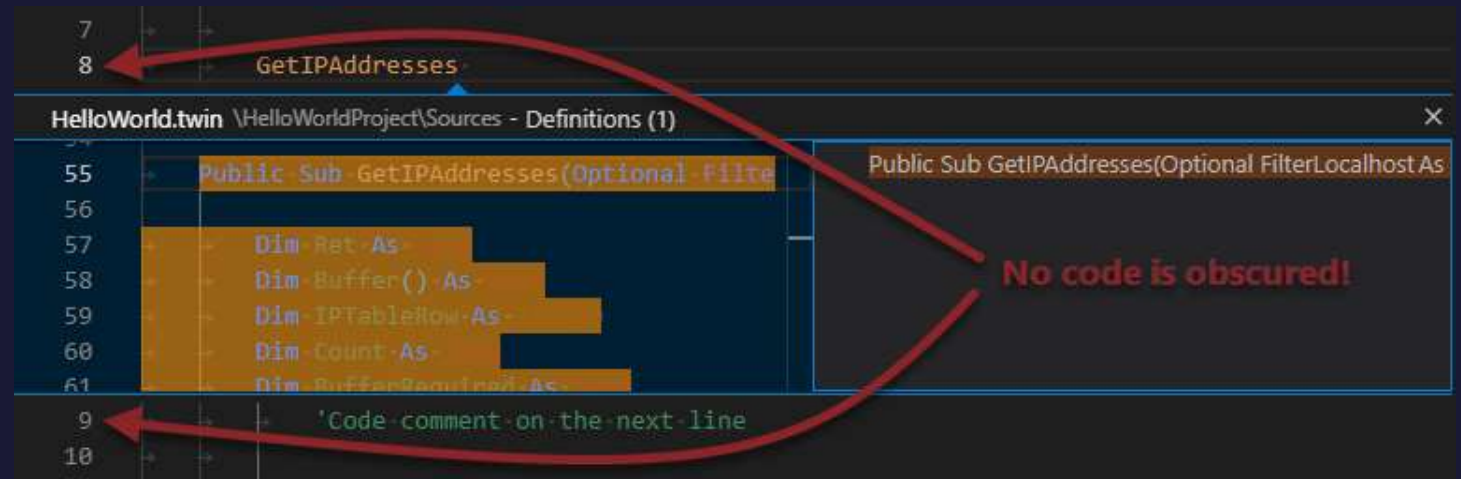
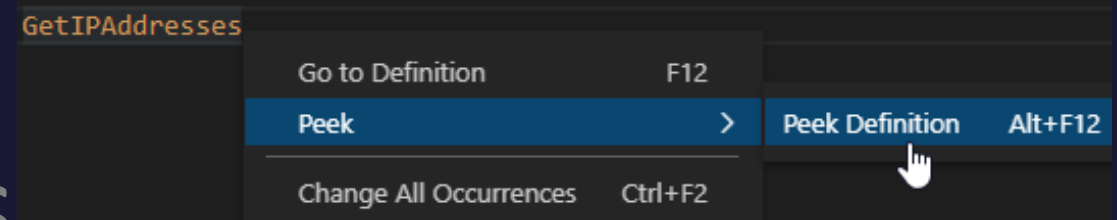
IDE (Visual Studio Code)

- Semantic highlighting
- Inline parameter hints
- Code folding
- Outline view



IDE (Visual Studio Code)

- Semantic highlighting
- Inline parameter hints
- Code folding
- Outline view
- Peek definition



Debugging

- Debug Console

```
DEBUG CONSOLE
→ ?123
123
(time taken: 0.0001244s)
→ ?123 * 2
246
(time taken: 0.0001346s)
→ ?GetAccessVersionInfo
Microsoft Access 16.0
(time taken: 0.4862512s)
> ?"Next command to run"
```

BUILD CONFIGURATIONS > __BUILDS > win32

Debugger: Console Log Time Taken *Inherited from [Defaults]*

if YES, the debugger will report the time taken to execute the line of code immediately beneath the result

Yes ▾

Debugging

- Debug Console
- Breakpoints

```
21 Public Sub TestBreakpoints()  
22     Debug.Print "A"  
23 End Sub
```

Nothing to break on here!

```
155 Sub InlineBreakpoints()  
156     Const FeatureIsWorking As Boolean = False  
157  
158     If FeatureIsWorking Then IWillNotBreak  
159 End Sub
```

▼ BREAKPOINTS

●	✓	HelloWorld.twin	\HelloWorldProject\Sources	
●	✓	HelloWorld.twin	\HelloWorldProject\Sources	28
●	✓	HelloWorld.twin	\HelloWorldProject\Sources	32
●	✓	HelloWorld.twin	\HelloWorldProject\Sources	42
●	✓	HelloWorld.twin	\HelloWorldProject\Sources	44
●	✓	HelloWorld.twin	\HelloWorldProject\Sources	83

Toggle Activate Breakpoints

Debugging

- Debug Console
- Breakpoints
- Locals window

```
▼ VARIABLES
  ▼ Locals
    Foo: Empty
    i: 0
    s: vbNullString
```

```
▼ Locals
  ▼ Auto: {object: 186437976}
    Make: "Ford"
    Model: "Mustang"
    Year: 1968
  ▼ Lot: {object: 186269456}
    {element: 0000}: {object: 186437976}
    {element: 0001}: {object: 186438016}
  ▼ Vehicle: {object: 186438016}
    Make: "Chevy"
    Model: "Chevelle"
    Year: 1971
```

Debugging

- Debug Console
- Breakpoints
- Locals window
- Watch window



Debugging

CALL STACK

COMPILER [Win32] v0.9.1622 RUNNING

MAIN_THREAD [EXECUTING] BREAKPOINT

Go	HelloWorld.twin	151:1
You	HelloWorld.twin	150:1
Can	HelloWorld.twin	149:1
Low	HelloWorld.twin	148:1
How	HelloWorld.twin	147:1
Limbo	HelloWorld.twin	146:1

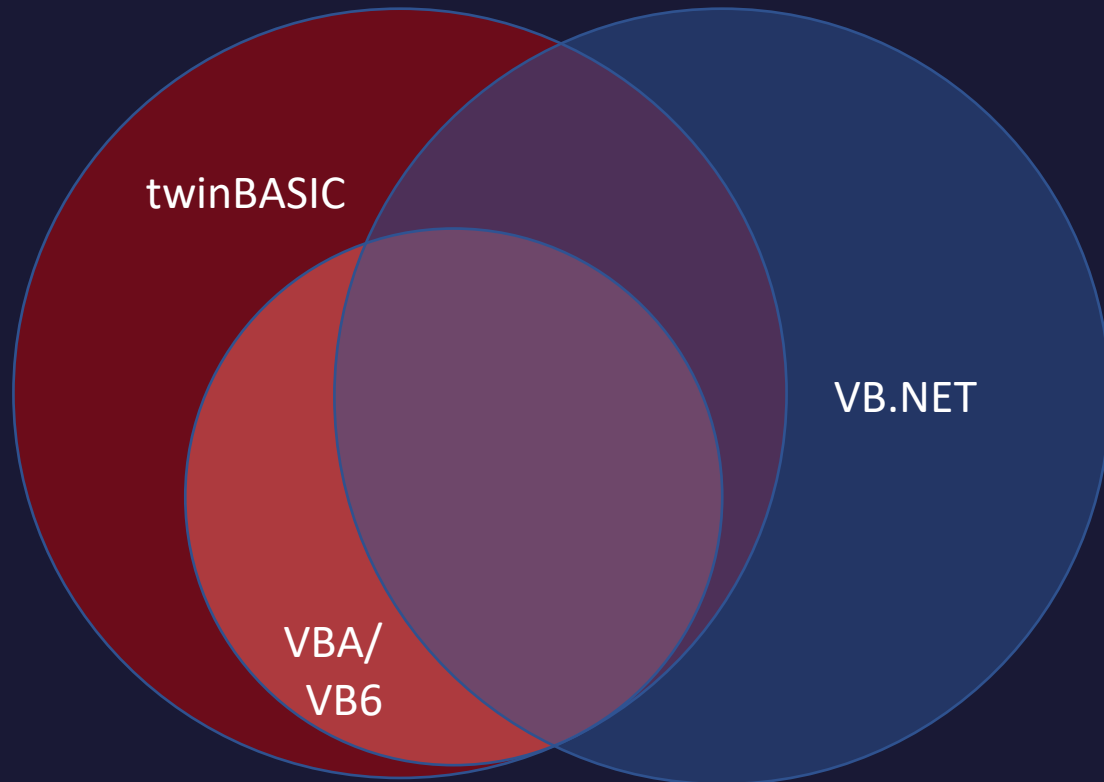
145
146
147
148
149
150
151
152

```
Module DemoCallStack
... Public Sub Limbo() : Call How : End Sub
... Public Sub How() : Call Low : End Sub
... Public Sub Low() : Call Can : End Sub
... Public Sub Can() : Call You : End Sub
... Public Sub You() : Call Go : End Sub
... Public Sub Go() : Debug.Print "Very Low" : End Sub
End Module
```

- Call stack window

Compatibility

- VBA/VB6 vs. VB.NET vs. twinBASIC



Compatibility

- VBA/VB6 vs. VB.NET vs. twinBASIC
- Pareto Principle: the 80/20 Rule

*The difference between
100% compatible and 99% compatible
is way more than 1%.*

Compatibility

- VBA/VB6 vs. VB.NET vs. twinBASIC
- Pareto Principle: the 80/20 Rule
- Quirks Preserved



Compatibility

- VBA/VB6 vs. VB.NET vs. twinBASIC
- Pareto Principle: the 80/20 Rule
- Quirks Preserved
- Test, Test, Test

Compatibility

Getting Help

(Screenshot from twinbasic.com)

twinBASIC issue tracking is now live at github. We've been inundated with bug reports and feature requests from many different sources. Going forward, it would help immensely if you could report bugs and post feature requests over at the repository which will help us better keep track and manage requests.

[twinBASIC - GitHub](#)

[twinBASIC - GitHub Issues](#)

Report issues here

Alternatively, you can get in touch with the lead developer on twitter:

 Follow @WaynePhillipsEA

- Report Bugs



New Features

- Initialize on declare

```
Dim i As Integer = 1  
Dim s As String = "Default"  
Dim d As Date = Now()
```

New Features

- Initialize on declare
- Return syntax in functions

```
Function GetCurrentHour() As Byte
    Return Hour(Now)
End Function
```

```
74 Module TestReturnCalls
75     ... Function Traditional() As String
76         ... Traditional = "Traditional Function Syntax"
77         ... Debug.Print "Some cleanup code"
78     ... End Function
79     ...
80     ... Function NewReturnSyntax() As String
81         ... Return "New Return Syntax"
82         ... Debug.Print "Some more cleanup"
83     ... End Function
84     ...
85     ... Sub TestReturnSyntax()
86         ... Debug.Print Traditional()
87         ... Debug.Print NewReturnSyntax()
88     ... End Sub
89 End Module
```

DEBUG CONSOLE

```
TestReturnSyntax
Some cleanup code
Traditional Function Syntax
New Return Syntax
(time taken: 0.000119s)
```

This line never executes!

Filter (e.g. t

New Features

- Initialize on declare
- Return syntax in functions
- Parameterized constructors

Not compatible
with COM-visible
classes!

```
Class oVehicle
  Public Make As String
  Public Model As String
  Public Year As Integer

  Public Sub New(Make As String, _
                Model As String, _
                Year As Integer)
    Me.Make = Make
    Me.Model = Model
    Me.Year = Year
  End Sub
End Class
```

```
Sub CtorTest()
  Dim Vehicle As oVehicle
  Set Vehicle = New oVehicle("Ford", "Mustang", 1968)
End Sub
```

New (ByRef Make As String, ByRef Model As String,
ByRef Year As Integer)

New
in oVehicle

no further information available. to add documentation here,
add Attribute VB_Description = "" inside the declaration
of the procedure.

New Features

- Initialize on declare
- Return syntax in functions
- Parameterized constructors
- Method overloading

```
Function Halve(Text As String) As String  
    Return Left(Text, Len(Text)\2)  
End Function
```

```
Function Halve(Value As Single) As Single  
    Return Value / 2  
End Function
```

```
Function Halve(Value As Currency) As Currency  
    Return Round(Value/2, 2)  
End Function
```

DEBUG CONSOLE

```
→ ?Halve("twinBASIC")  
twin  
→ ?Halve(CSng(0.51))  
0.255  
→ ?Halve(CCur(0.51))  
0.26
```

New Features

- Initialize on declare
- Return syntax in function
- Parameterized constructors
- Method overloading
- Short circuiting Booleans

```
Sub ShortCircuit()
```

AndAlso

```
If False And 1/0 Then Debug.Print "I will error at runtime"  
If False AndAlso 1/0 Then Debug.Print "No runtime error"
```

OrElse

```
If True Or 1/0 Then Debug.Print "I will error at runtime"  
If True OrElse 1/0 Then Debug.Print "No runtime error"
```

Ternary If() statement

```
Debug.Print IIf(True, "I will error at runtime", 1/0)  
Debug.Print If(True, "No runtime error", 1/0)
```

```
End Sub
```

New Features

- Initialize on declare
- Return syntax in functions
- Parameterized constructors
- Method overloading
- Short circuiting Booleans
- Syntax conveniences

- `obj IsNot Nothing`

- `Continue For`
- `Continue While`
- `Continue Do`
- `Exit While`

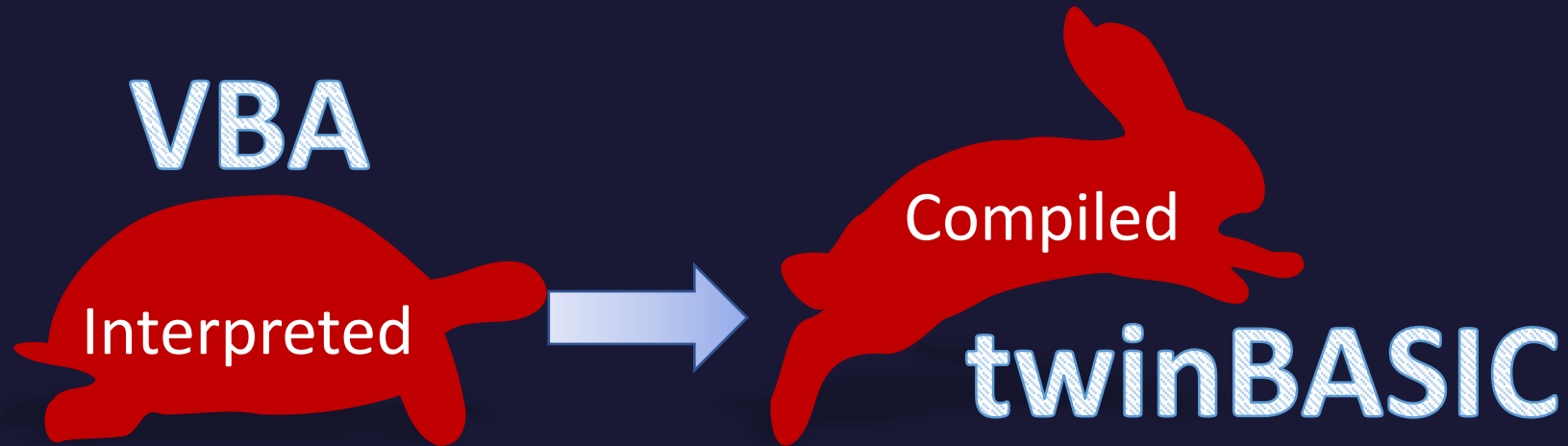
- `i += 42`
- `i -= 26`
- `s &= "a"`

...and then some

- Generics
- Unicode everywhere (DeclareWide)
- WebEx/vbWatchdog libraries (coming soon)
- Per-procedure Handles/Implements
- Multiple modules in one file
- New datatypes: LongLong, LongPtr, Decimal

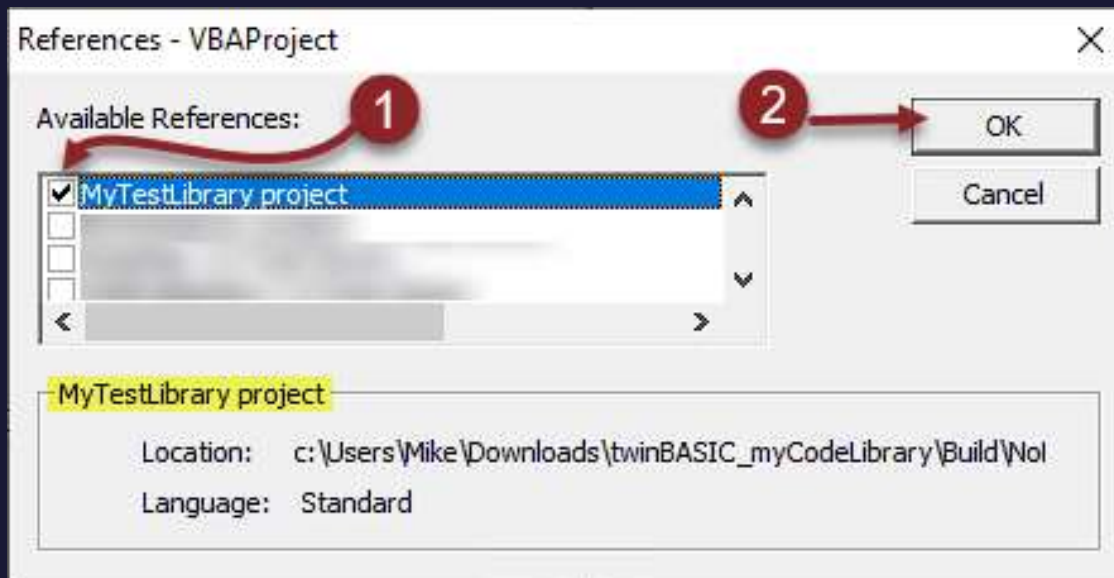
Compilation Benefits

- Native code is faster




Compilation Benefits

- Native code is faster
- Create ActiveX DLLs



Compilation Benefits

- Native code is faster
- Create ActiveX DLLs
- Lightweight executables

Name	Size
 HelloWorldProject_win32.exe	8 KB



Compilation Benefits

- Native code is faster
- Create ActiveX DLLs
- Lightweight executables
- Requires no runtime

	Runtime (MB)	Hello, world (KB)
MS Access	315.6	N/A
Java	69.74	N/A
.NET Framework	58.7	N/A
.NET Core	52.4	N/A
Python	8.03	N/A
VB6	1	N/A
Go	0	1,900
AutoHotkey	0	1,082
twinBASIC	0	8

Current Limitations

- Generic Error Descriptions

```
231 Sub GenericErrorText()  
232     Dim Coll As Collection  
233  
234     Coll.Add "Item one"  
235 End Sub
```

Exception has occurred.
Unspecified error
Error Number: -2147467259 (&H80004005)

```
Sub GenericErrorText()  
    Dim Coll As Collection  
  
    Coll.Add "Item one" ' <-- Error here
```

Microsoft Visual Basic

Run-time error '91':
Object variable or With block variable not set

Current Limitations

- Generic Error Descriptions
- No `Like` operator

expected operator between the operands

View Problem (Alt+F8) No quick fixes available

```
If "Test Case" Like "Test*" Then Debug.Print "True"
```

Current Limitations

- Generic Error Descriptions
- No `Like` operator
- No Global or Module-level variables

Released yesterday!

```
Sorry, global variables are not yet supported in this preview!
```

```
View Problem (Alt+F8) No quick fixes available
```

```
GlobalVariable As String
```

Current Limitations

- Generic Error Descriptions
- No `Like` operator
- ~~No Global/Module-level variables~~
- **Debug.Print** syntax not supported

```
Module PreviewLimitations
  Sub DebugPrintSyntax()
    Debug.Print "Bitness: " & 32, "Year: "; 2021
  End Sub
End Module
```

syntax error. no handler for this symbol
View Problem (Alt+F8) No quick fixes available

Current Limitations

- Generic Error Descriptions
- No `Like` operator
- ~~No Global/Module-level variables~~
- `Debug.Print` syntax not supported
- No built-in file operations
Use `FileSystemObject` for now

Future Plans

- GUI framework

**Full backwards compatibility
with existing VB6 forms!**

Future Plans

- GUI framework
- Git VS Code integration
Via a Git provider for VS Code

Future Plans

- GUI framework
- Git VS Code integration
- Multi-threading

Requires new syntax

Future Plans

- GUI framework
- Git VS Code integration
- Multi-threading
- 64-bit support

Will require a commercial license

Future Plans

- GUI framework
- Git VS Code integration
- Multi-threading
- 64-bit support
- Cross-platform support (via LLVM)

My Bold Prediction

- 1999: VBA6 => COM Add-in support
- 2010: VBA7 => 64-bit support
- 2025: VBA8 => **twin BASIC**

Wayne gets paid



Questions?

Links and Resources:

nolongerset.com/devcon-2021/