

Visual Basic Cheat Sheet

Visual Basic 17.13 · .NET 10 (LTS) · Visual Studio 2026 · checked 2026-10-19 · [albertoirurueta.github.io/docs](#) → Programming Languages → Visual Basic Reference

Toolchain and project

```
dotnet new console|classlib|winforms|wpf -lang VB -o Name
dotnet new mstest|xunit|nunit -lang VB -o Name.Tests
dotnet new sln; dotnet sln add A B; dotnet add A reference B
dotnet build | run | test | publish -c Release | pack
dotnet msbuild -getProperty:OptionStrict
<Project Sdk="Microsoft.NET.Sdk"><PropertyGroup>
  <OutputType>Exe</OutputType>
  <TargetFramework>net10.0</TargetFramework>
  <RootNamespace>Shelf.Cli</RootNamespace>
  <OptionStrict>On</OptionStrict>
  <OptionInfer>On</OptionInfer>
  <OptionCompare>Binary</OptionCompare>
  <LangVersion>latest</LangVersion>
  <UseWindowsForms>true</UseWindowsForms>
</PropertyGroup></Project>
```

SDK defaults: OptionStrict **Off**, OptionExplicit and OptionInfer On, OptionCompare Binary; set them in Directory.Build.props. MyType selects the My members. Implicit imports: Microsoft.VisualBasic, System, Collections.Generic, Diagnostics, Linq, Xml.Linq, Threading.Tasks. LangVersion: 15.5, 16.9, 17.13, latest (17.14 is rejected).

File skeleton and lexical rules

```
Option Strict On ' first lines; or set in the .vbproj
Imports System.Text
Imports Alias = System.Collections.Generic.List(Of String)
Namespace Shelf.Cli
  Module Program
    Sub Main(args As String())
      Console.WriteLine($"Hello, {args.Length}!")
    End Sub
  End Module
End Namespace
' comment REM comment ''' <summary>doc</summary>
Dim total As Integer = 1 + 2 ' implicit line
    continuation (VB 10)
Dim s = "a" & _ ' explicit _ only where
    "b"
    still needed
Dim a = 1 : Dim b = 2 ' : separates statements
Dim [End] As Integer ' [Keyword] escapes a reserved word
```

Names are **case-insensitive** (name = Name assigns to itself). Block statements end with End X; a line break ends a statement.

Types, literals and conversions

type	bytes	suffix
Byte SByte	1	none
Short UShort	2	S US
Integer UInteger	4	I % UI
Long ULong	8	L & UL
Single Double	4 8	F ! R #
Decimal	16	D @
Char Boolean	2/1-2	"a" c True False
String Date Object	ref	\$ #10/20/2026#
&HFF 5017 6B1010 1_000_000 ' hex, octal, binary, digit separators		
"say ""hi"" ok" \$" {x:N2} {y,8}" Nothing = default value Dim n As Integer?		
CInt CLng CDbt CDec CStr CBool CDate CChar CByte CShort CUShort CObj CType(x, T) DirectCast(o, T) TryCast(o, T) Integer.TryParse(s, v) x.ToString()		
CInt(2.5) = 2 CInt(3.5) = 4 ' banker's rounding; Fix/Int truncate/floor		
CInt(True) = -1 ' True = -1; Option Strict On: no narrowing		

Operators, high to low precedence

```
^ | unary - + * / \ | Mod | + - | & | << >> |
= <> < > <= >= Is IsNot Like TypeOf x Is T | Not |
And Also And
Or OrElse Xor
7 / 2 = 3.5 7 \ 2 = 3 -7 Mod 2 = -1 2 ^ 3 = 8.0 (Double)
s1 & s2 a AndAlso b a OrElse b x Is Nothing x IsNot Nothing
s Like "a#[x-z]*" If(c, a, b) If(a, b) ' IIf evaluates both branches
obj?.Name arr?(0) NameOf(x) GetType(T) AddressOf M Await t
x += 1 x -= 1 x *= 2 x \= 2 x ^= 2 x &= "s" x <=< 1

And/Or do not short-circuit; use AndAlso/OrElse. = is assignment and comparison; no ++, no ??.
```

Control flow

```
If a Then ... ElseIf b Then ... Else ... End If
If a Then x() Else y()
Select Case n : Case 1, 2 : Case 3 To 5 : Case Is > 9 : Case Else : End Select
For i As Integer = 0 To 10 Step 2 ... Next
For Each x In items ... Next
Do While c ... Loop Do ... Loop While c
Do Until c ... Loop
While c ... End While Exit
For|Do|While|Sub|Function|Try Continue
For|Do|While
With obj : .Name = "x" : End With
Using f As New StreamReader(p) ... End Using
SyncLock gate ... End SyncLock
GoTo label avoid
' For bounds are evaluated once; the loop variable is not visible after Next
```

Procedures and lambdas

```
Sub Log(msg As String, Optional level As Integer = 0) ... End Sub
Function Add(a As Integer, ByRef b As Integer, ParamArray r() As Integer) As Integer
  Return a + b ' or: Add = a + b (older style)
End Function
Log(level:=2, msg:="x") ' named arguments; ByVal is the default
Sub Counter() : Static n As Integer : n += 1 : End Sub
Function F(x As Integer) ... Function F(x As String) ... ' overloads
Dim sq = Function(x As Integer) x * 2 ' single-line lambda
Dim act = Sub(m As String) Console.WriteLine(m)
Dim f2 = Function(x As Integer) Return x + 1 End Function ' multi-line lambda
Iterator Function Evens() As IEnumerable(Of Integer) : Yield 2 : End Function
<Extension()> Function Shout(s As String) As String ... ' in a Module
Async Function GetAsync() As Task(Of String) ... Return Await t
```

Types and members

```
Public Class Book : Inherits CatalogItem : Implements IComparable(Of Book)
  Private _isbn As String : Public Shared Count As Integer
  Public Const Max As Integer = 9 : Public ReadOnly Id As Integer
  Sub New(isbn As String) : MyBase.New() : _isbn = isbn : End Sub
  Public Property Title As String = "t" ' auto-property
  Public ReadOnly Property Len As Integer
  Get : Return Title.Length : End Get
  End Property
  Default Public Property Item(i As Integer) As String ... End Property
  Public Overrides Function ToString() As String ... End Function
End Class
Structure P : Public X As Integer : End Structure
Module M : Public Sub Main() ...
Interface IStore : Function Find(id As Integer) As Book : End Interface
Enum Color As Byte : Red = 1 : Blue : End Enum
<Flags()> Enum Perm ... End Enum
Delegate Sub Handler(x As Integer) Partial Class C ... End Class
Dim b As New Book("x") With {.Title = "t"} New List(Of Integer) From {1, 2}
Dim t = (Id=-1, Title:="x") Dim anon = New With {.A = 1, Key .B = 2}
```

Public Friend Private	Protected Protected Friend Private Protected
Overridable Overrides NotOverridable	MustOverride MustInherit NotInheritable
Shadows Overloads Shared	Me MyBase MyClass

Generics and events

```
Class Repo(Of T As {Class, New, IComparable(Of T)}) : End Class ' Structure, Class, New
Function Max(Of T As IComparable(Of T))(a As T, b As T) As T
Interface ISource(Of Out T) Interface ISink(Of In T) ' variance
Event Returned As EventHandler(Of EventArgs)
RaiseEvent Returned(Me, EventArgs.Empty)
Private WithEvents btn As Button
Sub OnClick(s As Object, e As EventArgs) Handles btn.Click
AddHandler btn.Click, AddressOf OnClick
RemoveHandler btn.Click, AddressOf OnClick
Custom Event X As EventHandler : AddHandler(v) ... RemoveHandler(v) ... RaiseEvent ... End Event
```

Arrays, collections, LINQ and XML

```
Dim a(4) As Integer ' upper bound: 5
elements Dim b = {1, 2, 3}
ReDim Preserve a(9) Erase a Dim m(2, 3) As Integer Dim j()() As Integer
List(Of T) Dictionary(Of K, V) HashSet(Of T) Queue(Of T) Stack(Of T)
Dim q = From b In books Where b.Year > 2000 Order By b.Title Descending
    Select b.Title, b.Year
From b In books Group By b.Year Into Count()
Aggregate b In books Into Sum(b.Pages), Avg = Average(b.Pages)
From a In xs Join c In ys On a.Id Equals c.Id Let n = a.Name.Length
books.Where(Function(b) b.Year > 2000).Select(Function(b) b.Title).Distinct().Skip(1).Take(5)
Dim x = <book id="1"><title><%= t %></title></book>
Imports <xlns:s="urn:shelf"> x.<title>.Value x.<id> x...<author>
```

Errors, My and runtime library

```
Try
  Throw New InvalidOperationException("msg")
Catch ex As IOException When ex.Result = 5
Catch ex As Exception
  Throw ' rethrow keeps the stack; Throw ex resets it
Finally
End Try
Class ShelfException : Inherits Exception : End Class ' custom exception
On Error GoTo h / On Error Resume Next / Err.Number Err.Description ' legacy; never with Try
My.Application.Info.Version My.Settings.Name My.Resources.Logo My.UserName ' console: few My members
My.Computer.FileSystem.ReadAllText(p) ' WinForms projects only; else System.IO
Mid(s, 1, 3) is 1-based; s.Substring(0, 3) is 0-based ControlChars.CrLf vbCrLf
Dim p As New TextFieldParser(path) With {.Delimiters = {","}} ' CSV
Environment.Exit(0) Err.Raise(5) Format(d, "yyyy-MM-dd") Len(s) UCase(s)
```

Directives, attributes, async and threads

```
#If DEBUG Then ... #ElseIf TRACE Then ... #Else ...
#End If #Const Mode = 1
#Region "Name" ... #End Region #Disable Warning BC42024 ... #Enable Warning BC42024
<Serializable()> Class C <Assembly: InternalsVisibleTo("T")> <Obsolete("x")> Sub S() Sub Log<CallerMemberName> Optional m As String = Nothing)
Sub Main() : RunAsync().GetAwaiter().GetResult() : End Sub ' no async Main in VB
Async Function RunAsync() As Task
  Dim r = Await Task.WhenAll(t1, t2) Await Task.Delay(100, ct)
End Function
Async Sub only for event handlers ct.ThrowIfCancellationRequested()
SyncLock gate ... End SyncLock Interlocked.Increment(n) Parallel.ForEach(xs, Sub(x) ...)
Task.Run(Sub() ...) Await Using / Await For Each do not exist in VB
```

Interop one-liners

```
Declare Auto Function MessageBox Lib "user32.dll" (h As IntPtr, t As String, c As String, f As Integer) As Integer
<DllImport("Lib")> Private Shared Function getpid() As Integer : End Function
Dim xl = CreateObject("Excel.Application") ' late-bound COM (Windows)
#< -> VB: reference entire project; VB cannot declare required/init members or generic math
```

Best practices and gotchas

Do: Option Strict On; & not + to join strings; AndAlso/OrElse; Is Nothing; If() not IIf; Using for IDisposable; Try not On Error; TryParse over CInt; Await never .Result.

Gotchas: Dim a(4) has 5 elements; / gives Double, \ integer division; ^ is Double; Nothing on a value type is its default; overflow checks are on; Static and uninitialized loop locals keep old values; runtime functions are 1-based, String methods 0-based; Option Compare drives string comparison; dates and numbers are culture-sensitive; Name = name in a constructor assigns the parameter to itself; MsgBox/InputBox need Windows Forms.

Legacy → **modern** (VB 17.13 on .NET 10; the version is where the modern form arrived)

VB6 Integer/Long (16/32-bit) → Short/Integer (Long is 64-bit) (7.0)
Variant → Object + Option Strict On (7.0)
Type ... End Type → Structure (7.0)
On Error GoTo/Resume Next → Try/Catch/Finally (7.0)
Set obj = ... → obj = ... (7.0)
ByRef by default → ByVal default; write ByRef (7.0)
GoSub/Return → Sub/Function procedures (7.0)

Open ... For Output As #1 → StreamWriter/File.WriteAllText (7.0/8.0)
FileSystemObject → System.IO (7.0)
Scripting.Dictionary → Dictionary(Of K, V) (8.0)
IIf(c, a, b) → If(c, a, b) (9.0)
_ on every continued line → implicit line continuation (10.0)
System.Data.SqlClient → Microsoft.Data.SqlClient (.NET)
EF6 / LINQ to SQL → Entity Framework Core (EF Core 10)

BinaryFormatter → System.Text.Json (removed .NET 9)

HttpRequest → HttpClient (.NET 6 SYSLIB0014)

Web Forms → C# ASP.NET Core + VB libraries (ASP.NET Core)

Upgrade Assistant → GitHub Copilot modernization (VS 2026)

VBScript → PowerShell (7)