DEVELOPERfusion
the UK community for developers

Printed from http://www.developerfusion.co.uk/show/30/

Rated
Viewed 258335 times

Actions
Save as Favourite
Share with a Friend
PDF Version
Report Problem

Contents

1. Introduction
2. What is VB?
3. What can I do with it?
4. Where can I get it?
5. Getting Started
6. Common Words
7. Programming!
8. Properties
9. User Input
10. Running your Project
11. The Code Window
12. Coding
13. Conclusion

Complete VB Beginners Tutorial - Introduction

Author James Crowley

Introduction

Until now, the tutorials at VB Web have focused on specific topics and points, and complete beginners could easily be overwhelmed. This tutorial is aimed directly at beginners. I will explain what VB is, where you can get it from, and get you started making your first VB program.

Just one warning - you are going to be blasted with tonnes of new words to learn. But don't worry... even if you don't remember them now, you will soon become very familiar with them, and use them throughout your Visual Basic programming career! To make this easier, important words are in bold. These words will be used throughout the site, and probably throughout every other Visual Basic sites.

Hopefully by the end of this tutorial, you will be in a better position to understand just what the tutorials at VB Web and other sites are going on about!

What is VB?

First off, what is VB? VB stands for Visual Basic, and is a High-Level Programming Language. A programming language basically allows you to create programs or applications, such as Microsoft Word. These can then be run on a number of operating systems, depending on which language you choose. Visual Basic is specifically for Windows 95 or above.

A High-Level programming language essentially means a language that is (relatively) easy to learn, and the code you write is similar to English words. In comparison, a Low-level language would mainly involves working with assembly language (ie what the PC's own language). It would involve far more work creating a program using a Low Level language, so nearly everyone uses High Level programming languages now. Visual Basic, Java, C++, Pascal, and nearly every other language are now high level languages.

Visual Basic has its roots in a language called BASIC, back in the days of Amstrads, and when Bill Gates was running Microsoft from his garage. (BASIC actually stands for Beginners All-Purpose Symbolic Instruction Code if you really want to know!). Since then, Visual Basic has rapidly evolved, and today, Visual Basic one of the most popular programming languages around. Fortunately for you, it is also one of the easiest, and is ideal for beginners.

What can I do with it?

Now you know what Visual Basic is, what can it do? Over the last few years, the selection of options available for this has grown rapidly, and now you can do almost everything C++ can to some extent, and in half the time. To get some idea, take a look at the list below:

- File Editors/Word Processing
- Database Programs
- Internet Applications
- Web Browser
- Custom Applications
- Data Manipulation
- File Management
- Games
- Spreadsheets
- and more!

What's more, if VB can't easily do what you want it to do, ActiveX Controls (or just Controls) are available to let you do this. These are essentially plug-ins that allow you to extend the functionality of Visual Basic, and can be bought from 3rd Party vendors. Some are also available free, from VB Web, and other sites.

As you can see, there are tonnes of options available to you. All you need is the knowledge to do it - which this site will help you gain.

Where can I get it?

If you've got to this stage, and decided Visual Basic is for you, then you will need to know where you can get it from, and how much it will cost. For this, there is a large number of options.

N.B. VB=Visual Basic, VS=Visual Studio

VB Edition	Cost	Description
Visual Basic for Applications	N/A	This version is standard with all versions of Microsoft Office 2000, and earlier versions to some extent. It's main use is for creating Macros to automate reptitive tasks, but can also be used for getting to grips with VB. Like the Learning Edition, a number of controls are unavailable, and you won't be able to distribute your application unless your users have Office too.
Learning/Standard Edition	~ £74	The basic functionality of Visual Basic is present, however important controls such as the Common Controls (that let you add Toolbars, Status Bars, Tab controls, List Views and TreeViews), the RichTextBox (which lets create word processing-type applications), Inet Control (which lets you access remote internet servers), and a number of other controls are not available. Occasionally, this edition is available on a cover CD. If this is the case, the option of creating an EXE file, which you can then distribute to users without Visual Basic is usually disabled.
VB (Professional)	£390	Visual Basic Professional includes all the features of the Standard edition, plus additional ActiveX controls, integrated Data Tools, and a DHTML editor. This edition includes everything nearly every developer will ever need!
VB (Enterprise)	£900	Visual Basic Enterprise Edition is aimed for those in a team setting, usually based around Server work... It includes all the features of the Professional edition, plus Back Office tools such as SQL Server, Microsoft Transaction Server, Internet Information Server, Visual SourceSafe, SNA Server, and more.
VS (Professional)	£790	This package includes Visual Basic Professional, but also includes the other Visual Studio products - Visual C++, Visual J++ and Visual InterDev.
VS (Enterprise)	£1136	This package includes Visual Basic Enterprise, but also includes the other Visual Studio Enterprise products - Visual C++, Visual J++, Visual InterDev, BackOffice etc.
VS (Professional) - Student Price	£76	This is exactly the same as Visual Studio Professional, but available at a massively reduced price for Students (under 16, or working towards a recognised qualification) and Teachers. Please note that when you purchase this copy, you will not receive the software. You then need to fill out a form, get it signed by your school or college, send it off, and then wait for up to 28 days... That is the downside of this edition!

As you can see, there are a wide number of choices. Obviously, if you are a student, I strongly recommend you go for the last option! Otherwise, it is really up to you. In my opinion, you will quickly find the missing controls in the Standard Edition a big blow, and so, if you can afford it, I would recommend the Professional Edition - this will cover most users needs completely. Upgrades from a previous version are available, usually at about half the price of the full version, however note that it is not possible to upgrade from a Standard Edition Version 6 to an Professional Edition Version 7 (as far as I am aware).

The next issue is where you can get Visual Basic from. Almost any software retailer should be able to provide you with a copy. I would suggest that you use whoever you are happy with - if you have a company you have already purchased software off, then you might as well use them. The prices I have given are for a guide only, and may vary... if you have the time, shop around! If you have no idea who to use, you can always try Jungle.com, or Simply.co.uk (for the UK only). You can always go to one of these sites to check the current price.

Getting Started

Now that you have bought your copy of Visual Basic, whatever its edition, we can take our first look at where you will be spending the next 10 years of your life! (on your PC anyway) I am going to assume that you have successfully installed your copy - this does not usually pose any problems.

Double-click on the Visual Basic icon, and after a little while you will be greeted with the New Project screen. From here, you can then create a new VB project, open an existing one, or re-load one you have recently opened. For the sake of this tutorial, select 'Standard EXE' and click Open. And there you go, your new VB application - or at least an application that displays an empty form. In VB, any dialog that a user is shown is known as a form; when you click save, and a save dialog appears, that is a form. When you type the text in a Word document, or open the options dialog, or anything else, that is a form too!

Before we start doing anything, we need to do something that anyone (with any sense!) who regularly uses Microsoft applications does - save your work! To do this, click the Save icon on the toolbar. You will then be prompted for two filenames. The first is a location for the Form. All good programmers use a 'naming convention'. This basically means that the programmer will use a standard format for all files, or controls, or anything else. Therefore, before naming the file, we want to stick to this convention. These can vary, but normally they follow these two rules:

- Use uppercase letters for the start of each word - spaces are not allowed for names when programming! Some programmers also use a _ (underscore). For example, if I wanted to name something 'my first form', it would be named My_First_Form.
- Add three letters before the name, so that you can instantly recognise what it is. When programming, you end up using an awful lot of names for different things. Naturally, because the things we are naming can be different sorts of things, we need to be able to tell what sort they are, without having to trawl through our code to find out. How do we do this? Using three meaningful letters (or sometimes one, if a programmer wants to have a really hard time coming up with different letters).

In this case we are saving a form, so three letters that are often used are frm. Consequently, as this is an example form, we put the letters 'frm' and 'Example' together, getting 'frmExample'. Enter this in the first Save As dialog, and click Save. The next dialog that appears is for saving the Project file. For projects, we normally use 'prj' as our three letters, so save this one as 'prjMy_First_App'. (In case you are wondering what App is, it is an abbreviation of Application).

Common Words

Now that we have saved our project, we can get on and do some programming. But first, I have created a list of common words that you will need to know. If you come across a word in this tutorial that you don't know, refer back to this.

Application	When you 'program', you are creating a program. This is also known as an Application. Word, Excel and Lotus Notes are all applications.
Code	This is what you type when you are programming!
IDE	The *IDE* or *Integrated Development Environment* is simply the main Visual Basic window. This is where you create and run your Visual Basic Applications.
Form	This is what programmers often call a dialog box, or as you probably know it, a window.
Form Window	In the Visual Basic window, you can see a form or window, within another window. This, is the *Form Window*, and is where you create your own form, and add your own controls and text to it. (See Figure 1)
Control	A *control* is something you place on a *form*. When you see a box to enter text, or have a drop-down box of choices, or a button to click, these are all controls.
Toolbox	This is the window shown down the left-hand side of the *IDE* , which lists all the *controls* that you can add to the *form*. (See Figure 2)
Project Window	This is the window on the right hand side of the Visual Basic window, which lists all the items in your project, and lets you choose a form to modify, or edit its code. (See Figure 3)
Property	When programming, every *control* has properties. These allow you change the appearance of a control, or the text it displays.
Properties Window	The properties window, not surprisingly allows you to modify these properties (See Figure 4).

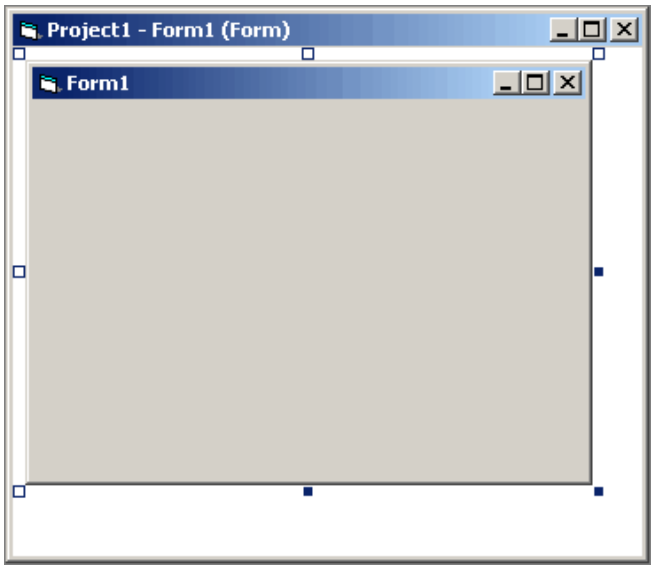


Figure 1 - The Form Window

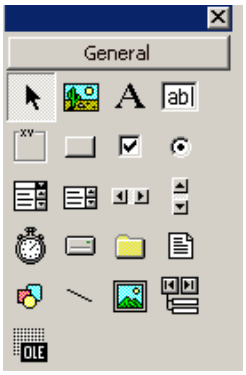


Figure 2 - The Toolbox

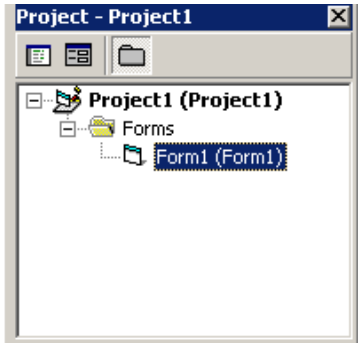


Figure 3 - The Project Window

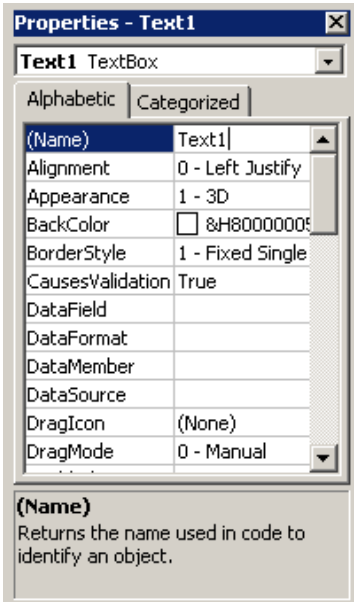


Figure 4 - The Properties Window

Programming!

We can now *finally* get on to actually doing something! To start off, we are going to create a simple program that asks you for your name, and then says hello. Although this might not sound much, it should get you going on the basics of VB.

To begin, ensure that you have the Form Window we described in the last section. If you can't see it, double click on the text Form1 on the right hand side of your screen (in the Project Window). Next, take a look at the left hand toolbar. This displays the controls that you can currently use in your project (ie what you can add to the form). Click the icon shown below:



This is known as the Label control. As its name suggests, it is used to 'label' other controls, and provides a means of adding text to your form. To add the control to your form, move the mouse to over the Form and hold down the left mouse button. Keep the button down, and move the mouse - you will see a rectangle form, and change shape as you move the mouse. This is showing you where the new control will go. When you are happy with the shape, let go, and you will see the text 'Label 1' appear.

Once you have placed a control on your form, you are free to move it around to wherever you want, by simply clicking and dragging. If you want to change the shape of the control, just like in almost any Office, or graphics program, simply select it, hold down the mouse over one of the blue boxes that appear, and drag.

Properties

Now that you have added the Label to your form, and selected it, you can change its properties using the properties window:

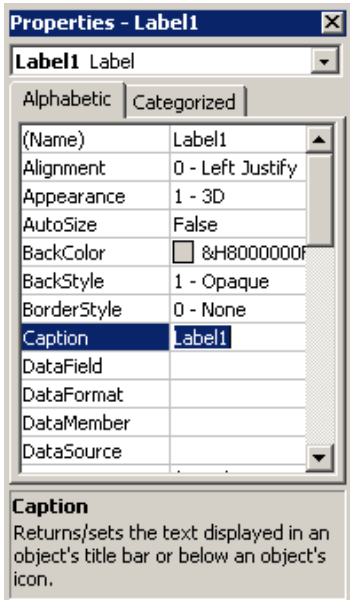


Figure 1 - The Properties Window

This window lists lots of different properties that you can use to change how a control looks and behaves. For example, the Appearance property sets whether a control should look 'Flat' or 3D. For the moment, you don't need to worry about these, but you can come back later and work out what they do. If at any stage you're not sure, simply press F1, and the VB help file will appear.

First, we need to give the control a meaningful name. This is so that when we start writing code, we can refer to this control, and remember which it is. The current name of 'Label1' is not particularly helpful. To change the name, we modify the text next to the (Name) box at the top of the properties list. In keeping with Visual Basic convention, the name of Label controls usually start with lbl, so lets call this control lblPrompt.

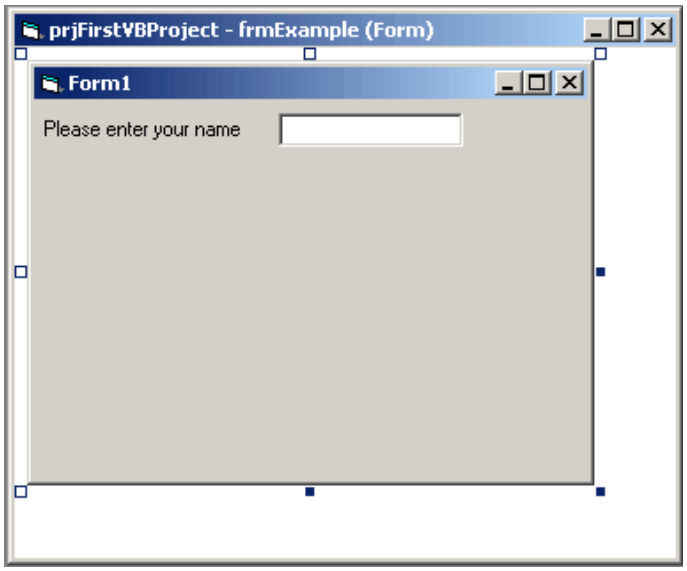
To change the text that appears for the label, you change the Caption property in the same way. We can change the caption of this label to 'Please enter your name'. Once you have done that, you might find that if you look back to the form, you can't see all of the text. If this is the case, select the control, and resize it so that you can.

User Input

As we have now written some text asking the user (the person using the program) for his or her name, we need to give them somewhere to type it. To do this we use the TextBox control. This is the 

icon in the Toolbox. Just like before, select this icon, and add the textbox to your form, next to the 'Please enter your name' text. Again, we can use the Properties window to change a few things. First, rename the textbox to txtName (as txt is the conventional prefix for textboxes). The other thing we need to change is its Text property; because the user needs to enter some text, the textbox needs to start out empty, rather than having 'Text1' displayed in it. To do this, scroll down to the Text property in the Property Window, and delete the text.

By now, your form should look something like this:



For the next stage you need to add one more control; the CommandButton control. This simply a button that the user can click, in this case to say that he/she has entered their name. To add the control, click the  icon, and add the control to your form, next to the text box. Name this control cmdOK, and set its Caption property to 'OK'.

You will notice that to change the text on a control, sometimes the property is called Caption, and sometimes it is called Text. The reason for this is that in a Text box, a user can change what appears in it (and therefore it is Text). On a button or label control however, the text is static. This means that it cannot be changed by the user.

Running your Project

See, its not that bad, is it?! Now for the moment of triumph. Save your work, and click the Run button on the top toolbar (this looks like a blue 'play' button). After a few seconds, you will see your form appear in front of you, just like any other window. You are now in what is known as run-time; this is when your VB application is actually running.

If you want, you can enter your name in the textbox, and click OK. However, nothing happens! That's because so far you haven't actually written any code, which will tell Visual Basic what to do when the button is clicked. This is what we will do next.

Click the Close button on your form, and you will be returned to what is known as design time, i.e. when you are designing your VB application.

The Code Window

Once you have been returned to your form, double click on the OK button on your form. A new window will appear, and some VB code will be automatically inserted. This is known as the Code Window. Let's take a look at the code VB has inserted for you:

```
Private Sub cmdOK_Click()

End Sub
```

Although it might seem strange at first, it is actually quite logical. First off, the words `Sub` and `End Sub` mean that this is a Procedure. Procedures are blocks of code, that can be executed (or run) by Visual Basic. The word `Private` defines the group of procedures that can call it, but you don't need to worry about that yet.

The most important part is `cmdOK_Click()`. If you remember, cmdOK is the name of the button that you added to your form. The text after the _ is known as the event. An event, just like in the outside world is something that happens. In Visual Basic, 'event' occur all the time. When the mouse is moved over the form, or when the user enters some text into a textbox, all of these trigger events.

In this case, the event is `Click`, and occurs, not surprisingly when cmdOK, the button, is clicked. Therefore, any code you enter into this procedure will be run when the OK button is clicked.

Now take a look at the two drop-down boxes at the top of the window:



As you can see, the text in these boxes match what is written. The box on the left lists all the controls that are on your form. If you click it, you will see the names of the three controls that you added to the Form. The right hand box lists all the events that can occur for this control. If you click it, you will see a long list of possible events for the button; the one that you are currently editing is displayed in bold (Click). If you select another item in the list, Visual Basic will automatically add a new procedure. Try this if you want, and then delete the text that VB added afterwards.

Coding

You are now going to write your first bit of Visual Basic code! In the cmdOK_Click() procedure, enter the following:

```
MsgBox "Hello " & txtName.Text & "!"
```

What this code does is displays what is known as a Message Box, which says 'Hello x!', where x is the text in the Text box. Message Boxes are used throughout Windows for displaying messages and asking questions in a standard window. Convieniently, it means that you don't have to create a form yourself to display a simple message.

To see this in action, save your work, and click the Run button on the toolbar again (you can also press the F5 key). Enter your name into the text box, and click OK. You should see something like this:



Click OK, and click the close button on the form to return to design time. Let's take another close look at the code. First of all, the `MsgBox` text is actually a Function or Procedure, that VB provides as standard. This MsgBox function displays a Message Box dialog. After that, there is `"Hello " & txtName.Text & "!"`. This is actually the text displayed in the Message Box, placed in quotes. However, we can't simply write

```
"Hello James!"
```

as we don't know what the user will enter into the textbox, and therefore what name to display. If you knew exactly what the message would be, you could do that. In this instance, we need to get the text from the Text box on our form. This is done using `txtName.Text`. `txtName` refers to the text box on the form (because we called it txtName). The period (`.`) means that we are 'accessing' a property from this control. `Text` refers to the property we want to read.

When we do this, Visual Basic returns the Text property; what is entered in the Text box. As the `txtName.Text` is actually VB code, and not simply the text we want displayed, it needs to be outside of the quotes. If we simply wrote

```
MsgBox "Hello txtName.Text!"
```

you would get



which is certainly not what we wanted. Instead, we close the quote after Hello (and a space, so that there is a gap

between Hello and the name). Next we use the ampersand (&) symbol to say that we want the text Hello, and then something else. We then place the code txtName.Text to retreive the text in the Text box, and then another ampersand and "!" to add an exclamation mark to the text, ending up with

```
"Hello " & txtName.Text & "!"
```

Conclusion

When you are happy with your project, you can compile it to an EXE file (which you can then run from explorer, or on another PC) by clicking File | Make ProjectName.exe...

Hopefully by now, your understanding of Visual Basic will have grown enormously. I realise that ideally, this tutorial could continue for another 100 pages, and hopefully I will get some time to extend it. Below are a few challenges to enhance your application. Hints for these tasks are at the bottom of the page.

- Display a message asking the user a question
- Disable the OK button when the user has not entered a name
- Provide a Cancel button
- Remove the Maximize and minimize buttons
- Display another form after the user has entered their name, and display their name there.
- 'Remember' the name that the user entered.

Below are also a few suggestions for expanding your knowledge of VB!

i) take a look at some of the other Commands described at VB Web, such as the MsgBox and InputBox pages... see if you can ask the user a question using the information provided there.
ii) you can then move on to some of the Conditional statements.... executing code when a certain condition is true. Take a look at the If...Then...Else statement and Select...Case, or even the Number functions page.
iii) explore the other tutorials at VB Web and on other sites to expand your knowledge
iv) use the Visual Basic help files! This is a massive resource, and also enourmously under-used.
v) Finally, play around, and have fun! This is the best way to learn how to program with VB. Write a few lines, run it, learn from your mistakes, and find out how to enhance it. If you are not sure how to do something, after looking at the VB help, join our discussion list, and send the email there. There are hundreds of programmers on the list, all of whom are willing to answer your questions. Before you know it, you will be answering others queries too.

Good luck with your programming career,

James Crowley

Hints for the tasks...

- Display a message asking the user a question
Hint: Look at the MsgBox function page
- Disable the OK button when the user has not entered a name
Hint: Look at the buttons `Enabled` property and the text boxes `Change` event
- Provide a Cancel button
Hint: Use `Unload Me` in the Click event
- Remove the Maximize and minimize buttons
Hint: Look at the forms `MaxButton` property
- Display another form after the user has entered their name, and display their name there.
Hint: Add another form to your project by clicking Project|Add Form. You can display another form by calling *`formname`*`.Show`, and hide one using *`formname`*`.Hide`
- 'Remember' the name that the user entered.
Hint: Use the registry!

James first started writing tutorials on Visual Basic in 1999 whilst starting this website (then known as VB Web). Since then, the site has grown rapidly, and James has written numerous tutorials, articles and reviews on VB, PHP, ASP and C#. In October 2003, James formed the company Developer Fusion Ltd, which owns this website, and also offers various development services. In his spare time, he's a 3rd year undergraduate studying Computer Science in the UK. He's also a Visual Basic MVP.

Related Content

Making a picture browser in VB 6

PDF Generated by ABCpdf 5.0