Contents

1 **Getting started** 7

Introducing Visual Basic 8
Installing Visual Studio 10
Exploring the IDE 12
Starting a new project 14
Adding a visual control 16
Adding functional code 18
Saving projects 20
Reopening projects 21
Summary 22

2 **Setting properties** 23

Form properties 24
Meeting the properties editor 25
Editing property values 26
Coding property values 28
Applying computed values 30
Applying user values 32
Prompting for input 34
Specifying dialog properties 36
Summary 38

3 **Using controls** 39

Tab order 40
Using Button 41
Using TextBox 42
Using ComboBox 43
Using Label 44
Using PictureBox 45
Using ListBox 46

Using CheckBox 48
Using RadioButton 49
Using WebBrowser 50
Using Timer 52
Summary 54

4 Learning the language 55

Elements of a program 56
Declaring variable types 58
Understanding variable scope 60
Working with variable arrays 62
Performing operations 64
Branching code 66
Looping code 68
Calling object methods 70
Creating a sub method 72
Sending parameters 73
Creating a function 74
Doing mathematics 75
Generating a random number 76
Summary 78

5 Building an application 79

The program plan 80
Assigning static properties 82
Designing the interface 84
Initializing dynamic properties 86
Adding runtime functionality 88
Testing the program 90
Deploying the application 92
Summary 94

6 Solving problems 95

Real-time error detection 96
Fixing compile errors 98
Debugging code 100
Setting debug breakpoints 102
Detecting runtime errors 104
Catching runtime errors 106
Getting help 108
Summary 110

7 Extending the interface 111

Color, Font & Image dialogs 112
Open, Save & Print dialogs 114
Creating application menus 116
Making menus work 118
Adding more forms 120
Controlling multiple forms 122
Playing sounds 124
Playing multimedia 126
Summary 128

8 Scripting with Visual Basic 129

Introducing VBA macros 130
Creating a Word macro 132
Creating an Excel macro 134
Running advanced macros 136
An introduction to VBScript 138
Enforcing declarations 139
Validating input 140
Merging text files 142
Getting registry data 144
Summary 146

9 Harnessing data 147

Reading text files 148
Streaming lines of text 150
Reading Excel spreadsheets 152
Reading XML files 154
Creating an XML dataset 156
Reading RSS feeds 158
Addressing XML attributes 160
Summary 162

10 Employing databases 163

An introduction to databases 164
Designing a database 166
Creating a database 168
Adding database tables 170
Defining table columns 172
Making table relationships 174
Entering table data 176
Creating a database dataset 178
Adding form data controls 180
Binding meaningful data 182
Building custom SQL queries 184
Summary 186

Index 187

1 Getting started

Welcome to the exciting world of Visual Basic programming. This chapter introduces the Visual Studio Integrated Development Environment (IDE) and shows you how to create a real Windows application.

8 Introducing Visual Basic
10 Installing Visual Studio
12 Exploring the IDE
14 Starting a new project
16 Adding a visual control
18 Adding functional code
20 Saving projects
21 Reopening projects
22 Summary

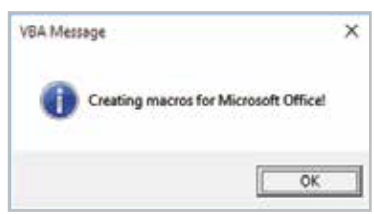
Introducing Visual Basic

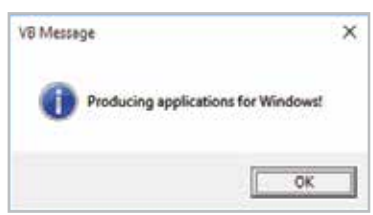
In choosing to start programming with Visual Basic you have made an excellent choice – the Visual Basic programming language offers the easiest way to write programs for Windows. This means you can easily create your own programs to give maximum control over your computer, and automate your work to be more productive. Also, programming with Visual Basic is fun!

Like other programming languages, Visual Basic comprises a number of significant "keywords" and a set of syntax rules. Beginners often find its syntax simpler than other programming languages, making Visual Basic a popular first choice to learn.

Although writing programs can be complex, Visual Basic makes it easy to get started. You can choose how far to go. Another advantage of Visual Basic is that it works with Microsoft Office applications, and with the Windows Script Host within the Windows operating system – so the possibilities are immense.

- **Visual Basic (VB)** – quite simply the best programming language for the novice or hobbyist to begin creating their own standalone Windows applications, fast.
- **Visual Basic for Applications (VBA)** – an implementation of Visual Basic that is built into all Microsoft Office applications. It runs within a host, rather than as a standalone application.
- **Visual Basic Script (VBScript)** – a derivative of Visual Basic that can be used for Windows scripting.





You can download the projects from this book at **www.ineasysteps.com/resource-centre/downloads/**

The **New** icon pictured above indicates a new or enhanced feature introduced with the latest version of Visual Basic and Visual Studio.

The evolution of Visual Basic

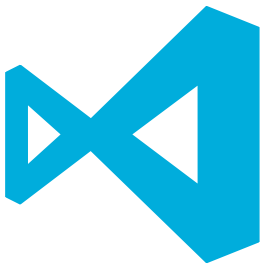
- Visual Basic 1.0 released in May 1991 at the Comdex trade show in Atlanta, Georgia, USA.
- Visual Basic 2.0 released in November 1992 – introducing an easier and faster programming environment.
- Visual Basic 3.0 released in the summer of 1993 – introducing the Microsoft Jet Database Engine for database programs.
- Visual Basic 4.0 released in August 1995 – introducing support for controls based on the Component Object Model (COM).
- Visual Basic 5.0 released in February 1997 – introducing the ability to create custom user controls.
- Visual Basic 6.0 released in the summer of 1998 – introducing the ability to create web-based programs. This hugely popular edition is the final version based on COM and is often referred to today as "Classic Visual Basic".
- Visual Basic 7.0 (also known as Visual Basic .NET) released in 2002 – introducing a very different object-oriented language based upon the Microsoft .NET framework. This controversial edition broke backward-compatibility with previous versions, causing a rift in the developer community. Subsequent editions added features for subsequent .NET framework releases.
- Visual Basic 8.0 (a.k.a. Visual Basic 2005).
- Visual Basic 9.0 (a.k.a. Visual Basic 2008).
- Visual Basic 10.0 (a.k.a. Visual Basic 2010).
- Visual Basic 11.0 (a.k.a. Visual Basic 2012).
- Visual Basic 12.0 (a.k.a. Visual Basic 2013).
  (version numbering of Visual Basic skipped 13 to keep in line with the version numbering of Visual Studio itself).
- Visual Basic 14.0 (a.k.a. Visual Basic 2015).

All examples in this book have been created for Visual Basic 14.0, although many of the core language features are common to previous versions of the Visual Basic programming language.

Visual Basic derives from an earlier, simple language called BASIC, an acronym –
Beginners
All-purpose
Symbolic
Instruction
Code.
The "Visual" part was added later as many tasks can now be accomplished visually, without actually writing any code.

Installing Visual Studio

In order to create Windows applications with the Visual Basic programming language, you will first need to install a Visual Studio Integrated Development Environment (IDE).



Microsoft Visual Studio is the professional development tool that provides a fully Integrated Development Environment for Visual C++, Visual C#, Visual J#, and Visual Basic. Within its IDE, code can be written in C++, C#, J# or the Visual Basic programming language to create Windows applications.

Visual Studio Community edition is a streamlined version of Visual Studio specially created for those people learning programming. It has a simplified user interface and omits advanced features of the professional edition to avoid confusion. Within its IDE, code can be written in the Visual Basic programming language to create Windows applications.

Both Visual Studio and Visual Studio Community provide a Visual Basic IDE for Visual Basic programming. Unlike the fully-featured Visual Studio product, the Visual Studio Community edition is completely free and can be installed on any system meeting the following minimum requirements:

Component	Requirement
Operating system	Windows XP Windows Vista Windows 7 Windows 8/8.1 Windows 10
CPU (processor)	1.6GHz or faster
RAM (memory)	1024MB (1GB) minimum
HDD (hard drive)	4GB available space, 5400RPM speed
Video Card	DirectX 9-capable, and a screen resolution of 1024 x 768 or higher

The Visual Studio Community edition is used throughout this book to demonstrate programming with the Visual Basic language, but the examples can also be recreated in Visual Studio. Follow the steps opposite to install Visual Studio Community edition.

1. Open your web browser and navigate to the Visual Studio Community download page – at the time of writing this can be found at **visual-studio.com/en-us/products/visual-studio-community-vs.aspx**

2. Click the "Download Community 2015" button to download a **vs_community.exe** installer file to your computer

3. Click on the **vs_community.exe** file to run the installer

4. Accept the suggested installation location, then click **Next**

5. Choose the **Custom** type of installation, then click **Next**

6. Check only the **Microsoft SQL Server Data Tools** feature to be added to the typical setup, then click **Next**, **Install** to begin the download and installation process

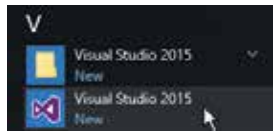
Choosing a different destination folder may require other paths to be adjusted later – it's simpler to just accept the suggested default.

The Visual Studio 2015 setup process allows you to install just the components you need.

You can run the installer again at a later date to modify Visual Studio by adding or removing features. The **Microsoft SQL Server Data Tools** are required by the database example in the final chapter of this book.

Exploring the IDE

1. Go to the Start menu, then select the Visual Studio 2015 menu item added there by the installer



2. Sign in with your Microsoft Account, or simply click the **Not now, maybe later** link to continue

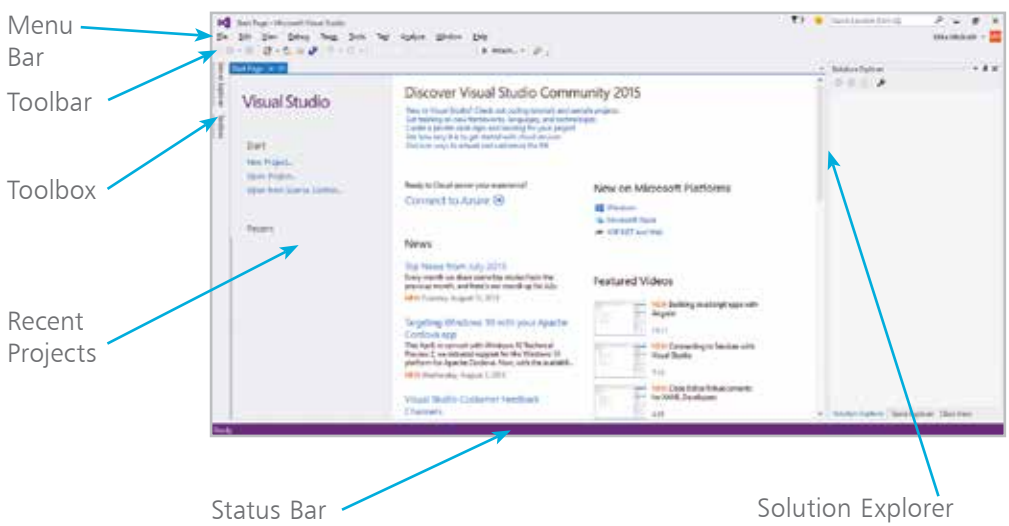
3. Choose your preferred color theme, such as **Light**, then click the **Start Visual Studio** button

The first time Visual Studio starts, it takes a few minutes as it performs some configuration routines.

You can change the color theme later – choose the **Tools**, **Options** menu then **Environment**, **General**.

The Visual Studio Integrated Development Environment (IDE) appears, from which you have instant access to everything needed to produce complete Windows applications. From here you can create exciting visual interfaces, enter code, compile and execute applications, debug errors, and much more.

The Visual Studio IDE initially includes a default Start Page, along with the standard IDE components, and looks like this:



Start Page elements

The default start page provides these useful features:

- **Start** – provides links you can click to begin a new project or reopen an existing project.
- **Recent** – conveniently lists recently opened projects so you can quickly select one to reopen.
- **News** – feeds the latest online news direct from the Microsoft Developer Network (MSDN).

You can return to the Start Page at any time by selecting **View**, **Start Page** on the menu bar.

Visual Studio IDE components

The Visual Studio IDE initially provides these standard features:

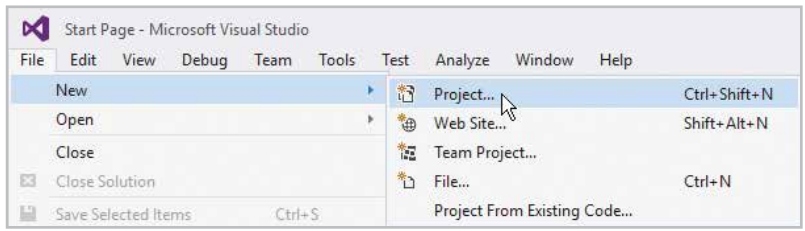
- **Menu Bar** – where you can select actions to perform on all your project files and to access Help. When a project is open, extra menus of Project and Build are shown in addition to the default menu selection of File, Edit, View, Debug, Team, Tools, Test, Analyze, Window, and Help.
- **Toolbar** – where you can perform the most popular menu actions with just a single click on its associated shortcut icon.
- **Toolbox** – where you can select visual elements to add to a project. Place the cursor over the Toolbox to see its contents. When a project is open, "controls" such as Button, Label, CheckBox, RadioButton, and TextBox are shown here.
- **Solution Explorer** – where you can see at a glance all the files and resource components contained within an open project.
- **Status Bar** – where you can read the state of the current activity being undertaken. When building an application, a "Build started" message is displayed here, changing to a "Build succeeded" or "Build failed" message upon completion.

The menus are once again in title-case, rather than the ALL CAPS style of the previous version.

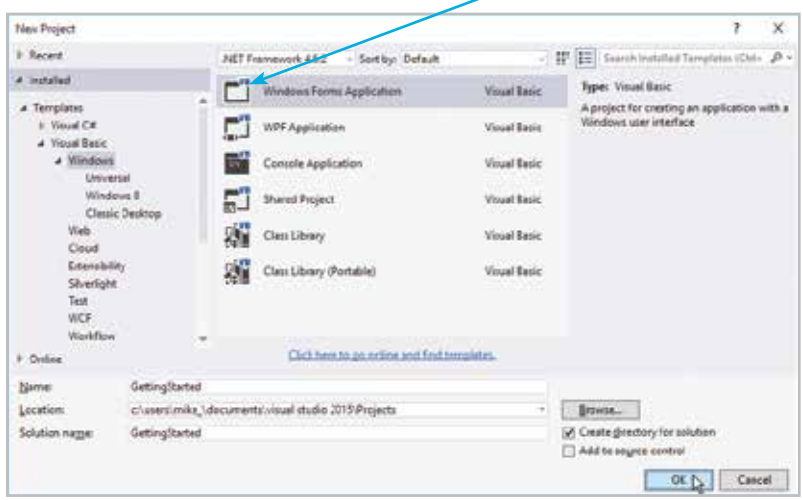
Online elements of the Start Page require a live internet connection – if the hyperlinks do not appear to work, verify your internet connection.

Starting a new project

1 On the menu bar click **File**, **New**, **Project**, or press the **Ctrl + Shift + N** keys, to open the New Project dialog box



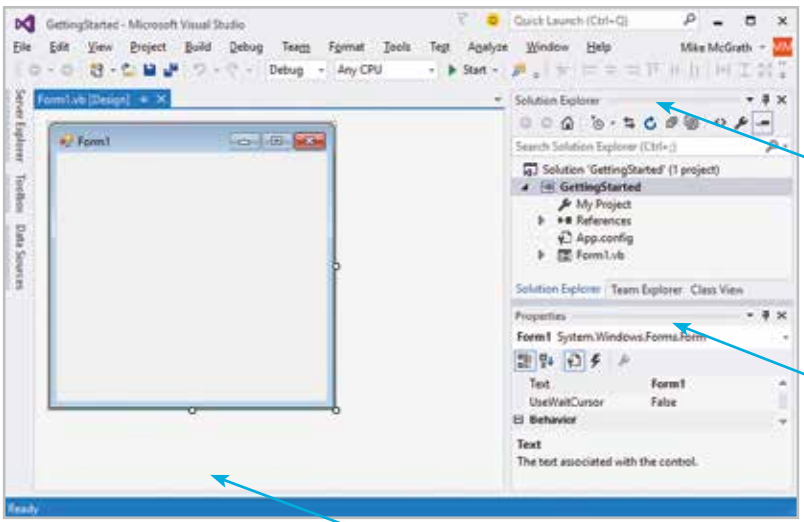
2 In the New Project dialog box, select the **Windows Forms Application** template icon



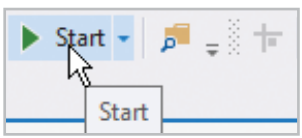
3 Enter a project name of your choice in the **Name** field, then click on the **OK** button to create the new project – in this case the project name will be "GettingStarted"

Visual Studio now creates your new project and loads it into the IDE. A new tabbed **Form Designer** window appears (in place of the Start Page tabbed window) displaying a default empty Form. You can select **View**, and then the **Solution Explorer** menu, to open a Solution Explorer window that reveals all files in your project. Additionally, you can select **View**, **Properties** menu to open a **Properties** window to reveal all properties of your Form.

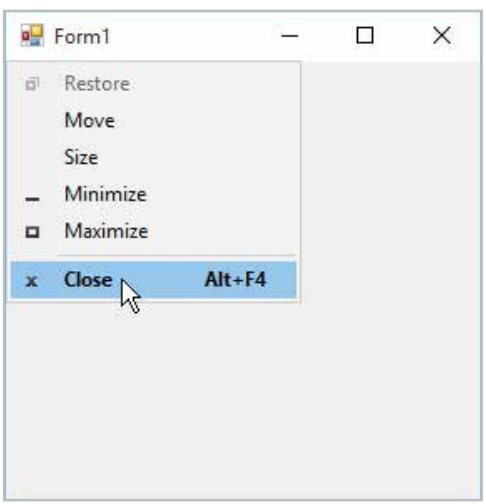
The **New Project** dialog automatically selects the **Windows Forms Application** template by default as it is the most often used template.



The **Form Designer** is where you can create visual interfaces for your applications, and the **Properties** window contains details of the item that is currently selected in the Form Designer window.



The Visual Studio IDE has now gathered all the resources needed to build a default Windows application – click the **Start** button on the toolbar to launch this application.



The application creates a basic window – you can move it, minimize it, maximize it, resize it, and quit the application by closing it. It may not do much but you have already created a real Windows program!

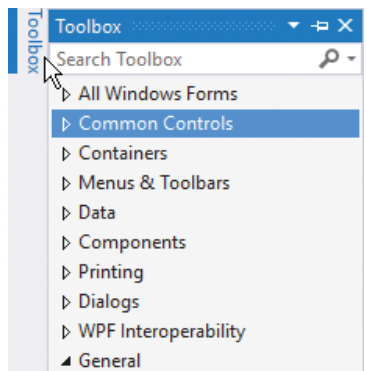
You can alternatively run applications using the **Debug**, **Start Debugging** menu options.

Adding a visual control

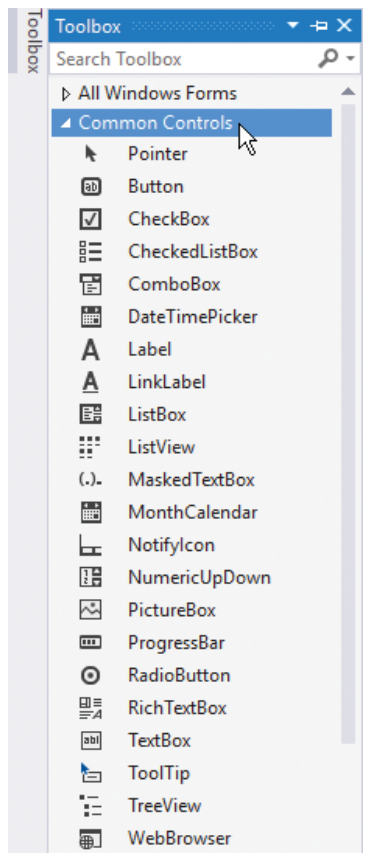
The **Toolbox** in the Visual Studio IDE contains a wide range of visual controls which are the building blocks of your applications. Using the project created on the previous page, follow these steps to start using the Toolbox now:

The **Toolbox** will automatically hide when you click on another part of the IDE, but it can be fixed in place so it will never hide, using the pin button on the Toolbox bar.

1 Place the cursor over the vertical **Toolbox** tab at the left edge of the IDE window, or click **View**, **Toolbox** on the menu bar, to display the Toolbox contents. The visual controls are contained under various category headings beside an expansion arrow

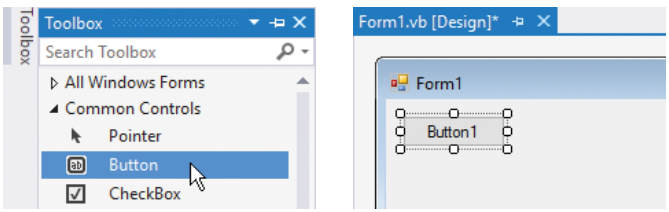


2 Click on the expansion arrow beside the **Common Controls** category heading to expand the list of most commonly used visual controls. Usefully, each control name appears beside an icon depicting that control as a reminder. You can click on the category heading again to collapse the list, then expand the other categories to explore the range of controls available to build your application interfaces

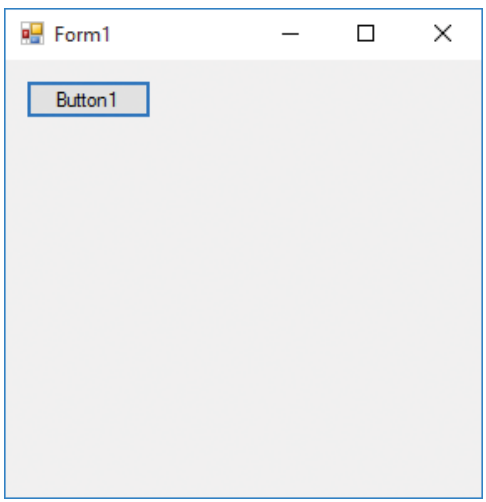


Any pinned Window in the IDE can be dragged from its usual location to any position you prefer. Drag it back to the initial location to re-dock it.

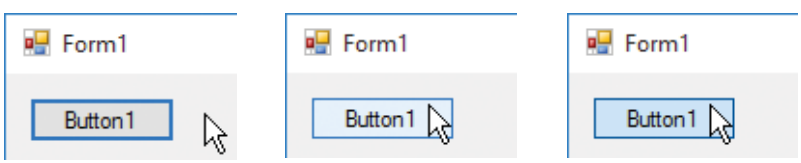
3 Click and drag the **Button** item from the Common Controls category in the Toolbox onto the Form in the Designer window, or double-click the Button item, to add a Button control to the Form



The Button control appears in the Form Designer surrounded by "handles" which can be dragged to resize the button's width and height. Click the **Start** button to run the application and try out your button.



The Button control behaves in a familiar Windows application manner with "states" that visually react to the cursor.



Default State Hover State Down State

A **Button** is one of the most useful interface controls – your program determines what happens when the user clicks it.

This **Button** control performs no function when it's clicked – until you add some code.

Adding functional code

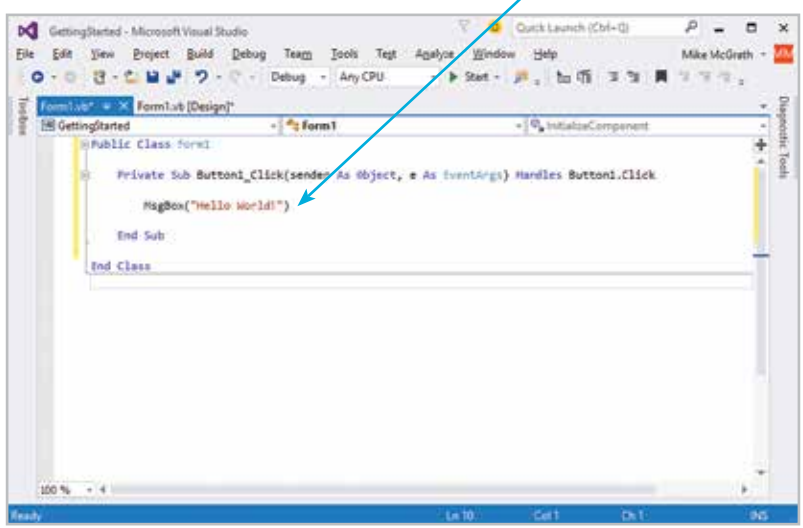
The Visual Studio IDE automatically generates code, in the background, to incorporate the visual controls you add to your program interface. Additional code can be added manually, using the IDE's integral **Code Editor**, to determine how your program should respond to interface events – such as when the user clicks a button.

Using the project created on the previous page, follow these steps to start using the Visual Studio Code Editor:

Switch easily between the **Code Editor** and **Form Designer** (or Start Page) by clicking on the appropriate window tab.

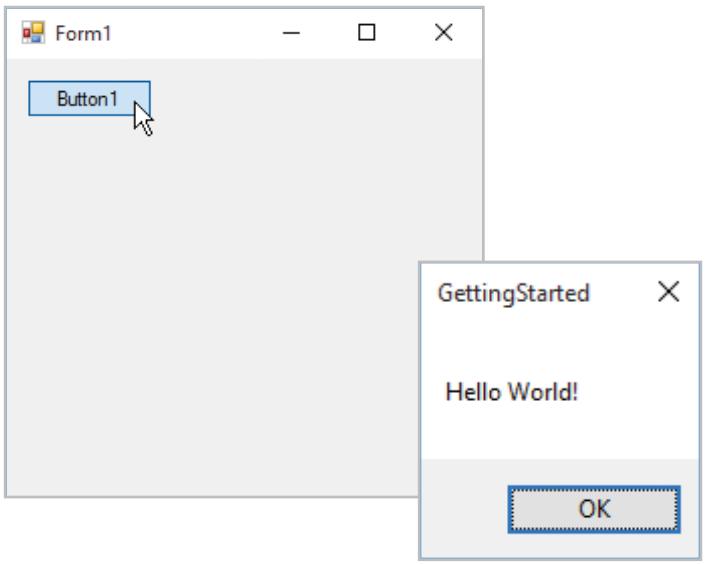
1 Double-click on the **Button** control you have added to the default Form in the Designer window. A new tabbed text window opens in the IDE – this is the **Code Editor** window

2 The cursor is automatically placed at precisely the right point in the code at which to add an instruction, to determine what the program should do when this button is clicked. Type the instruction **MsgBox("Hello World!")** so the Code Editor looks like this:



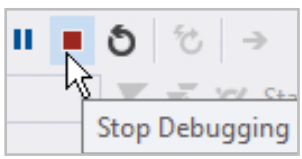
The **Solution Explorer** and **Properties** windows are closed here for clarity. You can reopen them at any time from the **View** menu.

3 Click the **Start** button to run the application and test the code you have just written, to handle the event that occurs when the button is clicked



Use the **View** menu on the menu bar to open the **Code Editor**, **Form Designer**, or any other window you require at any time.

4 Click the **OK** button to close the dialog box, then click the **X** button on the Form window, or click the **Stop Debugging** button on the menu bar, to stop the program



Each time the button in this application is pressed, the program reads the line of code you added manually to produce a dialog box containing the specified message. The action of pressing the button creates a **Click** event that refers to the associated "event-handler" section of code you added to see how to respond.

In fact, most Windows software works by responding to events in this way. For instance, when you press a key in a word processor a character appears in the document – the **KeyPress** event calls upon its event-handler code to update the text in response.

The process of providing intelligent responses to events in your programs is the very cornerstone of creating Windows applications with Visual Basic.

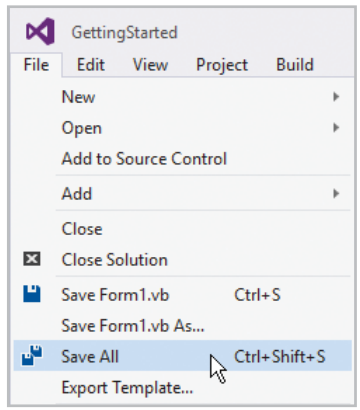
Saving projects

Even the simplest Visual Basic project comprises multiple files which must each be saved on your system to store the project.

Follow these steps to save the current New Project to disk:

You can click **File**, **Close Solution** on the menu bar to close an open project – a dialog will prompt you to save any changes before closing.

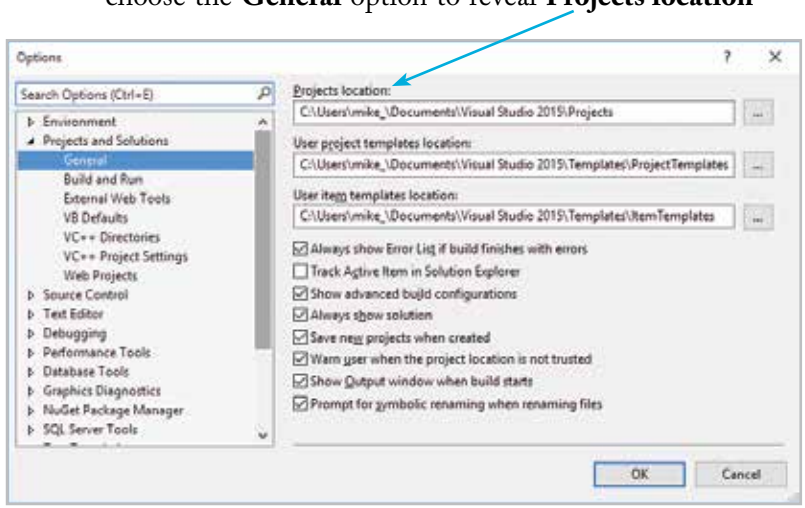
1. Click the **Save All** button on the toolbar, or click **File**, **Save All** on the menu bar, or press **Ctrl + Shift + S**



2. Your project is now saved at its default save location

3. To discover or change the save location click **Tools** on the menu bar, then select the **Options** item

4. Expand **Projects and Solutions** in the left pane, then choose the **General** option to reveal **Projects location**

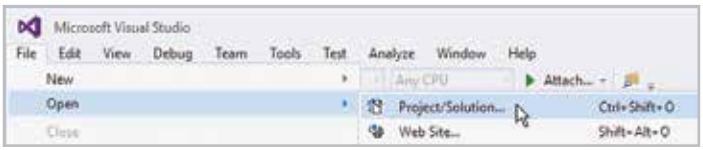


Find the **Debug** folder in your saved project directory containing the application's executable (**.exe**) file – you can double-click this to run your program like other Windows applications.

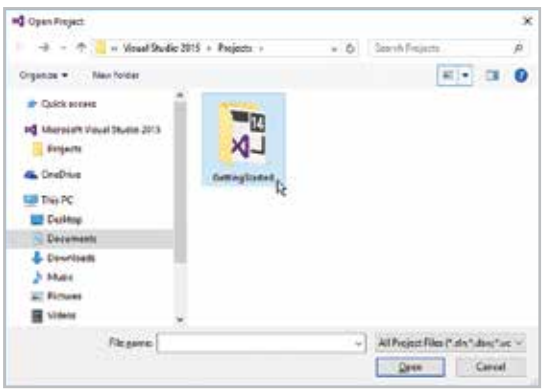
Reopening projects

Use these steps to reopen a saved Visual Basic project:

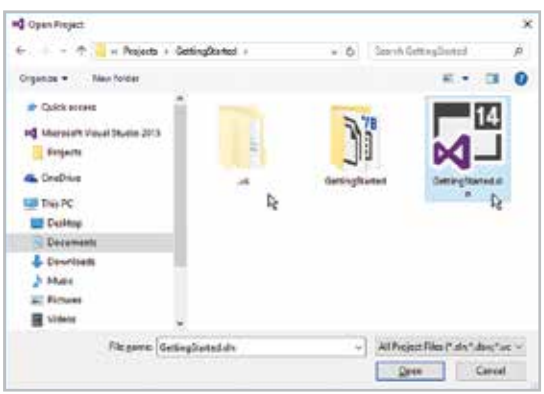
1. Click **File**, **Open**, **Project/Solution** on the menu bar to launch the **Open Project** dialog



2. In the **Open Project** dialog, select the folder containing the project you wish to reopen, and **Open** that folder



3. Now, select the Visual Basic Solution file with the extension **.sln** to reopen the project, or alternatively, open the folder bearing the project name, then select the Visual Basic Project File with the extension **.vbproj**



Only have one project open at any given time to avoid confusion – unless several are needed to be opened together for advanced programming.

If you don't see the Form Designer window after you have reopened a project, click the **Form1.vb** icon in **Solution Explorer** to make it appear.

Summary

- The **Windows Application Template** in the New Project dialog is used to begin a new Windows application project.
- A unique name should be entered into the **New Project** dialog whenever you create a new Visual Basic project.
- The **Form Designer** window of the Visual Studio IDE is where you create the visual interface for your program.
- Visual controls are added from the **Toolbox** to create the interface layout you want for your program.
- A control can be dragged from the **Toolbox** and dropped onto the Form, or added to the Form with a double-click.
- The **Visual Studio IDE** automatically generates code in the background as you develop your program visually.
- The **Code Editor** window of the Visual Studio IDE is where you manually add extra code to your program.
- Double-click on any control in the **Form Designer** to open the Code Editor window at that control's event-handler code.
- The **Start** button on the Visual Studio toolbar can be used to run the current project application.
- Pressing a Button control in a running application creates a **Click** event within the program.
- Code added to a button's **Click** event-handler determines how your program will respond whenever its Click event occurs.
- Providing intelligent responses to events in your programs is the cornerstone of programming with Visual Basic.
- Remember to explicitly save your working project using the **Save All** button on the toolbar, to avoid accidental loss.
- Select the solution file with the **.sln** extension in your chosen saved project directory to reopen that project.