CHAPTER 1

INFOCUS

UNDERSTANDING WORD VBA

Visual Basic (also known as ***VB***), is a programming language that was developed many years ago by Microsoft to help people program their computers. Microsoft Office has a number of applications that each have a derivative version of ***VB*** which has become known as ***Visual Basic for Applications***, or ***VBA*** for short.

VBA is an object-oriented programming language that performs operations by manipulating ***objects***, which in turn have ***methods***, ***properties*** and ***events***. Each application in the Microsoft Office suite, including Microsoft Word, has its own specific set of objects.

In this session you will:

- ✓ gain an understanding of programming in ***Microsoft Word***
- ✓ gain an understanding of ***VBA*** terminology
- ✓ learn how to display the ***Developer*** tab
- ✓ gain an understanding of the ***VBA Editor*** screen and its features
- ✓ learn how to open, close and switch to the ***VBA Editor***
- ✓ gain an understanding of objects in ***VBA***
- ✓ gain an understanding of the ***Microsoft Word*** object hierarchy
- ✓ learn how to view the ***Word*** object model
- ✓ learn how to use the ***Immediate*** window
- ✓ learn how to work with object collections
- ✓ learn how to set property values
- ✓ learn how to use the ***Object Browser***
- ✓ learn how to program with the ***Object Browser***
- ✓ gain an understanding of how to get help for ***VBA***
- ✓ gain an understanding of the codes used in this chapter.

PROGRAMMING IN MICROSOFT WORD

Microsoft Word has had some form of programming language available to it ever since it was first released. Its current programming language, ***VBA***, is a very powerful and versatile tool. It is not difficult to learn and use, but it can sometimes seem complicated until you fully grasp its background and what it is designed to do.

The Difference Between Macros And VBA

When spreadsheets were introduced to computing, you could program them to perform repetitive procedures. These programs were simply lists of commands from the standard menu that you wanted to run in a particular sequence. This early form of a program was known as a ***macro***.

The term ***macro*** is still used today – indeed you'll see it on the ***Developer*** tab of the ribbon in Microsoft Word. However, these macros are now based on a programming language (VBA) rather than a list of simple commands.

Why Use VBA?

The main reason for including a programming language in an application like Microsoft Word is to extend its capabilities beyond those that you can find on the ribbon. There are several reasons people decide to program in VBA:

- **Repetition** – The primary use of VBA is to automate operations, thus making repetitive tasks more efficient. For example, if you need to add the same header to several documents every day, or ensure the same font is used for the heading of each document, you can write code in VBA to automate these tasks and get them done quickly and accurately.
- **Limitation and Interaction** – With carefully written VBA coding you can guide inexperienced users through a workbook, giving them access to specific areas and precluding them from others, providing them with specialised dialog boxes (known in VBA as ***forms***) and prompting them for key information.
- **Cross-application Communication** – You can use VBA to write programs that communicate across applications. For example, using VBA you can take Word data and charts and put it into a PowerPoint presentation, or insert sales figures from Excel into a Word document.

Time Versus Effort

VBA isn't always the best tool to use to complete a task. If you only need to perform an operation once (even a complex one), it would be unnecessary to write a program for the operation as it would be an inefficient use of time. Also, there are many actions that can quickly and easily be performed using a command or commands that currently exist on the ribbon. Therefore, before writing a VBA program, check to make sure that the task can't already be done in other ways. This is one of the reasons why it is important to have a good grasp of Microsoft Word before you attempt VBA.

VBA TERMINOLOGY

Visual Basic for Applications is a derivative of the programming language ***Visual Basic***. Each Office application has its own particular kind of VBA depending on the objects and operation of the application. For example, Microsoft Excel uses worksheets while Microsoft Word works with documents. The following notes explain some of the key concepts in VBA programming.

Object Oriented And Procedure Driven

Visual Basic for Applications and ***Visual Basic*** are both ***object-oriented*** programming languages because they work with ***objects***. Most of these objects appear on the screen, hence the term ‘visual’.

They are also ***procedure-driven*** languages using commands and structures from the ***BASIC*** programming language to bind object statements into workable applications.

Objects, Properties, Methods And Events

In VBA an ***object*** is anything in an application that you can see and manipulate in some way.

For example, you can manipulate a document by adding pictures, deleting text, and so on. A document is therefore an example of an object. Just to add a little complexity, tables are also objects, as are columns and paragraphs. These are ***child*** objects of the ***parent*** object – the document. This way of organising objects into a hierarchy is known as an ***object model***.

Objects can be manipulated in one of three ways. You can:

- change the way an object looks or behaves by changing its ***properties***
- make an object perform a task by using a ***method*** that is associated with the object
- run a procedure whenever a particular ***event*** happens to an object.

Objects therefore have ***properties***, ***methods*** and ***events***.

The Car Analogy

Let’s look at a simple analogy to get a better understanding of *objects*, *properties*, *methods* and *events*.

Consider a car: it is an ***object*** because you can see it and manipulate it. It has both ***properties*** and ***methods*** and they are as follows:

- the ***properties*** are its physical characteristics such as its make, model, colour, and so on
- the ***methods*** define what you can do with the car such as reversing, accelerating, turning, stopping, and so on

Events are the actions that happen to the car that generate an automatic response from the car. For example, if you remove the keys from the ignition while the car’s headlights are on (event), most cars will sound a warning alarm or turn off the lights (response).

The Active Object

In VBA, ***active*** describes the object item that you’re currently working on.

For example, the document that you’re currently using in Word is the ***active*** document. The object that is currently active is said to have the ***focus***.

This is an important concept to understand because most of your VBA programming will be performing an action on a particular object. If you don’t identify that object correctly, you may find that Word shifts focus behind the scenes to a different object and your program will fail.

DISPLAYING THE DEVELOPER TAB

Before you start working with VBA, you will need to ensure that the ***Developer*** tab is displayed on the ribbon in Excel. This tab allows you to access the commands to display the ***VBA Editor*** and run macros, among others. The ***Developer*** tab is not displayed by default; you must enable it using the ***Word Options*** dialog box.

Try This Yourself:

Open File

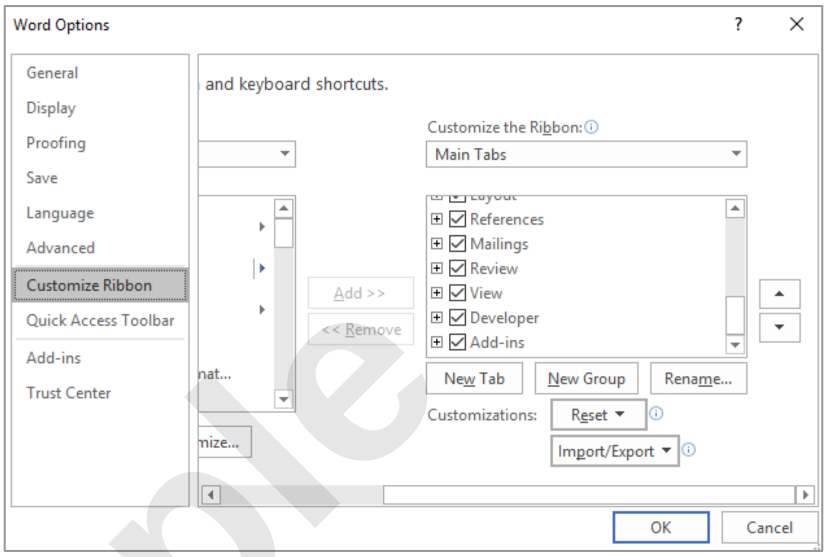
Before starting this exercise you MUST ensure Word has started and a new blank document is displayed…

1. Click on the ***File*** tab to display the ***Backstage***
2. Click on ***Options*** to display the ***Word Options*** dialog box
3. Click on ***Customise Ribbon*** in the left pane to display the options for customising the ribbon
4. In the right pane, click on ***Developer*** so it appears ticked
5. Click on **[OK]** to save the settings and return to Word

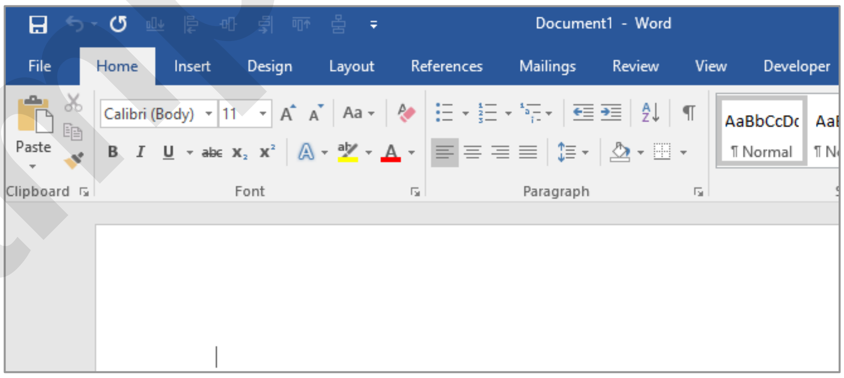
The Developer tab is now displayed on the ribbon…

6. Click on the ***Developer*** tab to see the available commands

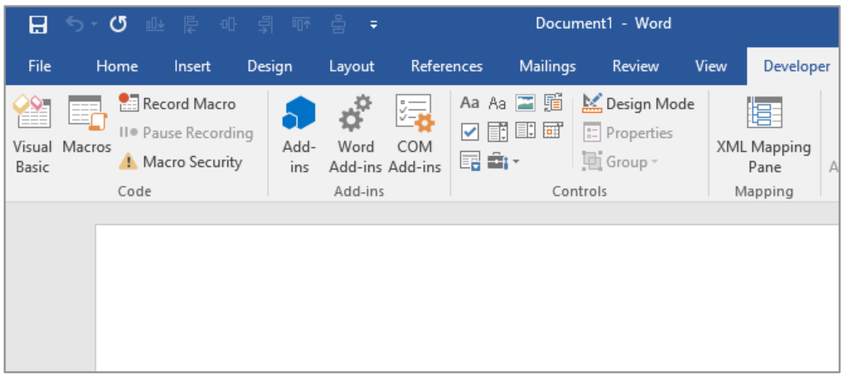
4



5



6



For Your Reference…

To ***display*** the ***Developer tab*** on the ***ribbon***:

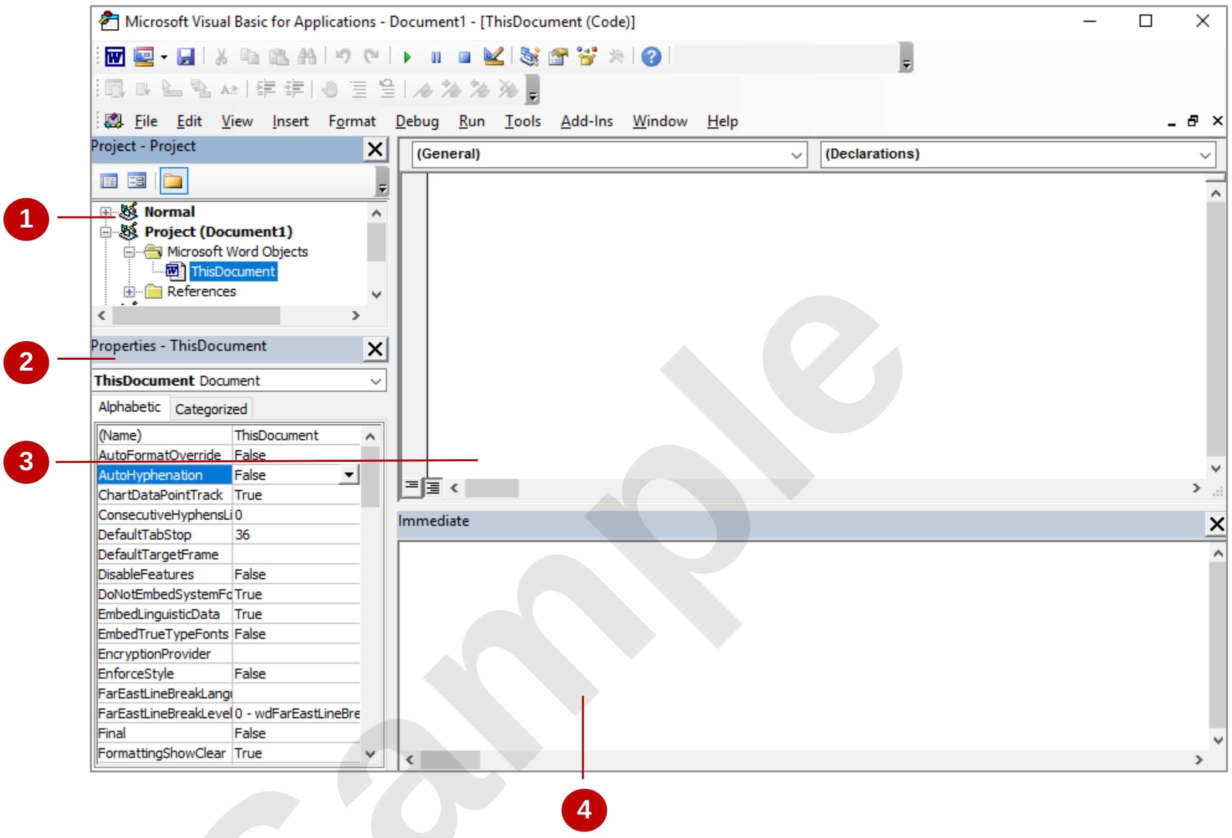
1. Click on the ***File*** tab
2. Click on ***Options***
3. Click on ***Customise Ribbon***
4. Click on ***Developer*** in the right pane so it appears ticked, then click on **[OK]**

Handy to Know…

- Once the ***Developer*** tab is displayed, it will remain on the ribbon until you choose to remove it again. You can do this by displaying the ***Customise Ribbon*** screen of the ***Word Options*** dialog box and clicking on ***Developer*** so it appears unticked.

THE VBA EDITOR SCREEN

The ***VBA Editor*** is a separate application designed to help you create and edit VBA procedures. The ***VBA Editor*** comprises four main components: the ***Project Explorer***, the ***Properties Window***, the ***work area*** in which you create VBA code in ***code modules*** and build ***UserForms***, and an area for special ***panes*** that can be displayed.


Microsoft Visual Basic for Applications - Document1 - [ThisDocument (Code)]
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Project - Project
Normal
Project (Document1)
Microsoft Word Objects
ThisDocument
References
(General)
(Declarations)
Immediate
1
2
3
4

Properties - ThisDocument

ThisDocument Document

Alphabetic | Categorized

(Name)	ThisDocument
AutoFormatOverride	False
AutoHyphenation	False
ChartDataPointTrack	True
ConsecutiveHyphensLi	0
DefaultTabStop	36
DefaultTargetFrame	
DisableFeatures	False
DoNotEmbedSystemFo	True
EmbedLinguisticData	True
EmbedTrueTypeFonts	False
EncryptionProvider	
EnforceStyle	False
FarEastLineBreakLang	
FarEastLineBreakLevel	0 - wdFarEastLineBre
Final	False
FormattingShowClear	True

1	***Project Explorer***	The ***Project Explorer*** displays the contents of the current VBA project which, in the example above, is called ***Project***. (In general terms, a ***project*** is an Office file and all of its associated VBA items, including its macros and user forms). Contents of a project can include ***forms***, ***objects*** such as documents, and ***modules***. Using the tools in the ***Project Explorer's*** toolbar you can display the code window (***View Code***) so you can write and edit code associated with the selected item; display the object window (***View Object***) for the selected item, an existing document or user form; and hide or show the object folders while still showing the individual items contained within them (***Toggle Folders***).
2	***Properties Window***	The ***Properties Window*** lists the properties for the object that is currently selected in the ***Project Explorer*** or in the work area (in the case of a ***form***) and displays its current settings.
3	***Work Area***	The ***work area*** is the larger area to the right of the ***Project Explorer*** and ***Properties Window***. It is in this area where the text editor is used to create VBA code in a ***module*** (a window that is designed to hold programming code) and where user forms are built. (A ***UserForm*** is a window or dialog box that makes up part of an application's user interface.)
4	***Pane Area***	The ***Pane Area*** is used to display additional mini-windows or panes that can assist in the development and debugging of programs. The example above shows the ***Immediate*** pane.

OPENING AND CLOSING THE EDITOR

The ***Visual Basic Editor*** is a separate application that can be accessed from any Microsoft Office application. When you are programming you can keep the Editor open and switch between the Editor and the document as required. Alternatively, you can close the Editor and return to the document, then re-open the Editor when you need it.

Try This Yourself:

Open File

Before starting this exercise you MUST open the file Understanding Word VBA_1.docm…

1 If the macro security warning appears while opening the file, click on **[Enable Macros]**

The document should now be open…

2 Click on the ***Developer*** tab, then click on ***Visual Basic*** in the ***Code*** group to open the ***VBA Editor***

The Editor will open in a separate window…

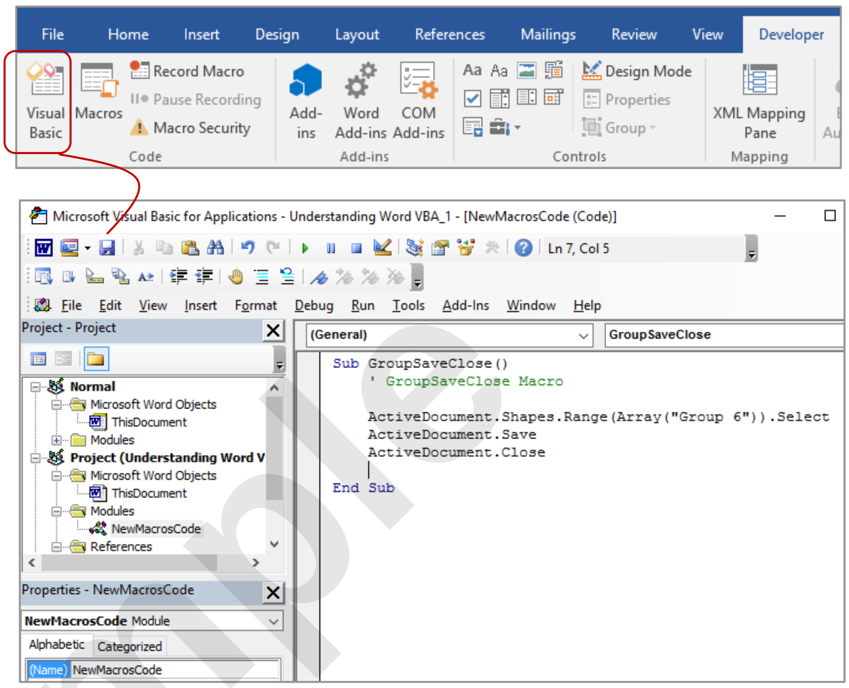
3 Select **File** > **Close and Return to Microsoft Word**

The Editor will close and you will return to the Word document…

4 Press Alt + F11 to reopen the Editor window

5 Press Alt + Q to close the Editor window

6 Close the document

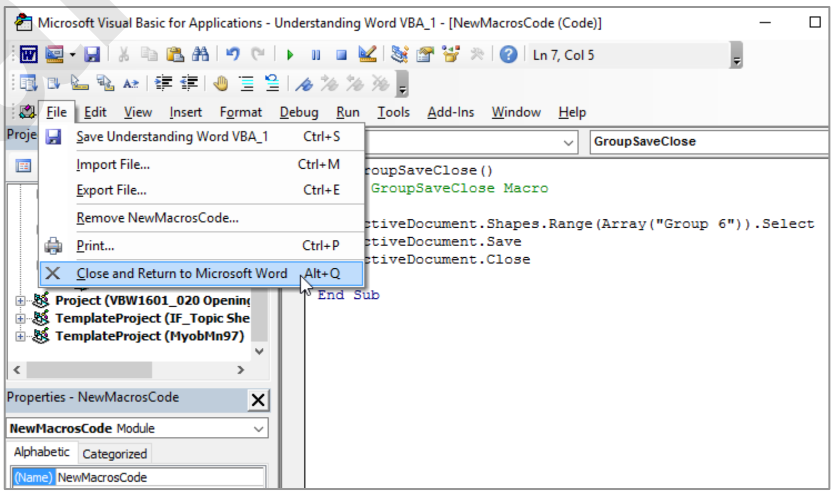


```
Sub GroupSaveClose()
' GroupSaveClose Macro

    ActiveDocument.Shapes.Range(Array("Group 6")).Select
    ActiveDocument.Save
    ActiveDocument.Close

End Sub
```

2



Microsoft Visual Basic for Applications - Understanding Word VBA_1 - [NewMacrosCode (Code)]
Ln 7, Col 5
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Save Understanding Word VBA_1 Ctrl+S
Import File... Ctrl+M
Export File... Ctrl+E
Remove NewMacrosCode...
Print... Ctrl+P
Close and Return to Microsoft Word Alt+Q
Project (VBW1601_020 Opening
TemplateProject (IF_Topic She
TemplateProject (MyobMn97)
Properties - NewMacrosCode
NewMacrosCode Module
Alphabetic
Categorized
(Name) NewMacrosCode
GroupSaveClose

```
oupSaveClose()
 GroupSaveClose Macro

tiveDocument.Shapes.Range(Array("Group 6")).Select
tiveDocument.Save
tiveDocument.Close

End Sub
```

3

For Your Reference…

To ***open*** the ***VBA Editor***:

- On the ***Developer*** tab, click on ***Visual Basic*** in the ***Code*** group, or
- Press Alt + F11

To ***close*** the ***VBA Editor***:

- Select **File** > **Close and Return to Microsoft Word**

Handy to Know…

- Alt + F11 acts as a toggle between the Editor and the document. When you use Alt + F11 to swap back to the document, it keeps the Editor open rather than closing it.

UNDERSTANDING OBJECTS

In VBA, an ***object*** is anything in an application that you can manipulate in some way. For example, your code may apply a theme to a document, change the style of a paragraph, apply bold to a range of text, maximise a window or set a specific application option. Each of these items – document, paragraph, range, window and application – is an ***object***.

Object Collections

You can have several Word documents open at any time. Each of these documents is a separate ***object*** that belongs to a ***collection***. A ***collection*** is simply a set of similar objects. For example, Word's ***Documents*** collection is the set of all open ***Document*** objects. Because collections are objects themselves, they have their own properties and methods that you can use to manipulate one or more objects in the collection.

The members of a collection are called ***elements***. You can refer to an individual element by the object's name or its index value (the ***index*** is the number that Word lists beside the filename in ***Switch Windows*** in the ***View*** tab). For example, you can programmatically close a document named ***Intro.docx*** using the following two commands (where ***Intro.docx*** is the first document opened in the current Word session):

```
Documents("Intro.docx").Close or
Documents(1).Close
```

Notice how the object (***Documents***) and the method (***Close***) are delimited by a full stop (**.**).

If you don't specify an element – such as ***("Intro.docx")*** or ***(1)*** – VBA assumes you are working with the entire collection.

Object Properties

Every object has a defining set of characteristics. These characteristics are called the object's ***properties*** and they control the appearance and position of the object. For example, the ***Window*** object has a ***WindowState*** property that you can use to display a window as maximised, minimised or normal.

When you refer to a property you use the syntax **Object.Property**. For example:

```
Window.WindowState
```

Properties appear in a listing with a property icon .

Object Methods

An object's ***method*** describes what you can do with the object. For example, you can ***save*** (method) the ***active document*** (object).

When you refer to a method you use the syntax **Object.Method**. For example:

```
ActiveDocument.Save
```

Some methods have ***arguments*** and you use the syntax **Object.Method argument1, argument2**, ...

```
Documents.Save NoPrompt :=True, OriginalFormat :=wdOriginalDocumentFormat
```

The above statement will save each open document in the ***Documents*** collection in their original format without first prompting the user (unless a document has not been previously saved). Including the two arguments with this statement is optional; if you don't include them Word will prompt the user to save any documents that have been changed since they were last saved. Note, however, that you cannot specify only one of the expected arguments – it's either both or none.

Notice also, in the example above, how the ***:=*** operator assigns ***values*** to the named arguments ***NoPrompt*** and ***OriginalFormat***. The names of the valid arguments, plus the ***constants*** (such as ***wdOriginalDocumentFormat)*** that can be used with them, are listed in the Help system for each particular method.

The names of the valid arguments, plus the ***constants*** that can be used with them, are listed in the ***Help*** system for each particular method. ***Methods*** appear in a listing with the method icon .

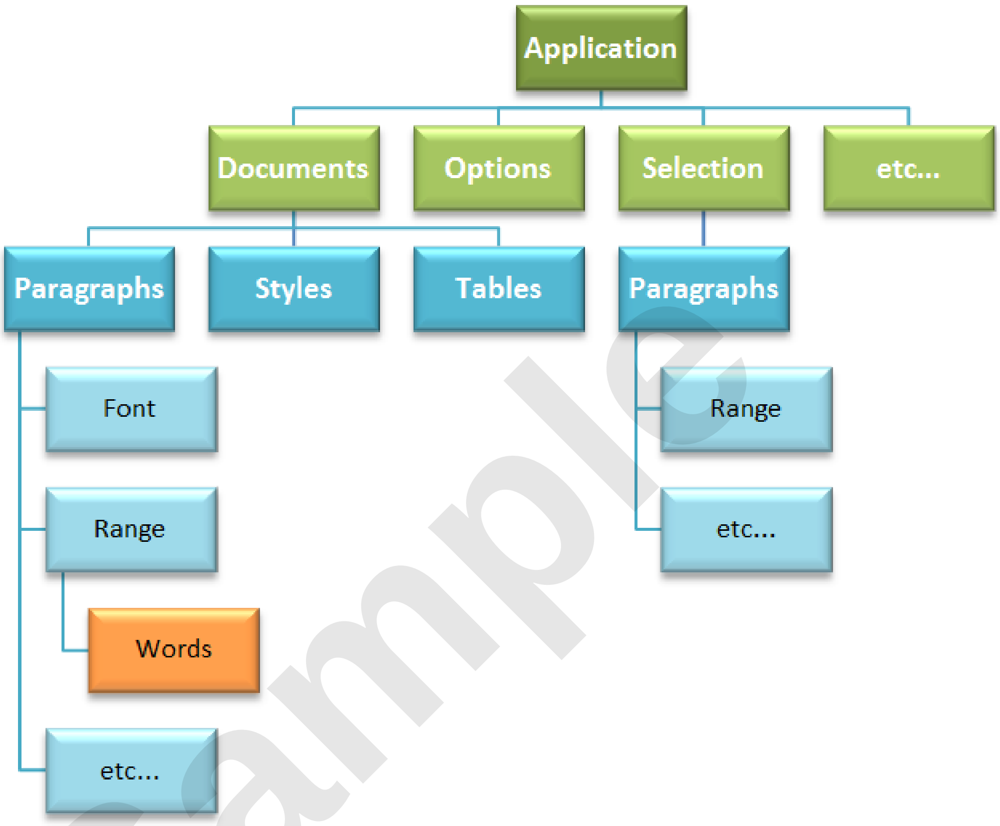
Object Events

An ***event*** is something that happens to an object. The opening of a document in Word is an example of an event. Although Word has an ***Open*** method that you can use to open a document, this ***method*** only initiates the procedure; the actual process of the file being opened is the ***event***.

For example, you may write a procedure (which is called an ***event handler***) that will display a message box each time a specific document is opened.

The Object Hierarchy

Each Office application contains its own set of objects. These objects are arranged in a ***hierarchy*** with the most general (the *Application* object which refers to the program) at the top. In Word, this object contains more than 30 objects – we've shown only three below. The lower levels progress through more specific objects, such as ***Documents***, ***Tables***, ***Styles*** and so on.



Specifying Objects

To specify an object in the hierarchy you usually start with the uppermost object and add the lower objects, separated by full stops. For example, to insert a new paragraph before a selection of text, you could type:

```
Application.Selection.Paragraphs(1).Range.InsertParagraphBefore
```

One of the most confusing aspects of objects and properties is that some properties also act as objects. When you look in Help you will see that the ***Application*** object has a ***Selection*** property, but you will also see that ***Selection*** is an object in its own right – it represents the current selection in a window or pane, or the insertion point if nothing is selected.

In other words, lower-level objects in the object hierarchy are really just properties of their parent objects. Therefore, because the ***Selection*** object implicitly refers to the ***Application*** object you can reduce the above statement to:

```
Selection.Paragraphs(1).Range.InsertParagraphBefore
```

VIEWING THE WORD OBJECT MODEL

The ***object model*** for Word is a very extensive hierarchy of ***objects*** and ***collections***. In previous versions of Word you could readily find a pictorial representation of the entire model; however, in Word you can only view pictorial representations of the individual objects of the model. Nevertheless, this is still a great way of finding the methods and properties of an object.

Try This Yourself:

Open File

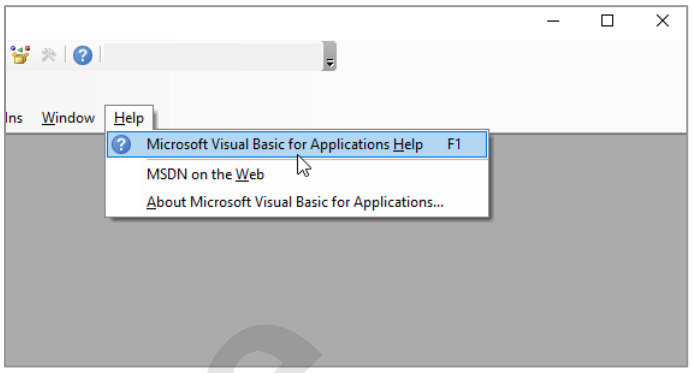
Before starting this exercise ensure Microsoft Word has started…

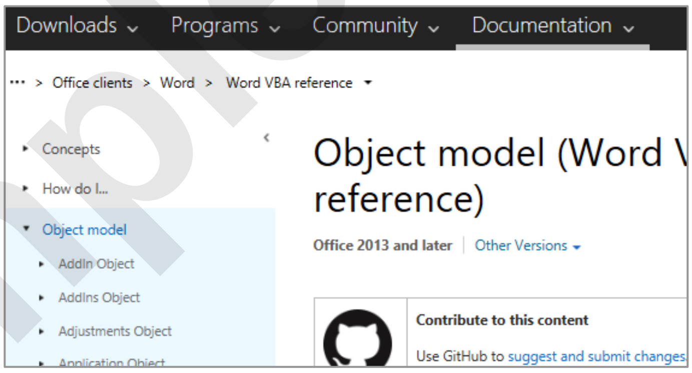
1. Open a new, blank Word document, then click on the ***Developer*** tab, then click on ***Visual Basic*** in the ***Code*** group to open the ***VBA Editor***
2. In the Editor, select **Help** > **Microsoft Visual Basic for Applications Help**, as shown, to open the ***Word*** page in a new browser window
3. Click on ***Word VBA reference*** in the left pane, as shown, then click on ***Object model*** in the left pane to display the list of objects in the model

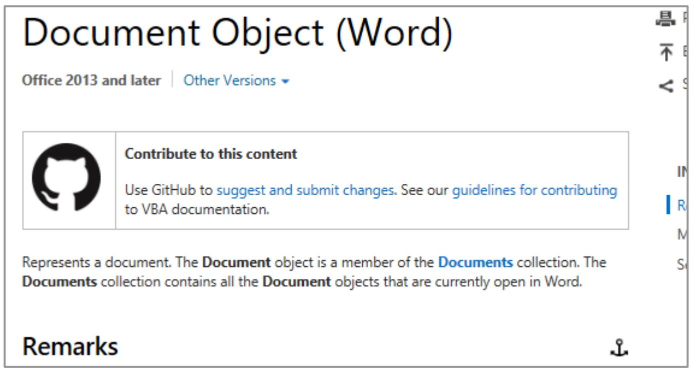
   Objects that appear with an arrow have an additional level of hierarchy…
4. Scroll to and click on ***Documents Object*** in the left pane

   This displays the methods and properties for the Documents object…
5. Close the browser tab

   Leave the document and the Editor open for the next exercise

2

Ins
Window
Help
Microsoft Visual Basic for Applications Help F1
MSDN on the Web
About Microsoft Visual Basic for Applications...

3


4


Use GitHub to suggest and submit changes. See our guidelines for contributing to VBA documentation.

Represents a document. The **Document** object is a member of the **Documents** collection. The **Documents** collection contains all the **Document** objects that are currently open in Word.

For Your Reference…

To ***access*** the ***Word object model***:

1. Select **Help** > **Microsoft Visual Basic for Applications Help**
2. Click on ***Word VBA reference***
3. Click on ***Object model***

Handy to Know…

- It is more important to know how to find the object model than it is to memorise the entire structure. Understanding how these objects, methods and properties work together will come with time and experience.

USING THE IMMEDIATE WINDOW

The Editor has an ***Immediate*** window that lets you type instructions and test expressions. It runs statements immediately as if they were run from a procedure. You can also precede variables and expressions with a ***question mark*** to perform ***what is*** operations. For example, ***?Documents.Count*** asks "*How many documents are open?*".

Try This Yourself:

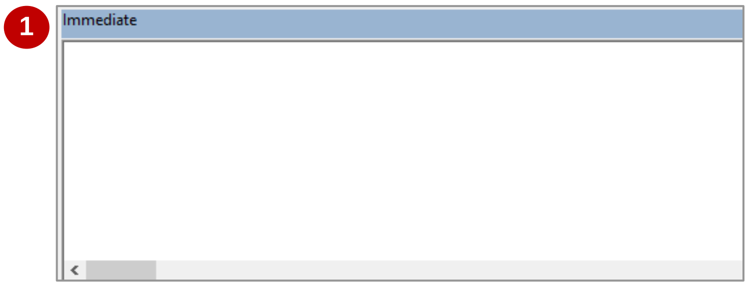
Continue using the previous file with this exercise or open a new blank document and open the VBA editor…

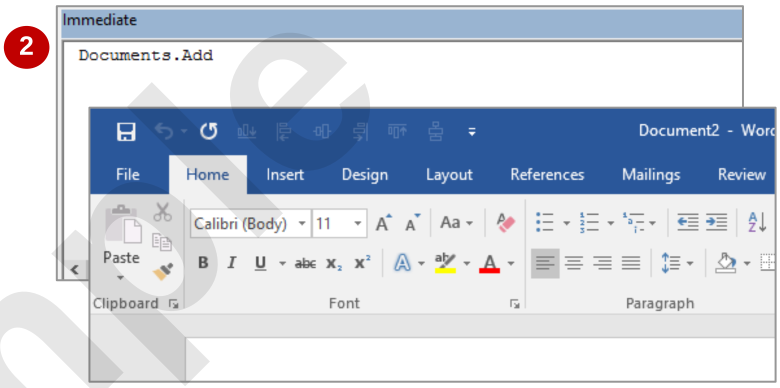
1. Select **View** > **Immediate Window** to display the ***Immediate*** window as a pane in the Editor
2. Type **Documents.Add**, then press Enter to create a new document
3. Click on ***Documents.Add***, then press Enter to run the command again

   You can see the documents that are open in the Project Explorer…
4. Repeat step ***3*** three more times to open a ***collection*** of documents
5. Click under the command, type **Documents.Open "C:\Course Files for Microsoft Word 2016 VBA \Alpheius.docx"**, then press Enter to open the document called ***Alpheius.docx***

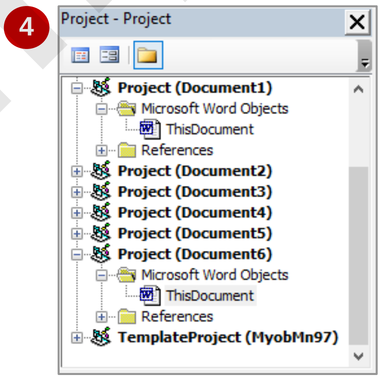
   It will replace Document1 in the list…
6. In the ***Immediate*** window, type **?Documents.Count** and press Enter to display the number of documents that are open

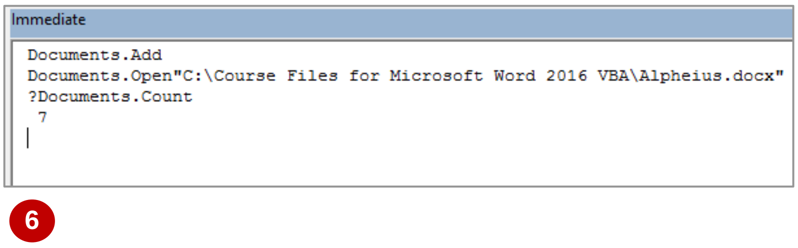
   Leave all open documents displayed for the next exercise





```
Documents.Add
```





```
Documents.Add
Documents.Open"C:\Course Files for Microsoft Word 2016 VBA\Alpheius.docx"
?Documents.Count
 7
```

For Your Reference…

To ***display*** the ***Immediate Window***:

- Select **View** > **Immediate Window**

To ***use*** the ***Immediate Window***:

1. Type a command in the window
2. Press Enter

Handy to Know…

- Computer programming requires a much higher degree of accuracy than you may be used to. For instance, if you have typed the file and file path details incorrectly, or if the files are in a different location on your computer, the programming instructions you type will result in an error message.

Working With Object Collections

In VBA some objects belong to ***collections***. The ***Documents*** collection is the set of all the ***document*** objects open in the ***application***. All of the ***templates*** currently available belong to the ***Templates*** collection. Each item in a collection is known as an ***element*** and must be referenced as part of the collection, either by name or by its position in the collection (known as the ***index***).

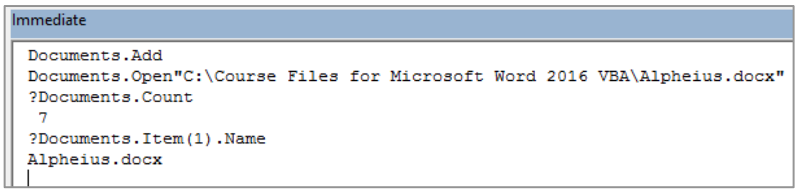
Try This Yourself:

Continue using the previous files with this exercise…

1. Ensure the cursor is positioned on a blank line in the ***Immediate Window***, type **?Documents.Item(1).Name**, then press Enter to return the name of the first document
2. Type **Documents.Item(1).Activate**, then press Enter to make this the active document
3. Type **Documents.Item(2).Activate**, then press Enter to make the second document active
4. Type **?Documents.Item ("Alpheius.docx").Saved**, then press Enter to see whether the file has been saved – *True* if it has, *False* if it has not
5. Type **Documents.Item ("Alpheius.docx").Close**, then press Enter to close the document

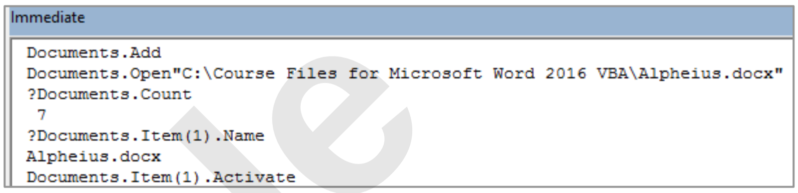
   The document Alpheius.docx will disappear from the list…
6. Type **Documents.Close**, then press Enter to close all documents – don't save if prompted

   Leave the Editor open for the next exercise



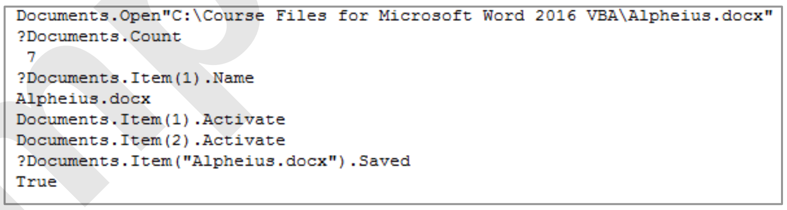
```
Immediate
Documents.Add
Documents.Open"C:\Course Files for Microsoft Word 2016 VBA\Alpheius.docx"
?Documents.Count
 7
?Documents.Item(1).Name
Alpheius.docx
```

1



```
Immediate
Documents.Add
Documents.Open"C:\Course Files for Microsoft Word 2016 VBA\Alpheius.docx"
?Documents.Count
 7
?Documents.Item(1).Name
Alpheius.docx
Documents.Item(1).Activate
```

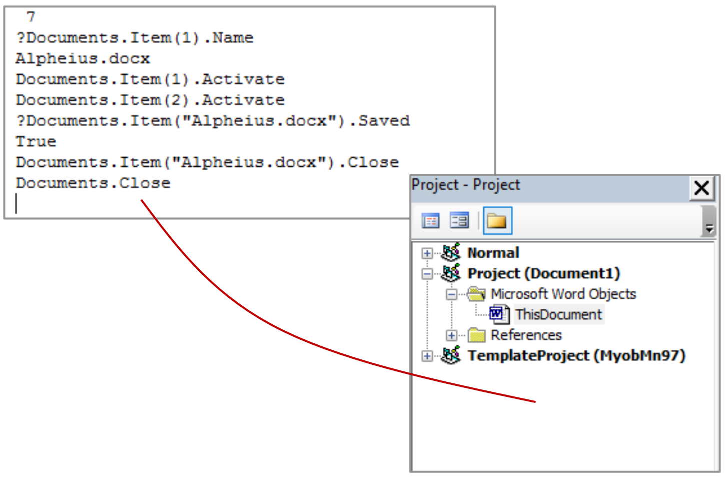
2



```
Documents.Open"C:\Course Files for Microsoft Word 2016 VBA\Alpheius.docx"
?Documents.Count
 7
?Documents.Item(1).Name
Alpheius.docx
Documents.Item(1).Activate
Documents.Item(2).Activate
?Documents.Item("Alpheius.docx").Saved
True
```

4

6



```
 7
?Documents.Item(1).Name
Alpheius.docx
Documents.Item(1).Activate
Documents.Item(2).Activate
?Documents.Item("Alpheius.docx").Saved
True
Documents.Item("Alpheius.docx").Close
Documents.Close
```

For Your Reference…

To ***work*** with ***collections use*** the ***structures***:

CollectionName.**Item**(*index or "name"*).*Method*

CollectionName.**Item**(*index or "name"*).*Property = value*

Handy to Know…

- You can find out information about the active document by using the object ***ActiveDocument***. For example, ***?ActiveDocument.Name*** will display the name of the active document.

SETTING PROPERTY VALUES

Properties affect the way an object looks or behaves. The properties for objects can be listed by typing the name of the ***object*** followed by a full stop. VBA includes a built-in ***list*** system (known as ***IntelliSense***) that will list methods, properties and values as you type your command. ***Property values*** are set by specifying the ***object***, the ***property***, an ***equal sign*** and the new ***value***.

Try This Yourself:

Open File

Before starting this exercise ensure your cursor is positioned in the Immediate Window in the VBA Editor…

1 Type **Documents.Add** and press Enter to create a new blank document

2 Type **ActiveDocument.Close**

Notice how a list of methods and properties appears when you press the full stop or get to the end of a method or property…

3 Press Enter, then type **Documents.Add** and press Enter

4 Type **ActiveDocument.Saved**=

IntelliSense will list the possible values for the Saved property. Let's use the logical value False to force Word to display a prompt to save the unsaved document when it is closed…

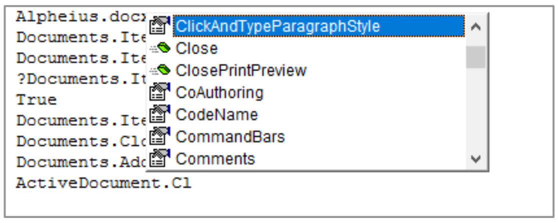
5 Type **False** and press Enter, then repeat step ***2*** and press Enter

This will close the document but now you'll be prompted to save it…

6 Click on **[Don't Save]**

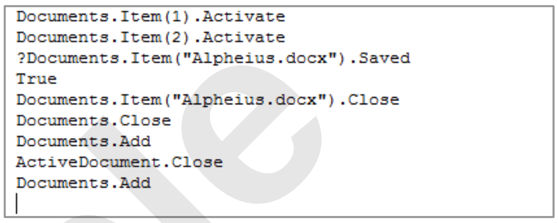
Leave the Editor open for the next exercise

2



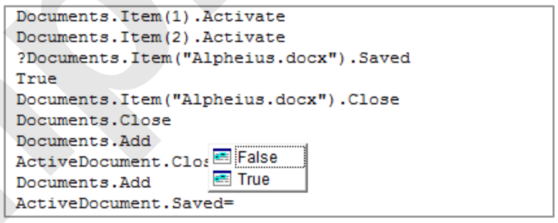
```
Alpheius.doc
Documents.Ite
Documents.Ite
?Documents.It
True
Documents.Ite
Documents.Clc
Documents.Adc
ActiveDocument.Cl
```

3



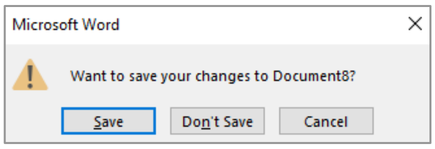
```
Documents.Item(1).Activate
Documents.Item(2).Activate
?Documents.Item("Alpheius.docx").Saved
True
Documents.Item("Alpheius.docx").Close
Documents.Close
Documents.Add
ActiveDocument.Close
Documents.Add
```

4



```
Documents.Item(1).Activate
Documents.Item(2).Activate
?Documents.Item("Alpheius.docx").Saved
True
Documents.Item("Alpheius.docx").Close
Documents.Close
Documents.Add
ActiveDocument.Clos
Documents.Add
ActiveDocument.Saved=
```

5



Microsoft Word

Want to save your changes to Document8?

For Your Reference…

To ***set*** a ***property value use*** the ***structure***:

Object.Property = Value

- ***Value*** can be any one of VBA's recognised data types, including: ***Numeric values*** (e.g. *ActiveDocument.Range.Font.Size = 14*) or ***String values*** (e.g. *ActiveDocument.Range.Font.Name = "Arial"*)

Handy to Know…

- It can take some time to get used to working with properties. Some properties can only be read, while others can be changed. When you change a property you are said to be ***setting*** its value. When you read a property you are said to be ***getting*** its value.