Become an expert
Learn more about where your career can take you

Communicator

The Institute of Scientific and Technical Communicators
Winter 2016

Help your business value documentation

Give feedback in the classroom and in practice

Discover nanotechnology

Find out about Doc-to-Help improvements

Document content and manipulation

Changing your document's content via VBA.

By Mike Mee

Introduction

Welcome to the seventh article in my series about macro programming for Word.

This article will cover a number of methods that can be used to change the content of a document in numerous ways.

By the end of this article, you will have a series of 'quick fix' macros to add to your own library – whether you tweak them to do anything extra, or maybe you convert them into proper functions, is entirely up to you.

Looping through the paragraph's contents

The first port of call is to generate a loop that will go through all of the paragraphs in your document. I will also expand on it afterwards to loop through all of the individual sentences within a paragraph.

I will use the `For .. Next` type of loop, which was covered in my article in the Spring 2016 issue of *Communicator*.

The loop example below does nothing at all, **except** loop through each paragraph in the current document and store in the variable `objPara`, the range of data found in the current paragraph.

```
Sub LoopParagraphs()
Dim lngPara As Long
Dim objPara As Object

With ActiveDocument
For lngPara = 1 To .Paragraphs.Count
Set objPara =.Paragraphs(lngPara).Range

' Using objPara, you can now manipulate
the content and formatting of the current
paragraph

Next lngPara
End With
Set objPara = Nothing
End Sub
```

The loop is now 'in charge' of your paragraph. You can do whatever you want to manipulate the content and formatting of the paragraph. So long as your code appears before the `Next lngPara` command.

The `Set objPara = Nothing` is there to just reset the definition of the object variable. This is also known as 'tidying up' the memory that VBA has used up during the loop.

The following simple examples show the kind of changes you can now make to the content of each paragraph. Each example needs to be inserted into the main loop above.

Changing the format of the paragraph

You can alter the formatting of the font used in each paragraph, by adapting the code below as you see fit.

For example, to change the font in all of the paragraphs to Times New Roman at 14 points, insert this code before the `Next lngPara` statement and run the code.

```
objPara.Font.Name = "Times New Roman"
objPara.Font.Size = 14
```

There are other tweaks you can do to the font in each paragraph, such as Emboss, Shadow, DoubleStrikeThrough or Outline. There are many others available too.

To use the examples listed above, they need to be set to `True` or `False`, as required, using the following lines within the paragraph loop:

```
objPara.Font.Emboss = True
objPara.Font.Shadow = True
objPara.Font.DoubleStrikeThrough = True
objPara.Font.Outline = True
```

There are other format tweaks that are possible, you can find out which ones via the Intellisense option in VBA's editor.

Make a backup of your source document first though, just in case you forget to undo the changes your code makes.

Counting the characters found

To count all of the characters in the current paragraph, you use the `Len` command on `objPara`. This code needs to be inserted before the `Next lngPara` command in the previous listing.

```
lngChars = Len (objPara.Text)
Msgbox "Number of characters in paragraph "
& objPara & " is = " & lngChars
```

Remember to include a `Dim lngChars As Long` at the start of the adapted procedure, otherwise it won't work.

You could also change the Msgbox command to be a `Debug.Print` command and the information will appear in the Immediate Window instead within VBA.

When you run the code using this enhancement, a message box will appear showing you how many characters are in each paragraph, so only use it on small documents or you will need to press the Ctrl & Break keys to escape out of the loop.

Watch out for this

One final note: If you adapt the code to remove any paragraph(s), then the routine will fail if your code attempts to remove the very last paragraph in the document.

To prevent this from happening, you will have to use a reverse `For .. Next` loop which counts down from the very last paragraph to the first, like this:

```
For lngPara = .Paragraphs.Count To 1 Step -1
```

There is a more in-depth explanation as to why you need to go through the paragraphs in reverse in *Communicator*, Summer 2016.

Adding or removing indents

As a technical communicator, there are times when you receive a document that is formatted using various indentations that need fixing. Normally you would have to go through all of the paragraphs and reset them individually.

However, there are VBA functions to allow you to change the indent of one, or all, paragraphs in your document. These functions are listed in the following paragraphs, with the parameters included to all reset of the indents:

```
Sub ResetIndents(ByVal strIndent As String)
Dim lngPara As Long

Debug.Print "Reset Indents = " & strIndent

With ActiveDocument
For lngPara = 1 To .Paragraphs.Count

With .Paragraphs(lngPara).Format
Select Case strIndent

Case "Right"
  .CharacterUnitRightIndent = 0
  .RightIndent = CentimetersToPoints(0)

Case "Left"
  .CharacterUnitLeftIndent = 0
  .LeftIndent = CentimetersToPoints(0)

Case "First"
  .CharacterUnitFirstLineIndent = 0
  .FirstLineIndent = CentimetersToPoints(0)

Case "All"
  .CharacterUnitRightIndent = 0
  .RightIndent = CentimetersToPoints(0)
  .CharacterUnitLeftIndent = 0
  .LeftIndent = CentimetersToPoints(0)
  .CharacterUnitFirstLineIndent = 0
  .FirstLineIndent = CentimetersToPoints(0)

End Select

End With

Next lngPara
End With
End Sub
```

Figure 1. A subroutine which resets your defined indent

Reset all first line indents:

```
With objPara
.CharacterUnitFirstLineIndent = 0
.FirstLineIndent = CentimetersToPoints(0)
End With
```

Reset all left indents:

```
With objPara
.CharacterUnitLeftIndent = 0
.LeftIndent = CentimetersToPoints(0)
End With
```

Reset all right indents:

```
With objPara
.CharacterUnitRightIndent = 0
.RightIndent = CentimetersToPoints(0)
End With
```

Using the coverage of functions from the previous article, Figure 1 shows a subroutine that you can call with a simple parameter to reset whichever indent – or all of them – as you require.

You can add this subroutine into any of your modules that contains your own functions. Just call it when needed via this single line of code:

```
ResetIndents(parameter)
```

Replace the `parameter` with either `First, Right, Left` or `All`. Ensure that you put double-quotes either side of the parameter value you use.

```
ResetIndents("All")
```

However, because VBA can be a bit picky – and depending on where you copy this code – you will need to include the name of the module first. For example, if the above code was put into `ThisDocument`, then you would need to change the call to:

```
ThisDocument.ResetIndents (parameter)
```

As a future expansion to this code, you could add other items that might need formatting. Or amend the zero values and use it to format each paragraph with exact indent settings. The choice is yours.

Adding a border to all images

This code example shows a loop which goes through every image (also known as, inline shapes) and changes the border to be a 3D embossed red line, with a width of 100 points.

Whilst you can make changes to the borders of any image in your document, you will need to add the `.Enable = True` parameter, otherwise the changes won't be applied.

```
Sub BorderAroundAllImages()
Dim lngImage As Long
With ActiveDocument

' Start the loop
```

```
For lngImage = 1 To .InlineShapes.Count

' Check if the current shape is a picture
If .InlineShapes.Item(lngImage).Type =
wdInlineShapePicture Then

With .InlineShapes.Item(lngImage).Borders

' Parameters to alter
.Enable = True
.OutsideColor = vbRed
.OutsideLineWidth = wdLineWidth100pt
.OutsideLineStyle = wdLineStyleEmboss3D
End With

End If
Next lngImage
End With
End Sub
```

You can change the type, colour and width of the line by altering the parameters accordingly.

These are the built-in colours that VBA recognises: `vbRed, vbGreen, vbYellow, vbBlue, vbMagenta, vbCyan,` or `vbWhite`. You can set your own colours by using the hexadecimal values for the RGB components instead, such as `&HFFB27F` to produce a shade of orange.

The `OutsideLineWidth` parameter can handle widths of `25, 50, 75, 100, 150, 225, 300, 450` and `600` points. Remember to include the `wdLineWidth` part first and, if the width is less than `100`, include the leading `0`, for example, `wdLineWidth050`.

Similarly, you could change the line style parameter too. Microsoft lists 25 possible values for the `wdLineStyle` parameter in Word 2013 and above. The range of values is dependent on the version of Word that you are using, so check that your copy of Word can use some of the parameters first.

If you want to try something a little more taxing, you could convert this routine so that it is like the `ResetIndents` subroutine. It should handle the width, colour and line style as parameters. Ensure it resides inside your library of other modules that you have created.

Removing the border

This is the reverse of the above function. It will scan through all of the inline images and remove any borders, regardless of the colour or size.

```
Sub RemoveBorderFromImages()
Dim lngImage As Long

With ActiveDocument
' Start the loop
For lngImage = 1 To .InlineShapes.Count

' Check if the current shape is a picture
If .InlineShapes.Item(lngImage).Type =
wdInlineShapePicture Then

.InlineShapes.Item(lngImage).Borders.Enable =
False

End If
Next lngImage
End With
End Sub
```

Note that by setting `.Enable = False`, Word does not just switch off the lines, but it resets all of the other parameters to their defaults.

To see this in action, change the value of `.Enable` to be `True` and run the code again. Thin black lines will be applied instead of the 3D embossed red ones that were set previously.

Changing the styles of a paragraph

Maybe the company you work for has a set style that must be applied to each paragraph. As we all know, quite often these can be altered via Word's Paste option. Especially when we forget to use Paste Special and the normal Paste function brings in the other document's style(s) too.

Earlier in the article, the code example showed you how to loop through all of the paragraphs and change the text in some way.

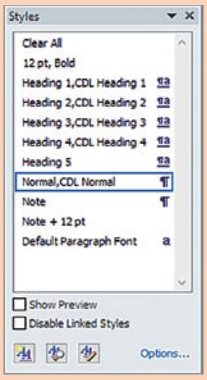
The next code example does something similar, but instead it loops through the styles that are used in each paragraph and displays them, via the `Debug.Print` command, in the Immediate Window.

To show an example of 'fixing something', the code includes a simple test to see if any of the styles found match 'Heading 5'. If yes, then these paragraph styles are changed to the 'Normal' style.

```
Sub LoopThroughStyles()
Dim lngPara As Long
Dim strStyle As String

With ActiveDocument
For lngPara = 1 To .Paragraphs.Count

' Get current paragraph style name
strStyle = .Paragraphs(lngPara).Style

Debug.Print "Paragraph #" & lngPara & " has
" & strStyle & " style applied to it."

Select Case strStyle
Case "Heading 5"
  .Paragraphs(lngPara).Style = "Normal"

End Select

Next lngPara
End With
End Sub
```

You can adapt the above macro and it will, in the end, save some of the time that you would normally waste hovering between the document's content and Word's Format Painter button.

There are manual ways to do this, via the Draft or Outline views and the 'Style area pane width in Draft and Outline views' setting in Word's Advanced Options or, the Styles window. However, this is an article about speeding things up with macros to save you time.



Style panel

Further reading

Mee M (2015) 'Variables and screen output/input' *Communicator*, Autumn 2015: 44-47

Mee M (2016) 'Creating VBA loops: part 1' *Communicator*, Spring 2016: 34-36

Mee M (2016) 'Creating VBA loops: part 2' *Communicator*, Summer 2016: 52-55

Microsoft 'Declaring Variables' https://msdn.microsoft.com/en-us/library/office/gg264241.aspx?f=255&MSPPError=-2147217396 (accessed April 2016)

Next article

If you get stuck trying to recreate any of the subroutines covered in this article, you can contact me via email, Twitter or put a post on the ISTC forum. C

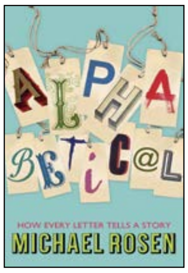
Mike Mee MISTC is technical author with 14 years' experience in the software industry.
E: mugukmail@gmail.com
T: @Mug_UK
W: www.mikestoolbox.co.uk – my toolkit for Word 2007-2016 (the full source code is also available on the website).

Book review

Alphabetical

How every letter tells a story

By Michael Rosen
John Murray (2013), Hardback: 336 pages. ISBN 978-1848548862.
Reviewed by Jean Rollinson FISTC.



"This book is recommended to anyone with an interest in the origins of letters and language."

The subtitle of this book really tells it all: the story we can attach to each letter of the alphabet. But as it's by Michael Rosen there's a lot more to this book than that.

Each chapter begins with a brief history of how the letter came to look the way it does; the differences between the uppercase and lowercase versions; how to pronounce the letter name; and how to pronounce the letter sound. Then for each letter, Mr Rosen picks a word (related to language) beginning with that letter that he can tell us about. For example 'A is for alphabet', 'G is for Greek' and 'W is for Webster' are some of the chapter titles.

This is such a simple idea that it's surprising it's not been done before, but then maybe it takes a writer like Michael Rosen to see the possibility. Imbued with his usual wit and delight in words, this is a wonderful journey through the familiar letters of our alphabet. As the *Metro* quote on the back cover says 'Thinking about the alphabet is like walking along the pavement and finding a Roman villa hidden beneath your feet.' And I know exactly what they mean. We all take letters for granted. We see them every day and everywhere. It takes someone special to spot that they are interesting and exciting and can tell us so much about our history and the development of civilisation.

I would definitely recommend this book to anyone with an interest in the origins of letters and language. C

Rating: ★★★★★

About the author:
Michael Rosen is one of the most popular contemporary poets and authors of books for children. His titles include *We're Going on a Bear Hunt*, which was the winner of the Smarties Book Prize, *Michael Rosen's Sad Book*, and *Totally Wonderful Miss Plumberry*. Also the presenter of "Word of Mouth" on BBC Radio 4, he received the Eleanor Farjeon Award for services to children's literature in 1997.

Member news

New Members

Member	
Peter Chambers	Castleford
James Kewley	Cramlington
Sean Reed	Newcastle Upon Tyne
Christa van Gelderen	The Netherlands
Graeme West	Glasgow
Matt Woolrich	Tewkesbury
Associate	
Maggie Bate	Twyford
Junior	
Kathryn Hadfield	Cambridge
John Kennedy	Scotland
Ellen Rutter	Shipley
Aleksandra Wieczorek	Poland
Student	
Aaliya Ali	Twickenham
Peter Anton	Guernsey
Timothy Greenhalgh	Stockport

Transfers

Fellow	
Peter Hirons	Southampton
Catherine Sharp	Northern Ireland
Member	
Elizabeth Stewart	Derbyshire
Junior	
Jake Cahill	Surrey

Rejoiners

Member	
Peter Ford	Telford

See what else we offer...

If you enjoyed this article, visit our website to see what else we do.

The Institute of Scientific and Technical Communicators is the largest UK body representing information development professionals, serving both our members and the wider technical communication community.

What the ISTC offers

Professional development and recognition

Resources and opportunities to develop and diversify skills, stay up to date with trends and technologies, and get recognition for achievements.

Our CPD (Continuous Professional Development) framework enables you to provide evidence of your learning in all its forms, and our Awards programme gives you the opportunity to showcase excellent work.

Communicator professional journal

Communicator is the ISTC's award-winning quarterly professional journal, covering the breadth of technical communications, offering in-depth articles, case studies, book and product reviews.

Now you've read a sample article, would you like to see more? The journal is free to our members and is also available on subscription.

ISTC Community

The ISTC offers opportunities to network, exchange expertise, and stay in touch with the UK technical communication industry – through a range of online groups, local events, and InfoPlus+ (our monthly newsletter).

Find us on LinkedIn, Eventbrite and YouTube.

Technical Communication UK

The ISTC hosts Technical Communication UK, the annual conference that aims to meet the needs of technical communicators, their managers and clients, from every corner of the industry.

Open to all, visit www.technicalcommunicationuk.com for the latest news.

ISTC Resources

The ISTC offers access to discounted essential resources, including British Standards publications.

Institute of Scientific and Technical Communicators
The home of technical communication excellence in the UK

T: +44 (0)20 8253 4506
F: +44 (0)20 8253 4510
E: istc@istc.org.uk
W: istc.org.uk