- Light Relief
  - Humor!

# Word Tutorials

Literature

  - **Book reviews and recommendations**
  - Links to other sites

- Word Tutorials
  - Bend Word to Your Will (written for Word for the Macintosh, this article has broad applicability to all versions of Word).
  - How to save yourself hours by using Outline View properly
  - What do all those funny marks, like the dots between the words in my document, and the square bullets in the left margin, mean?
  - Typographical Tips from Microsoft Publisher
  - Some of the most useful Word keyboard shortcuts
  - Word commands, and their descriptions, default shortcuts and menu assignments
  - How can I insert special characters, such as dingbats

## Recommended books on how to use Word

## Links to other sites

## Light relief

## *How to Use Word*

Bend Word to Your Will (written for Word for the Macintosh, this article has broad applicability to all versions of Word).

How to save yourself hours by using Outline View properly

What do all those funny marks, like the dots between the words in my document, and the square bullets in the left margin, mean?

Typographical Tips from Microsoft Publisher

Some of the most useful Word keyboard shortcuts

Word commands, and their descriptions, default shortcuts and menu assignments

How can I insert special characters, such as dingbats and accented letters, in my document?

Setting tabs

Ruler of all you survey: How to make the best use of Word's rulers

The strait and narrow – using columns

## *Troubleshooting*

How can I recover a corrupt document or template – and why did it become corrupt?

How to recover a Master Document

Why Master Documents corrupt

## *Using Tables*

I have a "Name" column which I want to split into "FirstName", "LastName" – how can I do it?

## *Forms, Fields and Macro Programming*

How to add pop-up lists to any Word document, so you can click your way through changes in seconds

Creating a macro with no programming experience using the recorder

Run a macro automatically when Word starts or quits

Run a macro automatically when a document is created, opened or closed

Getting to grips with VBA basics in 15 minutes

The art of defensive programming

Why variables should be declared properly

How to cut out repetition and write much less code, by using subroutines and functions that take arguments

When to use parentheses to enclose subroutine and function arguments

Early vs. Late Binding

- How to customise the Control Toolbox in the VBA Editor
- Intercepting events like Save and Print
- Writing application event procedures

# Bend Word to Your Will

Article contributed by Clive Huggan

**Word Document: Zip 1 mb**

**Template: Zip 185 kb**

This article was last updated in September 2007. Since Clive updates it more than once a year, you may wish to come back!

You can download "Bend Word to Your Will" in Word document format (as a compressed Zip file) from the links above.

By clicking here on Table of Contents, you can see what's in the document before you decide whether to download it. For the most part it's structured like a dictionary with self-contained articles, so even though it's about 200 pages long, it isn't difficult to find information – just as with a dictionary! It's intended to be used on-screen rather than to be printed out, because the articles have clickable hyperlinks for instant access to related topics.

Clive has been continually improving these notes since 2001 to increase his speed and efficiency, especially in working on long and/or complex documents – although the notes are equally useful for shorter documents. He's particularly interested in reducing the chances of corruption in his documents because in Clive's professional work they are distributed back and forwards between many people, on PCs and Macs. His emphasis is on configuring Word to make it suit your unique needs, but without making the documents themselves very complex.

From the top of this page you can also download a template (a compressed Zip file) that's related to the main “Bend Word to Your Will” document. If you are not too experienced yet with Word, you will probably not need to download the template – you can always come back later for it. It contains all the styles used in the main document (but so does the main document itself), plus some of the formatting tools, a skeleton for a long document you can create using those tools, and some notes that Clive sends to colleagues when he attaches toolbars to a document to make it easier for others to format it. (The template does not yet include the most recent formatting changes made in the "Bend Word to Your Will" document.)

Please note: If you download the template and install it in the appropriate place and attach a document to it, the template will change the behaviour of Word a little: that's the whole purpose of it. If you want to avoid this, simply keep the template in a different folder from the Microsoft Office “Templates” or “My Templates” folders until you have read the part of “Bend Word to Your Will” which explains how templates are used. If you only want to use a few of the template’s features, it’s easy to do so by transferring them across using Word’s Organizer; you don’t have to install the template to do this.

Finally, this article contains the author's recommendations for working with Word based on his particular experience. Everyone is different, and although most of this material is totally consistent with the Word:Mac MVPs' ideas of good working practice, some of the techniques represent the most suitable approach for the work that Clive does, just as other articles represent other contributors' method of working. If you want to find out about alternative approaches to a topic, we (including Clive) will welcome your questions or comments in the newsgroup news:msnews.microsoft.com/microsoft.public.mac.office.word

This is an example of an article submitted by someone who was not a Microsoft MVP at the time it was published (he is now!). We are very keen to receive such contributions, and we invite you to make one, no matter how small!

You should also be aware that Clive retains Copyright for this article, although he has placed no restrictions on non-commercial use. Clive also has his own Disclaimer. Both statements impose terms in addition to the standard Terms of Use and Disclaimer used on this website and are to be read together.

Return to Top

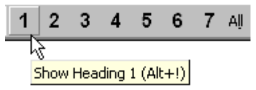
# How to save yourself hours by using Outline View properly

Article contributed by Dave Rado

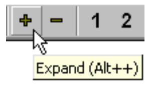
Word's Outline View is wonderful for long documents and – used properly – can cut the time taken to write a typical report, proposal, thesis, or dissertation by as much as 50%.

In order to make use of it, you should create all your headings use Word's built-in Heading Styles (which you can redefine to look the way you want) – using Heading 1 for your Chapter headings, Heading 2 for subheadings and so on. If not familiar with using styles, see John McGhie's article on this site **Creating a Template – The Basics (Part II)**, as well as the excellent **Shauna Kelly** article **Understanding Styles**.

- Outline View lets you view all your Headings collapsed to any heading level you want.

  To set the collapse level, you can either click on the numbers on the outline toolbar (so if you want to view only your Heading 1 paragraphs, click on the **1** button, etc); or you can use the **+** and – buttons on the outline toolbar to collapse and expand just the selected Heading(s).

  So if you click on a Heading 1 paragraph, and then click on the **+** button, it will expand to show you the Heading 2 paragraphs under that Heading 1 paragraph, but not any of the other Heading 2 paragraphs in the document.

- Outline View is an excellent way of getting to the section you want in a long document extremely quickly – switch to Outline View, click on the **1** button to show just the Heading 1 paragraphs; click in the Heading 1 paragraph you're interested in and expand it to see its subheadings, click on the subheading you're interested in and expand, until you're where you want to be. Then switch back to Page layout or Normal view. Much quicker than it sounds, it means you can find your way around a 500 page document just as easily as if it were a 5 page document.

- It makes it incredibly easy to restructure your document. Just drag and drop a heading to move not only that heading, but all its associated subheadings and body text. Or if you don't like drag and drop, use the up and down arrow buttons on the Outline toolbar, or press **Alt + Shift + up** arrow or **Alt + Shift + down** arrow.

  Because this doesn't involve the clipboard, it means you can move 200 pages-worth of information from the end of a 500 page document to the beginning in less than a second, as opposed to probably 15 minutes if you'd had to select it all in the normal way and then use cut and paste.

  If you want to move a Heading 1 (with all its subheadings and body text), click on the **1** button before dragging; to move a Heading 2, click on the **2** button, and so on.

  Best of all, you aren't restricted to moving one heading at a time – whatever you've selected when you drag and drop gets moved. So you can move five contiguous Heading 1 paragraphs with all their subheadings and body text, (which, in a typical document, means well over 100 pages) in one go, and in less than a second.

  Split the pane (Window + Split) if you can't see all your headings at once, in order to drag headings from one section to another.

Terms of use Disclaimer

- To promote subheadings to main headings or demote main headings to subheadings, you can either press the left and right arrow buttons on the Outline toolbar, or you can press **Alt + Shift + left** arrow (on the keyboard) to promote, or **Alt + Shift + right** arrow to demote. You can use these shortcuts in any view, not just in Outline View – a big time saver.

  Again, when you promote or demote a heading, any subheadings and sub-sub-headings associated with it also get promoted or demoted.

  And again, you aren't restricted in how many headings you can promote or demote simultaneously. If you select five Heading 1s and click on the demote button, they will all be converted to Heading 2s, their subheadings will all be converted to Heading 3s and so on (and the Heading Numbering scheme you defined under Format Bullets and Numbering will follow suit).

  Take a scenario where someone else has written a report which you want to incorporate in a report you're writing. So the title of their Report needs to become a Heading 1 in your document, which means you have to convert all their Heading 1 paragraphs to Heading 2 paragraphs, all their Heading 2 paragraphs to Heading 3 paragraphs and so on. Using Outline View this takes seconds. *Without* Outline View it could take hours, literally, if it was a long report.

  Or take another scenario: you are sending a letter to a client and need to include your company's Environmental Policy. But the only document you have containing the Environmental Policy has it as a subsection of the main company policy document. So in this case, having pasted it into your letter, you need to *promote* all the headings by one level – a laborious task *without* Outline View but a 5 second task with it.

- You can create a skeleton report very quickly, by typing all your headings in Outline View and then switching to Normal or Page Layout view to add the body text. Whereas, in Normal or Page Layout view, if you type a heading and press Return you get body text, in Outline view you get another heading (which you can then promote or demote as necessary using the **Alt + Shift +** left and right shortcuts).

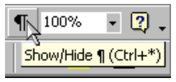
- A Word outline can be pasted straight into PowerPoint; and this is a very quick way to prepare the guts of a presentation; Heading 1 paragraphs are converted automatically to slide titles, Heading 2's to main bullets, Heading 3's to sub-bullets.

# What do all those funny marks, like the dots between the words in my document, and the square bullets in the left margin, mean?

Article contributed by Suzanne S. Barnhill and Dave Rado

Occasionally a new user of Word is alarmed to discover that his previously pristine document is full of strange symbols – dots, arrows, paragraphs marks, and the like. For experienced users, the usual reaction of such a user seems almost comical because experienced users know how invaluable the display of nonprinting characters can be both in formatting and in troubleshooting documents. "Nonprinting characters" is Word's term for anything that takes up space or has a formatting function but does not appear on the printed page: spaces, tabs, paragraph breaks, and the like. Even if you prefer to work most of the time without seeing them, you should know how to display them and what they mean.

By default, the Formatting toolbar in Word has a button with the ¶ icon. If you hover your mouse over it, the ScreenTip says "Show/Hide ¶." (In Word 2007 this button is in the Paragraph group on the Home tab.)



This button toggles between display of all nonprinting characters and whatever specific nonprinting characters you have chosen to display as an alternative. This requires a little explanation. If you look at the **View** tab of **Tools | Options**, you will see six check boxes under "**Nonprinting characters** " ( "**Formatting marks** " in Word 2000 and above). The last one of these is "**All**. " When you toggle the toolbar button on and off, this box is checked and unchecked. When it is unchecked, all you will see will be the nonprinting characters represented by whichever of the other check boxes you have checked. Usually this is none, but there might be times when, for example, you would want to see just Hidden text or just paragraph marks and none of the others. (Note that "Paragraph marks " includes line breaks and text-wrapping breaks, discussed below). (In Word 2007, these check boxes are at **Office Button | Word Options | Display.**)

The shortcut key for "**ShowAll**" is **Ctrl+*** (**Ctrl+Shift+8**). If you've ever turned on display of nonprinting characters unintentionally, it may have been by accidentally pressing this key combination when you were trying to type an asterisk. The same key combination will also toggle the display off.

So what do all these marks represent?

## Paragraph marks

The paragraph mark or pilcrow (¶) represents a paragraph break. You should see one at the end of each paragraph (if there is not one, you'll likely find that you have a problem). Ordinarily you should not see one anywhere else. By this I mean that you should not be ending lines with paragraph breaks, nor should you be using "empty paragraphs" to create "blank lines" between paragraphs (in most cases this is better accomplished with Space Before or After).¶

The ¶ contains all the paragraph formatting. You can select it, copy it, and paste it onto another paragraph to copy and paste formatting (though there are other ways to do this as well). The last ¶ in the document contains formatting for the entire document (header/footer and margin information, for example) or for the last section if there are more than one.¶

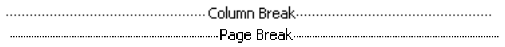
## Line breaks

A right-angle arrow pointing to the left ↵ represents a line break, inserted with **Shift+Enter**. You can use a line break to start a new line without starting a new paragraph.

A right-angle arrow between two vertical lines |↵| represents a text-wrapping break. This new break type, introduced in Word 2000 and intended primarily for Web pages, is used to force subsequent text below an adjacent text-wrapped object. For example, if you have a caption beside a picture and end it with a text-wrapping break, the text following the caption will start below the picture regardless of how long or short the caption is.

## Pagination breaks

More obvious in their meaning are manual column, page, and Section Breaks. To delete these, you can simply select them and press the delete key (or you can use Find and Replace). The examples below show how they appear in Word 2003 and earlier; the display is a little different (but still recognizable) in Word 2007.



::::::::::::::::::::::::::::::Section Break (Continuous)::::::::::::::::::::::::::::::

▪ Finally, you will sometimes see a small black bullet ▪ in the margin next to a paragraph.

This indicates that the paragraph is formatted with the “**Keep with next**,” “**Keep lines together**,” “**Page break before**,” or “**Suppress line numbers**” property.

These settings are found on the **Line and Page Breaks tab** of the **Paragraph** dialog (**Format | Paragraph** in Word 2003 or earlier; accessed through the dialog launcher in the bottom right corner of the **Paragraph** group on the **Home** tab in Word 2007); if you double-click on the “bullet” itself, you will bring up this dialog with the Line and Page Breaks tab selected. Word's built-in Heading styles by default are formatted as “Keep with next,” so you will always see these bullets next to them.

### Space characters

In most fonts, and certainly all Windows “core fonts”, a• small• raised• dot• represents• an• ordinary• space• (some• fonts,• such• as• Arial Special G1, *don't* include a character to represent a space;▯and▯some▯use▯a▯large▯square, which can be very distracting).

Be sure you don't have space • • characters • • where • • they are • • not • • needed. If you are tidy-minded, for example, you won't want a string of them at the end of a paragraph where your thumbs relaxed on the spacebar while you stopped to think.• • • • • • • • • • • • •

A degree symbol ° represents a nonbreaking space (**Ctrl+Shift+Spacebar**), which you can use to prevent words from being separated at the end of a line.

This is useful for keeping dates together (so you don't end up with September
5, 2000), as well as initials such as J. P.
V. D. Balsdon.

En and em spaces (on the **Special Characters** tab of the **Symbol** dialog) are also represented by the degree symbol, but there is extra space to the left of the symbol for an en space °and extra space both left and right for an em ° space.

### Tabs

An arrow pointing to the right → represents a tab character, where you have pressed the Tab key. As explained in the article on setting tabs, in a well-formatted document you should not see more than one of these in a row. → → →

### Hyphens

A conditional hyphen (one that is printed only if it falls at a line break, entered with Ctrl+Hyphen) is shown as ¬.

A nonbreaking hyphen (Ctrl+Shift+Hyphen), which is useful for phone numbers and any hyphenated compound you don't want to break at the end of a line, is displayed as a dash whose length is intermediate between an en (–) dash and an em (—) dash. This is one of the most confusing symbols because it is very difficult to tell, with nonprinting characters displayed, whether you have actually entered a nonbreaking hyphen or a dash.

### Cell markers

In tables you will see one additional character, the universal monetary symbol (¤), which displays variously at various point sizes and magnifications but upon close inspection is seen to be a circle with four lines radiating from the corners. ¤

| | |
|---|---|
| This is the end-of-cell marker. It is a little like the paragraph mark in that it contains paragraph formatting for the last (or only) paragraph in the cell, but it also holds formatting for the cell.<br>The same mark at the end of each row is the (wait for it) end-of-row marker, which serves a similar purpose with regard to row formatting.¤ | ¤ |

### Hidden text

One other type of “nonprinting character” that is toggled by the Show/Hide button ¶ is Hidden text.
This·is·signalled·by·a·dotted·underline.

Even when it is displayed, Hidden text is not printed unless you check the box for it on the Print tab of Tools | Options (in Word 2007, this setting is at **Office Button | Word Options | Display: Printing options**)

There are a number of clever formatting tricks you can do by formatting text (especially paragraph breaks) as Hidden, but you must hide it in order to see how the document will look when printed. It is especially important to hide it before generating a table of contents or index; if there is enough of it to affect the pagination, then the page numbers in your TOC or index may be incorrect. Note in particular that the **XE** (index entry) fields used to generate an index and the **TC** fields that can be used to generate entries in a TOC are formatted as hidden text, so it is important to hide these before generating the index and TOC.

### Coloured underlines

In addition to the dotted underline indicating hidden text, Word uses a variety of different types of colored underlines —solid, dotted, and wavy—to give information about the text. For an explanation of the meaning of these various underlines, see the Help topic “***What do the underlines in my document mean***?” or the Microsoft Knowledge Base article “**Unusual marks that may appear in a Word document**.”

### Anchors

Another very important nonprinting character is the **anchor** symbol – when working with **floating objects** it's often crucial to know where these are

⚓ ¶

There are two other types of nonprinting characters that are not (usually) toggled with the Show/Hide ¶ button:

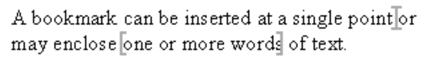
### Field codes

A **field** is a set of codes that instructs Microsoft Word to insert text, graphics, page numbers, and other material into a document automatically. For example, the DATE field inserts the current date. Ordinarily Word displays the result of a field (the date, page number, number of pages, number of words, etc.), but if you press **Alt+F9** or select **Toggle Fields** on the shortcut menu, you will see the code itself, enclosed in curly braces, such as `{ DATE \@ "dddd, MMMM d" }`.

As mentioned above, there are two types of field codes that are toggled with the Show/Hide ¶ button rather than the **Toggle Fields** command. Both **TC** (table of contents entry) and **XE** (index entry) fields are formatted as Hidden text; when you insert either type of field, the display of nonprinting characters is toggled on by default so that you can see these codes.

### Bookmarks.

When the box for “Bookmarks ” is checked on the **View** tab of **Tools | Options** (in Word 2007, find this setting at **Office Button | Word Options | Advanced: Show document content**), user-defined bookmarks are indicated by heavy square gray brackets. A single-point bookmark has the brackets reversed so that it looks like a capital I.


A bookmark can be inserted at a single point I or
may enclose [one or more words] of text.

Even if you choose to keep nonprinting characters hidden most of the time, displaying them can be very helpful in troubleshooting obstreperous documents. If your pages are not breaking as you like, perhaps it is because you have too many (or the wrong) paragraphs set as “Keep with next.” If your printer is adding a blank page at the end of your document, it could be that you have a string of empty paragraphs at the end that are forcing an extra page. Accidentally deleting the paragraph break before a manual page break or Section Break can cause very peculiar problems. And if an automatic number insists on being bold even though you have applied bold formatting to only a part of the numbered paragraph, it could be that you need to select the paragraph mark and unbold it, since automatic bullets and numbering take on the formatting of the paragraph mark. All of these problems are much easier to diagnose if you can see what you're dealing with.

In general, it's best to proofread your documents twice; once for content, with nonprinting characters off (as they can be distracting when reading); and a second time with nonprinting• characters • visible, • so • that • you • can • check • for redundant ↵
line breaks, space • • • characters • and • the like.

Send all feedback to **word@mvps.org** Top - **Ctrl+Home** Previous - **Alt+left arrow** Page Updated:

# Typographical Tips from Microsoft Publisher

Article contributed by **Suzanne S. Barnhill** and **Dave Rado**

The fact that you have come to this Web site suggests that you are a user of Microsoft Word. In one form or another, Word is ubiquitous. If you buy a new computer, chances are good that it will come with a trial version of Office installed. Word is a powerful word processing program that incorporates many of the features of a page layout application, but there are times when a page layout or desktop publishing application is what is needed. If you are using any edition of Office other than the Standard or Home & Student edition, you have such a program: Microsoft Publisher.

Often Publisher will be better suited to your needs than Word, so it is well worth becoming familiar with it. It is a very user-friendly application, generally aimed at a more casual, less professional level of user than Word. But even if you use only Word, Publisher can be useful to you. Because once upon a time, at least, it came with an excellent manual. The *Microsoft Publisher 97 Companion* is a 328-page book (compare this to the 19 pages devoted to Publisher in *Discovering Microsoft Office 2000 Premium and Professional*), and it contains much material that can be equally helpful to Word users.

For example, the chapter “The Look of Words” discusses “what fonts are, how to choose them, and how to get the most from them.” The following tips, guidelines, and rules of thumb are excerpted from that chapter [with some comments interspersed]. We have not attempted to reproduce all the illustrations that appear in the actual manual, but even the text alone is helpful.

## What's a font?

A font is all the characters of a single design – that is, a complete set of all the letters in the alphabet, plus numerals and symbols.…Each font belongs to a family. Just as the members of a human family have similar but usually not identical features, so do the members of a font family. One font might be italic, another bold, and yet another extra-bold. But they all have a common structure that ties them together. Some font families are large (like Rockwell); others have just one member (like Comic Sans MS).

### Typeface and font

As you read other books about fonts, you'll encounter the terms *type* and *typeface*, which are generally used interchangeably with *font* in the desktop publishing environment.

### Traditional fonts

In traditional typesetting, before the advent of computer fonts, different sizes of [a typeface] were considered to be separate fonts. Unlike your computer, which can transform most fonts from tiny to huge with the click of a mouse, every size of every font resided on its own specifically sized piece of metal.

### Set in stone

Roman stonemasons chiseled a *serif* into each stroke they made in slabs of stone (the “paper” of that time) to correct the uneven appearance of the letters and to add a graceful finishing touch to the blunt chiseled edges of the letters.

*Serifs* entice the eye to move from letter to letter. It's this quality that makes serif fonts superbly readable and so suitable when there is a great deal of text to read.

*Sans* is French for *without*, so *sans serif* fonts have no serifs. Their clean lines make them ideal for headlines and other large text that must catch the reader's attention.

*Script* fonts look hand-lettered – the letters appear to be joined visually if not physically. Script is useful mostly for invitations to formal or ceremonial occasions, though some of the newer, less formal scripts are suitable for personal correspondence.

## Font personality

Fonts are like clothing for words. As a man in a black leather jacket, jeans, and rugged boots conveys a very different image and feeling than he does in a three-piece business suit, silk tie, and Italian shoes, so it is with fonts.

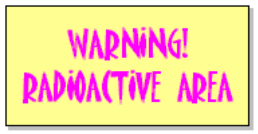


Figure 1

The same words dressed in a different font deliver a very different message. For example, a mischievous, informal font, like the one on the left, can undermine a serious message.

## Mixing and matching fonts

Choosing fonts is a lot like planning a wardrobe. You start with some basic fonts that convey the idea and the feeling of your business or organization, and then from time to time you accessorize with fonts appropriate for a particular publication. It also probably goes without saying that the right font means one that you're comfortable with.

### Choosing a basic look

- Lay the foundation with a couple of fonts that work for your business and stick with them. Those fonts will form the basis of your visual identity (along with a logo, if you have one, and the kind of paper you use), making all your publications instantly recognizable.

**Figure 2: The font for Coca-Cola is so distinctive that even if you saw it in Russian, you would know what it was selling**

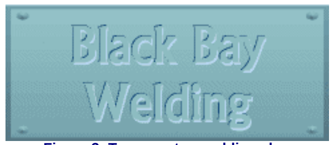


**Figure 3: To promote a welding shop, use a forceful, bold font**

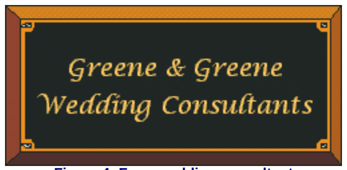


**Figure 4: For a wedding consultant, use something graceful and formal**

- The goal of good design is communication. Since the bottom line of good communication is clarity, make sure that you're clear about your audience, the point you want to make, and the tone you want to set. Then choose a font that suits that message.
- If you're making a publication that will be distributed by fax, avoid fonts with elaborate and delicate detail (like Harrington). The faxing process can fill in the open spaces of the letters and muddy the detail. [Note: this is one instance where sans serif body text can be better than serif.].
- Think about the paper you're using, too. If you use newsprint, textured paper, or other papers that soak up ink, avoid delicate, detailed fonts.

### Mixing fonts

Sometimes one font isn't compelling enough to pique and hold the reader's interest. The contrast between two fonts can attract the reader's eye and help clarify the organization of the piece. For example, designers usually use a different font for body text – the text that forms the main body of the publication – than for headlines (also known as display text). The following rules will help you mix fonts successfully.

- It's a good idea to limit yourself to two font families in a small publication like a business card.…In a longer publication, use a maximum of three fonts. You won't go wrong if you combine a serif font for body text with a sans serif bold for headings.
- In one publication, don't mix two serif fonts from two different families, two sans serif fonts, or two script fonts. If they're similar enough, the mix can look like a mistake.
- Play it safe. If you can't decide whether fonts from two families look right together, use just one font family. Use a lighter weight for body text and a heavier, bold font at a larger size for display text.
- As you gain experience mixing fonts, there will be times when you know you've broken the rules, but you like what you see anyway. Go with it! It means that you've developed your “eye” enough to create your own look.

## The look of body text

As its name suggests, body text is the text that forms the main body of your publication. In a newsletter or a book, it's the story. In a smaller publication, it's everything that isn't either a headline or special text like captions and fancy first letters.

## Choosing a font for body text

First and foremost, body text must be readable. If the words aren't readable, you've lost your audience. More subtly, pay attention to the emotional impact of the font. In the same way that background music in a movie affects the feeling you have about a scene, the font you choose for body text affects the mood of your publication and the response of your audience to it. Also, depending on how much you have to say and how much space you have to say it in, you might need to consider the compactness of the font. Once you've chosen a body font, stick with it throughout your publication; otherwise you can confuse your reader.

**Readability**

- We read words by their shapes and not letter by letter, so readable fonts are those that make it easy for the reader's eye to scan words and lines – to read blocks of text. Serifs give words distinctive shapes that the eye recognizes more easily than it recognizes sans serif shapes. But blocks of sans serif text can be readable as body text, especially when given plenty of white space.
- *Don't use script fonts* or **FONTS MEANT FOR HEADLINES**.

  DON'T USE ALL CAPITAL LETTERS or *all italics for body text*. They can all be practically impossible to read at the smaller sizes of body text.
- A medium-weight font – not too heavy and not too light – is most readable. If you have a text-intensive page, consider a light font like Goudy Old Style. If the text is sparse, use a dark font like Rockwell.
- Consider the height of the lowercase letters (the "x-height," as it's known to typographers). The "x-height" refers to the main body height of the lowercase letters, excluding ascenders and descenders – "x" in particular because in the letter x all the terminals of the letter touch a line of measurement. At small point sizes, fonts with larger x-heights tend to be more readable.

**Feeling**

You might not think of black text as having color. But black text combined with white space "colors" the page. Consider the way the color feels to you and if that's how you want your readers to feel.

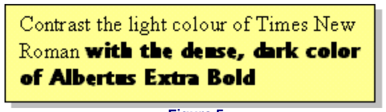

Contrast the light colour of Times New Roman **with the dense, dark color of Albertus Extra Bold**

Figure 5

A light-colored body text has a soft, calm feeling; dark text is stronger and more emphatic. Take your printer into account, too – if it prints on the dark side, you can compensate by choosing a lighter font.

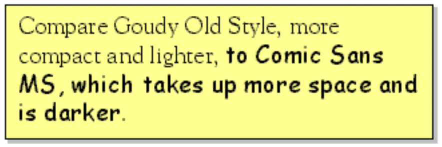

Compare Goudy Old Style, more compact and lighter, **to Comic Sans MS, which takes up more space and is darker.**

Figure 6

**Compactness**
The number of characters per line varies from one font to another even when the same font size is specified. If you have a lot of information to fit into a small amount of space, choose a compact font.

## Formatting body text

You can save yourself a lot of time and effort and easily keep the formatting consistent throughout your publication if you use text styles, particularly if your publication has more than a handful of pages.

### Font size

Make all body text within a publication the same size – between 10 and 12 points is the norm; smaller text can be difficult to read. Obviously this doesn't apply if your publication is a poster that's meant to be read at a distance or a newsletter for people with low vision.

### A quick lesson about points, picas, font sizes, and leading

When you're working in Publisher, it's helpful to understand typographic measurements and how they're used. One inch (2.54 centimeters) can be broken down into points and picas, as follows:

- One inch = 72 points
- One inch = 6 picas (72 points ÷ 6 picas = 12 points to 1 pica)

It seems simple enough when you use these measurements to determine spacing between lines (leading) – you can easily use a point scale or a pica scale to measure and verify the number of points of spacing between lines on your hard copy. It can get confusing, though, when you look at font size, which is also described in points (see the "E"s in the previous margin graphic). [The illustration shows the letter E at 11, 12, 14, 18, and 24 points.] You might say to yourself, “That 12-point E doesn't look as if it's 12 points tall!” And you'd be right – it's not. Font sizes are measured completely differently than leading, even though they're both specified in points, and the only way you can verify font size on your hard copy is by measuring it against the designated sizes that you'll find on an E-scale. Most art-supply stores carry precision rules, which combine a pica scale, a point scale, and an E-scale in one handy tool.

**Space between characters**
You can adjust the spacing of letters, or characters, in your text for a single word or an entire paragraph.

- You can lighten the “color” of a text block or improve the readability of small font sizes or sans serif type by increasing the letter spacing.

- You can darken a text block or squeeze a few more words into a circumscribed space by decreasing, or tightening, the letter spacing.

**Space between sentences**
Current publishing convention puts one space, not two, between sentences. You can see the spaces on your screen by choosing Show Special Characters from the View menu. [For more on displaying nonprinting characters in Word see **What do all those funny marks, like the dots between the words in my document, and the square bullets in the left margin, mean?**]. If you habitually press the spacebar twice at the end of every sentence, you can remedy the situation by using the Replace command from the Edit menu.…

The habit of putting two spaces between sentences is a holdover from the days of typewriters when the typed characters were *monospaced* (Courier New is an example of such a font). That is, the space each letter occupied was as wide as the font's widest letter. So, because this created big spaces between some letters, you needed an even *bigger* space to show that you'd reached the end of a sentence. Today, however, most fonts are *proportional* – “skinny” letters take up the least amount of space and “fat” ones the most, and so one space after the period is enough to make the division between sentences clear.

**Space between lines**
Space between lines is to words what air is to people. Publisher automatically puts some vertical space between single-spaced lines, and in most cases this is adequate. But there will be occasions when you'll want to fine-tune it. Too little or too much space makes the text hard to read. Following are some suggestions for changing the line spacing while keeping the text readable.

- If lines are longer than 14 or 15 words, make the line spacing 2 or 4 points larger than the font size. [In Word, you can use Exactly line spacing to add leading; note, however, that Single line spacing is almost always more than the nominal point size to begin with. This varies from one font to another, but for Times New Roman it is approximately 120% of the font size, so that line spacing for 10-point Times is 12 points. For long lines, you would therefore set the line spacing to 14–16 points.]

- Make the line spacing 2 or 3 points larger than the font size for sans serif fonts (like Arial or Comic Sans MS) or dark fonts (like Rockwell).

**Space between paragraphs**
Just as you add a space after the period to separate sentences, you must signal where a new paragraph begins. There are a couple of ways to do this. The first method is the most economical, allowing room for more words on the page; the second adds more white space.

- For a classic, traditional look, indent the first line of each paragraph: 0.15" is a useful standard. It's customary, though, not to indent the first paragraph under a heading because it's obviously a new paragraph. [First-line indents are also thought to work better in combination with justified text, whereas extra space between paragraphs is more effective for ragged-right text.]

- For a contemporary, uncluttered look, don't indent the first line and do leave a space between paragraphs. Don't press the ENTER key twice, though – that generally creates too much space. Instead, adjust the space before or after each paragraph (but not both) – approximately half a line of additional spacing is the norm.

**Line length**
Readability is affected by line length. A very general rule is an average of 10 to 12 words per line for a serif font and 8 or 9 for a sans serif font. [Another rule of thumb is approximately 1½ alphabets or between 40 and 60 characters.]

**Alignment**
You can change the alignment of selected text in a flash. Simply click one of the alignment buttons on Publisher's [or Word's] Format toolbar – the buttons are shown below next to the descriptions of the four ways to align text:

**Flush left**. All the lines align on the left margin, with the right edge uneven, or “ragged,” as in this book. This alignment is recommended for most body text because it's the most readable.

**Flush right**. All the lines align on the right margin, with a ragged left edge. The eye struggles to find the beginning of each line, making reading difficult, so this alignment's uses are limited – for example, right-aligning a caption along the left side of a picture.

**Centered**. Text is not easily readable, but works well in short amounts as in invitations and announcements. It also offers a touch of formality.

**Justified**. Text is aligned on both the left and right margins. Don't justify text if the lines are short or the font is large; it can create unsightly gaps and slow down reading. (If you do justify text in other circumstances, be sure that Publisher's automatic hyphenation is turned on.)

### Adding emphasis

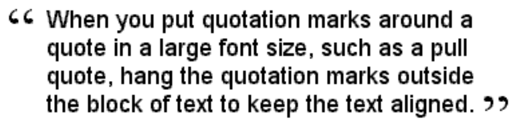
Well-set type is smooth and rhythmic, and adding emphasis should be done in a subtle way, without disrupting the even texture of the page. You can achieve this if you follow the basic guidelines below.

- For emphasis in body text, use italics instead of underlining. A second color often works well. Underlining (a holdover from typewriter days) can cut off the descenders and interfere with readability [it also destroys the cleanness of the overall page design]. Too much boldface creates a distracting “chocolate chip” effect. Use italics instead of boldface or underlining.
- Use all capital letters only in small doses.
- Use SMALL CAPS instead of REGULAR-SIZE CAPS, especially for A.M., P.M., and acronyms like UNICEF or RAM. All caps are difficult to read, they're not as legible for headlines, and they use up a lot of space. Small caps don't distract the eye because they're close in size to body text.
- Set off a section that you want to emphasize, such as a long quotation, with some space before or after the paragraph.
- Italicize book or magazine titles. Use quotation marks around the titles of shorter works like songs or magazine articles.

## Punctuation and special characters

**Apostrophes and quotation marks**

Publisher automatically gives you typographer's apostrophes ( ’ ) and single ( ‘ ’ ) and double ( “ ” ) quotation marks (often called “smart quotes”), rather than foot ( ' ) or inch ( " ) marks. [Word also does this by default.] If you do want inch or foot marks for measurements now and then, that's easy, too: simply hold down the CTRL key while you type the quotation mark or apostrophe. [In Word, press CTRL+Z to reverse the AutoFormat correction.] These contribute greatly to your publication's professional look.…If you'll be typing lots of measurements, it's easy to turn the smart quotes option off [do this in Word on the AutoFormat As You Type tab of Tools | AutoCorrect. Alternatively, you can assign keyboard shortcuts to typographer's foot and inch marks – the prime and double-prime characters found at 0162 and 0178 in the Symbol font and also included in the expanded character set of many Unicode fonts.]



“When you put quotation marks around a quote in a large font size, such as a pull quote, hang the quotation marks outside the block of text to keep the text aligned.”

Figure 7

**Dashes**

There are three kinds of dashes, each a bit longer than the other.

[In the US,] you don't need to put spaces before or after dashes.

- Use the hyphen (-) for hyphenating words.
- Use the en dash (–) where you would use “to,” as in “business hours are 10 A.M.–5 P.M.,” in a range of numbers (pages 17–25), or to link certain compound adjectives like “the Tokyo–Hong Kong flight” or “anti–blood clotting serum.”

- [In the US,] use the em dash (—) instead of parentheses—as is done here—to set off a parenthetical phrase. On the typewriter, two hyphens stood in for this dash, so Publisher cleverly substitutes an em dash when you type two hyphens. [So does Word. Although an em dash is used as described in most of the English-speaking world, it has become increasingly common in the UK to use instead an en dash with a space on either side – like this (and guess where this article was edited!). For that reason, if you type a space, then a hyphen, then another space, Word converts this into a spaced en dash – but not until you finish typing the following word. Also, note that (even in the UK), an em dash is used to denote that speech has been broken off in mid-flow, as in:

  > The Queen turned crimson with fury, and after glaring at her for a moment like a wild beast, began screaming “Off with her head! Off with—”
  >
  > “Nonsense!” said Alice very loudly and decidedly, and the Queen was silent.]

## The look of display text: headings and headlines

Display text – headings and headlines – is meant to catch the reader's eye by being distinctive, evocative, and big: as a general rule, display text is larger than 14 points.

However, display text works differently in a publication such as a book or a report than it does in an advertisement in a brochure or magazine. In the former case, the readers are already involved and the display text helps them to see the organization of the publication and to navigate in it. In the latter case, the display text must capture their imagination so that they'll read the advertisement.

### Choosing a display font

Start by playing around – set the headings in different fonts and see how they feel. Display fonts have a big impact on a publication because of their size, so it's important that they convey the mood you want.

- To attract and hold the reader's attention, there must be a contrast between display text and body text. Create contrast by choosing a headline font from a different category (remember serif, sans serif, and script) than that of your text font – that's what the designers of this book did [they used Britannic Bold for headings and Times New Roman for body text]. Or use only one font in your publication and create contrast by makings the headings bigger and bolder than the text.

- Choose a font that's legible. Because short bursts of text like signs or headlines can't rely on the meaning of the surrounding words for understanding, it must be easy for readers to identify individual letters rather than words and phrases. Many designers consider sans serif fonts to be more legible at large sizes than serif fonts.

- Once you've chosen the heading font, use the same one throughout the publication and use it consistently.

### Formatting display text

In Publisher, you can set display text as plain text or you can use WordArt for display text. [WordArt is also available in Word.]

Use plain, unadorned text for publications such as reports, in which display text helps organize the page, or in elegant or formal publications like invitations or awards. Plain text is very legible, which makes it desirable for critical information like road signs or telephone numbers. Plain text also gives you control over the spacing between letters, which is important when letters are set large. You can achieve the contrast you want by relying on a larger size, a heavier weight, or a different font. And finally, you might have fewer printing problems when you use plain text.

Use Publisher WordArt when you want splashy effects to heighten contrast. You can put white text on a black (or colored) background, pour text into a shape, add shadows to letters, or twist and turn text in a variety of ways. [Some of these effects are also available for plain text in Word.] Once you have placed it in your publication, however, you cannot directly edit the WordArt. To make changes, you have to reactivate the WordArt program by double-clicking the WordArt frame.

You'll want to limit the use of WordArt, though, simply because it is so memorable. Too much of it is like having a dinner entirely of desserts or a birthday every day of the year. A maximum of half a dozen words is a good rule.

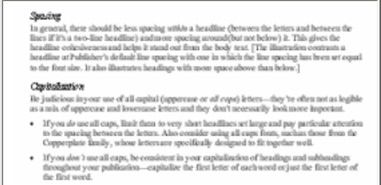
#### Size and weight

To generate impact, set main headings between two and four times larger than body text – even five times larger for greater punch. Let your eye be the judge, but headlines that are much larger and darker than body text attract the reader's attention immediately. [For long documents such as books and reports, headings are not usually so large; Word's default Heading 1 is 14-point Arial, while Body Text is 10- or 12-point Times New Roman; Arial is admittedly heavier than Times, however, with a greater x-height.] You can also use color every effectively in headings.

Another effective way to set off subheads and make their associated text clear is by placing a horizontal line (also called a *rule*) above the heading. [You can do this in Word by applying a top border in the Format | Borders and Shadings dialog.]

Make sure that the size of the heading reflects its importance – the more important the heading, the larger its size. Larger sizes help readers figure out where to look on the page, and smaller subheads pull them along through the story. [The Headings in this article illustrate the point.]

**Spacing**
In general (see Figure 8), there should be less spacing *within* a headline (between the letters and between the lines if it's a two-line headline) and more spacing around (but not below) it. This gives the headline cohesiveness and helps it stand out from the body text.



*Spacing*

In general, there should be less spacing *within* a headline (between the letters and between the lines if it's a two-line headline) and more spacing around(but not below) it. This gives the headline cohesiveness and helps it stand out from the body text. [The illustration contrasts a headline at Publisher's default line spacing with one in which the line spacing has been set equal to the font size. It also illustrates headings with more space above than below.]

*Capitalization*

Be judicious in your use of all capital (uppercase or *all caps*) letters—they're often not as legible as a mix of uppercase and lowercase letters and they don't necessarily look more important.

- If you *do* use all caps, limit them to very short headlines set large and pay particular attention to the spacing between the letters. Also consider using all caps fonts, such as those from the Copperplate family, whose letters are specifically designed to fit together well.
- If you *don't* use all caps, be consistent in your capitalization of headings and subheadings throughout your publication—capitalize the first letter of each word or just the first letter of the first word.

Figure 8

**Capitalization**
Be judicious in your use of all capital (uppercase or *all caps*) letters – they're often not as legible as a mix of uppercase and lowercase letters and they don't necessarily look more important.

- If you do use all caps, limit them to very short headlines set large and pay particular attention to the spacing between the letters. Also consider using all caps fonts, such as those from the Copperplate family, whose letters are specifically designed to fit together well.

- If you don't use all caps, be consistent in your capitalization of headings and subheadings throughout your publication – capitalize the first letter of each word or just the first letter of the first word. [Having said that, in business reports, the usual convention for headings is to Capitalise the First Letter of Most Words, but for Words Such as “the” to be Lowercase.]

- *WHATEVER YOU DO, DON'T SET SCRIPT FONTS IN ALL CAPS!*

## The look of fancy first letters, captions, and other special text

There are other kinds of text that don't readily fall into the body text and display text categories – sidebars (information that might interest a reader, but that isn't essential), fancy first letters that attract the eye to the page, captions with pictures, and pull quotes (excerpts from the body of the story, often set large). Use these elements in moderation, however, or your publication could look tacky and confusing. And once you've chosen the font and style for any of these elements, use them consistently throughout the publication.

### Fancy first letters

It's easy to create a fancy first letter (also called a *drop cap* or *raised cap*) in Publisher. If the font doesn't work with your publication, you can change it to match the heading font, or use another font that complements the style of your publication. [In Word, use Format | Drop Cap to make a drop cap or a large initial in the margin; for a raised cap, merely increase the font size.]

If you have more than one fancy first letter on a page, scatter them and make sure they don't line up along the top of the page. A fancy first letter often takes the place of a heading, so you don't want it at the bottom of the page either.

### Captions and callouts

When you mix fonts in your publication, it's a good idea if the font you use for captions and callouts is related either to the body text font or to the display font, as shown in the following illustration [which shows Arial Bold headings and a smaller Arial Italic for captions and callouts]. You'll notice, though, that the designers of this book used a third font (Blueprint) for the captions and margin notes –proving that it's okay to break the rules sometimes!

### Pull quotes

A pull quote extracts a short statement from the story to add visual interest to the page and to attract the reader's attention.

- Make the pull quote a different font and at least 6 points larger than body text to attract the reader's eye.

- Put the pull quote on the same page (or on the facing page) with the text it was pulled from, but not in the same paragraph. After reading the same thing twice, readers might think you made a mistake.

- Set pull quotes off from the rest of the text with ample white space and a border above or below the frame. An appealing way to call attention to a pull quote is to enclose it in giant quotation marks, as shown in **Figure 7** above. If you do this, be sure to hang the quotation marks outside the even margin of the quote. You can find giant quotation marks in some symbol fonts, or you can increase the point size of quotation marks in the font used for the pull quote; it may be necessary to use Format | Font | Character Spacing to lower the quotation marks if the latter method is used.

**Sidebars**
A sidebar contains information that isn't vital to understanding the main text, but that adds interest or additional information. Sidebar text can be the same font and size as body text or the same font (in a larger size) as the captions. Put the sidebar within a page or two of the related text. Set the sidebar off from the rest of the text by boxing the frame or shading it, or both.

## Using text styles for  fast and consistent formatting

Text styles are formatting descriptions that you can use to quickly apply text formatting on a paragraph-by-paragraph basis. Text styles contain all the formatting information that you need: font and font size, indents, line spacing, tabs, and special formatting such as numbered lists or character spacing. By using text styles, you can make even complex formatting easy by just choosing the style name. And by using text styles for specific purposes – a heading style for headings, a list style for lists, and a body style for body text – you can maintain a consistent design throughout your story, your entire publication, and even multiple publications.

Text styles can come from a variety of sources. Some text styles are created by Publisher, such as the Continued-On Text style, which Publisher uses when a “Continued on page…” notice is automatically included in your text frame. [Word also uses some styles automatically; among these are Header, Footer, Caption, Footnote Reference, and Footnote Text.] You can create other text styles in your word processor as you write your text, and those styles will be included in Publisher when you import the file. You can also define your own styles in Publisher, import text styles from other publications, or use text styles that have been saved in a template. [All of these are true in Word as well, and there are even more good reasons for using styles in Word.]

**See also:**
**Creating a Template – The Basics (Part II)**
**Understanding Styles**.

Printer-friendly version:

# Some of the most useful Word shortcuts

*Or how to save yourself hours by giving your mouse a rest!*

Article contributed by Dave Rado

**(to return to top, press Ctrl+Home)**

Terms of use Disclaimer

This list doesn't attempt to be comprehensive, but is a list of the shortcuts which save me the most time.

You can also get a complete list of keyboard shortcuts by selecting **Tools + Macro + Macros**, where it says “Macros in”, select “Word Commands”, select the command called “ListCommands” and press “Run”. Or you might find the following more useful: **Word commands, and their descriptions, default shortcuts and menu assignments**

## General keyboard time-savers

1. **If you look at the menus**, you will see many of Word's keyboard shortcuts displayed next to the command – for instance, **Ctrl+C** next to Edit + Copy, **Ctrl+V** next to Edit + Paste, **Ctrl+F** next to Edit + Find, etc. Learning and using these shortcuts will save you many hours, allowing you to spend more time with your family! (And many of them work in *all* Windows applications).

   One menu shortcut which is *not* displayed but which I find very useful is **Ctrl+F2** for Print Preview.

   One which *is* displayed but which is so useful and so often missed that it's worth mentioning specifically is **Ctrl+Z** to Undo. Keep pressing **Ctrl+Z** to Undo as far back as you want – if you go too far, press **Ctrl+Y** to redo.

2. **To access the menus** with the keyboard press **Alt** plus the underlined letter on the main menu. Then type the underlined letter in the drop-down menu. E.g. type **Alt+ V, P** to go into Page Layout View, or **Alt+ V, O** to go into Outline View. If you have the mouse in your hand it's quicker to use the mouse (and then the toolbars come into their own), but when touch-typing, accessing the menus with the keyboard saves a lot of time.

3. **To apply or remove Bold**, Italic or Underline press **Ctrl+B**, **Ctrl+I**, or **Ctrl+U**. Use **Ctrl+L** to left-justify text, **Ctrl+E** to centre it, **Ctrl+J** to justify it, and **Ctrl+R** to right-justify it.

4. **To return to your last edit point**, press **Shift+F5**. For instance, if you have copied and want to return to where you were in order to paste. Press **Shift+F5** again to go to up to the last three edit points, or a fourth time to return to where you started.

   Also use this when you first open a document, to go straight back to where you were last editing it.

5. **To change the case of any text**, select the text and press **Shift+F3**. Very useful, for instance, if you have accidentally LEFT YOUR CAPS LOCK ON!

   Keep pressing **Shift+F3** to toggle between ALL CAPS (or “UPPERCASE”), no caps (or “lowercase”), and First Letter In Caps (which Word misleadingly refers to as “Title Case” – a true example of Title Case would be “First Letter in Caps”, but to achieve this level of intelligence you need a macro).

   You get more options if you use the Format + Change Case dialog, though.

6. **You can repeat most commands** and actions by pressing **F4**. This is much more useful than you might think.

   For example, apply borders to a table. Go to your next table, select it, and press **F4** to apply the same borders. (Or do the same with rows within a table).

   Convert a picture from Floating to Inline, then use **F4** to do the same with all other pictures.

   Apply a Style somewhere, then use **F4** to apply the same Style to all other paragraphs in the document which need that Style applied.

   Select one table row, right-click, Insert Rows. Select the new row and **F4**. Select the two new rows and **F4**. Select the four new rows and **F4** – and so on.

   In Word 97, you can use **F4** in combination with the Table + Cell Height and Width dialog to make each column in one table exactly the same width as the equivalent column in another table – a trick I use constantly. In Word 2000 and higher, the Table Properties dialog doesn't support F4, a serious retrograde step; but fortunately you can fix this. See: **How to sidestep the problems of the Word 2000 (and higher) Table Properties dialog** for details.

   Before you start, make sure you can see both tables (split the window if necessary). Then select a cell in one table, select **Table + Cell Height and Width**, choose the "Column" Tab and press **OK**; select a column in the other table and press **F4**. The width you "captured" from the first table will now be applied to the other one. (You can even use this trick if the two tables are in separate documents.)

   You can use the same principle to left or centre-align multiple tables, apply table indents, etc. Apply the formatting you want to one table, using the Cell Height and Width dialog (or if it is already applied, simply display the dialog and press **OK**), and just click in the other tables you want to apply the formatting to and press **F4**.

   I also use **F4** for applying bold to the first couple of words in each item in a bulleted list (easier on the fingers than **Ctrl+B**); for merging cells in several different rows; for making the Page Setup identical in two different sections of a document (see **Working with sections**), or in two different documents – the list of time-saving uses for it goes on and on!

7. **You can repeat the last Find or Goto** by pressing **Shift+F4**.

8. **You can cycle through all open Word documents** by pressing **Ctrl+F6** (or you can cycle backwards by pressing **Ctrl+Shift+F6**.

   In Word 2000 you can also use Alt+Tab, which cycles through all open applications. Word 2000 uses SDI (Single Document Interface) which makes each Word document behave as if it were a separate instance of Word, although it isn't.

9. This one is more esoteric, but very useful if you customise commands a lot, and several people have emailed me with this tip.

   If you have a numeric keypad, press **Alt+Ctrl+Num+** (hold down **Alt** and **Ctrl** and press the **+** key on the numeric keyboard). If you *don't* have a numeric keypad, **assign a shortcut key** of your own to the Word command **ToolsCustomizeKeyboardShortcut**. Either way, when you press the shortcut, the mouse cursor will change into a 4-headed squiggle:
   ⌘

   Now if you press another shortcut key combination, the "Customize Keyboard" dialog will display and show you which command or macro is currently assigned to that shortcut. (for instance, if you press **Ctrl+F4** while the squiggly cursor is visible, the dialog will display the **DocClose** command).

   Alternatively, if you invoke the squiggly cursor and then select any **menu** item, the "Customize Keyboard" dialog will display and show you which command or macro that menu button is assigned to. Unfortunately, this doesn't work for toolbar buttons, but you can temporarily Ctrl+Drag a toolbar button onto a menu (select Tools + Customize first), and then use the squiggly cursor to find out what command or macro the button is assigned to.

   And unfortunately, it works less reliably with custom menu buttons than it does with built-in ones – according to squiggly cursor, several custom buttons that I've assigned to macros are actually assigned to the ToolsMacro command!!! That's a bug.

## Styles

1. **To remove manual formatting**: Press **Ctrl+Spacebar** to remove character formatting. Press **Ctrl+Q** to remove paragraph formatting. These shortcuts return the formatting to the default for the Style in use. To return the selection to the "Normal" style, press **Ctrl +Shift+N**.

   If you've been emailed a document by another company and need to get it into your "Corporate style", and if it contains a lot of manual formatting (as they usually do), print it, and then press **Ctrl+A** (Select All), **Ctrl+Spacebar** and **Ctrl+Q.** If the document uses styles, but the styles are in a mess (as they will be if the author had the default "**Autoformat as you Type**" settings on), press **Ctrl+Shift+N** as well. Then apply styles. Doing this can save you hours per document, literally.

2. **Avoid formatting text manually** as much as possible – use **Styles** instead.

   **But** where you *need* to format manually, you can use **Ctrl+Shift+C** to copy formatting and **Ctrl+Shift+V** to paste it. Having copied formatting, you can use **Ctrl+Shift+V** as often as you like – even across multiple documents – without having to copy again until you close Word.

   If a **paragraph marker** is selected when you copy, this will copy and paste the paragraph formatting; otherwise it will just copy and paste the character formatting.

   You can also use **Ctrl+Shift+C** and **Ctrl+Shift+V** to copy & paste such things as drawing object lines and fills – in both Word and PowerPoint.

   The **Paintbrush** on the Toolbar does more or less the same thing, (although it's much harder to use, and you have to *double*-click on it if you want to apply the same formatting multiple times); and in Excel and Visio, where unfortunately Ctrl+Shift+C and Ctrl+Shift+V don't work, the Paintbrush can be a huge time-saver for things like reapplying cell properties and shape fills.

3. **To create Headings**, hold the **Alt+Shift** keys down, and while keeping them held down, press the **Left** or **Right** arrow on the keyboard – Left arrow to create a main Heading, or promote an existing one, Right arrow to create a subheading or demote an existing one. No need to select anything first, just click in the paragraph which you want to apply the formatting to.

   This one is *very* useful in any View but especially in **Outline View**, as it allows you to promote and demote a large number of Headings at once.

   Alternatively, you can press **Ctrl+Alt+1** to create a Heading 1, **Ctrl+Alt+2** to create a Heading 2, etc. But unfortunately, in Europe, **Ctrl+Alt+4** has been hijacked for the Euro symbol.

   Personally I much prefer the **Alt+Shift** method anyway; easier on the fingers, only one shortcut to remember; and you don't need to think about which Heading Level you want to apply, you only have to think about whether you want the heading to be the same level as the previous one (Left arrow), a higher level (Left arrow twice) or a sub-heading of it (right-arrow). When going through a long document applying headings, this shortcut saves me hours!

   **Alt+Shift+Left** or **Right** arrow can also be used to promote and demote outline numbered or bulleted lists – not just Headings.

   If the Headings don't look the way you want them to, *don't* format them manually! Redefine the Styles instead (Format + Style + Modify).

   Incidentally, some people also like using **Alt+Shift+Up** and **Down** arrows to change the order of their Headings in Outline View, so give that a try. Personally, I prefer using drag and drop.

4. **Never!** use manual page breaks – they're a maintenance nightmare. Instead, on the Format + Paragraph + "Line and Page Breaks" tab:

   **a)** Select "Keep with next" to keep paragraphs together. For example, on the top few rows of a table or the top few paragraphs of a bulleted list, or an inline picture which needs to stay with its caption – although in the latter case it would be better to build this into your Picture style definition.

   **b)** Select "Keep Lines together" to prevent a paragraph from ever being split over two pages (but see also "Widow/Orphan Control", covered below).

   **c)** Select "Page break before" to force a page break where needed – again, build that into your style definitions where possible, e.g. for the Heading 1 style in a long document, and for the style you use for your Table of Contents title.

   "Widow/Orphan control" prevents one line of a paragraph being "orphaned" at the top or bottom of a page – but this is built in to your style definitions by default anyway.

And a last point while on the subject of styles, make sure that under Format + Style + Modify, the "Automatically Screw Up Update" setting is turned off. Unfortunately, it is turned on by default for the List Bullet and TOC styles. Turn them off!

## Moving around and selecting things

1. **To move around a document**:

   **a)** Use **Ctrl+Left** or **Right** arrows to move one word; use **Home** and **End** to move to the start or end of a line; **Ctrl+Home** or **Ctrl+End** to move to the beginning or end of a document, **Ctrl+Up** or **Down** Arrow to move one paragraph, and the **Up** or **Down** arrows to move one line. And of course, **PgUp** and **PgDn** to move one screen, but you knew that one!

   **b)** Press **F5** and in the left-hand pane of the **Goto** dialog, select “Section” to go to the next section, “Table” to go to the next table, etc. Or you can use the browse button near the bottom of the scroll bar, pictured on the right (although I never do!). The browse object changes when you use Find/ Replace or select a different object in the Go To dialog, which I find infuriating (see the **1c** for more on this).

   **c)** By default, **Ctrl+PgDn** is assigned to the **BrowseNext** command, and **Ctrl+PgUp** is assigned to **BrowsePrevious**. If the last thing you used **Goto** for was to go to a table, they will go to the next or previous table.

   Some people like this, because it allows them to go backwards as well as forwards, whereas **Shift+F4** only goes forwards (although you can also use **Shift +F5** to go back). But personally, I find it *infuriating*, because 90% of the time I just want to go to the next or previous page, not to a table, thank you. So I've assigned the GoToNextPage command to Ctrl+PgDn and the GoToPreviousPage command to Ctrl+PgUp (which you can do using Tools + Customize+ Keyboard). I'd be lost without these ones (as would most of my users!).

2. **To select text:**

   **a)** Hold the **Shift** key down and use the key combinations covered in **1a**. For example, to select everything from the insertion point to the end of the document in order to delete it, press **Ctrl+Shift+End** – *much* quicker than any other way! Similarly, **Shift +Up** or **Down** Arrow selects one line, **Shift+PgUp** and **PgDn** selects one screen, and so on.

   And if you've selected too much, just keep the Shift key down while you use the same keys covered in **1a** to de-select what you don't want.

   This works just as well when selecting cells in Excel, incidentally, and is probably even more of a timesaver in Excel than it is in Word.

   **b)** Another good way of selecting text is to click where you want the selection to start, use the scrollbars (or wheelie if you have one on your mouse) to scroll until you can see where you want your selection to end, and with the **Shift** key held down, click again.

   As with **2a**, you can de-select if you've selected too much by keeping the Shift key held down and clicking again.

   **c)** A third method, which I never use but some people swear by, is to either double-click on the status bar where it says “**EXT**”, or press **F8**. This puts you into Extended Selection mode, which means you can use the same shortcuts as in **2a** but *without* having to hold the Shift key down.

   While in Extended Selection mode, you can also press **Enter** to select to the end of the paragraph; or press **period** (**.**) to select to the end of the sentence. Keep pressing **Enter** or **period** to select more paragraphs or sentences. (I got this one from **Beth Melton**).

   And again you can de-select if you've selected too much, but this time without needing to keep the Shift key held down.

   Pressing **F8** twice selects a word, three times a sentence, four times a paragraph and five times the entire document.

   To get out of Extended Selection mode, press **Esc** or double-click on the status bar again.

   **d)** **Double-click** on a word to select it, **triple-click** to select the paragraph. **Ctrl+Click** to select a sentence.

   Or **click once** in the left margin (which MS refers to as the “Selection Bar” in Help) to select one line, **double-click** to select a paragraph and **triple-click** to select the document (although I usually use **Ctrl+A** to select the document).

   **e)** To select a block of text (for instance, to quickly remove manually typed bullets), **either** hold the **Alt** key down while you drag; **or** press **Ctrl+Shift+F8** and then move the arrow keys on the keyboard.

   The latter method works much better than the former if selecting large blocks spanning multiple pages (I picked this one up from the newsgroups).

**f)** The following tip was supplied by Klaus Linke: sometimes you may have selected some text by moving *down* the document, and you may suddenly realise that you want the selection to *begin* further *up* the document (or the other way round); and Word won't let you do this without deselecting and starting again!

You can get around this problem by assigning the following macro to a keyboard shortcut. Press your shortcut to change the selection direction (so that for example, if you can't currently extend the start of the selection upwards, you *will* be able to do so after pressing the shortcut); then press the same shortcut again to reverse the direction back to how it was:

```
Sub ChangeSelectionDirection()
    Selection.StartIsActive = Not (Selection.StartIsActive)
End Sub
```

**3.** **To move and select within a table:**

**a)** **Alt+Mouse Click** selects a column. **Alt+Double-click** selects the table.

**b)** **Alt+PgDn** goes to the bottom of a column, **Alt+PgUp** to the top. Press **Shift** as well, to *select* to the top or bottom of the column.

**c)** To select a row, click in the left margin of the document; drag down or up to select multiple rows.

**d)** To change the order of your rows, you can use the **Alt+Shift+Up** and **Down** arrows (no need to select anything). Or you can drag and drop.

**4.** **To move paragraphs of text without resorting to cut and paste**, you can use drag and drop. If you are moving paragraphs to a position that is off-screen, split the window first (Window + Split).

Alternatively, if you are in Page/Print View or Normal View, and your cursor is **not** in a table, you can move the current paragraph(s) up or down the document using the **Alt +Shift+Up** and **Down** arrows. Whereas in Outline View, this moves Headings and all their subsidiary text, in Page/Print View or Normal View it just moves the current paragraphs (but in a table it moves the current rows instead).

**5.** **To edit text while you're in Print Preview**, click on the page to zoom in, then click on the Magnifier button on the Print Preview toolbar to switch into edit mode. Click on the Magnifier again when you want to quit edit mode and zoom back out.

**Word of warning:** Be careful **not** to type unless you are in Edit mode! Word lets you do this, and because there's no visible insertion point, you will have no idea where the text you type is going to be inserted! This is a bug.

Oh, and one last thing – don't forget about your **right mouse button**!

# Word for Windows commands, and their descriptions, default shortcuts and menu assignments

Article contributed by Dave Rado

Word has a built-in command *ListCommands*, which produces a table of all the Word commands with their current key and menu assignments. However, it does not list the commands using their actual names; nor does it include descriptions of what the commands actually do.

Various sites on the web list the default shortcuts, but again, most don't list the commands using their correct names; most don't list those commands which are not assigned to a shortcut by default – but which you might well want to assign to one, or to assign to a menu; or which you might want to intercept using a macro; and some don't list full descriptions of what the commands do.

**WordCommands.zip** (50k) extracts to an Excel file, **WordCommands.xls**, which contains a list of **all** interceptable Word commands (Word 97 and above), using their correct English names; and with various filters applied to the spreadsheet so that you can easily switch between viewing all commands; or only those assigned to a shortcut by default; or only those assigned to either a shortcut or a menu by default (select View + Custom Views, then select a view from the list).

WordCommands.zip

WordCmndsPDF.Zip

Get Acrobat Reader

If you don't have access to Excel, **WordCmndsPDF.zip** (136k) contains 3 PDF files, one for each of the Custom Views.

Each command is listed with a description of what it does; in many cases a much fuller description than that which Word itself displays when you scroll through the commands (by selecting Tools + Macros + Macros, and selecting "Word commands" under "Macros in").

The hyperlinks at the top right of the sheet are to help you with navigation.

If you just want to know about the most useful shortcuts, and why they are so useful, see
**Some of the most useful Word shortcuts**

If you want to assign some of the unassigned commands to a menu or shortcut, see:
**How to assign a Word command or macro to a hot-key**
**How to assign a Word command or macro to a toolbar or menu**

If you want to create a macro to intercept one of the commands, see:
**Intercepting events like Save and Print**

# How can I insert special characters, such as dingbats and accented letters, in my document?

Article contributed by Suzanne S. Barnhill

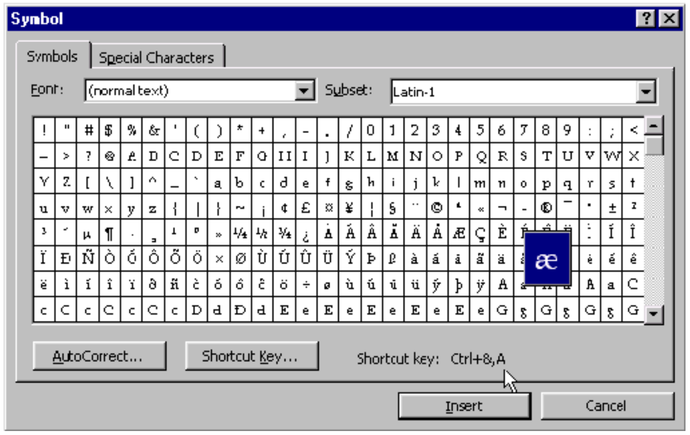
Terms of use Disclaimer

Many Word users don't realize how easy it is to insert special characters. There are at least four ways to do it: through the **Symbol dialog**, using **shortcut keys**, automatically with **AutoCorrect**, or by **direct keypad entry**.

## The Symbol dialog

If you choose Symbol… on the Insert menu, you will bring up the Symbol dialog, shown below. (If you have a slow system and/or one with many fonts installed, you may find that this dialog takes an appreciable time to appear the first time you use it in a Word session, but after that it should pop up instantly.)

In the font list in the Symbol dialog, "(normal text)" means the font you are currently using. For more information about the other fonts listed, see **Fonts in the Symbol dialog** (below).


Font: (normal text)
Subset: Latin-1
Shortcut key: Ctrl+&,A

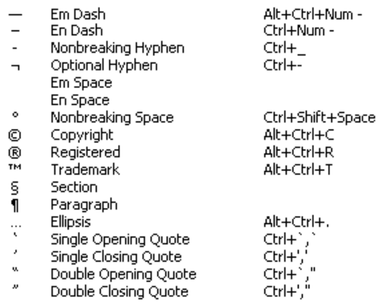
To insert a character, double-click on it, press Enter, or click the Insert button. The dialog stays open so that you can insert more than one character, and you can “step out” of the dialog to move the insertion point before choosing another character and inserting it.

## Shortcut keys

Word has also made it very easy for you to insert many of these characters without recourse to the dialog - in particular special characters such as ® and international characters such as é. It does this through built-in shortcut keys. When you select a character in the dialog to which a shortcut key has been assigned (either by Word or by you, the user), it is displayed at the bottom of the dialog. The characters to which Word has assigned shortcut keys are broadly categorized as either “special characters” or “international characters.” Memorize the shortcuts for the characters you use often, use the dialog for the rest.

### Special characters

Note that the Symbol dialog has two tabs: “Symbols” and “Special characters.” The latter both lists the shortcut key (if any) for each of a variety of characters and lets you insert it directly (by selecting it and double-clicking or pressing Insert). The list is as follows:



| | | |
|---|---|---|
| — | Em Dash | Alt+Ctrl+Num - |
| – | En Dash | Ctrl+Num - |
| - | Nonbreaking Hyphen | Ctrl+_ |
| ¬ | Optional Hyphen | Ctrl+- |
| | Em Space | |
| | En Space | |
| ° | Nonbreaking Space | Ctrl+Shift+Space |
| © | Copyright | Alt+Ctrl+C |
| ® | Registered | Alt+Ctrl+R |
| ™ | Trademark | Alt+Ctrl+T |
| § | Section | |
| ¶ | Paragraph | |
| … | Ellipsis | Alt+Ctrl+. |
| ‘ | Single Opening Quote | Ctrl+`,` |
| ’ | Single Closing Quote | Ctrl+',' |
| “ | Double Opening Quote | Ctrl+`," |
| ” | Double Closing Quote | Ctrl+'," |

In the above list, note the following:

- In the shortcut keys for the em and en dashes, “Num -” means the minus sign on the numeric keypad, as opposed to the hyphen on the top row of the keyboard (that is the key used in the shortcuts for the nonbreaking and optional hyphens). If you are using a laptop computer that doesn't have a numeric keypad or for some other reason don't have easy access to the numeric keypad, you might want to assign different keyboard shortcuts to these symbols.
- In the shortcut keys for the various quotation marks, ` (accent grave) is the key at the top left of your keyboard (it also has the tilde ~ on it); ' and " are the apostrophe and shifted apostrophe (quote). These keyboard shortcuts use what is called a “setup key.” The comma in the shortcut shows that you press, say, Ctrl+` and release. The status bar will display the combination you have pressed. You then press the remaining character. (As you will see, this technique is widely used in producing international characters.).
- Other useful shortcuts that are not included in this list are Ctrl+@, Spacebar to produce the degree symbol (°) and Ctrl+/, c to produce the cent sign (¢).
- On the Symbols tab, under “(normal text)”, there are a number of fractions, which you can assign to shortcut keys if you don't want to use the **Autoformat** method of inserting fractions.

You may wonder why some of these shortcuts are needed. For example, if you have “smart quotes” enabled on the AutoFormat As You Type tab of Tools | AutoCorrect, you will get these characters automatically. But sometimes Word guesses wrong and gives you ” when you want “; and Word always gets it wrong when you need two opening quotes in a row. In such cases, it is convenient to be able to force Word to give you what you want.

Note that there are no assigned shortcut keys for some of the characters. You can assign your own shortcuts if you like; for example, I have Alt+Ctrl+M and Alt+Ctrl+N assigned to the em and en spaces. To assign a shortcut, just select the desired symbol and press the Shortcut Key… button. The Customize Keyboard dialog opens with the insertion point in the “Press new shortcut key” box. Just enter the key combination you want to use and press Assign. If you want this shortcut to be available in all your documents, press Close. If you are using a template other than Normal.dot and want the shortcut key available only in documents based on that template, select it in the “Save changes in” list before closing the dialog.

You can use this same technique to assign a new shortcut to a character (even if Word already has a built-in one). The one you assign will take precedence over the built-in one. If you later decide you don't need this shortcut, select it in the “Current keys” list in the Customize Keyboard dialog and press Remove. Word will then revert to the built-in shortcut.

## International characters

Word also provides built-in shortcuts for many of the accented and other special characters needed to type foreign words. If you are using a language other than English exclusively or primarily, there are more efficient ways to type (for more information on this, see Word's Help under “characters, international”), but for the occasional foreign (or domesticated) word that needs an accent, these shortcuts are very handy. Word provides a complete list of these shortcuts in the Help article “Type international characters” (reached via “international characters, type international characters” or “characters, special, type international characters”). The list is as follows:

| To produce | Press |
|---|---|
| **à, è, ì, ò, ù**<br>**À, È, Ì, Ò, Ù** | Ctrl+` (accent grave), the letter |
| **á, é, í, ó, ú, ý**<br>**Á, É, Í, Ó, Ú, Ý** | Ctrl+' (apostrophe), the letter |
| | |

| | |
|---|---|
| **â, ê, î, ô, û<br>Â, Ê, Î, Ô, Û** | Ctrl+Shift+^ (caret), the letter |
| **ã, ñ, õ<br>Ã, Ñ, Õ** | Ctrl+Shift+~ (tilde), the letter |
| **ä, ë, ï, ö, ü, ÿ<br>Ä, Ë, Ï, Ö, Ü, Ÿ** | Ctrl+Shift+: (colon), the letter |
| **å, Å** | Ctrl+Shift+@, a or A |
| **æ, Æ** | Ctrl+Shift+&, a or A |
| **œ, Œ** | Ctrl+Shift+&, o or O |
| **ç, Ç** | Ctrl+, (comma), c or C |
| **ð, Ð** | Ctrl+' (apostrophe), d or D |
| **ð, Ð** | Ctrl+' (apostrophe), d or D |
| **ø, Ø** | Ctrl+/, o or O |
| **¿** | Alt+Ctrl+Shift+? |
| **¡** | Alt+Ctrl+Shift+! |
| **ß** | Ctrl+Shift+&, s |

Note that in the above shortcuts, unlike many of the others, you get a different symbol depending on whether the combining letter is capital or lowercase.

## AutoFormat and AutoCorrect

Many symbols are or can be entered in Word automatically through the action of AutoFormat and AutoCorrect.

### AutoFormat

We have already mentioned the “Replace as you type” option to replace “straight quotes” with “smart quotes.” Other options are to replace “Fractions with fraction characters” and “Symbol characters with symbols.” The example given for the latter is replacement of -- (two hyphens) with a dash. Note that this works only when the two hyphens are not preceded or followed by a space. If you include spaces, they may sometimes be converted to an en dash. On the other hand, a hyphen is not converted to an en dash (even in many places where it would be appropriate) unless it is preceded and followed by spaces (and the spaces remain around the en dash), so keyboard shortcuts may still be needed for ultimate control. And remember that whenever Word converts anything you type into something you don't want, you can reverse just the AutoCorrect or AutoFormat with Undo (Ctrl+Z).

### AutoCorrect

Many special characters are defined as AutoCorrect entries. Since these all sort to the top of the AutoCorrect list, it is easy to review them. They are also summed up in this list, found in Word's Help under the topic “Create arrows, faces, and other symbols automatically” (“symbols, creating automatically”):

| Type | To create |
|---|---|
| (c) | © |
| (r) | ® |
| (tm) | ™ |
| ... | … |
| --> | → |
| :) or :-) | ☺ |
| :\| or :-\| | 😐 |
| :( or :-( | ☹ |
| <-- | ← |
| <== | ⬅ |
| <=> | ⇔ |
| ==> | ➔ |

Note that some of these (such as ©, ®, ™) overlap Word's built-in shortcut keys. This gives you more than one way to accomplish the same thing. Also, the shortcut keys give you backup in case you want to delete the AutoCorrect entries. For example, perhaps you often create lists beginning with (a), (b), (c), and you get tired of having the list become (a), (b), ©. So you delete the AutoCorrect entry for (c). But you can still create © using Alt+Ctrl+C.

Note also that the remaining entries (the "dingbats") are characters from the Wingdings font. They can be entered from any font and will not change if you change fonts.

You can create an AutoCorrect entry for any special character. Just select the character in the Symbol dialog, press the AutoCorrect… button, and type the combination of letters or symbols you want to be replaced by the given character. Note that the entry is stored as "formatted text" and therefore will be entered in the selected font regardless of what font you are using in your document.

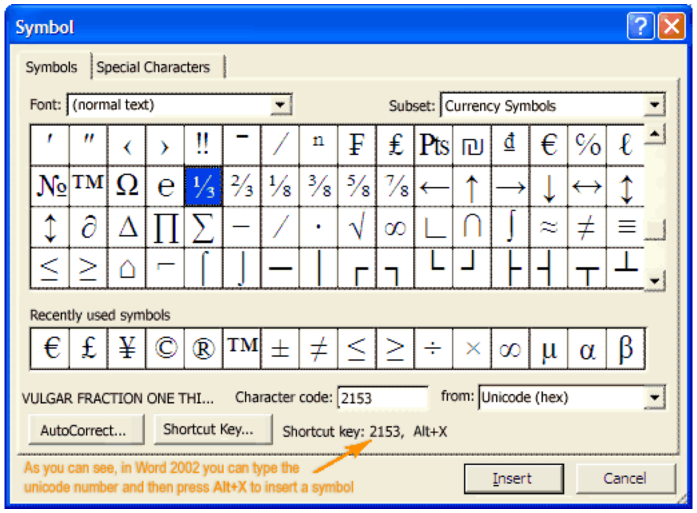
## Direct keypad entry

The oldest way to insert special characters in Word, and still one of the most dependable, is to enter the character number on the numeric keypad. The 256-character ANSI character set actually contains about 224 "characters"; the first 32 positions (character numbers 0–31) are reserved for other keyboard functions and printer control commands such as Escape, Backspace, Tab, Line Feed, Carriage Return, and so on. If you know the number of the character you want, you can enter it by pressing the Alt key and typing the number, preceded by enough leading zeroes to pad it to four digits, on the numeric keypad. For example, to insert the ¥ character, you would enter Alt+0165. The advantage to this method is that it works in virtually any Windows application, not just Word.

But how can you find out the number of the character in question? If you select **Insert | Symbol** in Word 97 and above, this information is available from the status bar. When you select a character in the Symbol dialog, the status bar displays (for example) "Insert Times New Roman character 165."

In **Word 2000**, it also displays the Unicode number for character numbers 160 and above – for example, "Insert Times New Roman character 165, (Unicode: 00A5).". For character numbers greater than 255, it displays the Unicode number only, and not the character number (unfortunately).

**Word 2002** displays the Unicode or character number (your choice) in the dialog itself:



In Word 2002, you can insert characters directly from the keyboard if you know the Unicode number, by typing the Unicode number and pressing **Alt+X** (this also works in certain dialogs, such as Find and Replace).

Another way is to use the Windows Character Map. This applet is one of Windows' Accessories and can be found at **Start | Programs | Accessories | Character Map** in Windows 95 and at **Start | Programs | Accessories | System Tools | Character Map** in Windows 98. If you don't find it at either of those places on your system, you can use Windows Find to search for Charmap.exe. You can then create a shortcut to that file from someplace easily accessible. If you use it a lot, you may want to put it directly on the Start menu (by placing a shortcut in the Windows\Start Menu folder).

## Some other points worth noting

### Fonts in the Symbol dialog

The first time you use this dialog, the Font box will probably be displaying “(normal text).” That means the characters that will be inserted will come from the font you are currently using. Moreover, if you change the font of your document, the character you inserted will be changed to the new font.

If you scroll down the font list, you will see quite a lot of other fonts listed, but not all the fonts you have installed (that is, not all that are listed in Word's main Font list). The fonts presented in this dialog (aside from “(normal text)”) are supposed to be “decorative” fonts – that is fonts whose character set is different from that of the standard alphanumeric font (the ASCII or ANSI character set). These are often called “symbol,” “dingbat,” or “pi” fonts. Two of the “Windows core fonts” – Symbol and Wingdings – are such fonts, and are by far the most frequently used (Zapf Dingbats is another commonly used one). If you have Internet Explorer installed, you probably also have Webdings. Word, Office, and other Microsoft applications install other fonts of this type, and others may come with your printer.

But you may see fonts listed whose character set is identical to, say, Times New Roman (though the letters may be very ornamental, they are not “decorative” in this specific sense). And you may have “dingbat” fonts that are not listed. There is evidently a marker in font files that tells Word whether or not to include them in this category; some fonts have it unnecessarily, and some qualifying fonts are missing it. But you can force any installed font to appear in the list: just type in the font name exactly as it appears in Word's font list and press Enter or click anywhere in the character grid. The characters in that font should then appear. (Occasionally all the characters in a font will appear in the Symbol dialog as squares. This problem may or may not be solved by updating your display driver.)

### How Word deals with symbols when you change fonts

There is a difference in the way Word treats the characters you insert from the Symbol dialog. As already stated, if you insert a character from “(normal text),” it is treated as interchangeable with the same character in any other font. This should not cause problems unless you change to a font that does not include these characters. For example, some older, cheaper fonts contain only the characters that can be entered from the keyboard and perhaps a few others. If you have inserted an accented letter that is not included in that font, it will be displayed and printed as a small square. Also, the new Unicode versions of Windows core fonts contain many more characters than the standard ANSI character set, including characters such as:

♫

These also will not will translate properly to older fonts that contain only the ANSI characters. Keep this in mind in deciding whether to insert a symbol that is part of the extended character set in Times New Roman or Arial or to use the same symbol from, say, the Symbol or Wingdings font.

When you insert a symbol from Symbol or Wingdings, Word treats it differently from a "(normal text)" character. In earlier versions of Word, these symbols were inserted as Symbol fields, which protected them from being updated when the font was changed. In newer Unicode-aware versions of Word, these characters are recognized as being different by having different glyph numbers from the standard character set (more on this later). If you insert one of these symbols and change the font of the paragraph it is in, it will not be changed. But if you insert a character from one of the fonts whose character set is the same as that of "(normal text)" (that is, one of those fonts that shouldn't be in the list to begin with), Word recognizes this and will change it to a new font whether you want it to or not. Unfortunately, this also applies to the bona fide symbol fonts that Word has not seen fit to include in the font list. (For a way around this, see **How to protect symbols from updating when you apply a different font to a paragraph**, below.)

## What is Unicode?

A complete explanation of Unicode is beyond the scope of this article, but a rudimentary knowledge of it is helpful in understanding how fonts work in recent versions of Word. More information on the Unicode standard can be found in the **Unicode Introduction** at the Agfa Monotype Corporation Web site. (Agfa Monotype Corporation supplies many of the fonts distributed with Microsoft products, including Times New Roman and Arial; and they co-develooped the Arial Unicode font with Microsoft.)

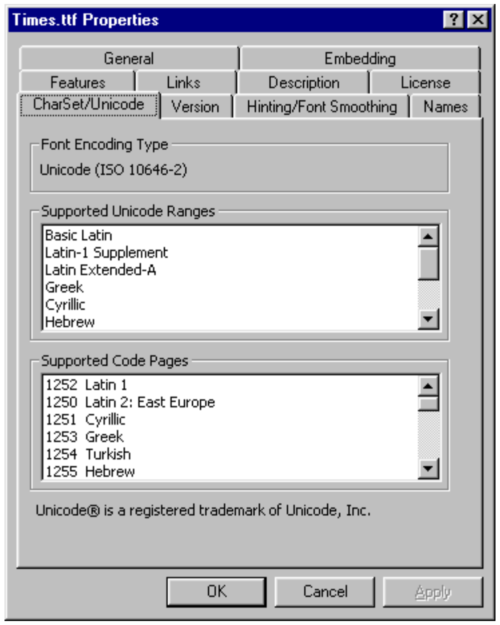
There is also a good **Unicode Introduction** on the unicode.org site.

According to the former article:

> Unicode is a worldwide character encoding standard designed to enable the global interchange of multilingual digital information. The inventors of Unicode had the goal of supporting all the world's scripts while accommodating existing national and international character sets.
>
> Most computer users in the West are accustomed to character sets based on the Latin-1 standard (ISO 8859 series), which contains only Latin-script characters for Western Europe. While Latin-1 supports about 200 characters, Unicode supports 65,000 characters.

According to this source, "A base-level Unicode-based conformant font would include: Pan-European Latin, Cyrillic, Greek, Hebrew, and Arabic." If you look at Times New Roman or Arial in the Symbol dialog (provided you have the Unicode-based versions), you will see that they do indeed contain these characters. These character sets comprise 1,140 or so of the 65,000 characters supported by Unicode.

You can get specific information about which character sets a font contains by installing the **Font properties extension**, which you can download for free. Once you've installed it, right-click on any font displayed in the Fonts folder (c:\windows\fonts) and choose Properties. The "CharSet/Unicode" tab displays whatever information is available in the font. The Properties for Times New Roman are shown below.


Times.ttf Properties

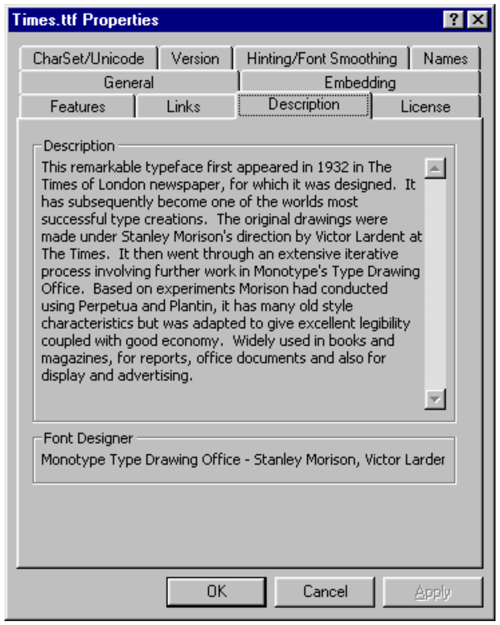
Font Encoding Type
Unicode (ISO 10646-2)

Supported Unicode Ranges
Basic Latin
Latin-1 Supplement
Latin Extended-A
Greek
Cyrillic
Hebrew

Supported Code Pages
1252 Latin 1
1250 Latin 2: East Europe
1251 Cyrillic
1253 Greek
1254 Turkish
1255 Hebrew

Unicode® is a registered trademark of Unicode, Inc.

As an aside, the other tabs the Font properties extension gives you access to are also very interesting – for instance, this is the “Description” tab for Times New Roman:


Times.ttf Properties

Description
This remarkable typeface first appeared in 1932 in The Times of London newspaper, for which it was designed. It has subsequently become one of the worlds most successful type creations. The original drawings were made under Stanley Morison's direction by Victor Lardent at The Times. It then went through an extensive iterative process involving further work in Monotype's Type Drawing Office. Based on experiments Morison had conducted using Perpetua and Plantin, it has many old style characteristics but was adapted to give excellent legibility coupled with good economy. Widely used in books and magazines, for reports, office documents and also for display and advertising.

Font Designer
Monotype Type Drawing Office - Stanley Morison, Victor Larder

The Unicode standard distinguishes between a “character” (such as a particular letter of the alphabet) and a “glyph” (the rendering of it in a particular font). “A character set [is] an ordered collection of characters, while a font is an ordered collection of glyphs.” The characters are considered interchangeable. Therefore, if you select a character in the “(normal text)” display in the Symbol dialog and press the Shortcut Key… button, the Customize Keyboard dialog displays, “Inserts the x character.” It says this even when it is incapable of displaying the character, in which case it shows: “Inserts the ? character.”

But when you select a symbol from one of the other (“decorative”) fonts in the dialog and press the Shortcut Key… button, the dialog displays, for example, “Wingdings: 61649,” where 61649 is the Unicode number of the character. Note, however, that this number is not a unique identifier of that specific symbol in that particular font. All symbol fonts use the Symbol character set, which has the range 61472–61695. So this number will be the same for the symbol in the same position in the character set in any symbol font.

In Word 97, there is no way in to either ascertain the Unicode number of characters in the basic character set or to enter them manually through the numeric keypad as you can the ANSI character set.

As discussed above, in Word 2000 you can ascertain the Unicode number from the status bar, but you cannot insert an upper Unicode character directly in the document using that number (without resorting to a macro).

In Word 2002 you can insert Unicode characters from the keyboard using Alt + X.

### How to protect symbols from updating when you apply a different font to a paragraph

As noted earlier, you can force the Symbol dialog to display any font, but Word does not recognize as "decorative" any fonts that it doesn't list, and so characters from these fonts are not protected from updating. For example, you can insert a hedera (vine leaf) from the Minion Ornaments font (a Type 1 PostScript font that is not listed), but if you change the font of an entire paragraph, the character will become an n. One way (possibly the only way) to prevent this from happening is to insert the character as a Symbol field. The syntax for this field is { SYMBOL 0xxx \f "Font Name" }. To enter the vine leaf character in this way, then, you would insert this field: { SYMBOL 0110 \f "Minion Ornaments" } –- assuming you had the Minion Ornaments font installed. You can also enter the Unicode character number using the \u switch, but you still have to specify the font since the Unicode numbers are the same for all fonts using the "Symbol character set."

### Printing problems

Sometimes the symbols appear correctly on the screen but have the wrong character or a box when printed. Sometimes this can be fixed by changing settings in the printer driver (e.g. to print as graphics or by changing font substitution settings).

## Related articles

**Finding and replacing non-printing characters (such as paragraph marks), other special characters, and text formatting**

**Finding and replacing symbols**

**Inserting Hebrew vowels in a document, using the Hebrew language version of Word**

# Setting tabs

Article contributed by Suzanne S. Barnhill and Dave Rado

Terms of use Disclaimer

Most Word users who are old enough to have used a typewriter will have had some experience in setting tab stops in order to position text to line up accurately without having to use multiple spaces. In Word (or any other word processor), this becomes even more vital because in most cases you will be using *proportional* fonts. Most typewriters produce text in which every character (including the space) is the same width; the equivalent in word processing terms is a *monospaced* font, such as Courier New. Although it is possible, using a monospaced font, to line up text using spaces alone, tabs are much more efficient. Tabs are also the only way to accurately line up text in a proportional font because the characters have varying widths.

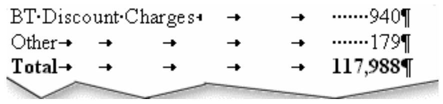
### Why set them?

By default, a Word document has built-in tab stops at half-inch intervals. You can change the default spacing in a given document using the spin box in the top right corner of the **Format | Tabs** dialog, but in general it is preferable to avoid using the built-in tab stops at all.

Have you ever tried pasting from someone else's documents into your own, or even just changing the page margins, or the font; and found that their tabbed lists no longer line up—so you have to waste a lot of time reformatting them? If so, it's because the person who created the document didn't use tabs properly.

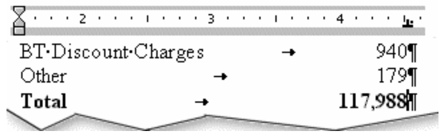
If you use the built-in tabs, (and even, as many people do, use the space bar to simulate right-aligned tabs); and therefore end up tabbing once on some lines, more on others, depending on how much text you're typing on a given line, then the tab positions will be determined by the document's margins and by the font in use.

The golden rules when it comes to using tabs are: do set the tab positions yourself, and don't press the tab key more than once between any two blocks of text: set a tab position instead. You can easily see whether a tabbed list has been created properly by clicking the Show/Hide ¶ button , so you can see where the tabs are.



BT·Discount·Charges→ → → ·······940¶
Other→ → → → → ·······179¶
**Total**→ → → → → **117,988¶**

**Figure 1: How not to use tabs**
Multiple tabs between the text blocks, spaces have been used to simulate right-aligned tabs – this list will go all over the place if the page margins or font are ever changed, or if the list is ever pasted into another document



BT·Discount·Charges → 940¶
Other → 179¶
**Total** → **117,988¶**

**Figure 2: These tabs have been set properly**
A left indent and a decimal tabstop have been set on the ruler, there are no redundant tabs or spaces in the document – this list will be very easy to maintain

## Types of tabs

The Tabs dialog (accessed via **Format | Tabs** in Word 2003 and earlier) lists five kinds of tab stops, as follows:

- **Left.** This is the type you are probably most familiar with, the one you get by default when you press the Tab key. Text is left-aligned at the tab stop position.
- **Center.** When you tab to a Center tab stop, the text you type is centered at the tab

stop position.

- **Right.** Text is right-aligned at the tab stop position.
- **Decimal.** This is a variant of the right-aligned tab stop. Text is aligned on the first non-numeric character (apart from the thousands separator) following a series of numbers. This type of tab stop is commonly used to align numbers with varying numbers of decimal places, but a decimal point is not required since the text is aligned where the decimal point would be if there were one. This makes it possible to align a mixture of negative numbers in parentheses and positive numbers without parentheses, numbers followed by footnote references (letters or symbols), or any other type of number followed by a non-numeric character.
- **Bar.** Bar tabs, which aren’t really tabs at all but have been included in this dialog because some Microsoft designer found it convenient, are an almost totally undocumented feature in Word. They can be very useful occasionally, but only you will be able to figure out when they might be useful for you. When you set a bar tab, you get a thin vertical line at the tab stop position in each line of a paragraph. You don't actually have to tab to it; it's there all the time, and it extends to the full height of the text line, making a solid line throughout the paragraph. It doesn't take the place of a paragraph border or cell borders in a table (although its original purpose was to simulate cell borders in tables), but every now and then it's just what you want.

## How to set them

Like many tasks in Word, setting tabs can be done in more than one way. Which one you use will mostly depend on which is most intuitive for you.

The **Tabs** dialog is the ultimate way to set tabs. Here you can type in an exact figure (down to hundredths of an inch), choose the kind of tab you want, and set a tab leader if desired. It is the only way to set a tab leader, which is typically a dotted line (period leader) filling the space taken up by the tab, used most commonly in tables of contents, but there are four different choices of tab leader available. It is also the only way to set a bar tab in Word 97 and earlier versions.

***Note:*** It is far from obvious how to access the Tabs dialog in Word 2007. The Help file will tell you to double-click on any tab stop on the ruler (see below); if you are using an unpatched copy of Word 2007, this is the ***only correct way*** to do it, even if you have to set a tab stop temporarily in order to use this method. The alternative method Help suggests is to open the **Paragraph** dialog (using the dialog launcher in the **Paragraph** group on the **Home** or **Page Layout** tab) and click the **Tabs…** button. The “gotcha” (in the original release of Word 2007) is that, if you access the Tabs dialog this way, the settings you choose (even though they may be intended for a single paragraph) will be set as the document defaults *and* will be applied to the Normal template (Normal.dotm). This egregious bug was corrected by Service Pack 1, so if you have applied SP1, this will not be a problem.

Unless you want to set a bar tab or a tab leader, however, it is rarely necessary to visit the Tabs dialog. Usually the easiest way to set tabs is using the ruler bar. If you do not have the ruler displayed, check that item on the **View** menu (check the Ruler box in the **Show/Hide** group on the **View** tab in Word 2007). On the left side of the ruler is a button with an icon that looks like an **L**.

This button shows which kind of tab stop is currently selected; the default is a left tab. Click anywhere on the ruler, and you will have set a tab stop; you will see a small L on the ruler. By default, the tab stop will be set at one of the ruler markings or halfway between them. If your ruler display shows inches, this means that tabs can be set only at intervals of 1/16" (which is usually close enough) *unless* you press the **Alt** key while dragging the tab marker; when you do this, you will see measurements displayed, and you can set tab stops just as precisely as in the Tabs dialog.

***Note:*** The built-in tab stops (at half-inch intervals by default) are not shown on the ruler in any way, so it may not be obvious to you that whenever you set a custom tab stop, all the built-in tab stops to the left of it are deleted so that when you press Tab you go directly to the tab stop you set. The built-in stops to the right of your custom stop remain.

At least three other types of tab stops can be set using the ruler. If you click on the **L** button, it will change to the icons that represent **center**, **right**, and **decimal** tab stops. These are, respectively, an upside-down **T**, a backwards **L**, and an upside-down **T** with a dot (see Figure 2 for an example of a decimal tab stop). When you have reached the kind you need and click on the ruler bar, you will place that type of tab stop.

In Word 2002 and above, the bar tab has also been added to the rotation, along with buttons for first-line and hanging indents. (My personal feeling about the last two is that they are entirely superfluous—it is much easier just to drag the corresponding markers on the ruler—and rather tricky to use, not to mention that they result in requiring additional clicks on the button to get back to the left tab stop.)

You can move these tab stops as needed; just click on one of the markers and drag it where it is needed. This is very handy if you want to set up a simple table using tabs: you can enter data such as the following:

Item**<tab>**Item**<tab>**$Number
Another item**<tab>**A very long item**<tab>**number
And so on.

After you've finished the whole list and can see where the tab stops need to be, you can select the entire block and place the tab stops as needed.

***Note:*** Like every other kind of paragraph-level formatting, tab stops in Word affect just the paragraph in which they are set and any other paragraphs that may later be created by pressing **Enter**. This means that you can set tab stops in a paragraph while you are writing it and keep those same settings for as long as you keep writing, until you change them in another paragraph, **but**, if you have already entered text and go back and set tab stops, they will affect only the paragraph where the insertion point is located **or** any block of text you have selected. This is really a very powerful tool, but it is not always intuitive for beginners.

One of the things I especially like about Word is that you can set tab stops beyond the right page or paragraph margin, which makes it very easy to do, say, invoices with a multi-line entry that wraps short of the right page margin (because the paragraph has a right indent), while the money amount is at a right tab stop at the right page margin (beyond the paragraph margin). This is also useful for long entries in tables of contents.

## Tabs and Tables

You might think that using tables would be a substitute for tabs, and to a large extent it is, but you can also set tab stops in tables; the trick is that you have to use **Ctrl+Tab** to get to the tab stop, because **Tab** alone takes you to the next table cell. Also, if you set a decimal tab in a table, the cell contents (which must be left-aligned), will jump to that tab automatically: you don't have to enter a tab character.

A borderless table has one other major advantage over a tabbed list—the text in it can word-wrap. You may very well start out using a tabbed list, then realize halfway through that you need the text to word-wrap. *Provided* you have used tabs properly, this is almost a one-step process: on the **Table** menu, select **Convert | Text to Table**, confirm the desired number of rows and columns in the ensuing dialog, press **Enter**, and you're done (in Word 2007, find the **Convert Text to Table** command by clicking the **Table** button in the **Tables** group on the **Insert** tab). But if you tabbed more than once between any two blocks of text, the resulting table will be a mess.

## First-line indents

One more word of advice about tabs: If you are accustomed to indenting the first line of a paragraph using a tab, don't do it. The proper (and timesaving) way to do this in Word is to use a first-line indent. You can set this in the **Format | Paragraph** dialog (accessed via the dialog launcher arrow in the **Paragraph** group on the **Home** tab in Word 2007) or by dragging the first-line indent marker on the **ruler bar**. This is the top triangle of the three buttons to the right of the tab selector button. If you hover your mouse over it, the ScreenTip will say "First Line Indent." (The other two are for Left Indent—there's a corresponding Right Indent on the other side—and Hanging Indent.)

# Ruler of all you survey: How to make the best use of Word's rulers

Article contributed by Suzanne S. Barnhill and Dave Rado

Terms of use Disclaimer

## Overview

The way in which most users set a hanging indent is as follows. They type until they get to the beginning of the second line of the paragraph, and they press **Tab**. Then they type to the beginning of the next line and press **Tab**. And so they continue. They end up with paragraphs which (with non-printing characters displayed) look something like this:

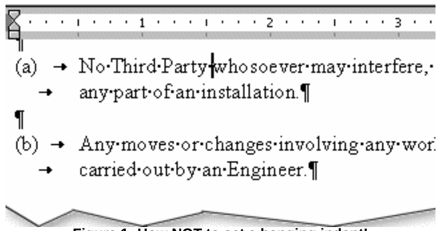
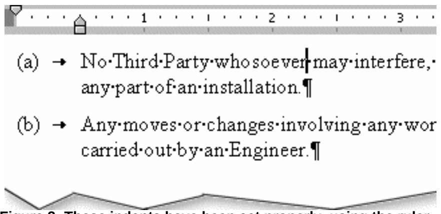

(a) → No·Third·Party·whosoever·may·interfere,·
→ any·part·of·an·installation.¶
¶
(b) → Any·moves·or·changes·involving·any·wor
→ carried·out·by·an·Engineer.¶

**Figure 1: How NOT to set a hanging indent!**
**This document will be a maintenance nightmare**

Later, if they need to add or delete a word somewhere, or paste it into another document with different margins, the text will go all over the place, and some poor soul will have to spend a great deal of time reformatting the document.

To avoid spaghetti indents, set indents using the ruler:


(a) → No·Third·Party·whosoever·may·interfere,·
any·part·of·an·installation.¶

(b) → Any·moves·or·changes·involving·any·wor
carried·out·by·an·Engineer.¶

**Figure 2: These indents have been set properly, using the ruler.**
**The text can be modified or pasted into any other document without any reformatting being required. This document will be easy to maintain.**

If you work for a company of any size at all, then simply getting your staff to use the ruler properly will save them many hours every week, and significantly increase your company's profitability.

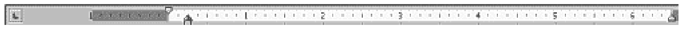
## What rulers? Where?

Although Word's rulers, both horizontal and vertical, are displayed by default, you or the person who set up or previously used your computer may have hidden them to maximize usable screen area. If you don't see anything at the top or side of your screen that looks like a ruler, here's what to do:

1. On the **View** menu, select **Ruler**. This setting toggles the ruler's display. The result will be to display the horizontal ruler in all views and the vertical ruler subject to the conditions described in step **2**.

2. If you're in Normal view, you will never see the vertical ruler, but if you are in Page Layout (Print Layout) view and still don't see it, go to the View tab of **Tools | Options** and check the check box for "Vertical ruler" (at the bottom under "Window"). Note that selecting this option doesn't force display of the vertical ruler; it just adds it to the rulers that are toggled on and off with the **View | Ruler** menu item.

Want the best of both worlds? Want to see the ruler only when you need to use it? Toggle the ruler display off on the View menu, and check the box for "Provide feedback with animation" on the General tab of **Tools | Options**. You will then see just the edge of one or both rulers. When you carefully mouse over this edge, the ruler will slide into view.



## Using the horizontal ruler

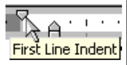
Using the horizontal ruler can save you a lot of trips to the **Format | Tabs**, **Format | Paragraph**, and **File | Page Setup** dialogs (or make it easier to get there). It is also useful when working in tables or columns.

## Setting tabs

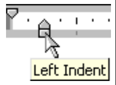
For a full explanation of how to set tabs using the ruler, see Setting tabs.

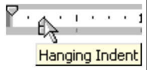
When you need to use the **Format | Tabs** dialog to refine your tab formatting by adding tab leaders or fine-tuning the location of tab stops, you can access this dialog from the ruler. With the mouse pointer on a tab marker or the ruler itself, right-double-click with your mouse; or left double-click on the grey portion at the bottom of the ruler. The chief drawback to this method is that it results in setting a tab stop if you have clicked where there was not already a tab marker.

## Setting paragraph margins and indents

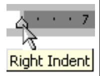


If you hold your mouse over the triangular sliders at either end of the ruler, you will see (provided you have ScreenTips enabled) that they are identified as Right Indent, Hanging Indent, and First Line Indent. The rectangular slider is Left Indent. A little experimentation in a previously entered text paragraph will quickly reveal what they do.





You'll see that the Hanging Indent drags the Left Indent Marker with it, but if you carefully grab just the Left Indent alone, it will drag both the Hanging Indent and the First Line Indent markers, resulting in indenting the entire left margin.

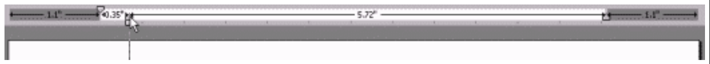


Despite the name, you can also use these sliders to "outdent" (or negatively indent) a paragraph.

You will probably have figured out that the white area in the center represents the text area of the page and the darker portion on the right (and left if you're in Page Layout/Print Layout view) represents the page margins. If you drag a marker into the grey area, the paragraph will extend into the left or right margin.

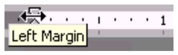
It might appear that the left margin markers cannot be dragged to the left in Normal view, but in fact trial and error will show that they can. If you want to be able to see this portion of the ruler before you start dragging, press the **Shift** key while you click on the left arrow on the horizontal scroll bar. (If you do not see the horizontal scroll bar at the bottom of your screen above the status bar, you need to check the appropriate box on the View tab of **Tools | Options**.)

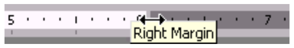
By default, ruler sliders snap to "detents" or "click-stops" at intervals (the interval is 0.06" if you have selected inches as your unit of measurement). But you can override this behavior by pressing **Alt** while dragging. Not only does this allow the sliders to move freely, but the relative measurements are also displayed as you drag (as illustrated below).



## Setting page margins

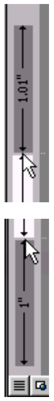
The horizontal ruler can also be used to set left and right page margins, but only in Page Layout (Print Layout) view. If you hover your mouse just above the Left Indent or Right Indent marker, you will see that the pointer changes to a double-headed horizontal arrow, and the ScreenTip says "Left Margin" or "Right Margin."





Dragging will then change the corresponding margin. Once again, pressing **Alt** will allow you to see the relative measurements as you drag.

*You should note an important difference between this action and the action of setting tabs or paragraph margins or indents using the ruler.* When you change paragraph formatting or set tabs without text selected, your actions apply to the paragraph in which the insertion point is located. If you have multiple paragraphs selected, the formatting is applied to all the selected text. Because page margins are a section property, however, when you change margins, **whether or not you have text selected**, the margins will be changed for the entire document (or the current section if there is more than one). This may surprise former WordPerfect users, who have been accustomed to being able to change page margins for selected text (Word accomplishes the same thing by changing the paragraph margins).



## Using the vertical ruler

In the same way, you can change top and bottom margins using the vertical ruler. When you are in the header or footer pane, you can also change the header or footer margin (as well as the top or bottom margin, respectively).

If you prefer to set margins in the Page Setup dialog, or if you have other settings to make in that dialog, you can easily access it by double-clicking on one of the grey borders of either the horizontal or the vertical ruler (be careful to avoid the white area in the center; although double-clicking the centre of the white bit also brings up the Page Setup dialog, clicking nearer the bottom of the white area will set a tab stop).

## Sizing objects using the rulers

When your insertion point is in a text box or you have a frame or floating graphic selected, you will see that the display on both horizontal and vertical rulers changes to reflect the size of the selected object.

You can size the object by dragging in the ruler in the same way that you drag to change page margins. When your pointer shows the ScreenTip “Adjust Left” or “Adjust Right,” “Adjust Top” or “Adjust Bottom,” you can drag to change the size. You may wonder why in the world you would want to do this when it's just as easy to drag the borders of the object itself. The difference is that if you press **Alt** before dragging, you can see the resulting size of the object. Better still, if you just press **Alt** and press and hold the left mouse button without dragging, you can see the current size of the object without a trip to the **Format | Object** dialog.

## Using the rulers to format tables and columns

In the same way, you can resize newspaper-style columns and table rows and columns using the rulers. Although it is generally easier to drag row and column borders within a table (especially given the risk of dragging paragraph margins instead of column margins on the ruler), there is no other way to size newspaper-style columns outside the **Format | Columns** dialog. And here, too, you can press **Alt** and press and hold the left mouse button to get a display of the current dimensions of rows and columns.

For newspaper-style columns, what you can do with the ruler depends on whether or not you have checked “Equal column width” in the Columns dialog; if this option is not checked, you can adjust the width of each column and the distance between columns independently; if it is checked, then dragging one marker drags all equally. For more on working with columns, see **The strait and narrow: using columns**.

## Using the horizontal ruler as a diagnostic tool

If you choose not to display the horizontal ruler, you are missing out on one of the most informative features of the Word workspace. Because it displays paragraph indents and tabs, it can be very helpful in troubleshooting problems with paragraph formatting. And if you've ever had text disappear entirely from a table cell because of a negative right paragraph indent combined with right-aligned text, you'll know how helpful the ruler can be in troubleshooting table formatting. Naturally there will be times when you want to eliminate clutter (though really this is what Print Preview is for), but as a general rule, you would be well advised to keep the rulers visible.

Send all feedback to **word@mvps.org** Top - **Ctrl+Home** Previous - **Alt+left arrow** Page Updated:

# The strait and narrow: using columns

Article contributed by Suzanne S. Barnhill and Dave Rado

Terms of use · Disclaimer

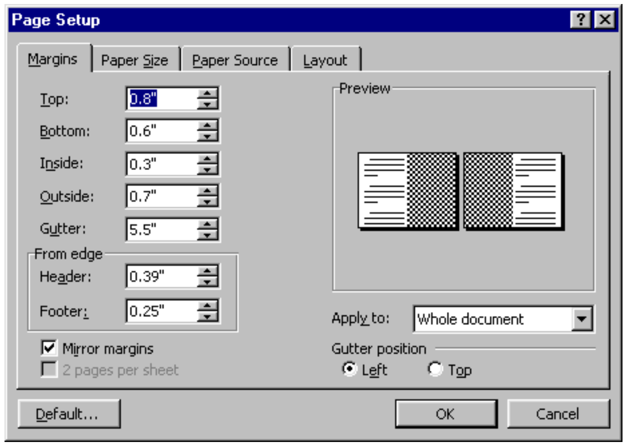
When you open a new blank document in Word, you begin typing at the left side of the screen/ page and continue typing to the right margin, where Word wraps your text back to the left so you can start again. All your lines of text are full width. But sometimes you need to divide your text into two or more columns.

## Types of columns (tables, snaking columns, or ??)

Word gives you several different ways of lining up columns of text. Which one you use depends on what you want the text to do.

### Tabs

If you want to enter text in one column, then text in another column on the same line, then more text in the first column (but on the next line), and so on, and if each of the entries will fit on one line, you may be able to align the columns using tabs. What you're doing here is creating a “tabbed table”:



Page Setup

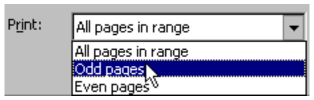
Margins | Paper Size | Paper Source | Layout

Top: 0.8"
Bottom: 0.6"
Inside: 0.3"
Outside: 0.7"
Gutter: 5.5"
From edge
Header: 0.39"
Footer: 0.25"
[✓] Mirror margins
[ ] 2 pages per sheet
Apply to: Whole document
Gutter position: [✓ Left] Top

Just be sure you set a specific tab stop for each column location; don't use Word's built-in tabs.

### Tables

If you want to be able to enter some text in one column, then text in another column aligned with it, then text in the first column again, then the second (and possibly a third, fourth, or more), and if the text in each column must be able to wrap to the next line, then what you want is a table. A table allows you to align text vertically as well as horizontally. Whenever you need to “synch” the columns, you begin a new row.



Print: All pages in range
All pages in range
Odd pages
Even pages

Table columns are especially good for things like opera scores, where you need the text in the original score and the translation to line up with each other vertically.

### Text box columns

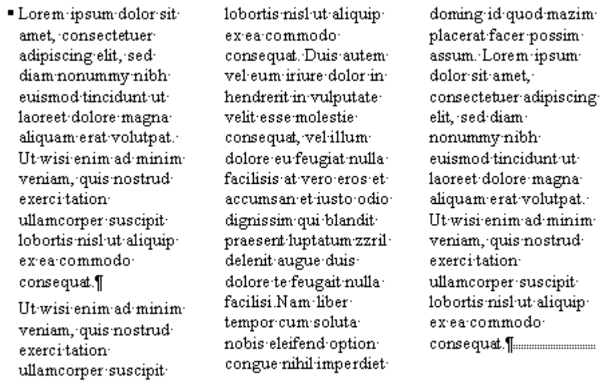
It is possible to get text to flow from one Text Box to another. To do this, click on the first Text Box, then click on the **Link** button on the Text Box toolbar that will now be visible, and click on the second textbox.

The idea of Text Box columns is that, unlike snaking columns (which are covered in detail below), they can be used to cater for articles which start on page 1 of a publication and continue in the middle of page 14. An example is to be found in the (appallingly formatted!) Newsletter Wizard that is supplied with some versions of Microsoft Office (Word 2007 has no wizards, but there are plenty of equally appalling templates in the **Template Gallery** at Microsoft Office Online).

However, Text Box columns *don't* work well and are best avoided. If you need the ability to flow text between non-consecutive pages, you will save yourself a lot of heartache by using a DTP package such as MS Publisher or Adobe PageMaker. For most purposes, however, snaking columns work admirably.

### Snaking columns

If you want the text to fill the first column and then snake into the next and fill it, continuing from Column A to Column B, then Column A on the next page, then you want newspaper-style columns, the subject of this article. (Note that this type of columns is not appropriate for text to be aligned vertically across the page; for that you need to use a table).



▪ Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.¶

Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.¶

If you can make any sense of the above passage,
you should have no trouble learning to work with columns!

### Parallel columns

If you are coming to Word from WordPerfect, you may be used to having another option: parallel columns. Word doesn't offer this option, but you can achieve the same effect using a one-row table with the required number of columns. Be aware, though, that Word can be uncomfortable with very long single-row tables, which may lead to document corruption. Usually you will want to start a new row occasionally, anyway, to line up text in the two (or more) columns.

## How many columns?

The most common reason for using newspaper-style columns is to create (as the name suggests) something like a newspaper. Dividing the page width into narrower lines allows you to use smaller type without creating an unreadable line length. (A general rule of thumb is to have no more than 1½ alphabets, or 40–60 characters, in a line.) For some documents, such as newsletters, the number of columns can vary, and columns may be of unequal widths. For other documents, such as a three- or four-panel brochure, it makes sense to have one column of text for each panel (though sometimes a heading or other design element or even an entire column will spread over two or more panels). However many columns you decide to have, and whether they are of equal or unequal widths, you can easily create them in Word.

## Getting started

Sometimes you will want just one portion of your document to have multiple columns. We'll discuss later how to accomplish that. For now, however, let's assume that your entire document will be multicolumn. For illustration purposes, let's say that it is a four-panel brochure on legal-sized paper in landscape orientation.

### Planning your columns

Since the document will be folded, you will want to allow enough space between the columns to leave a reasonable margin on each panel. This margin should be the same as the outside margin, meaning that the space between columns will need to be double the outside margin.

By default, Word puts half an inch between columns, so you could make your outside margins a quarter of an inch, but that might be pushing the limits of your printer or a photocopier, so let's say you'll set 0.3" left and right margins and plan to have 0.6" between columns.

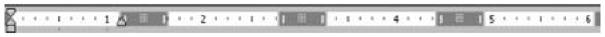
- Word 2003 and earlier: After setting your document margins in **File | Page Setup**, click on the Columns ▥ button on the Standard toolbar, drag to select four columns, and release. Your document will now be divided into four columns. You can also use **Format | Columns** to access the **Columns** dialog.
- Word 2007: After setting your document margins through **Page Layout | Page Setup | Margins** or in the **Page Setup** dialog (accessed via the dialog launcher arrow in the bottom right corner of the **Page Setup** group), click **Columns** in the **Page Setup** group and choose **More Columns…** This opens the **Columns** dialog (see figure below), where you can choose the number of columns.

### What you will see

If you are working in Page Layout (Print Layout) view and have text boundaries displayed, you will see that the text area has been divided into four rectangles (you will have to set Zoom to Page Width to see all four at once).

***Note:*** If you don't have text boundaries displayed, you can select this option as follows: In Word 2003 and earlier, check the box for "Text boundaries" on the **View** tab of **Tools | Options**. In Word 2007, this option is at **Office Button | Word Options | Advanced | Show document content**: "Show text boundaries."

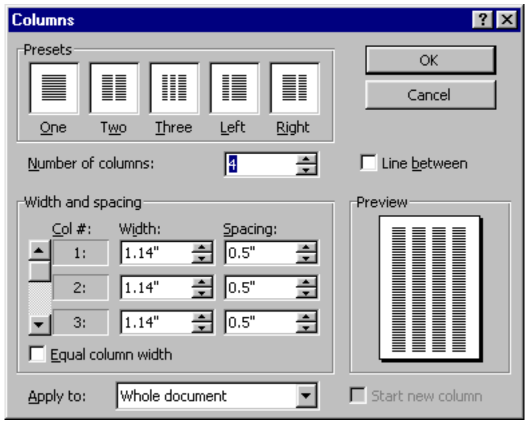
You will also see a change in the horizontal ruler.



If you are working in Normal (Draft) view, the only change you will see will be in the ruler, but when you type you will find that your line now wraps at a shorter length. For this reason, make sure that you don't have very large paragraph indents. If one of your styles, for example, has a one-inch indent both left and right (the default formatting for the Block Text style), there won't be much left of it in a column less than three inches wide!

### Completing the setup

At this point, you will still have only 0.5" between columns. To change this to 0.6", you will need to visit the **Columns** dialog. You can access it from the **Format** menu or **Page Setup** group on the Ribbon, as described above, but a quick way to access it is from the ruler: mouse over the space between columns until your pointer changes to a double-headed horizontal arrow and the ScreenTip says "Left Margin," "Right Margin," or "Move Column"; then double-click.



Number of columns: 4
Line between: ☐

Width and spacing

| Col #: | Width: | Spacing: |
|---|---|---|
| 1: | 1.14" | 0.5" |
| 2: | 1.14" | 0.5" |
| 3: | 1.14" | 0.5" |

Equal column width: ☐

Apply to: Whole document

Start new column: ☐

Check the box for "Equal column width." You'll notice that all the columns except Column 1 are now dimmed. Use the spin button to change the Spacing 0.5" to 0.6" (note that Word adjusts the Width automatically).

You've done all you need to do for now, but take a look around if you like before you OK out of the dialog. Notice that, if you don't choose "Equal column width," you can change each column individually, along with the space between columns. You can also put a line between columns if you like; this line is of a fixed weight (which cannot be changed), but you can have it in any color you like, so long as it's black.

## Working with snaking columns

You can now begin entering text. You will start typing in the first column. When it is filled, text will flow into the second column. If you're editing text near the top or bottom of a column, the constant text reflow may be distracting (and Word may find it challenging to update the display accurately). In such cases, you may find it easier to work in Normal (Draft) view, where you can type in a single long column.

This is all very well if you're typing continuous text, such as a newsletter article, but this is supposed to be a brochure, so you probably have several distinct portions. You may want to work in the second or third or fourth column before you finish the first. So how do you get there? Insert a column break with **Ctrl+Shift+Enter**. This will take you to the top of the next column. Note that column breaks, like page breaks, inherit their formatting from the following paragraph; this can sometimes be a problem. Also, sometimes you will eventually fill the first column so full that there is no need and indeed no room for a column break; in that case you may have to delete it. A column break in a filled column can be difficult to see in Page Layout (Print Layout) view, however, so you may need to switch to Normal (Draft) view to select and delete it.

## Multicolumn sections in single-column documents

Sometimes you will want to divide just a portion of your document into more than one column. This is useful for long, narrow lists that would otherwise waste a lot of paper. Since columns are a section property, you will need to insert section breaks before and after the text you want to divide into multiple columns. Because you want the multicolumn section to be on the same page with the single-column text, these will need to be Continuous section breaks.

Although you can insert these breaks manually, there is an easier way. If you wait till you have typed some or all of the text you want to format into multiple columns, plus at least one paragraph (it can be empty) of the following single-column text, then all you have to do is select the text to be multicolumn and choose the desired number of columns using the Columns button or dialog. Word will take care of inserting the section breaks for you.

***Note:*** If you need to insert a break manually, do it as follows: In Word 2003 and earlier, use **Insert | Break** and choose **Continuous** under “Section break types” in the **Break** dialog. In Word 2007, this dialog is available (as a gallery) via **Page Layout | Page Setup | Breaks**.

Whenever you have a multicolumn section in the middle of a page, Word will automatically balance the columns for you. If you want the columns to break differently (that is, unevenly), you can either insert a column break (**Ctrl+Shift+Enter**) or control text flow using the “Keep with next” and/or “Keep lines together” paragraph properties.

If a multicolumn section ends a page (because you have inserted a Next Page section break or a page break or formatted the next paragraph as “Page break before” or used “Keep with next” to force text to a new page), columns will not be balanced unless you insert a Continuous section break in addition to whatever other break you have used.

## Some snaking column gotchas

There are a few limitations and danger areas you should be aware of when working with columns.

### Columns within columns

When you have divided your document or a portion of it into multiple columns, you can do almost everything you can do in a single-column document or section. You can use all the normal paragraph formatting (including borders), you can insert tables, floating or inline graphics, text boxes, equations, and so on. The only thing you can't do is have multiple columns within a column. Not even if you use a text box, since columns aren't permitted in text boxes, either. If you need to have multiple columns within a column, you will have to use either a table or tabs to align the columnar material.

### Footnotes

Another limitation of columns is that they don't play well with footnotes. Footnotes in a multicolumn document will be wrapped to the column width. Moreover, if the footnote occurs in a multicolumn section in a single-column document, that section will insist on being on a page by itself even though you have used Continuous section breaks before and after it.

The workaround for this is so clumsy that you may well decide it is not worth it, but if you must combine footnotes with columns and insist on full-width footnotes (and especially if the multicolumn section is part of a primarily single-column document), here's how to solve the problem:

1. Insert the footnote in a single-column section as close as possible to the desired location of the actual footnote reference mark (so that the footnote will be on the same page and in the correct order).

2. Where you want the footnote reference mark, insert a cross-reference to the number of the footnote you just inserted. In Word 2003 and earlier, do this with **Insert | Reference | Cross-reference | Reference type: Footnote; Insert reference to: Footnote number (formatted)**. In Word 2007, the Cross-reference dialog is accessed via **References | Captions | Cross-reference** or **Insert | Links | Cross-reference**.

3. Select the actual footnote reference mark (in the single-column text) and format it as Hidden (**Ctrl+Shift+H**).

### Headers and footers

This is not a problem with columns specifically, but whenever you have a short section in the middle of a page (which often happens when you use columns for just a portion of a document), you may develop problems with page numbering or other header/ footer-related aspects. That's because headers and footers are a section property. Your multicolumn section has a header and footer, but because it isn't at the top of the page, there is no way to access its header; because it's not at the bottom of the page, you can't get to its footer. If you suspect that header/footer problems are originating in these inaccessible locations, you have little choice but to remove one or both of the Continuous section breaks, deal with the problems, and then restore the breaks by selecting the text and reapplying the column formatting. Alternatively, using the information in the section above, you could force the multicolumn section to a page of its own (temporarily) by inserting a footnote! Or, more prosaically, you could temporarily insert a page break. Fix the header or footer, then delete the footnote or page break.

### Headings spanning columns

If you want a heading to span all of your columns, you need only leave it in the single-column section before your multicolumn section. If there isn't a single-column section there already, you'll need to create one. This is easily done by selecting the heading paragraph, clicking on the **Columns** button on the Standard toolbar or Page Layout tab, and selecting 1 column; Word will then create the necessary section breaks for you.

But what if you want a heading to span just some of the columns? In our four-panel brochure example, suppose you want text to span two of the four columns. Once you have four columns, you can't redivide just part of the page into two or three columns. You will therefore need to put your heading text into a text box or frame and position it as needed (the problems of positioning text boxes and frames could provide material for another article, so I won't go into that here).

Whether you use a frame or a text box, it will have a border by default. Remove this as follows:

- Remove a border from a frame using the **Borders and Shading** dialog. Access this dialog from the **Format** menu in Word 2003 and earlier. In Word 2007, click the arrow beside the border button in the **Paragraph** group on the **Home** tab. In the dialog, choose **None**. Alternatively, choose **No Border** from the Borders palette or menu accessed via the button on the **Formatting** toolbar in Word 2003 and earlier or in **Home | Paragraph** in Word 2007.
- Remove the border from a text box in Word 2003 and earlier by selecting No Line on the **Colors and Lines** tab of the **Text Box** dialog, accessed by double-clicking the edge of the text box. In Word 2007, selecting a text box will display the contextual **Text Box Tools Format** tab; in the **Text Box Styles** group, click **Shape Outline** and choose No Outline.

If you use a frame, it doesn't seem to matter whether you set wrapping to None or Around. For a text box, the default wrapping style in many versions is None of In Front of Text, which will not work (nor will Through). Any other wrapping style seems to be satisfactory. You will need to experiment with the other layout options to find which ones work best for your situation.

You may want to wait till you're fairly far along in entering text before you insert a text box or frame because the effect of doing this is to reduce the text boundaries to the amount of text you have entered (instead of showing the four rectangles you saw when you first created the columns). This can be rather disconcerting.

## Summary

Avoid text box columns. Use snaking columns when you need text to flow from one column to another, and table or tab columns when you don't. Table columns are especially good for things like opera librettos, where you need the text in the original score and the translation to line up with each other vertically.

# Working with sections

*Or: Why Word appears to behave so illogically when you delete or move a section break*

*Or: How to preserve section formatting when pasting between documents*

Terms of use Disclaimer

Article contributed by Dave Rado

**(to return to top, press Ctrl+Home, or use Alt + Left Arrow to Go Back)**

## How Word sections work

Section breaks store the following information:

1. The Headers and Footers (and their properties) for the section.
2. The Page Setup for the section.
3. The Columns settings for the section.

When you delete a section break, or move an entire section to another part of the document, you get what seem to be very strange results. For instance, deleting a Continuous section break causes the preceding Next Page section break to convert to a Continuous one, or deleting a section break causes an important Header to disappear from the document, or causes the entire document to become landscape..

I agree it's confusing, but it's "by design". These are the rules to remember:

1. A section break **stores** the formatting (page setup, header/footers etc.) of the **preceding** section.
2. The final paragraph of the document contains an invisible section break
3. When you delete a section break, the properties stored in the section break are deleted, and the text which formerly preceded the section break takes on the properties stored in the next section break.
4. A section break **displays** the "Continuous" or "Next Page" property of the **following** section!

So let's say you have 3 sections.

- Section 1 has "Section Start Continuous" defined under Page Setup. The properties of section 1 are stored in the first section break. So the section break at the end of Section 1 **stores** the information "Section Start Continuous".
- Section 2 has "Section Start Next Page" defined under Page Setup. So the section break at the end of Section 1 **displays** the information "Section Break Next Page". Meanwhile the section break at the end of section 2 **stores** the information "Section Start Next Page".
- Section 3 has "Section Start Continuous" defined under Page Setup. So the section break at the end of Section 2 **displays** the information "Section Break Continuous" and the invisible section break at the end of the document **stores** the information "Section Start Continuous".

Now if you delete the second section break, the text which preceded it will take on the formatting of the next section (formerly Section 3, now Section 2), which has "Section Start Continuous" defined. So the first section break will now display "Section Break Continuous" whereas before it displayed "Section Break Next Page".

## Preserving section formatting when pasting between documents

The secret to preserving Headers and Footers, Next Page information, etc. when copying and pasting between documents is to temporarily add a section break at the end of the text you are going to paste or insert.

So for example, Instead of:

Some text
::::::::::::::::Section Break (Continuous)::::::::::::::::
Some text
::::::::::::::::Section Break (Continuous)::::::::::::::::
Some text

... add an extra section break temporarily, so it's like this:

Some text
::::::::::::::::Section Break (Continuous)::::::::::::::::
Some text
::::::::::::::::Section Break (Continuous)::::::::::::::::
Some text
::::::::::::::::Section Break (Continuous)::::::::::::::::

Copy up to and including the temporary section break, which thus preserves the section formatting of the text preceding it. Now paste into the other document. Close the first document without saving.

If the target document is completely blank, however, see also: **How is it possible to copy an entire document into another document without bringing across the header and footer?**

## If you really want to delete the temporary section break ...

Unfortunately, you can't then delete the “temporary” section break(s) from the document you pasted into, or you'll still lose the formatting. This can sometimes lead to a section break being the next-to-last character in the document, which can be awkward.

If you want to get rid of it, you first have to make sure that the section formatting of the final section is identical to that of the preceding one. To do this:

1. Go to the very end of the document, and go into the final section's header. If it's a continuous section break, you will first need to temporarily create a page break at the end of the document, so that you don't go into the previous section's Header.
2. Make sure that both the Header and Footer are set to “Same as Previous”. If they aren't, use the Header/Footer toolbar to set it to this. Then return to the main document.
3. Go to the penultimate section, select **File + Page Setup...** and press **Return**. (This makes Word “Remember” all the settings in the dialog).
4. Go to the final section and press **F4** (repeat last command). This applies the “remembered” settings to the final section.
5. If there are differences in the column formatting between the two sections, you'll also need to use the **F4** trick with the **Format + Columns...** dialog.
6. You can now safely delete the final section break (and the manual page break, if you inserted one).

## Preserving section formatting when using Insert + File

The rules section breaks follow when using Insert + File are even more Alice-in-Wonderland than elsewhere; but the fix is straightforward: the files you plan to insert must contain a continuous section break at the **start** of the document, as well as at the end. (Alternatively, just stick to copy & paste, which works more logically).

If you're using Insert + File, Word inserts the saved version of the document, so you would need to save the file you're inserting, having inserted the extra section breaks, in order to have the temporary section breaks included when the file is inserted. You can, if you want, subsequently delete the temporary breaks and save the file again.

Here is some information that a source of mine at Microsoft found in the Office 2000 bug database regarding this

> “We preserve the last section of the destination's section properties by copying them to the first section of the source. The workaround will work if the source document starts with a continuous section break. We can't fix this bug without breaking another scenario. I say we let it lie instead of reverting back to Word '95 behavior and breaking something else”.

I wrote back to him: “The workaround works, but I'm still trying to get my head around it! <g>”. He replied: “Don't bother trying to figure out the reasoning. I'm of the opinion that it really should work the way that you were originally trying to do it. There's just no way of getting a program manager to agree with me and change it now... :-)”

The problem you will get if you **don't** use this fix can be reproduced as follows:

1. Create a new document (Doc1) and add a next page section break.
2. Set up section 1 with 1" margins and section 2 with 2" margins. Save and close.
3. Create a second new document (Doc2) and give it 3" margins.
4. Select Insert + File, and insert Doc1 into Doc2.

What one would expect to get is as follows:

1. The final paragraph mark of Doc2 originally contains section formatting of 3" margins, so the final section of Doc2 should still have 3" margins following the InsertFile.
2. The section break inserted into Doc2 from Doc1 contains the section formatting of 1" margins, so section 1 following the insertion should have 1" margins.

In fact, after inserting Doc2 into Doc1, section 2 has 2" margins, and section 1 has 3" margins.

Inserting a section break at the end of Doc1 doesn't help. What happens then is that after inserting Doc1 into Doc2, section 1 has 3" margins, section 2 has 2" margins, and section 3 has 2" margins.

In other words, whatever you do, the section formatting of section 1 in Doc1 is lost when inserted into Doc2.

If you insert a section break into Doc2 prior to inserting Doc1, it makes no difference – the section formatting of Section 1 is lost whatever you do. And you don't get these problems if you copy and paste.

As previously mentioned, the only fix is to insert a continuous section break at the **start** of the document you want to insert.

If the target document is completely blank, however, see also: **How is it possible to copy an entire document into another document without bringing across the header and footer?**

## Merging sections

If you want to merge two contiguous sections within a document, the same logic applies as described above under **How Word sections work**; with the following implications

1. If you want the section formatting of the second section to take precedence, no problem: it will, automatically.
2. If you want the section formatting of the first section to take precedence, and if the second section is followed by a section break (that is, if the document contains three or more sections), just select the first section break, Edit + Cut, and, immediately before the next section break, select Edit + Paste. Then delete the next section break, leaving the one you just pasted in place. What was previously the first section's section break has now become the merged section's section break.
3. If you want the section formatting of the first section to take precedence, and if the document only contains two sections, you can **either** cut the section break and paste it into an empty paragraph at the very end of the document (but then you're stuck with a redundant section break); **or** make sure that the section formatting of the final section is identical to that of the preceding one, by using the shortcuts described under **If you really want to delete the temporary section break ...**. You can then safely delete the section break.

# How to control the page numbering in a Word document

*Article contributed by Bill Coan*

Word's page numbering scheme isn't directly obvious but it isn't needlessly complex, either. Indirect might be a good term for it. Once you understand how it works, all sorts of possibilities open up. Unfortunately, the built-in tools that simplify the insertion of page numbers also happen to make it more difficult to tell what's really going on. So, for the moment, forget everything you've learned or think you know about page numbers. Let's start at the beginning, since it won't take too long . . .

First some background, consisting of four big ideas:

## *Big Idea #1*

A new blank document consists of a single section. Each time you insert a Section Break into the document, the document gains a section.

In other words, if you create a new blank document, then choose Break on the Insert menu and select a Next Page Section Break, the document will thereafter consist of two sections: Material from the start of the document up through and including the break represents Section 1. Material after that represents Section 2. (In some cases, Word inserts Section Breaks automatically, such as when you change the number of columns in part of a document.).

## *Big Idea #2*

Page numbers are usually a section property, not a document property. A page's number is determined by only two factors:

1. The “start at” value assigned to the document section.
2. The location of the page within its document section.

In other words, if a document section has been told to “start at” page 5, the first page of that section will be page 5, the second page will be page 6, the third page will be page 7, and so on.

## *Big Idea #3*

Just because a page has a number doesn't mean the number will appear somewhere on the page. Far from it. In fact, ALL pages have numbers. The number never appears unless you tell Word to display it.

## *Big Idea #4*

The easiest way to tell Word to display a page's number somewhere on the page is to insert a field code as follows:

1. Position the cursor where you want the number to appear. (Often, this is the header or footer. To position the cursor in one of these locations, choose Header and Footer on the View menu or double-click the location in Page Layout view.)

2. Press **Ctrl+F9** to insert a pair of field braces: **{ }**. (Don't just type them. You must use Ctrl+F9.)

3. Between the field braces, type “PAGE” without the quotation marks. (This will look as follows: **{** PAGE **}** )

4. To control the numeric format of the number, add a “switch” by continuing to type until your field code looks like one of the following:

**{ PAGE * Arabic }**

**{ PAGE * alphabetic }**

**{ PAGE * ALPHABETIC }**

**{ PAGE * roman }**

**{ PAGE * ROMAN }**

5. Right-click anywhere between the braces and choose **Update Field**. If, having done that, the field code is still visible, switch field codes off by pressing **Alt+F9** (or by going to **Tools + Options + View**).

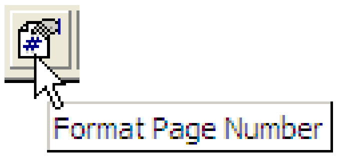
The “PAGE” field code is as uncomplicated as it looks. It has no effect on Word's page numbering scheme. It simply tells Word to display the number of the current page at the field location. As noted above, the page's number is dependent on:

1. The “start at” value assigned to the document section.
2. The location of the page within its document section.

This raises a question: How can I get your hands on the all-important “start at” value? After all, what good does it do to insert a PAGE field only to find out that Word thinks the second page of your document is page 102 because the “start at” value has been set to 101?

To change the start at value, proceed as follows:

1. Position the cursor in the section of the document whose start at value you want to change.
2. Go into the Header or Footer.
3. On the Header/Footer toolbar, click the Format Page Numbers button:



4. Click Start At and enter the desired value.

5. Click OK to close the dialog.

This method usually works **much** better than using the Insert + Page Numbers dialog.[1]

●●●

The following big ideas are presented for extra credit only. Please see me after class if you're worried about your grades.

## *Big Idea #5*

If a document is divided into multiple sections, the headers and footers for each section start out with their “same as previous” property turned on. This feature lets you insert a **{** PAGE **}** field into the header or footer for Section 1 and automatically have it appear in the headers or footers for all other document sections. (If you want to make changes to the header or footer for Section 1 without having those changes show up in the header or footer for all other document sections, you must turn off this feature by viewing each header and footer and clicking the Same As Previous button on the Header and Footer toolbar until the button is no longer depressed.)

## *Big Idea #6*

Each section of a document can have up to three headers and footers: the first page header and footer, odd page headers and footers, and even page (aka “primary”) headers and footers. The actual number of headers and footers available in a given section is determined by the Different First Page checkbox and Different Odd and Even Pages checkbox in the Page Setup dialog. To view these settings, position the cursor in the document section you're interested in, then choose Page Setup on the File menu. Changes made to the first page header have no effect on the odd page header or the even page header and vice versa all the way around.

## *Big Idea #7*

**Caution:** Just a word to the wise: consider carefully before using the following techniques. They can lead to problems in corporate documents, unless you include a warning to other uses that you are "calculating" the page numbers in this document. If you do not, great confusion can result, and other users may unsuspectingly ruin your page numbering or your document.

Now that you know about **{** page **}** fields, you're just two steps away from some truly amazing possibilities, made possible by two other fields, the formula ( **=** ) field, and the **If** field. Want to force Word to display a number that's one higher than the real page number? Try using the following field, where braces are inserted using Ctrl+F9:

**{ = { PAGE } + 1 }**

Want to tell Word to display a page number on pages one, two, and three, but not on any other pages? Try using the following field, where braces are inserted using Ctrl+F9.

**Note:** Make *sure* to leave spaces around the "<" sign:

**{ IF { PAGE } < 4 "{ PAGE }" "" }**

Want to tell Word to put the "real" page number on pages one, two, and three, but a higher-by-one page number on all other pages? Try using the following field, where braces are inserted using Ctrl+F9:

**{ IF { PAGE } < 4 "{ PAGE }" "{ = { PAGE } + 1 }" }**

Want Word to insert "Continued/..." on every page except the final page, where you want it to display "- End -" instead? Use

**{IF { PAGE } < { NUMPAGES } "Continued/..." "- End -" }**

But see Page X of Y displays or prints as Page 1 of 1, Page 2 of 2 etc. for some gotchas regarding the NumPages field and how to work around them.

---

**1.** But if you *do* use the Insert + Page Numbers dialog, be sure to click **Close** when you've finished, *not* OK, or you'll insert a second Page field! Furthermore, the dialog inserts the field in a frame, making it much harder to manipulate than if you insert it in the Header or Footer yourself as described in this article.

# Finding and replacing non-printing characters (such as paragraph marks) and text formatting

Article contributed by Dave Rado

## 1. Finding & Replacing non-printing characters

1. On the **Edit** menu choose **Replace**.
2. In the *Find & Replace* dialog, click in the *Find What* box and Click the **More** button.
3. Either type the character code you need (for instance, you can type **^p** in the *Find What* box to represent a paragraph mark); or, if you didn't happen to know that the character code you needed was ^p, click the **Special** button, and select an item from the list.

   If you use the **Special** button, a special code representing the non-printing character will be inserted in the *Find What* box. For instance, if you selected *Paragraph Mark*, **^p** will be inserted.
4. Do the same in the *Replace With* box. For instance, to replace all manual line breaks with paragraph marks, you would replace **^l** with **^p**. Or to delete all manual page breaks, replace **^m** with nothing. Once you know the special codes you can just type them; but the *Special* button is invaluable at first!

   **Note:** these character codes are case-sensitive: for instance, ^P is *not* valid.
5. If you leave the *Replace With* box empty, the search string in the *Find What* box will be deleted from the document or selection.
6. You can type additional text around the special codes. For instance, to delete all instances of a full stop [period] at the end of a paragraph, (in a bulleted list, for instance), you would select the area you want to do the replacement in (or not, if you want to do it in the entire document); in the *Find What* box, type:

   **.^p**

   leaving the *Replace With* box empty; then click **Replace All**.

## 2. Finding & Replacing formatting

### To replace text that has particular formatting

To find text that has particular formatting and replace it with different text, but without changing the formatting, you would need to carry out the following steps:

1. On the **Edit** menu choose **Replace**.
2. In the *Find & Replace* dialog, click in the *Find What* box and type the text you want to search for.
3. Click the **More** button.

4. Click on the **Format** button, and select the options as required.
5. In the *Replace With* box, type the text you want it to be replaced with and click **Replace All**.

For example, if you wanted to find all instances of the word “Hi” that were formatted in the Heading 1 style, and replace them with the word “Ho”, you would need to carry out the following steps:

1. In the *Find What* box, type “Hi”.
2. Click **More**, then **Format**, then **Styles**.
3. Find the *Heading 1* style in the list and click **OK**.
4. In the *Replace With* box, type “Ho”
5. Tick the *Find whole words only* checkbox
6. Where is says *Search*, make sure it's set to **All**.
7. Click **Replace All**.

### To replace formatting

To find all instances of the word “Hi” that were formatted in bold and remove the bold formatting, you would need to carry out the following steps:

1. In the *Find What* box, type “Hi”.
2. Click **More**, then **Format**, then **Styles**.
3. Find the *Heading 1* style in the list and click **OK**.
4. Leave the *Replace With* box empty.
5. Tick the *Find whole words only* checkbox
6. Where is says *Search*, make sure it's set to **All**.
7. Click **Replace All**.

To replace all instances of the *Normal* style with the *Body Text* style you would need to:

1. Leave both the *Find What* and *Replace With* boxes blank.
2. With your cursor in the *Find What* box, click **More**, then **Format**, then **Styles**.
3. Find the *Normal* style in the list and click **OK**
4. With your cursor in the *Replace With* box, click **Format**, then **Styles**, find the *Body Text* style in the list, and click **OK**.
6. Where is says *Search*, make sure it's set to **All**.
7. Click **Replace All**.

## 3. Finding & Replacing other special characters

- It's easy to find and replace characters such as ©, é, ä, or any character listed in the **Insert + Symbol** dialog with the **Font** box set to “(normal text)” and the **Subset** box set to “Basic Latin”, (or to put it another way, any character included in the **ANSI character set**). Just insert the character(s) into your document and then cut and paste them into the dialog.

  The easiest way is to insert both the “find” and “replace” strings into your document first, next to each other, so you can paste both – using **Ctrl+V** – into both the “Find what” and “Replace with” boxes and delete as appropriate.

  So if you want to replace Andre with André, type AndreAndré, paste that into both boxes of the dialog, and delete André from the first box and Andre from the second.

  For other symbols, such as **Upper Unicode** characters, and symbols from decorative fonts such as Symbol and Wingdings, things get a little more complicated, but it can be done. For the details, see **Finding and replacing symbols**.

- Sometimes when you paste in from other applications, non-printing characters paste in that *display* as paragraph marks but don't behave like “proper” paragraph breaks should – they behave like manual line breaks. The character code for a paragraph mark is 13 (as can be shown be selecting one and running a macro containing the line: MsgBox Asc(Selection.Text)).

  Replacing **^013** with **^p** fixes the problem.

- Sometimes you might want to replace all "keyboard quotes" in a document with “smart quotes”. The easiest way to do so is to simply replace " with " (i.e. replace a keyboard quote with itself) and then ' with ' (i.e. replace a keyboard apostrophe with itself), making sure you have “smart quotes” turned **on** (under Tools + AutoCorrect; “AutoFormat As You Type” tab).

  To replace “smart quotes” with "keyboard quotes", do the same Find and Replace operations, but with smart quotes turned **off.**

## 4. Controlling which sections of the document are acted upon

If you click on the **More** button and look in the *Search* box, it will be set to *Down*, *Up* or *All*.

By default, if nothing in the document is selected, it will be set to *All*, whereas if something is selected, it will be set to *Down*.

**If it's set to *Down (or Up)***, it will act upon the selection only; then ask if you want to continue searching the rest of the document. *Even if you click **Yes**, it will ignore the Headers and Footers.* (If nothing is selected it will search to the beginning or end of the document, then ask if you want to search the rest – but will still ignore the Headers & Footers).

**If it's set to *All***, it will check everything without asking, *including* the Headers and Footers. So if you want to do a replace on the entire document, you **must** remember to click **More** and then set the *Search* box to *All* if it isn't already.

If you're searching down, by the way, you can quit the dialog at any point and use **Shift +F4** to repeat the Find; and you can go down as well as up using the Browse arrows at the bottom of the vertical scrollbar to repeat the Find (this tip is more relevant to Finding than to Replacing, but seemed worth throwing in for good measure!).

By now you'll be starting to realise that to use the *Find & Replace* dialog properly you must **always** begin by clicking the **More** button! So why have the button at all, you may ask – it's a major **PITA** having to click it every single time! If you feel that way, may I suggest you email **mswish@microsoft.com**. If enough of us ask, who knows, they might listen!

## 5. Major “gotchas”

If you Find & Replace formatting, the settings you chose are sticky – they won't be reset until you do another Find & Replace and click the **No Formatting** button.

Settings such as *Find entire words only* are equally sticky.

When you do a replace and it finds nothing when you expected the opposite, the reason is *almost always* that you forgot to clear all those sticky settings. Another reason to email **mswish@microsoft.com** – but in the meantime, you can **assign a macro** to a toolbar to **clear the settings from the Find dialog**. Then you can just click on the toolbar button after each Find & Replace.

Or better still, you could write a macro to **intercept** the **EditFind** and **EditReplace** commands and include the code to clear the settings in your macro. Then you won't have to remember to click on a button.

## 6. Using wildcards

Using wildcards allows you to perform extremely powerful Find & Replace operations in Word. This is covered separately in the article: **Finding and replacing characters using wildcards**.

## 7. Related articles

**Finding and replacing characters using wildcards**

**How can I insert special characters, such as dingbats and accented letters, in my document?**

**Delete html tags or sgml tags or other bracketed tags (<example>) from a document without affecting other text**

**Finding and replacing symbols**

**How to replace text in quotation marks with italic or highlighted text minus the quotes**

**Transpose dates from mm/dd/yy to yy/mm/dd**
*Example: from 12/17/85 to 85/12/17*

**Eliminate carriage returns (paragraph marks) at the end of each line but not at the end of each paragraph**

**How to put a page break in front of each Heading 1 paragraph**

**How to modify the path or filename in a series of hyperlink fields**

**Replace any instance of the left square bracket character "[" that happens to be the fifth character in a paragraph**

**Remove all empty paragraphs from a document**

**How to use Edit Find to select everything from where the cursor is to the first found item**

# Finding and replacing characters using wildcards

Article contributed by Graham Mayor, with thanks also to Klaus Linke

## Contents

**(to return to top, press Ctrl+Home)**

## Overview

Wildcards are like the blank pieces in Scrabble, or like the Jokers you can use in some card games to stand in for any card. You are perhaps already familiar with the "*****" and "**?**" wildcards from file matching: In the **File + Open** dialog, you can display all files with the extension ".doc" by typing "*.doc", or all files "01062001.doc", "01072001.doc", "01122001.doc"... by typing "01??2001.doc".

But the wildcard feature in Word goes way beyond that, and can be very powerful.

To begin, you must first turn Wildcards **on** in the Find/Replace dialog.  To do so, bring up the **Find** dialog, click **More** and check **Use wildcards**.  In a macro, set **.Find.MatchWildcards = True**.  If you do not do this, Word treats the wildcard characters as if they were ordinary text.

As we'll see later, you can define ranges **[]**, groups **()**, repeats @, **{}**, anchors **< >** and exceptions **!**. With these regular expressions you can search for patterns in your text that have certain things in common (some pattern: for example, that they only contain certain characters, or a certain number of characters).

**Note:** Word uses "lazy" pattern matching: this means it will quit matching as soon as possible. Most Unix tools use "greedy" pattern matching (the algorithm tries to match as much text as possible), so if you have used such tools, beware!

The secret of using wildcard searches is to use a "pattern" that identifies the string of text that you wish to find, and ignores everything else. Wildcards are used to represent the characters or sequences of characters in that string.

Because different combinations of characters can be represented by a variety of wildcard combinations, there is often more than one way of identifying a particular string of text within a document. How you choose to represent that group of characters is therefore often a matter of individual preference; and the context of the text within the document will to a great extent dictate the most suitable combination to use on a particular occasion.

The following is a list of the characters that have a special meaning in wildcard searches **( [ ] { } < > ( ) - @ ? ! * \ )**.

**Note: wildcard searches are case sensitive.**

It doesn't help that the list of wildcard characters in Word's Help files is almost impossible to find! The wildcard characters are all listed and described in this article, but if you need to find them in Help, the topic is called: "Type wildcards for items you want to find". But you can't get to that article directly; you must first find the topic: "Fine-tune a search by using wildcard characters", which contains a link to it!

**Zen tip:** when using wildcard searches: don't wrinkle your brow or bite on your tongue while thinking it through – you have to keep up a regular expression. :-|

## The theory

### 1. ? and *

The two most basic wildcard characters are **?** and *****. These are essentially similar in use.

**?** is used to represent a single character and ***** represents any number of characters. On their own, these have limited use.

**s?t** will find sat, set, sit, sat and any other combination of three characters beginning with "s" and ending with "t". It will also find that combination of letters *within* a word, thus it would locate the relevant (highlighted) part of inset etc.

**s*t** will find all the above, but will also find "secret", "serpent", "sailing boat" and "sign over document", etc.

**Note:** Unlike **?**, which always looks for the missing character, ***** will also find occurrences of the two adjacent characters (here "s" & "t") with *no* text between them e.g. "streaker". It is a blunt instrument and must be used with care, or with other characters to tie it down to only the text you require. There is no limit to the amount of text that the ***** character might match. It can end up matching all of your text in a multi-megabyte document!

### 2. @

@ is used to find one or more occurrences of the previous character. For example, lo@t will find lot or loot, ful@ will find ful or full etc.

### 3. < >

Used with any of the above (or any other combination of wildcards and characters), you can use the tags **<** and **>** to mark the start and end of a word, respectively. Thus, building on the example we used for the "*****" character: **<s*t>** would find "secret" and "serpent" and "sailing boat", but **not** "sailing boats" or "sign over documents". It will also find "'set" in "tea-set" , but **not** "set" in "toolset".

Again, beware of using "*", as **<s*t>** will find any block of text from a word starting with "s" to the end of the next word in the document that ends with "t", which may not always be what you had in mind.

The **<>** tags can be used in pairs, as above; or individually, as appropriate. For instance, **ful@>** will find "full" and the appropriate part of "wilful", but will not find "wilfully".

### 4. [ ]

Square brackets are always used in pairs and are used to identify specific characters or ranges of characters. For example:

**[abc]** will find *any* of the letters a, b, or c. **[F]** will find upper case "F".

**[A-Z]** will find any upper case letter; **[0-9]** will find any single number; **[13579]** will find any odd numbers; **[0-9A-z]** will find any numbers *or* letters.

You can use any character or series of characters in a range **[ ]**, including the space character. Characters are processed in alphanumeric order – lowest first. If you are uncertain which character is lower than another, look in the Insert + Symbol dialog.

## 5. \

If you wish to search for a character that has a special meaning in wildcard searches – the obvious example being "?" – then you can do so by putting a backslash in front of it:

**[\?]** will **not** find "****" followed by any character; but will find "?" instead.

If you wish to find the backslash itself then you need to precede that with a backslash **[\\]**. It's best if you always put the double-backslash into its own range, as in **[\\]** – sometimes it doesn't work, otherwise.

As previously mentioned, the following is a list of the characters that have a special meaning in wildcard searches **( [ ] { } < > ( ) - @ ? ! * \ )**.

## 6. [!]

**[!]** is very similar to **[ ]** except in this case it finds any character **not** listed in the box so **[!o]** would find every character except "o". You can use ranges of characters in exactly the same was as with [ ], thus [!A-Z] will find everything except upper case letters.

## 7. { }

Curly brackets are used for counting occurrences of the previous character or expression.

**{n}** This finds exactly the number "n" of occurrences of the previous character (so for example, **a{2}** will find "aa").

**{n,}** finds at least the number "n" occurrences; so **a{2,}** will find "aa" and "aaaa").

**{n,m}** finds text containing between "n" and "m" occurrences of the previous character or expression; so **a{2,3}** will find "aa" and "aaa", but only the first 3 characters in "aaaa" ).

**Note:** Counting can be used with individual characters or more usefully with sets of characters e.g. [deno]{4} will match done, node, eden); or with bracketed groups: (ts, ){3} will match ts, ts, ts, .

(Unfortunately, there is no wildcard to search for "zero or more occurrences" in Word wildcard searches; [!^13]{0,} does not work).

## 8. ( )

Round brackets have no effect on the search pattern, but are used to divide the pattern into logical sequences where you wish to re-assemble those sequences in a different order during the replace – or to replace only part of that sequence. They must be used in pairs and are addressed by number in the replacement e.g.

(John) (Smith) replaced by \2 \1 (note the spaces in the search and replace strings) – will produce Smith John

or replaced by \2 alone will give Smith.

**Note:** The placeholders \1, \2 etc., can also be used in the search string to identify recurring text. e.g.

Fred Fred could be written (Fred) \1.

**Round brackets are perhaps the most useful aspect of complex wildcard search and replace operations.**

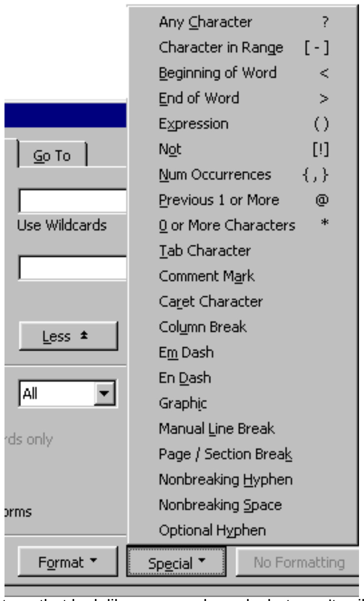
## 9. ^

The **^** ("caret") character is not specific to wildcard searches but it sometimes has to be used slightly differently from normal, when searching for wildcards.

In the Find and Replace dialog, if you click in the "Find what" and "Replace with" boxes, and click the "Special" button, you will see a list of supported special characters that you can use; select one and a code will be inserted in the box for you. The ones near the bottom of the list insert special **^** codes, such as **^t** for a tab. Once you know what the codes are, you can type them straight in without using the "Special" button.

Unfortunately, the range of options

available from the “Special” button when you do a wildcard search is more limited than in an ordinary search. Some notable examples, and their workarounds:



- You may wish to identify a character string by means of a paragraph mark ¶. The normal search string for this would be **^p**. ^p does **not** work in wildcard search strings! It **must**, however, be used in replace strings, but when searching, you must use the substitute code **^13**.

  If you use ^13 in a replace string, invalid characters, that look like paragraph marks but aren't, will be inserted – so beware!
- Wildcard searches will also not find footnote/endnote marks – substitute **^2**.
- In normal searches you can use **^u** followed by the character number to find a character by code (which is useful for finding non-keyboard characters). That doesn't work in wildcard searches (and even in the replacement text of normal searches). So you have to either paste the character itself into the search text, or use **the Alt-XXXX-trick** to type it (the latter doesn't work in Word 97).
- There doesn't seem to be any way to find fields in a wildcard search. Even **^19** (the field code character code) doesn't work in wildcard searches – this seems to be a bug.

For most other special characters and objects, you can use the same codes as in simple searches (^l = manual line break, ^g = graphic, ^^ = caret ...).

In many cases, characters can be addressed using their character numbers. The obvious example is ^13 for “paragraph”, as mentioned above. Unfortunately, this has not been implemented in a completely predictable or reliable manner, and if you have a need to search using character numbers, ensure that the search works with a test sample before committing to it.

**So much for the theory. How are the wildcards used in practice?**

## The practice

### Example 1: Transpose first name and surname

There are many occasions when you are presented with blocks of text or numbers etc., where the order of the text is not what you might require in the final document. Swapping the placement of forename and surname as above is one such example – and don't forget you can add to the replacement, even when using bracketed replacements

For instance, you may wish **John Smith** to appear as **Smith, John**.

Or, more likely, you may have a column of names in a table, where you wish to exchange all the surnames with all the forenames.

| |
|---|
| John Smith |
| Brian Jones |
| Alan Williams |

You could do them one at a time, but by replacing the names with wildcards, you can do the lot in one pass.

Let's then break up the names into logical sequences that can only represent the names:

- At its simplest, we have here two words – John and Smith. They can be represented by**<*>[space]<*>** – where **[space]** represents a single press of the spacebar.

- Add the round brackets **(<*>)[space](<*>)** and replace with **\2[space]\1**
- Run the search on the column of names and all are swapped. Run it again and they are swapped back.

**Note:** If you get it wrong, remember that Word's "undo" function (**Ctrl+Z**) is very powerful and has a long memory! 🙂

**If some of the names contained middle names and/or initials**

- If some of the names contained middle names and/or initials, you would first have to convert your table to text (separated by paragraph marks). Select **Table + Convert Table to Text**.

  Or if there is more than one column in your table, paste your "Name" column into a new document, and *then* convert that column to text.
- You could then replace:
  **(<*) ([! ]@)^13**

  with:
  **\2, \1^p**

  This would convert:

  | | | |
  |---|---|---|
  | John F. Kennedy<br>J. Smith<br>John Smith | ▶ | Kennedy, John F.<br>Smith, J.<br>Smith, John |
- Finally, convert the text back to a table. (Select **Table + Convert Text to Table)**.

  If there was more than one column in your original table, then when converting the text back to a table, be sure to select "Paragraphs", where it says "Separate text at". Then, in your original table, delete the old column and paste in the new one.

## Example 2: Transposing dates

Another useful example might be the changing of UK format dates to US format dates – or vice versa.

7th August 2001 to August 7, 2001

(For a similar example, see also **Transpose dates from mm/dd/yy to yy/mm/dd**.)

To give an example of how most of the wildcard characters could be used in one search sequence to find any UK date formatted as above, the following search pattern will do the trick:

**[0-9]{1,2}[dhnrst]{2} <[AFJMNSOD]*> [0-9]{4}**

Breaking it down:

- **[0-9]** looks for any single digit number, but dates can have two numbers; so to restrict that to two, we use the "count" function {}. We want to find dates with 1 or 2 numbers so:
  **[0-9]{1,2}**
- Next bit is the ordinal "th" – Ordinals will be "nd", "st", "rd", or "th" so identify those letters specifically:
  **[dhnrst]**

  There will always be two letters, so restrict the count to 2:
  **[dhnrst]{2}**
- Next comes the space. You can insert a space with the spacebar **[space]**.
- The month always begins with one of the following capital letters – AFJMNSOD. We don't know how many letters this month has so we can use the blanket "*" to represent the rest. And we are only interested in that word so we will tie it down with <> brackets:
  **<[AFJMNSOD]*>**
- There's another space **[space]** followed by the year. Years will always have four numbers so:
  **[0-9]{4}**
- Finally, add the round brackets to provide a logical break-up of the sequence:
  **(**[0-9]{1,2}**)(**[dhnrst]{2}**)**[space]**(**<[AFJMNSOD]*>**)**[space]**(**[0-9]{4}**)**

  and replace with:
  **\3[space]\1,[space]\4**

  (where **[space]** represents pressing the spacebar once) to re-order the sequence.
- US style manuals specify that ordinals should *not* be used in dates, but if you did want to keep the ordinals (so converting 7th August 2001 to August 7th, 2001), you could replace
  **(**[0-9]{1,2}[dhnrst]{2}**)**[space]**(**<[AFJMNSOD]*>**)**[space]**(**[0-9]{4}**)**

  with:
  **\2[space]\1,[space]\3**
- You can use the same logic in reverse to convert August 7th, 2001 to 7th August 2001; or to convert August 7, 2001 to 7 August 2001. Unfortunately you can't **add** the ordinals using a wildcard Find & Replace, if they're not there to begin with – you would need to use a macro if you wanted to do that.

## Example 3: Adding or removing the period in salutations

Assume you are parsing addresses and wish to separate the honorific from the name. American usage puts a period at the end (Mr., Mrs., Dr.) while British usage omits the period.

**([DM][rs]{1,2})( )**

will find Mr Ms Mrs Dr without the period and

**\1.\2**

will put one in.

or vice versa:

**([DM][rs]{1,2}).**

will find Mr. Ms. Mrs. Dr. with the period and

**\1**

will take it out.

## Example 4: Duplicate paragraphs (and rows)

- **(*^13)\1\1** will match any sequence of three identical paragraphs.
- If you replace:

  **(*^13)\1**

  with

  **\1**

  it will delete all consecutive duplicate paragraphs in the document. Repeat until nothing is found, to delete all duplicate paragraphs in the document (as long as you have sorted the text first).
- To delete duplicate rows in a table (provided you have no merged cells), you can convert the table to text (Table + Convert to Text, using a tab delimiter); delete the duplicate paragraphs using the above method, then convert the text back to a table.

## Example 5: Tags

**\<([!\<\>]@\>)*\</\1**

will match any well-formed XML element including start-tag and end-tag such as:

<p>some text</p>

or

<customer-name>John Smith</customer-name>

See also:
**Delete html tags or sgml tags or other bracketed tags (<example>) from a document without affecting other text**.

## Example 6: Formatting

By building up appropriate patterns, you can search for almost any combination of characters.

Of course you can also restrict your searches by specifying some style or formatting, or add formatting for the replacement text. See **Finding and replacing non-printing characters (such as paragraph marks), other special characters, and text formatting** for more on this.

A nice trick if you want to apply formatting to **a part** (but not all) of the search text is to put in “tags” in a first replacement.

In a find/replace, you can only change the formatting of the whole find-text; so you would need to do two find-replaces to get the job done. In the first, you would “tag” the text that has to be formatted; in the second, you format them (and remove the tags).

Find what:
**(something)(something else)(another string)**

Replace with:
**\1$$\2##\3**

and then remove the tags and apply the formatting in a second replace:

Find what:
**$$(*)##**

Replace with:
**\1 ((bold))**

Imagine, for instance, that you've got a text file, and headings are marked up by having 3 empty paragraphs before them and an empty paragraph after.

Find what:
**^13{4}([!^13]@^13)^13**

Replace with:
**^p<H1>\1**

Then Find what:
**\<H1\>(*)**

Replace with:
**\1 ((style "Heading 1"))**

This will remove the empty paragraphs and format the headings.

See also: **Apply the built-in Heading 1 paragraph style to all paragraphs containing text in ALL CAPS**

### More examples

See also:
**Replace any instance of the left square bracket character "[" that happens to be the fifth character in a paragraph**

**How to make urls (and delimiters such as \, /, : and @) wordwrap in Word**

**How to replace text in quotation marks with italic or highlighted text minus the quotes**

## Tips for advanced users

- You can paste any (Unicode) character (unfortunately **not** characters from decorative (Symbol) fonts) into your search expressions. So copying the first and last characters from the Greek or Cyrillic subsets into a search:

  **[;-ώ]**
  would match any Greek character:
  **α β γ δ ε ...**

  **<[Ё-ґ]@>**
  matches any Cyrillic word:
  **Вы можете помочь мне?**
  ("Can you help me please?")

  **Note**: in Word 97, the characters sometimes display in the dialog box as squares, but they do work.

- In Word 2000+, you can type in Unicode characters with the Alt-key (make sure NumLock is on, then hold down the Alt-key and type the numbers on the numeric keypad). Since all characters from decorative fonts (Symbol-, Wingdings-fonts ...) are kept in a special code page from &HF000 to &HF0FF, you can search for them with [Alt61472-Alt61695]. (See also **Finding and replacing symbols**).

- [a-c] will not **only** match "a", "b", "c", but **also**:
  àáâãäåæąāă

  [a-à] will **not** match all characters from U+0061 (a) to U+00E0 (à).

  For some discussion on Unicode sorting and wildcards see the **Unicode Regular Expression Guidelines**. In general, even if the sorting used in Word is not very transparent, you usually get the results you would expect.

- If you are using VBA and doing string comparisons, the "Like" operator (covered in VBA Help) gives you much of the functionality of a wildcard search.

- If you are using VBA, you might want to look at the RegExp object (from VBScript, which you can include in your macro projects, and which offers some features not included in Word wildcards). Choose Tools + References in the VB editor, and check "Microsoft VBScript Regular Expressions". You can get help if you call the VBScript editor and search its online help for RegExp. A list of supported wildcards is given in the help on the Pattern-property.

## Gremlins to be aware of (for advanced users)

- Sometimes Word will get confused if it encounters "escaped" brackets **\(** or **\)**. For example:
  **(\\)**
  will match any character, not only a backslash.

  **Workaround:** use
  **([\\])**
  instead.

- ([a-z]\() throws an error – it should find an "a(".

  **Workaround:** Use ([a-z][\(]) instead.

- Word starts matching again **after** the previous match/replacement; so for example **^13*^13** will match only every second paragraph in a Replace operation.

  If that isn't what you want, you could (in this example) use *^13 instead.

  Other times, if this isn't what you want, you have to do multiple Find & Replaces. For example, supposing you wanted to subscript all the numbers in any chemical formula such as C2H4. You could tag the numbers that have to be subscripted; you want to subscript any group of numbers ([0-9]@) that follow a letter > ([A-Za-z]), and that are not followed by another number ([!0-9]):

  Find what: **([A-Za-z])([0-9]@)([!0-9])**

  Replace with: **\1<sub>\2</sub>\3**

  That search will for example match “C2H” in “C2H4” (letter + number + non-number), and then continue searching **after** the “H”. So you need to run this replacement twice, before doing the Find and& Replace that applies the formatting.

  But sometimes, Word finding every second instance of your search string may **be** what you want, and you can make use of the feature. (For instance, if you want to format the text between two identical tags).
- Not a bug but still annoying: You have to escape any special character even if you type its code; so ^92 will have the same problems as typing the backslash.
- The construction **{0,}** (find zero or more of the preceding item) is refused as incorrect syntax. This concept is available in Unix regular expression matching, so it's a curious omission.
- You don’t always have to “escape” the special characters, if the context makes it clear that the special meaning isn’t wanted. [abc-] matches “-”, and [)(] matches “)” or “(”. This may sometimes make your searches behave differently from what you expected.
- See also:
  **Flush bad karma from Word's find facility after an unsuccessful wildcard search**
  and
  **How to prevent the built-in BrowseNext and RepeatFind commands from creating bad karma for wildcard searches**

# Is there life after "Reveal Codes"?

Article contributed by Suzanne S. Barnhill

One of the questions most commonly asked by migrants to Word from WordPerfect is, "Where is Reveal Codes?" or "Does Word have anything like Reveal Codes?"

There is nothing in Word directly comparable to Reveal Codes in WordPerfect. There is a very good reason for this. WordPerfect can be thought of (and I understand is) basically a text stream with codes interspersed (for more on this, see John McGhie's article on Word vs. WordPerfect). This is what you see when you Reveal Codes. You have codes or markers that turn on and off certain formatting characteristics. Word, on the other hand, is a series of nesting containers, characters inside words inside paragraphs inside sections inside documents. The formatting of these is by styles and by pointers at the beginning and end of the document. I am reliably informed that if you open a Word document in a hex editor, you see a forest of gibberish at the beginning and end that represent these codes and pointers (you can get a small idea of this by opening a document using the Recover Text from Any File setting under "Files of type" in the File Open dialog). So Reveal Codes, even if there were such a thing in Word, would not be very helpful.

But there are many helpful cues and clues in Word if you know how to use them. The feature commonly touted as Word's equivalent to Reveal Codes is the "What's This?" button on the Help menu. Click on that (or press Shift+F1), then click in a paragraph, and you'll get information about formatting applied both by the style and directly. Rarely, however, does this tell you much more than you can tell by just looking at the paragraph on screen.

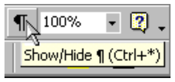
The "What's This?" button has been replaced in Word 2002 by the Reveal Formatting task pane, which gives you specific information about the text at the insertion point (font, paragraph, and even section formatting). To display the task pane, View | Task Pane (or click on the New or Styles and Formatting toolbar button), then click the down arrow at the top of the task pane and choose Reveal Formatting. If you check the box for "Distinguish style source," you can easily see what formatting is part of the style and what has been directly applied.

As a general rule, you want to avoid direct formatting. If you use styles to do your formatting and avoid manual formatting as much as possible, you will find that simply looking at the style name in the Formatting toolbar or Style Area will usually tell you all you need to know.

More importantly, use styles to do your formatting and avoid manual formatting as much as possible; you will then find that simply looking at the style name in the Formatting toolbar will usually tell you all you need to know.

Here are some more things that do help, in my experience.

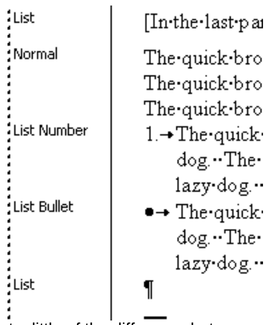
1. The most important one is display of nonprinting characters. The Show/Hide ¶ button on the Standard toolbar toggles this display on and off.



   The meaning of each of the nonprinting characters (or "formatting marks," as they are known in Word 2000 and 2002), is explained in **What do all those funny marks, like the dots between the words in my document, and the square bullets in the left margin, mean?** With these symbols displayed, it is much easier to find out that, for example, your document is printing an extra blank page because you have half a dozen empty paragraphs at the end.

2. The second most important is display of text boundaries in Print Layout view. You'll find this on the View tab of Tools | Options for that view. This is helpful not only in visualizing margins but also for seeing the outlines of graphics (even when they're not selected) and table cell boundaries (even when gridlines are not displayed). Some people evidently prefer the cleaner page appearance they get without this display, but for a truly WYSIWYG view, you can always switch to Print Preview.

3. While you're looking at the View Options, be sure to check the box for "Drawings" and clear the check box for "Picture placeholders." For more on this, see **I inserted some graphics in a document, but now I can't see them; or there is just an empty box where one should be; or my graphics won't print**. You will probably want to check the box for "Object anchors," which help you see what paragraph a floating graphic is anchored to, but you may prefer to display bookmarks only when troubleshooting because the heavy brackets are ugly and distracting (and hidden bookmarks are not marked, anyway). Other important View Options include the vertical ruler, status bar, and ScreenTips. Display of the horizontal ruler is toggled on the View menu.
4. In addition to the above, there are other visual cues. With the insertion point in a given paragraph or word, or with a portion of the text selected, you can look at the toolbars to see what style is in use, what font and point size are being used, and whether bold or italic formatting is in effect. Although you can usually tell just by looking at it whether text is bold or italic, problems can often arise from incorrect formatting of the paragraph mark itself. With the insertion point in a paragraph, you can see from the ruler whether any paragraph indents are in effect. If you select the entire paragraph, you can see (from the extent of the block selected) whether the paragraph has some Space Before or After.
5. One of the most useful tools when working with an entire document is the Style Area (see to the right). This is available only in Normal view. On the **View** tab of **Tools | Options**, set the style area width to a value greater than 0" (1" is usually sufficient unless your style names are very long). Word will then display the style for each paragraph in your document, and you can scroll through and check for incorrect or improperly formatted styles. Double-clicking on a style name in the Style Area will bring up the Style dialog box with the current style selected.



By using all these visual cues, and by understanding at least a little of the difference between the Word and WordPerfect object models, you will gradually be able to wean yourself from the desire for "Reveal Codes."

# How Word differs from WordPerfect

Article contributed by John McGhie

WordPerfect considers a document to be a “type stream.” If you picture WordPerfect sitting on the end of the printer cable, sending characters one-by-one, and every now and again inserting a COMMAND to change what the printer is doing, you'll get the idea. For example, WP sends the commands for “Arial” font and “bold”. It then expects the printer to print every character that way until it tells the printer to do something else.

Word, on the other hand, considers a document to be a “container.” Within this container are more containers and, within them, still more. Into each of these containers, Word inserts objects. The objects can be bits of text, or bits of pictures, or complete files created by other applications.

To make a document in Word, you create containers, fill them, and manipulate their properties to decide their position and appearance. For example, you create a container called a “paragraph”. Into it you place a string of text like this sentence. Then you apply properties to it such as, “Font=Times New Roman,” “Size=12,” “Face=Bold,” “Language=US English,” “Space before=12,” “Line height=1,” etc., etc., etc.... The properties of a container affect all the objects inside it, including other containers. For example, each character is a container that can have a font, size, and colour, but it inherits these properties from the paragraph that contains it unless you override the paragraph properties for that character.

Regrettably, long ago, Microsoft decided that this mechanism was far too complicated for us to understand, so they hid it from us. If you ever open a Word document in a hexadecimal editor and have a look, you will soon discover that they were right. A Word document internally is a maze of pointers to the various containers, which appear in the file in a sequence unrelated to the order in which they print.

I would make the following suggestions for a long and happy life with Word:

1. Forget reveal codes. They are not useful in Word. If you need them, click the “What's This?” button on the Help menu (or press Shift+F1), and then click the text you are interested in. That will show you most of the current properties. Also see: **Is there life after "Reveal Codes"?**
2. What you see on the screen is what you are going to get. If the text on the screen does not appear bold, then it does not have the bold property, regardless of what you feel it “should” have :)
3. As far as humanly possible, avoid direct formatting in Word. Word is designed to run on Styles. Learn to use them. It is so much easier to get one style correctly formatted than it is to get 174 paragraphs all looking the same. Ignore the format painter: it's a problem looking for a place to happen.
4. Do not accept Word the way it came out of the box. Customise the hell out of it. That's why it was designed that way. Start with the toolbars: piss off all the rubbish that comes on the standard toolbars and add your favourite styles and tools in their place, so everything you need is only one click away. Get into macros as soon as you can: there's no point in fiddling around typing things you can simply assign to a mouse-click.
5. Learn to use a different template for each document type. That way, you can make the template automatically set up all your styles, margins, spelling languages, etc., for the particular type of document you are making. Avoid basing documents on the “Normal” template. You can never control the contents of the Normal template, on your computer or on anyone else's. Documents attached to the Normal template will reformat themselves each time they are opened or passed to a different machine.
6. Don't be tempted to customise Word to work like WP. You can do it, but Word will fight you every inch of the way if you do, and you will have a very frustrating time of it.

Terms of use Disclaimer

7. Don't expect to read anything useful about Word in a paper manual. All the information is in the Help. Suffer that damned paper-clip and learn to use it: it's the fastest way to find anything. That's because there's too much information and it updates too frequently to be published on paper. I have found that anyone who has published a paper book about Word 97, for example, either hasn't understood it, or has done it at such a trivial level that it is not useful. Since I write books for a living, I can tell you that it is just not possible to describe Word in less than 1,800 pages, and it's just not possible to economically keep a book like that up-to-date.
8. Resign yourself to the fact that it will take six months of daily use to really tame the brute, but once you have, you won't go back.

See also:
**WordPerfect to Word converters (and why none of them are perfect)**
**Is there life after "Reveal Codes"?**

# The draw layer: a metaphysical space (and how to bring it back down to earth)

*Article contributed by Dave Rado and Bill Coan*

Word's draw layer is a metaphysical space where floating objects reside. It really isn't a layer, since floating objects can be sent behind the text layer or brought out in front of it. Either way, they continue to reside in the draw layer.

The relationship between the draw layer and the text layer is a bit unusual, to say the least. As noted, the draw layer is both behind and in front of the text layer. Any floating object can be right-clicked to bring up a menu that includes an “Order” command that lets you send the object behind or bring it out in front of the surrounding text (as well as behind or in front of any other floating objects).

Any objects which you can insert using the Drawing toolbar – for instance, Text Boxes – reside in the Drawing layer.

## *Some things to watch out for with objects in the drawing layer*

**1.** Any text residing in the Drawing layer is immune to many of Word's standard features such as Caption numbering, auto generated tables of contents, tables of figures etc.

If the above features are critical to your document, it is therefore usually better to use Frames rather than Text Boxes. Frames float, like Text Boxes, yet reside in the text layer, so are accessible to the above features.

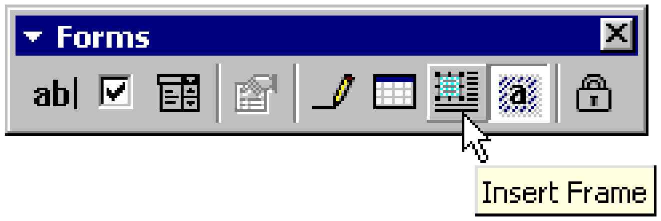
This is particularly an issue when you add a caption to a floating graphic. By default Word puts the caption in a textbox, which means it won't update and won't appear in the Table of Figures. To get round this you need to:

**a)** Convert the floating graphic to an inline graphic. Select **Format + Picture**, and if using Word 97, click the Position tab and uncheck Float Over Text; if using Word 2000~2003 click the Layout tab and select Inline with Text.

**b)** Delete the textbox, and reinsert the caption in the text layer.

**c)** If you really need the graphic to float, you can then select both the inline graphic and the caption simultaneously and put them into a frame.

To put them into a frame you need to use the **Insert Frame** command, which has been cunningly hidden on the Forms toolbar; you can get at it by right-clicking on any toolbar, selecting the Forms Toolbar, and clicking the Insert Frames button:



You can make the button more easily accessible for when you next need it by selecting **Tools + Customize**, and holding the Ctrl key down while you drag the button from the Forms toolbar to the Insert menu, where it rightfully belongs. Or see: How can I add the Insert Frame command to the Insert menu?.

**2.** Floating objects (even frames) can be a maintenance nightmare in any document that will ever need to be amended or pasted from. They should certainly be avoided in long documents, unless there is no alternative but to use them. Strange things tend to happen when you add text which precedes a floating object – such as part of it suddenly appearing in the footer of one page and the rest appearing in the header of the following page!

You can generally simulate text-wrapping quite convincingly by putting your (inline) graphic into one cell of a borderless, single-row, two-cell table; and putting your text in the other cell.

If you need to have call-outs pointing at various parts of a graphic, one way of getting around the problem is to do a screen capture of the complete drawing, paste it into a graphics package, crop it, then paste it back into Word as an inline graphic. Or another workaround is to create the drawing in another application such as PowerPoint, and paste it into Word inline. There are others as well, but these two are probably the simplest.

**3.** Floating objects use more memory. This can be an particular problem when printing. If your pages fail to print in full or print fuzzily, converting floating graphics to inline ones, and using tables rather than textboxes, generally fixes it.

Using tables rather than textboxes is usually to be preferred in any case – for some reason which I've never understood, inexperienced Word users tend to create their “tables” by putting several textboxes alongside each other! Please don't fall into this trap! Tables are far easier to create, maintain, and work far better as tables than textboxes do. Only use textboxes when tables really can't be used.

See also
I inserted some graphics in a document, but now I can't see them; or there is just an empty box where one should be; or my graphics won't print

and the following Microsoft Knowledge Base articles:

WD97: General Information about Floating Objects

WD2000: General Information About Floating Objects

# Cleaning up text pasted from the Web

Article contributed by Suzanne S. Barnhill and Dave Rado

The ease of copying and pasting text from Web sites and email greatly simplifies many tasks in Word, but problems often arise in making the pasted text conform to the style of the document into which it is pasted. One of the most common chores is getting rid of excess line breaks, which cause the text to wrap short of the right margin. There are several ways to work around this problem.

## Assessing the problem text

The most efficient method of reformatting short lines of text depends on whether the breaks are line breaks or paragraph breaks. So the first line of attack must be to display nonprinting characters using the Show/Hide button on the Standard toolbar. (For more on nonprinting characters, see **What do all those funny marks, like the dots between the words in my document, and the square bullets in the left margin, mean?**) If each line ends in a pilcrow or paragraph mark (¶), then AutoFormat may be all you need. If each line ends in a bent arrow ↵ (signifying a line break), you will need to use a different approach.

## Using AutoFormat

No matter what other options you have enabled on the AutoFormat tab of Tools | AutoCorrect, when you select a block of text with a paragraph break at the end of each full line, AutoFormat will delete all the paragraph breaks but the last.

Unfortunately, text pasted from the Web or email nowadays rarely has lines ending in paragraph breaks. But you can force this format by using **Paste Special** and selecting “Unformatted Text” (in Word 2002, if you have “Paste Options” enabled, you can just **Paste** and then select the “Keep Text Only” option). This pastes your selection with paragraph breaks instead of line breaks, and AutoFormat will then do the trick.

## Using Find and Replace

Sometimes, however, you will not want to paste as unformatted text. In that case, what you will most likely get is text with a line break at the end of each line. Provided there is an empty line at the end of each paragraph, cleanup is still relatively simple. It takes just two Find and Replace operations.

### First pass

1. Press **Ctrl+H** to open the Find and Replace dialog.
2. In the “Find what” box, type **^l^l** (those are lowercase Ls, representing two line breaks).
3. In the “Replace with” box, type **^p** (the code for a paragraph break).
4. Replace All. You will now have a paragraph break at the end of each true paragraph.

### Second pass

1. In the “Find what” box, type **^l**.
2. In the “Replace with” box, type a space.
3. Replace All.

This removes the line breaks and allows text to wrap naturally.

### Harder cases

If there is not an empty line between paragraphs, you will probably have to insert paragraph breaks by hand. If the amount of text is not large, you can scroll through and press Enter where a paragraph break is needed. Then replace each line break with a space. This will leave an extra space at either the beginning or the end of each paragraph. You can use Find and Replace again to replace **<space>^p** or **^p<space>** (as appropriate) with **^p**. (Note that "<space>" represents pressing the spacebar, you don't type "<space>")!

An alternative approach is to **Shift+Enter** to enter an extra line break at the end of each paragraph, then follow the instructions in the section above.

Even when the amount of text is very large, there is no really good alternative to manual editing. But if you Paste Special as Unformatted Text and run AutoFormat, you may find that Word is almost as clever as you are at finding where a paragraph ends.

Note that the methods described above are suitable only for simple text. If you have copied and pasted an entire Web page, with graphics, tables, and frames, much more work will be required to format it for use in a Word document.

### Other non-printing characters worth replacing

- Often when you paste from the web, and also from some other applications, characters come in which *display* as paragraph marks but don't behave like "proper" paragraph breaks should – they behave like manual line breaks!. So you might find that when you center a "paragraph", several other adjacent paragraphs also get centered. To cure this, do a Find and Replace; in the "Find what" box type **^013**, in the "Replace with" box, type **^p** and click "Replace All".
- When°pasting°from°the°web,°nonbreaking°spaces°often°come°in,°rather·than·ordinary spaces. To get rid of them, do a Find and Replace; in the "Find what" box type **^s**, in the "Replace with" box, insert a **<space>** character (press the spacebar), and click "Replace All".

If you want to automate any of the above steps you can record them using the macro recorder and play them back as needed.

## Neat tip

This following tip has appeared in Woody's Office Watch (WOW). When you cut and paste text from a Web site, there are often leading spaces at the beginning of each line. A very quick way to remove all these spaces is to select the text, center justify the selection and then left justify the selection. All the extra spaces will have disappeared.

Send all feedback to **word@mvps.org** Top - **Ctrl+Home** Previous - **Alt+left arrow** Page Updated:

# How to create a mail merge

Article contributed by Beth Melton and Dave Rado

Terms of use • Disclaimer

**(to return to top, press Ctrl+Home)**

## 1. Mail merge basics

This article will step you through the basics of creating a mail merge and contains links to some of the more advanced features.

Sometimes the term “mail merge” can be a little misleading. We assume from the title that the intent of mail merge is to produce letters for mass mailing purposes. That's not necessarily the case.

Mail merge is for simplifying repetitive documents and tasks. Mail merge can be used for creating many documents at once that contain identical formatting, layout, text, graphics, etc., and where only certain portions of each document varies. Mail merge is also used for generating mailing labels, envelopes, address lists, personalised training handouts, etc. As well as hard copy mailshots, it can be used to generate multiple emails and electronic faxes. And it can even be used to create a “friendly” front-end to spreadsheet or database information.

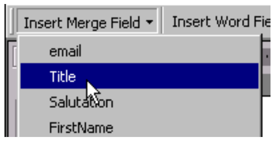
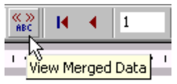
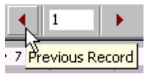
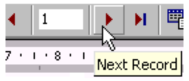
Whenever you need to assemble similar data, mail merge is the answer!

Mail merge primarily consists of two files, the **Main Document** and the **Data Source**. The Main Document contains the information that will remain the same in each record, and the Data Source contains all the variable information, in the form of *fields*. This is the information that will change in the Main Document when the merge is completed. Along with the information that remains the same, the Main Document also contains merge fields, which are references to the fields in the Data Source.

When the Main Document and Data Source are merged, Microsoft Word replaces each merge field in the Main Document with the data from the respective field contained in the Data Source. The end result is a third document, a combination of the Main Document and Data Source – although you can also mail merge directly to the printer; (or fax or email) – you don't need to create a merged document on screen; and you can also “preview” the mail merge without actually merging (using the **ViewMergedData** button).

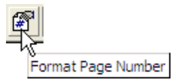
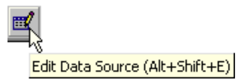
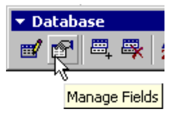
| Step | Comments |
|---|---|
| **1.** If you have an existing file you want to use as the merge document open it now. | For WordPerfect users this would be the primary file |
| **2.** Start the Mail Merge Helper by going to **Tools/Mail Merge**. | |

| | |
|---|---|
| **3.** Select the type of Main Document you wish to create.<br><br>You will be asked whether you want to use the Active Window or a New Document. To use the document you opened in Step 1, select **Active Window**.<br><br>If you select **New Document**, Word will create a blank document based on the Normal template. For most purposes it is much better to use a template specific to the task in hand rather than the Normal template; so **Active Window** is usually the better choice. | **Form Letters** – A Section Break is placed between records in the completed merge.<br><br>**Mailing Labels** – Records are merged to a table representing the label layout. If you use the wizard to create your labels, Word will insert a «Next Record» field at the start of every table cell except the first one. If creating your own customised label template, you will need to insert the «Next Record» fields yourself (you can use the "Insert Word Field" button to do this).<br><br>**Catalog** – Records are merged continuously; use for mailing lists, telephone directories, etc.<br><br>**Envelopes** – Similar to form letters except that an envelope paper definition is used. |
| **4.** Click **Get Data** to select your Data Source. There are four methods you can use for your Data Source: | |
| **a)** Create Data Source | This uses **Word** as the Data Source; setup your fields and type the data for the first time. The data is then stored in a Word table, which can easily be transferred to an Excel Data Source later if required.<br><br>Note that you may find even a Word Data Source easier to set up without the wizard once you know what you're doing, as it's nothing more than a Word table with the items in the Heading Row of the table representing the mail merge fields – although you may well need to use Normal View to view all the data in the table if you don't want the columns to be incredibly narrow. Some experienced users prefer to use the wizard for this reason. |
| **b)** Open Data Source | Lets you browse to and open a **Word**, **Excel**, **Access (or any supported database)** or **text file** Data Source. |
| **c)** Use Address Book | Use an **Outlook** or **Outlook Express** address book as the Data Source. (It can also link up to an Exchange *.pab and a Schedule+ Contacts list). |
| **d)** Header Options | If a Data Source from another program does not contain a Heading Row, or if field names in the Heading Row of your Data Source do not match the merge fields in your Main Document, you can use a separate Header Source and specify the field names.<br><br>● Field names must be listed in the same order as the corresponding information in the Data Source.<br><br>● Field names in the Header Source must match any merge fields you've inserted in the Main Document.<br><br>However, where possible, build the Heading Row into your Mail Merge Data Source rather than having a separate Header Source – it simply makes life easier.<br><br>Note that if your Data Source **does** contain a Header row, and if you nevertheless use a separate Header Source, the Header Row within the Data Source will be treated as a mail merge record – so be careful. |

| | | |
|---|---|---|
| **5.** | Select **Edit Main Document**.<br>Add all text that you want to appear with every record; and insert the appropriate merge fields using the **Insert Merge Field** button on the Mail Merge Toolbar.<br> Insert Merge Field<br>Insert Word Fie<br>email<br>Title<br>Salutation<br>FirstName | You can use as few or as many of the data fields as you need to, and you can use them in any order. Also, any field can be used more than once.<br>In a catalog merge, this text will appear with each record. If you want text to appear either before or after all records, it's best to add this to the completed merged document. |
| **6.** | You will want to add appropriate punctuation and spaces between the merge fields. | Although blank paragraphs are suppressed automatically when you merge, spaces *within* a paragraph are not.<br>Frequently (for instance, in an address) you will want to suppress spaces and punctuation following a field if the field is blank – for example, if you have "«Street, «City»", then if there is no street in a given address, you won't want the comma and space to display.<br>You can achieve this flexibility by using **IF** fields. See: **Making your mail merge "intelligent" by using IF fields**. |
| **7.** | As well as the obvious places, such as the address, you may also want to add mail merge fields at strategic points in the main body of the document – "«so-and-so» will be contacting you", etc.. | Frequently, you may want to make the results that are displayed dependant on a condition. For example, you may want to display "he" or "she", depending on gender; or you might want to display either: "please let us know your start date", or "you will commence employment on 01 January 2001", depending on whether the start date is known.<br>Again, see: **Making your mail merge "intelligent" by using IF fields** for details of how to do this. |
| **8.** | You can preview your data prior to performing the merge by clicking the **ViewMergedData** button on the mail merge toolbar. Doing this is *far* quicker than actually merging, if the Data Source is reasonably large.<br>You can skip though the records by pressing the **Next** and **Previous** buttons.<br>[You can also go to a specific record using the **Find** button (binoculars). This works fine under Word 2000 but can cause problems if you are using Word 97 – see: **Why is mail merge unavailable after using the Find Record feature in Word 97?**].<br>Having proof-read it in this way, you can merge straight to the printer if you want to, *without* having to create a merged document first. |  <br> <br> <br>A couple of minor "gotchas" with using **ViewMergedData**:<br>● Blank paragraphs are not suppressed until you do the merge. However, if you have the ShowAll button on, you will see a " symbol to indicate a paragraph that is going to be suppressed, instead of the usual ¶ symbol for a normal paragraph.<br>● Fields based on the total number of pages tend not to work as expected following a merge (e.g. Page X of Y), but this usually won't be apparent until you do a full merge. |

| | |
|---|---|
| **9.** You may well need to **filter** the records, in order to merge only to a subset of your Data Source.<br><br>You may also want to change the **sort** order for the merge without affecting the sort order of the Data Source.<br><br>To do this, you can use **Mail Merge Query Options** | For instance, you may be doing a follow-up mailshot to those people who didn't reply to the original mailshot. Or you may want to send a letter only to people whose birthday is this week.<br><br>For details of how to do this, see: **Turning Word into a pseudo-database by using Mail Merge Query Options**. |
| **10.** Prior to performing the merge, hold <Shift> and then select File/Save All. This will save the Mail Merge Main Document; and if the Data Source is a Word document, it will save that too. | For additional formatting techniques, tips and tricks, and special merges such as setting up an Address Book, phone lists, merging graphics, suppressing duplicates, One-to-Many Relationships, see the **Mail Merge home page** on this site. |
| **11.** Perform the merge. Merge choices are:<br>● Merge to New Document<br>● Merge to Printer<br>● Merge to Email<br>● Merge to Electronic Fax | For large mail merges, merging straight to the printer generally works better and is much faster than merging to a new document and printing that.<br><br>If merging to Email or Electronic Fax, click the **Setup** button on the **Merge** dialog when you're ready to merge. Then select the **Data field** which contains the email address or fax number, type the **subject line**; and in the case of Email, choose whether or not you want the document to be sent as an **attachment**. Finally, you may also want to use **Query Options** to filter out any records that don't contain an email address/fax number. |

## 2. General mail merge FAQ

| Question | Solution |
|---|---|
| Page numbering doesn't restart with each record in completed merge. | **1.** Go into the Header or Footer.<br>**2.** On the Header/Footer toolbar, click the Format Page Number button:<br> <br>**3.** Click Start At and enter the desired value.<br>**4.** Click OK to close the dialog. |
| How do I amend, add or remove mail merge fields? | **1.** Select **Edit Data Source** on Mail Merge Toolbar<br> <br>**2.** If the Data Source is an Excel spreadsheet or Access database, this will open it (or switch to it, if it's already open).<br><br>If the Data Source is a Word document, the Data Form dialog will now appear. Click the **View Source** button to open the Data Source.<br><br>**3.** To amend a field, edit it in the Heading Row; to add a field, add a column to the table; to remove a field, delete the relevant column.<br><br>Alternatively, if the Data Source is a Word Document, you can select **Manage Fields** on the Database Toolbar.<br> <br>This makes life a bit simpler, especially if it's your first time. Then save and Close the Data Source. |

| | |
|---|---|
| My email addresses aren't inserted as hyperlinks in the completed merge. | **1.** Select **Format/AutoFormat** and then **Options**<br>**2.** Turn off all formatting options that you do not want, but ensure that **Internet and network paths as hyperlinks** is turned **on**. |
| Merge to electronic fax option missing | **Follow this link** |
| How do I get an image such as our company logo into my mail merge labels? | Assuming you want the same picture in every label, just insert it **inline** into every table cell in the mail merge Main Document.<br>If you don't want the logo to print for cells where there is no data (so you can avoid wasting labels), you can use a combination of an **IF field** and an INCLUDEPICTURE field:<br>**{** IF **{** MERGEFIELD FirstName **}** <> "" **{** INCLUDEPICTURE "C:\\Temp\\Logo.tif" **\d *** MERGEFORMAT **}** "" **}**<br>You could also create an AutoText entry for the image or for the IncludePicture field. To use it, in the Label Setup dialog box, sample label text area, type the name for the AutoText entry and press <F3> to insert it.<br>Or you could create a label **template** with the image or field already included. |

## 3. Related articles on this site

**Mail Merge home page**

**Creating a Mail Merge Data Source**

**Making your mail merge “intelligent” by using IF fields**

**Turning Word into a pseudo-database by using Mail Merge Query Options**

## 4. Helpful mail merge Knowledge Base articles

**How to convert WordPerfect merge data documents to Word**

**How to convert WordPerfect merge data**

**How to use nested IF fields in a mail merge document**

**How to use form data as Mail Merge Data Source**

**Performing calculations in a mail merge field**

**How to design and set up Mail Merge Data Sources**

**How to convert WordPerfect 6.x data files and address books**

**How to create business cards in Microsoft Word**

**How to merge envelopes and labels if records in one column**

**How to merge conditional number of records to the same page**

**How to convert data in one column so that you can use the data in a mail merge**

**How to merge a single Data Source to multiple documents**

**How to use Microsoft Access data in Word**

**HOWTO: Automate Microsoft Word to Perform Mail Merge from Visual Basic**

Send all feedback to **word@mvps.org** Top - **Ctrl+Home** Previous - **Alt+left arrow** Page Updated:

# Creating a Mail Merge Data Source

Article contributed by Beth Melton

Microsoft Word supports many file formats which can be used as a Data Source for a mail merge. This article covers specifications and frequently asked questions on the most commonly used Data Sources, along with how to set up a Data Source in Word.

**(to return to top, press Ctrl+Home)**

## 1. Overview

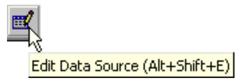
Break your fields down into the smallest meaningful value. For example, create a separate field for First and Last names, break the City, State, and ZIP of an address into separate fields as well. Combining information in separate fields in a merge document is easy; separating information in a field is difficult. For an example of *how* difficult, see: **I have a "Name" column which I want to split into "FirstName", "LastName", how can I do it?**

You can refer to the First name of an individual in a salutation and then use the First and Last in the inside address. However, using a separate “Salutation” field gives you much more flexibility, because it lets you use “Joe” or “Mr Bloggs” as appropriate.

If you wish to use barcodes, the delivery address and postal code must be placed in separate fields Should you wish to sort by a specific piece of your information, it must be placed in a separate field. This is important if you want to be able to view or print data in alphabetical order by last name for a mailing list or directory but in ZIP code order when creating mailing labels or envelopes to qualify for bulk mailing rates.

## 2. Using Word as a Data Source

| Step | Comments |
|---|---|
| **1.** In Step 2 of the Mail Merge Helper, select **Create Data Source**. | |
| **2.** Word displays a dialog box containing a default list of field names. You can use the fields provided and add/remove fields, or remove them all and create your own field names.<br>The order of the fields will become the data entry order. Use the Move Up/Move Down Commands to alter the order.<br>When all fields have been specified, select **OK** and Word will ask for a file name for your Data Source file. | Field Name specifications:<br>● Each field name must be unique.<br>● Field names can be up to 40 characters long.<br>● Word field names cannot contain spaces.<br>● No more than 63 fields can be used (Word 97 and later). If more are needed, then use an alternative Data Source. (In Word 95 and lower the limit was 31 fields). |

| | |
|---|---|
| | Your Data Source is stored in a Word table. The first row in the table is called the **Header** row. Each row contains one record, and each column one field. |
| **3.** Next you will be asked to set up the Main Document or to Edit the Data Source. Select E**dit Data Source**. | Word provides you with a data entry form to enter your records. Use either <Tab> or <Enter> to move from field to field. Once you enter the data in the last field, press <Enter> again to move to a new record. Otherwise select Add New Record. To move to a previous field use <Shift Tab>. |
| **4.** After all records have been entered you are ready to edit your Main Document. | To return to the Data Source, locate **Edit Data Source** on the Mail Merge Toolbar.  Edit Data Source (Alt+Shift+E) |

Note that you don't have to use the data entry form. If you find it more convenient (as many users do), you can work directly with the data table by pressing the View Source button in the entry form.

Moreover, you can create this data table independently before beginning the merge process. In a blank document, insert a table with as many columns as you want merge fields. The first (heading) row of the table should contain the merge field names. Don't leave a blank row below the Heading Row; if you do, you will have a blank record, and the first page, item, label, or envelope in your merge will have blanks instead of merged data. As you add records, the table will grow, and you can sort the data on any column. Save this just as you would any Word document. When using the Mail Merge Helper, instead of choosing Create Data Source in the **Get Data** step, you will instead use **Open Data Source** to select the document in which you saved the data.

## 3. Using Excel as a Data Source

A Data Source in Excel, referred to in Excel as a list, must meet the following specifications:

- Field names/column labels must be in the first row of the list, and field names must be unique.
- There **must not** be a blank row between the field names and the first record.
- If there is additional information on the worksheet, there must be at least one blank row or column separating the list from the other data.
- There should be no empty columns or rows within the list.
- If the field names/column labels are not in Row 1 of the worksheet, either create a named range for the list or add the AutoFilter feature to the list (**Data/Filter/AutoFilter**). The AutoFilter feature will automatically create a named range called _FilterDatabase. If you want to use lists from other worksheets, however, then using a named range is the best method. Note that if you ever add records to the bottom of the list you would need to redefine the named range. See Select Method (below) for additional details.

| Question | Solution |
|---|---|
| Word tells me it can't establish the DDE connection to Excel or Excel just won't start. | In Excel, go to **Tools/Options/General** and turn **off** "Ignore other Applications". |
| I can only see the first worksheet of my Excel workbook when I open my Data Source. | By default Word uses the **DDE method** to connect to Excel. In the Open Data Source dialog box, check the "Select Method" option to view other methods. **Select Method Notes:** **Excel files via ODBC**: Named ranges are displayed by default. To specify a worksheet, click Options and select System Tables. **Microsoft Excel Worksheet via Converter** will convert your worksheet into a Word document. If the file is saved in Word, it will permanently convert your Excel workbook into a Word document. You will no longer be able to open this file in Excel. Be cautious when using this method. |
| My ZIP codes are missing leading zeros. | In Excel, select **Format/Cells**, and on the **Number** tab in the **Category** section, select **Special**, and apply the Zip code format. |

| | |
|---|---|
| An AutoFilter is applied to my list in Excel and I'm getting blank records in Word. | In Word use **Tools/Mail Merge** and then select **Query Options**. Filter Records by one field using the "is not blank" criterion. Preferably this field should be one that would always contain an entry. |
| How you can use MS Query to merge from two Excel files in a single mail merge. | See KB Article **Q180599** |

## 4. Using Access (or any supported database) as a Data Source

An Access table or query is already in a format that Word will recognize as the Data Source.

In addition to Access, any database for which Word has a file converter (including Foxpro, dBase, and WordPerfect and Works databases, among others) can be used as a Data Source. Also, if you use VBA to open the Data Source, any ODBC-compliant database can (at least in theory!) be used as a Data Source. And finally, any database format supported by MS Query can (in theory) be used. In the case of bespoke Database Management Systems, it is generally simpler and quicker to export a comma-delimited text file from the database and use that as the Data Source.

| Question | Solution |
|---|---|
| Help! My phone numbers are not formatted correctly. | Use **Alt+F9** to turn on the display of field codes. Edit the merge field for the phone number and add the following format:<br>**{** MERGEFIELD strPhone \# "(###) ###'-'####" **}**<br>Note the apostrophes around the hyphen.<br>To prevent the phone number from breaking during word wrap, you can use a nonbreaking space, **Ctrl+Shift+ Space**, between the area code and prefix and a nonbreaking hyphen, **Ctrl+Shift+ Hyphen** between the prefix and number; in which case you won't need the apostrophes.<br>However, if some of the phone numbers are stored with the symbols and others are not, this will not correct the problem . In this circumstance, Word will display the calculated result of the phone number. So the best way to correct the phone numbers is in Access rather than in Word.<br>In an Access Query, add a calculated field that uses the Format function to correct the field. It would look something like:<br>Phone#: Format**(**[strPhone],"(000) 000-0000"**)** |
| Why does Word start another instance of Access? | This is Word is looking for a window titled Microsoft Access. If the database has an Application Title then a new instance of Access will be started. For a workaround, see Access MVP **Dev Ashish's** Web site. |
| I have pictures stored in Access but I can't get them to merge with Word – why not? | See: **How to merge in a picture from a database**. |

## 5. Using a Text File as a Data Source

Many database management systems (for example, most Human Resources systems) are difficult or impossible to merge directly to, but are very easy to export data from, as a comma delimited text file. The comma delimited file can then be used as a Data Source; or alternatively, can be imported into Excel and saved in Excel format, and the resulting spreadsheet can be used as a Data Source.

**Important:** It is always best to give the text file an extension of **.dat**. The ODBC drivers for Windows 95 are broken, but if you give your text file a "non-standard" extension, such as **.dat,** the ODBC driver will not recognize it, and Word's internal text converter (which works) will take over.

Specifications for using a text file as a Data Source:

- Field names should contain no punctuation, and spaces are discouraged.
- Each record ends with a paragraph mark.

- There should be no extra paragraph marks between records or at the end of the data records.
- Use the same field delimiter in both the header row and the rest of the data.
- For empty fields in the data record, include the field delimiter anyway with the exception of the last field in the last record.
- Use quotation marks if the field contains the same character you are using as the delimiter, if the text contains a paragraph mark, line break, or list separator that is specified on the Number tab in the Regional Settings of the Control Panel. Note that the default is a comma (,).
- If the information in the field contains quotation marks use double quotation marks ("" ""). Only one set of quotation marks will display in the completed merge.

Example:

```
"FirstName","LastName","Address1","Address2"
"John","Doe","1000 1st St.",""
"Timothy","Smith","1455 Anywhere Ln.","Apt. 1"
```

| Question | Solution |
|---|---|
| Accented or extended characters lost in mail merge | See KB Article **Q195688** |

## 6. Using Outlook as a Data Source

| Question | Solution |
|---|---|
| Some Contact List fields are not available to Word | **Word 97:** See MS KB Article **Q141874**.<br>**Word 2000:**<br>**1.** Start Outlook.<br>**2.** Under **Outlook Shortcuts**, select **Contacts**.<br>**3.** Select **Tools/Mail Merge**.<br>See MS KB Article **Q225000**for additional methods. |
| How to modify the layout of the address. | **Word 97:** See KB Article **Q134901**.<br>**Word 2000:** A supplemental macro, *Modify Address Layout*, is available in **Macros9.dot.** |
| Contact information does not appear in Address Book | To verify or install the Outlook Address Book Service:<br>**1.** In Outlook, select Tools/Services.<br>**2.** Verify Outlook Address Book service is in the list of available information services.<br>To add the Outlook Address Book:<br>**1.** Click Add.<br>**2.** In the Add Service to Profile dialog box, select the Outlook Address Book from the list of available services.<br>If the Outlook Address Book Service is not an available service, see KB Article **Q173072**. |
| Mapi32.dll is an invalid MAPI library.<br>Login failed, unspecified error.<br>Word was unable to open the Data Source. | Outlook Express has been set as your default email program.<br>In Outlook Express go to Tools/Options and either clear the checkbox for “Make Outlook Express my default Simple MAPI client.” or “Make Outlook Express my default e-mail program.”. |
| | |

| | |
|---|---|
| C:\~~\~~~_virtual_file_~~~.olk Is Being Used by Unknown. Do you want to make a copy? | This is a temporary file that is created when you use the Outlook Address Book. When you receive the error message click **No**. A second error message should appear. When you receive it, select **Options** and then **Remove Data/ Header Source**. Note that your Main Document is no longer associated with your Data Source. You will need to reattach the Data Source via **Tools/Mail Merge/Get Data.** |

## 7. Using Outlook Express as a Data Source

The Outlook Express address book was not designed for use in Word. You can open the Address Book in Outlook Express, select File/Export, and use a format that Word does support.

See Knowledge Base article **How to Use the Outlook Express Address Book in Mail Merge** for step-by-step instructions.

# Making your mail merge “intelligent” by using IF fields

Article contributed by Dave Rado

**(to return to top, press Ctrl+Home)**

## 1. Why and How

Almost any mail merge will work better if you use IF fields, as the frequently used scenarios discussed below attempt to illustrate.

At their simplest, they can be used to suppress unwanted spaces and commas in an address, if a field is blank. They can also be used in the main body of a letter or other mail merge document for things like inserting “his” or “her”, “he” or “she”, depending on gender. At a more sophisticated level they can be used to insert different series of boilerplate files, depending on certain conditions stored in the data file, which could be used, for example, in an Employee Contract template.

At its simplest, an IF field works as follows: If a condition is met display one result, otherwise display another. A field of this sort would look a bit like this:

**{** IF **[**Condition**]** **[**Display Result 1**]** **[**Display Result 2**]** **}**

Where the curly brackets represent field braces, which you must insert by pressing **Ctrl +F9** (you can't type them). A real example might be:

**{** IF **{** MERGEFIELD Gender **}** = "Male" "Him" "Her" **}**

Again, *all* of the curly brackets must be inserted by pressing **Ctrl+F9**.

IF fields can be made more powerful than this, because you can nest them within each other.

For example, you can create a field with logic such as: “If Condition 1 is met, then if Condition 2 is *also* met, display Result 1”. This is equivalent to saying “if both conditions are met, display Result 1”; but unfortunately you can't use “and” in Word fields.

Such a field would look something this:

**{** IF **[**Condition 1**]** **{** IF **[**Condition 2**]** **[**Display Result 1**]** "" **}** "" **}**

The easiest way to create nested fields of this type is to create them separately, then cut and paste one field into the other.

You can also simulate “Or” statements, by using the logic: “If Condition 1 *is* met, display Result 1; on the other hand, if Condition 1 *isn't* met, but Condition 2 *is* met, also display Result 1”. That is a rather tortuous way of saying “if either condition is met, display this”, but it *does* work.

## 2. Suppressing unwanted spaces and commas within a line of an address

A mail merge address frequently begins with a **Title** field (Mr/Ms/Mrs/Dr/etc). Typically, you would want the Title field to be followed by a FirstName field; and you would want a space between the title and the first name.

But what if some of the records have a blank Title field? You don't want the address to start with a space! So to prevent that happening you can use a construction like the following:

**{** MERGEFIELD Title **}{** IF **{** MERGEFIELD Title **}** = "" "" " " **}**

In other words, if there is a title, display it, and follow it with a space; if there *isn't* a title, don't follow it with anything.

Or looking at the construction in more detail:

**{** MERGEFIELD Title **}**

inserts the title (if ther is one).

**{** IF { MERGEFIELD Title } = "" "" " " **}**

means if the title is blank (""), insert nothing (""), otherwise insert a space (" ").

Further down, you might want to have (in the UK) “Town, County” or (in the US) “City, State” on a single line. If, for a particular record, both fields contain some data, you would want the two fields to be separated by a comma and a space. If either field is blank, you don't want a space or a comma. You can achieve this as follows:

**{** MERGEFIELD Town **}{** IF **{** MERGEFIELD Town **}** = "" "" **{** IF **{** MERGEFIELD County **}** = "" "" ", " **} }{** MERGEFIELD County **}**

This inserts the Town if there is one; if there is no town, it inserts nothing in front of the County field. If there *is* a town, but no county, nothing is inserted between the two fields. Finally, if there is both a town and a county, a comma and a space is inserted between the two fields.

●●●

You can, of course, apply this principle outside addresses. For example, suppose you want to output a line like:

**Subject:** <<Subject>>

when the database field <<subject>> is not blank, but suppress the line completely when it is blank. To achieve this, tick the “Suppress Blank Lines in Addresses” field in the Mail Merge dialog, and use the field:

**{** IF **{** MERGEFIELD Subject **}** = "" "" "Subject: " **}{** MERGEFIELD Subject **}**

## 3. Displaying gender-specific information in a letter

You may have a mail merge letter which ends: “Please feel free to ring «AccountMgr» if you have any queries. [He or She] will be glad to do anything [he or she] can to help.”

This is simple. In your Data Source, create a column called “AcctMgrGender”. Fill it in with “M” or “F” against every row. Then at its simplest, you could use:

**{** IF **{** MERGEFIELD AcctMgrGender **}** = "M" "He" "She" **}**

and

**{** IF **{** MERGEFIELD AcctMgrGender **}** = "M" "he" "she" **}**

respectively. However that would display “He” and “he” against any records where the “AcctMgrGender” field was blank – dangerous! One way of testing for this would be to use **Mail Merge Query Options** to display all records for which the field was blank; another would be to sort the Data Source on the “AcctMgrGender” field. But you could have an additional safeguard by using a field like the following:

**{** IF **{** MERGEFIELD AcctMgrGender **}** = "M" "he" **{** IF **{** MERGEFIELD AcctMgrGender **}** = "F" "she" "WARNING! YOU FORGOT TO PUT THIS PERSON'S GENDER IN THE DATA SOURCE!!" **} }**

## 4. Inserting different text, depending on various mail merge field conditions, in a document such as an offer letter or contract

The same principle can be taken to fairly high levels of sophistication. For example, consider that you have a recruitment spreadsheet in which one of the columns flags when an offer letter needs to be sent. You can use Mail Merge Query Options to create a mailshot to all those people to whom an offer letter needs to be sent. All sorts of details in the offer letter will, of course, vary, depending on information in the Excel spreadsheet.

Here is a relatively simple example that you can extrapolate from to create a sophisticated offer letter template.

> We are now pleased to confirm our offer for the position of **{** MERGEFIELD JobTitle **}** **{** IF **{** MERGEFIELD StartDate **}** = "" ". Please advise us of your earliest start date" ", with effect from " **}{** MERGEFIELD StartDate **}**.

Which will end up **either** reading something like this:

> We are now pleased to confirm our offer for the position of General Dogsbody. Please advise us of your earliest start date.

... **or** something like this:

> We are now pleased to confirm our offer for the position of General Dogsbody, with effect from 1 January 2001.

## 5. Inserting files at certain points in a document, depending on various conditions

You can combine IF fields MERGE fields and INCLUDETEXT fields into a single nested field to insert one of two (or more) files, depending on a flag you set in your Data Source, as follows:

> **{** IF **{** MERGEFIELD ReportType **}** = "PDD" **{** INCLUDETEXT I:\\Boilerplates\\PDD1.doc **}** **{** INCLUDETEXT I:\\Boilerplates\\Std1.doc **} }**

To insert part, but not all, of a file, you can mark with a bookmark the section of the file which you want to insert, then use a field like:

> **{** IF **{** MERGEFIELD ReportType **}** = "PDD" **{** INCLUDETEXT I:\\Boilerplates\\PDD1.doc MyBookmarkName **} {** INCLUDETEXT I:\\Boilerplates\\Std1.doc AnotherBookmarkName**} }**

## 6. Using (or not using) the MergeFormat switch

To quote from Help, the MERGEFORMAT switch applies “the formatting of the previous result to the new result. For example, if you select the name displayed by the field:

> **{** AUTHOR * MERGEFORMAT **}**

and apply bold formatting, Word retains the bold formatting when the field is updated when the author name changes.”

This may or may not the effect you want. If it *is*, just add the switch at the end of all relevant fields, e.g.:

> **{** IF **{** MERGEFIELD Gender **}** = "Male" "Him" "Her" * MERGEFORMAT **}**

By default, this switch is added automatically if you insert your fields using the **Insert +Field** menu. If you don't want it, you can de-select the “Preserve formatting during updates” checkbox; or you can display field codes (**Alt+F9**) and delete the switch manually.

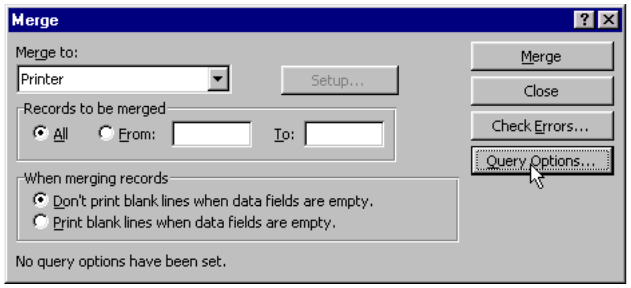
# Turning Word into a pseudo-database by using Mail Merge Query Options

Article contributed by Dave Rado

Terms of use · Disclaimer

(to return to top, press Ctrl+Home)

## 1. How to access Mail Merge Query Options


Merge
Merge to:
Printer
Setup...
Records to be merged
All
From:
To:
When merging records
Don't print blank lines when data fields are empty.
Print blank lines when data fields are empty.
No query options have been set.
Merge
Close
Check Errors...
Query Options...

"Mail Merge Query Options" is one of the most powerful features of Word's Mail Merge facility.

Purists might argue that the power it gives ordinary users isn't necessary because they should use Access queries for this sort of thing and link the merge to the query. But in my experience, many people who are very comfortable working with Word and Excel find Access (or any full-fledged database application) very difficult to work with, and can get the job done far more quickly and easily using a combination of Word and Excel. At the end of the day, getting the job done is what matters. The vast majority of the world's databases (in terms of number of databases, rather than in terms of amount of data) are stored in Excel spreadsheets.

You can access Mail Merge Query Options by clicking on the "Merge" button on the Mail Merge toolbar, and then on the "Query Options" button; or by selecting **Tools + Mail Merge + Query Options**. However, I find the feature so useful that I have added it to my Mail Merge toolbar.

One fact that is perhaps not immediately obvious is that when you create a query, the query is saved with the mail merge Main Document. This is very powerful, because it means you can, for example, have a number of different standard mail merge letters and labels all linked to the same Mail Merge Data Source, each with a different set of query options stored with them.

For example, a staff database (which could be an Excel Data Source) could contain a whole series of flags:

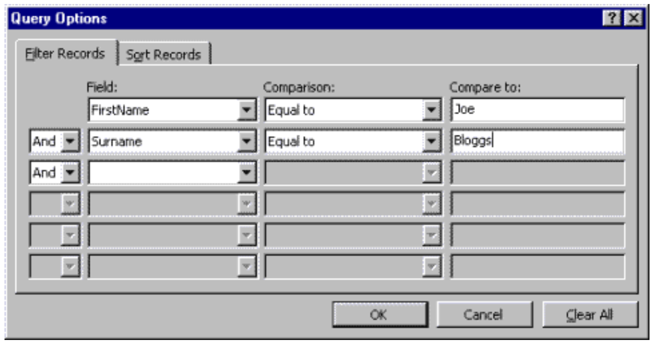
1. A column flagging when each person's birthday is getting close, so that a standard “happy birthday” mailshot can use that in its query; and another column flagging whether or not the letter's already been sent (so they don't get mailed twice)
2. A column showing their employee status, another showing their Line Manager, and if their employee status is “Contractor”, a column showing their contract expiry date. A standard letter could be saved with a query on these columns in order to mailshot their line managers when the Contract is about to expire

... and so on.

Here are a few examples of how you can make your mail merges more powerful using Query Options, starting with some very simple examples and getting more complex (and powerful) as we go on.

## 2. Re-sending a mailshot letter to just one person

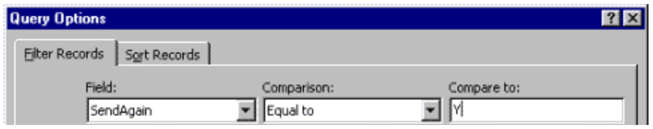
If somebody didn't receive your letter, the simplest way to re-send just their letter is to do a query on their first and last name as shown below.



|  | Field: | Comparison: | Compare to: |
|---|---|---|---|
|  | FirstName | Equal to | Joe |
| And | Surname | Equal to | Bloggs |
| And |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## 3. Re-sending a mailshot letter to several people (but not to the entire Data Source)

Create a field (a column if you are using a Word or Excel Data Source) called “SendAgain”. Put a “Y” in that column against the rows you want the letter to be re-sent to. Save, switch back to Word, close the document and open it again to refresh it, and set up the Query Options dialog as follows:



| Field: | Comparison: | Compare to: |
|---|---|---|
| SendAgain | Equal to | Y |

Or, for example, if you want to send a follow-up letter to people who haven't yet replied to the first mailshot, you could have a mail merge field called “Replied”. When a reply comes in, put a **Y** against the relevant record in the “Replied” column. When it comes to time to chase up the stragglers, you can just set up the Query Options to **Replied/Not Equal to/Y**, to send the follow-up letter to those who haven't yet replied.

## 4. Using Query Options to manage multiple different mail merges with a single Data Source

Take the following scenario:

> “We have a large mailing list that we use to post invitations (our community and education department), and send information to various individuals and charity organisations. We have different groups (one for xmas, invitations, etc); and one person can belong to many groups. At the moment they have to update a lot of duplicates because a person exists in many groups. I would like have one master list with addresses and some kind of a code or index. I would like to stay away from databases.”

This is easy. In your Data Source (your “master list”), create a column for each “group”; i.e. a column called “XmasLetter”, etc. To put someone into a particular group, simply put a “Y” in that column against that person's row. Then set up all your standard letters with the appropriate Query Options saved with the letter.

So for the standard Christmas letter, for example, the Query Options dialog should read **XmasLetter/ Equal to/Y**.

When you save the letter, the Query Options are saved as well, so from then on you can just open the appropriate standard letter whenever you need to do a mailshot, click on the "Merge" button, and you're done.

## 5. Using Query Options to select who to email, who to fax and who to send a letter to

You could have three Word documents attached to the same Data Source, each saved with different Query Options:

- The email would have its Query Options set to **EmailAddress/Is not blank**.
- The fax would have its Query Options set to **EmailAddress/Is blank** and **FaxNo/Is not blank**.
- The letter would have its Query Options set to **EmailAddress/Is blank** and **FaxNo/Is blank**.

## 6. Sending a letter to everyone whose birthday falls within the next 7 days

Assuming you have an Excel spreadsheet containing names, addresses and dates of birth:

1. Create a column called "Birthday". Use a formula such as the following to calculate the person's birthday from their date of birth:

   ```
   =(DAY(C2)&"/"&MONTH(C2)&"/"&YEAR(TODAY()))
   ```

   In the US you would swap the day and month round:

   ```
   =(MONTH(C2)&"/"&DAY(C2)&"/"&"/"&YEAR(TODAY()))
   ```

   The above formula assumes that the current row is Row 2, and that the date of birth is in Column C. It takes the day and month from their date of birth, shoves the current year at the end, and displays the result. Having created the formula in one cell, you can drag it down to Autofill the remainder of the column.

2. Create another column called "BirthdayFlag". Use a formula such as the following to calculate whether their birthday falls within the next 7 days:

   ```
   =IF(TODAY()>DATEVALUE(D2),"Whoops, too late!",IF(TODAY()+7>=DATEVALUE
   (D2),"Send now",""))
   ```

   If their birthday has already passed, it will display "Whoops, too late!". Otherwise, if their birthday falls within the next 7 days, it will display "Send now", and if it's more than 7 days until their birthday, it won't display anything.

   You could refine it further by adding a column called "LetterSent", and putting a Y in that column when you've sent the letter.

3. In Word create a standard "birthday letter", and in the Query Options dialog, set the **BirthdayFlag/Equal to/Y**; and **LetterSent/Is blank**.

   Create a second "happy belated birthday" standard letter with **BirthdayFlag/Equal to/Whoops, too late!**; and **LetterSent/Is blank**.

   Open both letters once a week, merge, and you're done – again, you only ever need to set the query options once.

The principle illustrated by this example can be used for all sorts of useful mail merges – for instance, for sending reminders to managers who have contractors working for them, when the contract period is about to expire.

## 7. Sending a letter to ask for information that is missing from a spreadsheet

Consider the following scenario:

> "I have a large Excel database with some missing information. I would like to create a form letter that asks the recipient for the missing information.
>
> What I'd like to do is write a letter that says, "For the patient X, we are missing the following information: Date of birth, Diagnostic, Dossier Number. For the patient Y, we are missing ... " etc., where Date of birth, Diagnostic, Dossier Number are all the names of columns in the Excel document."

For this you would need to use **IF fields** in the form letter, with the construction "If the Date of Birth field is blank, display the text "Date of birth", otherwise display nothing."

Trying to have several such fields on a single line, separated by commas, would require tortuous logic in order to get the IF fields to suppress all unnecessary commas, so it's easiest to use a bulleted list for this sort of thing:

For the patient X, we are missing the following information:

- **{** IF **{** MERGEFIELD DateOfBirth **}** = "" "Date of birth" "" **}**
- **{** IF **{** MERGEFIELD Diagnostic **}** = "" "Diagnostic" "" **}**

Note that, blank paragraphs which contain mail merge fields nested within an IF field are **not** suppressed when you merge. You can get round this problem by merging to a new document rather than directly to the printer, and then doing a Find and Replace replacing **^p^p** with **^p**, to remove all blank paragraphs from the merged document.

Finally, use Query Options to strip out records for which **no** information is missing. You could do this by setting the relevant fields in the Query Options dialog to "Is not blank". However, this only works with up to 5 fields (a limitation of the dialog). To get round this limitation, you can add a column in the spreadsheet called "IsBlank" containing a formula of the type:

```
=OR(ISBLANK(A2),ISBLANK(B2),ISBLANK(C2),ISBLANK(D2))
```

This formula will display "True" if *any* of the cells A2, B2, C2 or D2 are blank, and will display "False" if *none* of them are blank. You can then simply set up the Query Options dialog to merge with all records for which the "IsBlank" field is equal to True.

## 8. Using mail merge and Query Options to create a "reader-friendly" front-end for an Excel spreadsheet

Take the following scenario:

> "We are having a recruitment drive. We have advertised in various media and have also contacted various agencies. We are recording all the details of any job applications we receive in an Excel spreadsheet, together with interview dates etc.; but the spreadsheet has more than 30 columns, and trying to make head or tail of the information does our eyes in. What we need is a sort of database that allows me to see all the information about a particular candidate laid out nicely, so it's easy to read; and which lets me look at for example, only the candidates who have an interview tomorrow. I want it to be easy to look at this information both on screen and in a print-out. I'd rather **not** use Access"

This is very simple to set up using a mail merge. Design a mail merge form in Word (not with form fields, but *laid out* like a form), with the 30+ mail merge fields laid out logically on the page, making sure they all fit on a page, and attached to the Excel spreadsheet.

Save multiple copies of the form, one for each report you need.

For instance, one could be called "InterviewsTomorrow.Doc", and it would have a query saved with it which filtered just those records in which the interview date is tomorrow's date. To do this, you could add a column in the Excel spreadsheet called "InterviewTomorrow" containing a formula such as:

```
=IF(TODAY()+1=D2, "Y", "")
```

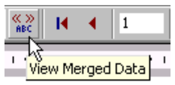
.. where column D contained the interview date. This would display "Y" if the interview date was tomorrow, and display nothing otherwise. Then you could set the Query Options to filter those records for which the field "InterviewTomorrow" is equal to **Y**.

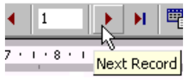
**As part of the query you can also store how you want the information to be sorted**, so each query could be sorted differently (for the "InterviewTomorrow" query you might want to sort by interviewer, and then by interview time, for example). In the Query Options dialog, click on the "Sort Records" tab.

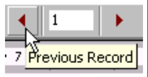
Once you've saved all the different versions of the form, each with its own unique query, you can simply open the documents whenever you like, to view or print the information, nicely laid out, and filtered and sorted appropriately.

To see the records on screen, **don't** do a merge. Instead, click on the ViewMergedData button on the mail merge toolbar to see the data on screen without merging; and use the

Next Record and Previous Record buttons to skip through the records:







This is much quicker and more convenient than doing a full merge, and gives you similar functionality to a conventional database application.

If you want to print the report, click on the Merge button and select “Merge to printer”.

# Using MacroButton fields

Terms of use Disclaimer

Article contributed by Graham Mayor, Jonathan West and Hak-lok Ng

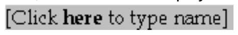
The macrobutton field can be used as a text marker within a template, or, as the name implies, it can be used to run a macro.

## Using MacroButton fields as a text marker

You can use a MacroButton field that doesn't actually run a macro but simply displays a prompt and allows the user to click on the prompt and type. To do this, insert a field like:

**{** MACROBUTTON NoMacro [Click **here** and type name] **}**

Change “Click here and type name” to whatever text you require. Press **F9** to update the field, which will also display the display text instead of the field code:


[Click **here** to type name]

See Microsoft's fax templates (which are supplied with Word) for examples of this. Also see:

**How to create a template that makes it easy for users to “fill in the blanks”, without doing any programming**.

## Using {MacroButton} fields to run a macro

For this, use a field like this:

**{** MACROBUTTON MyMacroName [Double-click **here** to run macro] **}**

See: **Run a macro when a user double-clicks a button in the document** for more details of how to create the field.

Macro button fields can make it very easy to set up fairly sophisticated templates with very little programming effort. For example, see:

**Using {Macrobutton} fields to insert information from the Outlook Address Book into documents such as letters**.

But they can be useful for all sorts of things – for some more examples, see:

**Using hyperlinks in protected forms**
**Enable a user to double-click text in a document to change its value**
**Organizing your macros**

Also see the checkboxes in the Microsoft Fax templates which are supplied with Word, where the macros associated with the fields insert AutoText entries, one AutoText entry being a MacroButton field containing a checked checkbox symbol, the other being a MacroButton field containing an unchecked one. If you copy those fields, and the macros and AutoText entries associated with them (using the Organiser) into your own templates, you can use them unmodified.

## “Passing arguments” to MacroButton fields

Macros assigned to MacroButton fields cannot take arguments. In fact if you want to be semantic, macros cannot take arguments, ever, because a macro is defined as a public subroutine that takes no arguments, which is why subroutines that *do* take arguments are not shown in the list when you select **Tools + Macro + Macros**.

However, depending on your situation, you can get round this in a number of ways, the best two (depending on the circumstances) being.

1. Your macro can read the value from a Custom Document Property, or a Document Variable.
2. You can insert a **Private** field within your MacroButton field. The first thing your macro should do is look for the code of the **Private** field (which by definition will be the second field of the Selection) and read the value that you want to pass

   This method is especially good if you have more than one MacroButton field in a single document which you want to call the same macro, but with the macro operating on a different variable in each case.

   For example, you could create a nested field as follows:

   **{ {** Private Hello world **}**Macrobutton TestMacro [Double-click to run macro]**}**

   ... which would display:
   [Double-click to run macro]

   ... and the macro could look like this:

   ```
   Sub TestMacro()

   Dim MyString As String
       'Ignore first 9 characters of the private field -
       the word 'Private', and the spaces
       MyString = Mid$(Selection.Fields(2).Code, 9)
       MsgBox MyString
   End Sub
   ```
3. Instead of a **Private** field, you could use an **Addin** field within your MacroButton field. An Addin field is very similar to a Private field but even more private - see **Using Addin Fields**.

   Note that the order matters; the following works as one would wish it to:

   **{** Macrobutton TestMacro [Double-click to run macro]**{** Addin **}}**

   ... but the following makes the MacroButton field's display text invisible:

   **{ {** Addin **}**Macrobutton TestMacro [Double-click to run macro]**}**

   The macro could look like this:

   ```
   Sub TestMacro()

   Dim MyString As String
       MyString = Selection.Fields(2).Data
       MsgBox MyString
   End Sub
   ```

Send all feedback to **word@mvps.org** Top - **Ctrl+Home** Previous - **Alt+left arrow** Page Updated:

# Creating a Template – The Basics (Part I)

Article contributed by Suzanne S. Barnhill

You can actually get some help in template creation from Word's online Help if you look under “templates, creating.” (In Word 2007, look at the “How to create a template” portion of the Help topic “Creating Microsoft Office Word 2007 templates.”) But first you need to understand **what a template is**, **what it is not**, and **when you need to create one**. Then we’ll look at the basics of **how to create a template**.

## What is a template?

In the most general sense, a *template* is a pattern or model on which something else is based. It might be a shape that you trace around or an outline of suggested content. In Word, however, the word *template* has a specific technical sense; it is a particular kind of file, with a different file extension from a document (.dot, .dotx, or .dotm instead of .doc or .docx).

Templates in Word are generally stored in a different location from documents, and you will rarely open one directly after creating it. Instead, you will use it as the basis for creating new documents.

Word comes with a number of built-in templates, but you may be unaware of them if you have never visited the dialog where they live. What you get when you press **Ctrl+N** to create a new document in any version is a Blank Document based on the default template, which is called Normal (Normal.dotm in Word 2007, Normal.dot in previous versions), but Word also offers templates expressly designed for specific types of documents: letters, reports, fax cover sheets, and the like. These are accessed as follows:

- Word 2000 and earlier: Select **New…** on the **File** menu. This opens the **New** dialog (see figure below).
- Word 2002 and 2003: Select **New…** on the **File** menu. This opens the **New Document** task pane, where you can either select a recently used template or click on “General Templates” (Word 2002) or “On my computer…” (Word 2003) to open the **Templates** dialog (which is just the **New** dialog with a different name).
- Word 2007: Click the **Office Button** and select **New**. This opens the **New Document** dialog, which initially displays “Blank and recent” templates—that is, the Blank Document and any other templates you have used recently. The **Recently Used Templates** pane will be empty until you have used other templates, which you can find under **Installed Templates** (the ones that ship with Word) or **My templates…** (the ones you have created).

Many, many more templates are available for download from the **Office Template Gallery**, which can be accessed directly from Word 2002 and above:

- Word 2002: In the **New Document** task pane, click “Templates on Microsoft.com.”
- Word 2003: In the **New Document** task pane, click on “Templates on Office Online.”
- Word 2007: In the **New Document** dialog, click on **Microsoft Office Online**.

## What a template is not

Although many of the templates you can download from Microsoft Office Online contain sample content, a template is not really about content but about structure and layout. A template is designed to provide specific page layout (page size and orientation, margins, number of columns, and so on), and styles for the types of paragraphs most likely to be used in the given type of document. It may also contain tools to facilitate using the included styles and other features. These tools include Building Blocks or AutoText entries, macros, keyboard shortcuts, and (in Word 2003 and earlier) custom menus and toolbars.

Some templates do contain boilerplate content: a **template for a letter**, for example, will perhaps have a letterhead on the first page, page numbering, and perhaps an automatic date field. In addition to custom styles for the parts of a letter (Inside Address, Reference Line, Salutation, Body Text, Complimentary Close, Signature, Copy List, and so on), it may have dummy paragraphs or text entry fields indicating where these parts go.

In general, however, the content of a document is up to the writer. Users often ask for “templates” for very specific content, such as a letter protesting an unfair dismissal or a letter to customers of a business thanking them for their patronage. You may actually find such samples among those available in the Template Gallery at Microsoft Office Online. Viewed from a layout perspective, however, such letters are just letters. They can be created using a generic letter template or from scratch, assuming the writer knows how to write a letter. What the user is really looking for is a model or sample document that would provide suggested wording for such a letter. That is not what a “template” is in Word.

On the other hand, users’ needs are sometimes are more related to layout: “a Request for Proposal template to hire a building designer for a residence” or “a restaurant evaluation sheet template” or “a flyer template for an AA – Al-Anon Event.” In such cases, finding a readymade template is unlikely, though it may be possible to find a generic template that can be adapted. In the last analysis, however, the user is still looking for a model or sample rather than a template—just something to copy or build on; even a sample document would suffice.

## When to create a template

There are several ways to create a new document in Word:

- Click **New** on the Standard toolbar (Word 2003 and earlier) or select Blank Document in the **New Document** dialog in Word 2007. Alternatively, press **Ctrl+N**. This will create a new Blank Document based on the Normal template, which contains all the styles available in Word. You can modify these styles as desired, and you can change the layout of the document in any way you wish.

***Note:*** By default the Quick Access Toolbar (QAT) in Word 2007 does not have a **New** button that automatically creates a new document based on the Normal template. To add the **New** button, click the arrow (**More** button) at the end of the QAT and then click **New**.

- Open an existing document as the basis for a new one. The natural tendency of most users of word processing applications is to create a document and use it as a model for future documents. That is, you format a letter the way you want all (or most) of your letters to look, save it, and then, when you want to write a letter, open this document and save it under another name as the starting point for your letter. While this technique is a practical approach in some instances, there is always a risk that you will forget to use Save As and will instead overwrite your original document.
- Create a new document using an existing document as a quasi-template. This is a way to reuse a document without risk because a document created this way is unnamed; the first time you click the **Save** button or press **Ctrl+S**, you will get the **Save As** dialog, which requires you to name the document and choose a place to save it.
- Word 2002 and 2003: In the **New Document** task pane, choose **New | From existing document…** Browse for the document in the **New from Existing Document** dialog and click on **Create New**.
- Word 2007: **New from existing…** is one of the options in the **New Document** dialog. Browse for the document in the **New from Existing Document** dialog and click on **Create New**.
- You can actually accomplish the same thing in any version of Word by right-clicking on a document file in the **Open** dialog or **Windows Explorer/My Computer** and choosing **New**.
- Create a new document based on a different template, either one of those that ship with Word (**Installed Templates**) or one you have created, by selecting it in the **New**, **Templates**, or **New Document** dialog.

The whole point of a Word template is to create a format that can be used over and over again. Accordingly, it is unnecessary and a waste of time to create a template for a single-use document. Creating a template for letters makes sense; creating a template for a letter protesting one’s unfair dismissal does not. A template for flyers for AA – Al-Anon events may make sense if the events are frequent and the flyers should be consistent in design; if the event is a one-off, a document will suffice.

So, before you set out to create a template, you should ask yourself whether it is something you would use repeatedly. Often this realization comes after you’ve recreated the same document format numerous times, changing margins, modifying styles, changing fonts. It occurs to you that you could save time in the creation of such documents if you didn’t have to make all these changes. That’s when you need a template.

In addition, there are advantages to true templates that cannot be achieved with documents used as templates. Although it is now possible to save macros, a customized QAT (toolbars and menus in earlier versions), and keyboard shortcuts in documents, Building Blocks (AutoText entries in earlier versions) must still be saved in templates. And the **New/Templates/New Document** dialog actually makes it easier to access templates than to search for documents.

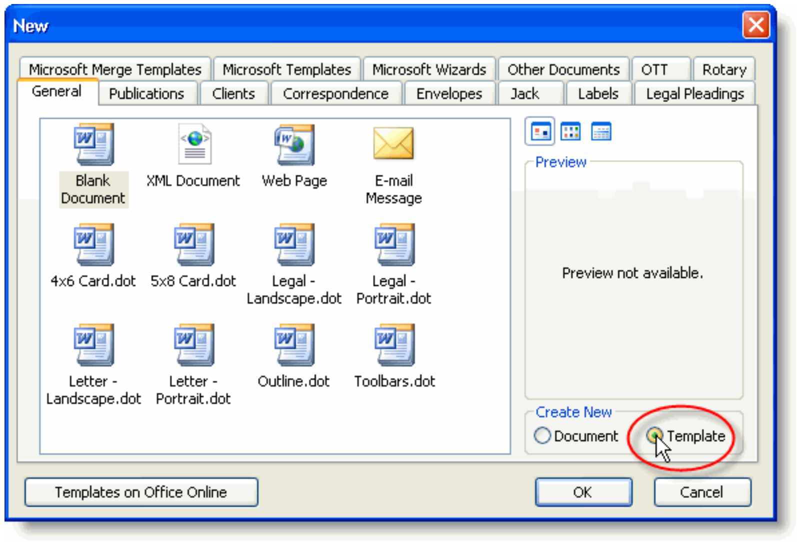
If you create a specific kind of document (such as letters) almost exclusively, your first impulse may be to just make the necessary changes to the Normal template, so that you get a document formatted the way you want when you click the **New** button. This can be a solution up to a point, but please note the caveats expressed in "**How to change the default settings for Word documents**." In particular, it is not a good idea to add a header or footer to the Normal template because this will affect the layout of labels.

For this reason it is usually a good idea to leave the Normal template alone (a lot of your settings, such as shortcut key assignments and—in previous versions—custom toolbars, will be stored there anyway, so that they are available to all templates) and make a custom template for each specific task you routinely do.

## How to create a template

There are two basic ways to create a template:

1. You can start from scratch, selecting New… from the File or Office Button menu and checking the radio button for Template instead of Document in the dialog box when you create a new file.



Create New: Document [✓ Template]

The above is the author's personal New dialog, which includes many custom tabs and templates.
The Installed Templates have been copied to dedicated folders and then uninstalled
(so as to reduce the number of tabs they require).

- You can base your template on any existing template, including the Normal template (represented in the **New** dialog by "Blank Document"). If you base a new template on Blank Document (which you will find in **My templates…** in Word 2007), it will not actually be based on the Normal template but rather on the default settings (that is, the out-of-the-box Normal template rather than the current version with any modifications you may have made).
- When you save the file, it will automatically be saved as a template, using the filename you assign. The default save location will be Word's Templates folder (or a subfolder you choose). In Word 2003 or earlier, the file format will be .dot. In Word 2007, the default template format is .dotx, but you can also choose .dot (if you want the template to be usable in earlier versions) or .dotm (if it contains macros).

2. You can create a template from a new or existing document. Whenever you have a document that has formatting you want to repeat in another document, you can Save As and under "Save as type" choose a template format.

- In Word 2003 and earlier, the only choice is "Document Template (*.dot)." In Word 2007, you have a choice of .dot, .dotx, and .dotm, as described above.
- In Word 2003 and earlier, the save location will be automatic (as with Method 1). In Word 2007, however, the file location is not automatically selected for you. You can save the template anywhere, but if you want it to appear in the **New Documents** dialog, you must save it in the Templates folder (accessed via the **Trusted Templates** link in the Places Bar of the **Save** dialog)

When you use the first method and base your new template on an existing one (other than Normal), the new template will inherit all the macros, custom toolbars or menus, toolbar or menu customizations, shortcut key assignments, and AutoText entries or Building Blocks that are stored in that template. Since these customizations are not stored in documents based on the template, a template created using the second method will not include them. A template created by either method will, however, contain the styles and layout of the parent template (though you may choose to modify them).

**Note:** Unless you have explicitly saved macros, custom toolbars or menus, toolbar or menu

customizations, shortcut key assignments, AutoText entries, or Building Blocks in a specific document template, they will be stored in the global template, Normal.dot or Normal.dotm. This means that they are available to all documents, regardless of what document template is attached, so they are not saved in a template based on the Normal template.
Another advantage to using the first method in Word 2007 (as noted above) is that Word defaults to the Templates folder when you save the template. If you use the second method, you must manually navigate to the Templates folder (or other save location).

## Using your new template

The next time you choose **File | New…** or **Office Button | New**, you will see the template you have saved. In Word 2007, it will be in the **Recent Used Templates** pane, at least to begin with; you can access it at any time from the **New** dialog that opens when you click on **My templates…** Double-click on your template and you will create a new document that incorporates the page setup, styles, and other formatting you have set, along with any boilerplate text you have left in the template.

In the **New/Templates** dialog, you will see a number of tabs. Except for General, which represents the root Templates folder, the labels on the tabs reflect the names of subfolders in the Templates folder. You can create additional subfolders of your own, and they will be added as new tabs (in Word 2000 and above, you will not see the tabs unless the folders actually contain templates). You can also move files from one folder to another, and they will be moved from one tab to another accordingly.

In Word 2000 and above, the installed templates (those that ship with Word) are saved in a different location from Normal.dot and user-created templates; you must create new subfolders (tabs) in the folder that is assigned for user templates. If you want to add your own templates to one of the tabs that contain installed templates, you must create a new subfolder the name of which is exactly the same as the label on the dialog tab. For more on this and other template issues, see "**Frequently asked questions about the location of Word 2002 templates**" (this Word 2002 article has links to versions for Word 2000 and Word 2003; the Word 2003 article actually covers Word 2007 as well).

Eventually you may get tired of having to go through a menu to access your custom templates, especially in Word 2002/2003, where this command opens the **New Document** task pane. When this happens, you can add a button to the Standard toolbar (Word 2003 and earlier) or QAT (Word 2007) to provide direct access to your templates:

- Word 97 and 2000: Open **Tools | Customize** and select the **Commands** tab. In the **File** category, select the **New…** command and drag it to the Standard toolbar. This button will run the FileNew command, which opens the **New** dialog, as contrasted with the existing **New** button, which runs the FileNewDefault command, which creates a new document based on Normal.dot. Since the button icons are identical, you may want to use the button image editor to modify one of them (right-click on the button and choose Edit Button Image), or, if you will be using your own templates most of the time, you may just want to drag the **New** button off the toolbar (you can still use **Ctrl+N** to create a new Blank Document).
- Word 2002 and 2003: The process is similar to that for Word 97 and 2000 except that the **New…** command opens the New Document task pane. You must therefore select the **All Commands** category and locate the **FileNewDialog** command. Because this command has no button image, when you drag it to a toolbar, it will have a text label: **Other…** If you want to use the button image from the **New** button, right-click on it and choose Copy Button Image, then right-click on the **Other…** button and choose first Default Style…, then Paste Button Image. Again, you may want to edit this image or remove the **New** button.
- Word 2007: In **Office Button | Word Options | Customize**, there are several "New" buttons you can add to the QAT, and they are somewhat confusingly labeled. If you select **All Commands** in the **Customize** dialog, you will see all four of them: New, New…, New Blank Document, and New Document or Template. "New" and "New Blank Document" both run the FileNewDefault command, creating a new Blank Document (based on Normal.dotm). "New…" opens the **New Document** dialog you get if you use **Office Button | New**. And "New Document or Template" takes you directly to the classic **New** dialog (accessed via **My templates…** in the **New Document** dialog) that displays both the installed templates and your custom templates in the classic tabbed **New** dialog.

Templates and styles in Word are your strongest allies in making creation of customized documents easy and straightforward. Note that when you add AutoText or Building Blocks or create customized toolbars or macros or keyboard shortcuts, you can choose whether to store them in the Normal template or in the template on which a given document is based. This allows you to have certain tools restricted to a given template, without cluttering up documents for which they are not suitable. The more you explore Word's capabilities, the more you will learn to do.

For more about the creation and use of styles in templates, see John McGhie's article **Creating a Template (Part II)**.

**Note:** I am indebted to Office MVP **Beth Melton** for technical editing and many useful suggestions that improved the flow of this article. The many flaws that remain are entirely my own fault (and due to my ignoring her advice).

# Creating a Template (Part II)

Article contributed by John McGhie

Terms of use Disclaimer

**Word version of this article downloadable here**

**This article can be downloaded as a Word document plus its template. If you save them to the same directory as each other, the document will retain its link to the template. The downloadable zip file is 75k.**

CreateATemplate.zip

**As well as being more printer-friendly than the web version, the document and template also allow you to see the principles preached in this article being put into practice.**

*See also* ***Creating a Template – The Basics (Part I)***

## Using Templates

A template is a repository for things you want to use frequently and for complex things that you want to do only once. In normal use, this means Layouts, Styles, Tool Bars, AutoText Entries and Macros.

This article tells you how to create a template to produce a software manual.  That's because:

- That is what I do for a living so I will know what I am talking about (sort of…);
- Such a template contains all the things you will need to produce any other kind of document;
- I happen to have one lying around here that I can give you to download;
- It's the template I used to produce this document, so you can download an example as well.

Mine is set up for metric A4 paper. If you change the paper size, you will have to change everything (and I mean “everything”) else as well. Sorry about that.

I am going to give all dimensions in metric units (except font sizes!). You may want to change your settings to metric now. That way you can use the dimensions I specify. At the end, you can change back to Imperial if you wish, and Word will convert everything into that measure for you. To do this got to **Tools>Options>General** and set “Measurement Units” to Centimetres.

Technical writers love to specify commands and dialog boxes very precisely; and even show you a screen shot of the dialog box. I can't do that here because I am writing to cover nine versions of Word. This was actually written on a beta version of Word 2004. If you work with me, you will find everything I mention in Word 6, Word 95, Word 97, Word 98, Word 2000, Word X, Word XP (Word 2002), Word 2003 and Word 2004. You may have to look for some things: things move around in the user interface from version to version.

I guess we should recognise that according to Microsoft's research, “normal” users do not use or even know about templates. When Word comes out of the box, it is set up to cater for users who do not understand word processing.

Word is set up to enable the simplest fastest way to produce a document if you have no idea of what you want or what you are doing. If you were in that state, you wouldn't be reading this. So this article assumes you are in a workplace, where you adhere to a Style Guide and a Formatting Specification. A Template is the repository that stores all the specifications and choices that implement your Style Guide and Formatting Specification.

It's also the place where you put all the things you use that are fiddly to create or required to conform exactly to specification.

Always change formatting with **Format>Style**. I may sometimes forget to say so, in which case please remember it for me! There is only one time in this whole exercise that you can apply direct formatting. Anywhere else, it's a total waste of time: remember: for most users, the only thing they can ever access in a template is the styles. If the settings are not in the styles, they're pointless.

By the end of this exercise, you will realise that Word's default settings are all designed for the knee-cap-level user, and that we have to spend a lot of time undoing them. {Begin Political Rant} I hereby give you permission to think unkindly of the Product Marketing Department, which took the world's finest word processor and ruined it in order to reinforce the misconceptions of people who should not be left unsupervised with a pencil!!! {end political rant}.

## Creating a Template

So: start Word, allow the default blank document to load, and choose **File>Save As**. Change the “Save as Type” box to “Document Template (.dot)”.

This is where Microsoft gives new users their first hint that they're getting in too deep. As soon as you change the Type to Template, you are dumped into your User Templates folder (although you can then change the path if you want to). There are good reasons for this. The first is that Word needs to know where this thing is so it can offer it to you when you need it; the second is that in this location word can take extra care that macro viruses do not try to add anything nasty to it.

At the moment, the document is still a copy of your Normal.dot template. Give it a file name and save it. Make the file name long and descriptive. It doesn't matter that it is long: you will never have to type it. You will do yourself a favour if you follow some kind of naming convention. I suggest “<Company Name> A4 Manual.dot” Adding your company name is really nice when you come to deal with a lot of templates. Specifying the paper size is goodness too: you are likely to end up with a series of templates for different paper sizes: it's nice to keep them together. You do not have to type the “.dot” bit: Word will add it for you.

So far, it appears that nothing has changed: you still appear to be looking at a copy of your Normal.dot (and you are). Behind the scenes, Word has made some critically important changes: the file now has a different internal structure, and it has gained extra objects to store things that documents cannot (or should not) contain.

## Table of contents

# How can I recover a corrupt document or template – and why did it become corrupt?

*Article contributed by Dave Rado and John McGhie*

**The advice in this article applies to all versions of Word, including those for the Macintosh.**

## *Why do documents corrupt?*

If you use any of the following features, your documents are likely to corrupt: Master Documents, Nested Tables, Versions, Fast Saves, Document Map, and saving to a floppy. For more on these, see: Tips and “Gotchas”.

In addition, saving when resources are low can cause corruptions. If you notice Word start to slow down noticeably it's *always* best to quit and restart Word immediately; to close any other applications that are open; and to clear the clipboard, by selecting any character and copying it.

Other signs that you are low on resources: fonts suddenly not displaying properly; the wrong application icons appearing on your Desktop or in Windows Explorer (e.g. Word's icon appearing where Excel's should be). If you get these symptoms, restart Windows immediately.

A corrupt printer driver can corrupt memory; and if you then save, this can corrupt the document. Symptoms: Word often crashes when printing (cure: reinstall the driver).

A corrupt template will corrupt any new documents based on it. A corrupt Normal.dot template is especially bad -- it spreads its evil almost like a virus to almost every new document you create.

If you create list numbering using the **Format + Bullets and Numbering** dialog, this is likely to lead to a corruption arising eventually, especially if you also have the “**Automatically update document styles**” checkbox ticked on the **Tools + Templates and Addins** dialog (less of a problem with Heading numbering than other types).

A bad sector on your hard disk can corrupt a document saved to that sector. Make sure you run Scandisk regularly. Running Defrag regularly will also help reduce the chances of running low on resources. Encourage users to save to the network. Make regular backups.

## *Where are corruptions stored?*

Corruptions are usually, but not always, stored in Section Breaks. The final paragraph mark in a document contains a hidden Section Break, so in a single-section document, corruptions tend to be stored in the final paragraph mark.

Corruptions can also be stored in *any* paragraph mark in a document; or in an end-of-cell or end-of-row marker within a table; or in a bookmark. (Corrupt bookmarks are very rare in Word 97 and above, unless you have been using EndNote).

If you find that certain commands such as **Edit>Find** don't work within a certain table, that table is probably corrupt.

If you find text mysteriously disappearing and reappearing as you page down past a particular paragraph, that paragraph's paragraph mark is likely to be corrupt (see the section on fixing corrupt templates).

## *How can I fix a corrupt document?*

If you have been using Master documents, see How to recover a Master Document.

If all new documents based on a certain template are showing symptoms of corruption, the template they are based on is almost certainly corrupt.

Otherwise:

### If using Word 2000 or Above:

Select **File + Save As Web** page, quit Word, reopen the htm file and save it back in Word format – that **usually** (but **not** always) gets rid of corruptions. The HTML/XML format forces Word to

completely re-create the internal structure of the document, either fixing or discarding the corrupt bits when it does. Best of all, in the case of Word 2000 and above, almost all of the formatting and page layout *is* preserved.

**Please note**: to preserve your formatting, you **must** select the plain **Save As "Web Page"** option, **not** the **Save As Web Page (Filtered).** If you use the Filtered option, you remove from the document all the formatting that an HTML browser cannot interpret: for example, page numbers and headers and footers!

If that doesn't fix it, the fixes described below apply.

**If using Word 97 or above**:

1. If you have isolated the corruption to a particular table, **either**:
   - Paste the table into Excel; delete the Word table; paste the Excel table back into Word, select the new table (**Alt+Double-click**), press **Ctrl+Spacebar** to remove the manual formatting, and reformat the table, **or:**
   - Select **Table + Convert Table to Text**, select the text that results, and select **Table + Convert Text to Table**. This has the advantage that you lose much less formatting than using the Excel method, but the disadvantage that if a corruption is stored in a paragraph mark within the table, it will remain.
2. If the table contains horizontally merged cells, it's best to recreate a few rows at a time – for instance, if using method (b), then after converting the table to text, select contiguous rows that have **equal** numbers of columns, convert *them* to a table, and keep doing this until you have converted all the text to individual tables (which will automatically merge themselves into a single table).
3. If you have isolated the corruption to a particular paragraph, select the text in the paragraph, but be careful ***not*** to select the paragraph marker (the paragraph marker is a property container, and that's where the corruption is stored). Paste into a new document. Delete the corrupt paragraph. Paste back from the new document to the old one.

disabled after displaying a message box

4. You can try saving as RTF, closing Word, reopening the RTF file and saving back as a Word document. Unfortunately, Word's RTF format is similar enough to Word's native format to preserve most corruptions.

5. If that doesn't work, delete any Section Breaks using Find and Replace, then **Select All (Ctrl+A), de**-select the final paragraph mark (**Shift + Left arrow**), copy, and paste into a new document. Then close the corrupt document and save the new one, overwriting the old one (in that order). Finally, log out or restart your operating system  before doing anything more (because document corruptions can corrupt memory).

6. If even that doesn't work, try saving in Word 2 format if you have this option (the Word 2 converter is no longer offered, but if you have upgraded from a previous version, you will still have it). Unfortunately, you will lose an awful lot of formatting if you do this, though.

7. If even *that* doesn't work, select **File + Open**, set the "**Files of type**" list box to "**Recover Text from Any File**", and open the corrupt document. Delete the gobbledygook at the end. You'll lose *all of* your formatting leaving only the text.

Note that the "**Recover Text from Any File**" setting is "sticky". In Word versions prior to 2002, you must immediately select **File + Open** again, change the  "**Files of type**" setting back to "Word Document" and open another document, while you remember.  If you forget, every file you open will have no formatting, and if you save it in this condition, it's gone forever.  See Whenever I open a document using File Open all my formatting is gone, and there is garbage at the end).

## *How can I fix a corrupt template?*

The best strategy is to keep a backup, macro-free version of all your templates. Then if a template become corrupt, you can copy any macros over to a copy of the backup template using the Organizer and you're back in business.

If you haven't done that, though, and if your template contains any macros, you could try running the VBA Code Cleaner.

If that doesn't fix it, recreate the template from scratch. If you want to copy the content from the corrupt template to the new one, follow the same steps as for fixing a corrupt document. It may be worth creating the new template based on a "*virgin*" copy of Normal.dot, just in case Normal.dot is also corrupt. With Word closed, rename your existing Normal.dot file and restart Word; a new copy of Normal.dot will automatically be created.

One more thing; under **Tools + Options + Save**, turn **on** the checkbox which says "**Prompt to save Normal template**", if it isn't switched on already (unfortunately, it is switched off by default). The only time you should ever save Normal.dot is when you have knowingly made a change to it that you want to save. Then you should save it by holding the **Shift** key down and selecting **File + Save All**. And be sure never to save Normal.dot when resources are low (see: Why do documents corrupt?).

Allowing Word to save Normal.dot whether you've consciously made changes to it or not, is OK in versions of Word later than Word 97/98.

# How to recover a Master Document

Article contributed by John McGhie

**(to return to top, press Ctrl+Home)**
**Note that this procedure must be followed in strict sequence in order for it to work.**

## 1. Overview

Notice how this article starts off with the cheerful assumption that you want to “recover” a Master Document? I bet you were hoping that we would tell you how to “fix” one! We can't. If you are having a problem with a master document, the problem **is** the master document. Any attempt you make to repair one will inevitably make your problem worse.

Do not be tempted to re-create a master document. If you re-create the master document, you will immediately re-create the problem. Master documents have been fatally buggy since Word 6, and remain so through Word 2000. If you use them you lose them. They must never be used for valuable text.

There is no way to successfully and safely use master documents. They always corrupt eventually.

To understand why, you need to understand the Microsoft Word document internal structure in some detail.

### Versions of Word

The Master Document “feature” appeared first in Word 6 for Windows. These instructions are written for Word 97 and above. The master document bug afflicts those products to a much greater degree, due to their more complex internal file format.

## 2. Why Master Documents corrupt ...

For information on why they corrupt, please see the article **Why Master Documents corrupt** .

## 3. And how to recover them

I strongly recommend that you read through this whole article first, before you attempt any of it. The procedure is long and potentially confusing. The command and menu names come from Word 2000. You may have to apply a little interpretation, particularly if you are working in a language other than English.

Please bear in mind that this procedure recommends that you do particular, specific actions in a tightly defined sequence using specific methods. That's because I have found that it works that way (and in many cases I found out the hard way that it ***doesn't*** work any other way). It is very tempting (particularly for me) to read such a procedure and say “I know all about Word, I don't need to follow all that!” Allow me to gently suggest that if you did, you wouldn't be here {*grin*}.

### Software Patches

Microsoft issues “service releases” for its major software about twice a year. The first one contains fixes to many of the issues that were discovered during beta testing. Between service releases, Microsoft may release one or more patches or fixes.

If you use the default numbering and bulleting buttons on the Formatting toolbar, then for Word 97 and 2000, one of the more important fixes is for the build-up of unwanted list templates. See Microsoft Knowledge Base articles **Q237274** and **Q241581** (“Error Message: ‘This Document May Be Corrupt’ After Switching Between Bullet and Number List Format”). This fix is included in Word 2002 as standard.

Unfortunately, though, the fix does not work unless the user opens a document contains so many list templates that *without* the fix, they would get the ‘This Document May Be Corrupt’ message (more than 1500 list templates). However, documents containing anything over 100 list templates are prone to other forms of document corruption. And in any case, the vast majority of document corruptions are not specifically list template related.

Bugs that have been corrected in the service releases or patches may have caused the problems you are having. If your software is not at the latest service level, there is little point in reading any further: it is more likely than not that your problem will simply re-create itself as soon as you save the document.

To obtain the patches, see **Obtaining Patches** , below.

### Scan for Viruses

So often, the cause of a document corruption is one of the myriad Word macro viruses floating around the world.

Ensure that you download the latest anti-virus definitions from your anti-virus vendor and scan your whole machine. Definitions more than a month old are too old, and you need to ensure you have your virus scanner configured for a deep scan.

### Create a Template

All the initial settings for a document and its styles are held in a file called a Template. If you do not understand templates, read “About templates” in the Word Help.

Document corruptions often begin in, or are copied to, the template the document is attached to.

The template that is most likely to corrupt is the default global template, **Normal.dot**. We need to create a new, known good template to work with. Until we do, we are likely to get the problem back at any instant, whenever the corrupt template is accessed (which can happen dozens of times during an editing session).

To create a template, see **Creating a template** , below.

### Create a base document

The first thing you need to do is re-create your whole file as a single long document.

Because the master document file format itself is unstable, unless you re-create your file as a single document, the problem you have will simply re-occur.

To create a base file see **Creating a base document** , below..

### Copy Text

Copying the text from your corrupt documents into your base document must be done very deliberately, with the utmost attention to detail.

Think of your problem as a “virus” (because it will behave a bit like one). It is an infected piece of code, sitting there in your file. Every time you access that piece of code, the infection will spread a little further.

You need to copy the text extremely carefully to ensure that you do not also copy the corruption into the new document. To copy the text, see **Copying text** , below.

### Formatting

While it is true that your troubles have been caused by a bug in Word, the reason the bug bit is because of the way you used Word.

When you begin working on long, complex documents, all the rules change. Normally, Word is very, very forgiving. There are 45 ways to skin any given cat, and one way is as good as any other, so Word invites you to use the way that suits you.

This is **not** true in long document work. There are very definite rules for what you can and cannot do. If you break the rules, you break your document. See **Repairing Formatting** , below.

## 4. Obtaining patches

First, determine whether your software is at the latest level. If you open the Help menu in any Microsoft Office product, you will see an **About** item on the menu. If you are using Office 2000 with Adaptive Menus, you may have to pause on the Help menu for several seconds until the *About* item is revealed.

Open this item. A dialog box appears. The top line of text in that dialog box provides the official product description for the software.

The end of the line contains some characters that tell you what service release you have. The level should be as shown in the **following link** .

It is **not possible** for **anyone** to solve your problem until you have the latest service release.

## 5. Creating a template

1. Close all the documents and reboot (We need a nice clean computer without any dead bits of Word to interfere with things.
2. Make sure you do not have anything else running for this exercise except your virus scanner. Close your email program (particularly: we don't want anything trying to talk to Word while we're doing this) and any other applications.
3. Find and re-name your **Normal.dot** file. This is often deeply buried in your user profile. To find it, look in Tools>Options>File Locations. If you attempt to Modify the User Template location, the File Location you see when the dialog box opens is where your Normal.dot is. For god's sake do not change the location! Write the whole path down, cancel all the way out, stop Word, navigate to the location in Windows Explorer and re-name the file. Leave Explorer open where it is, you will need to come back here. You cannot rename the file if Word is running. It does not matter what you name the file: I usually add a single letter to the front of the file name: “onormal.dot”.
4. Re-start Word. When it completes starting, stop it again. Word performs a check on startup to ensure it can find its **normal.dot**. If it cannot, it creates a new, blank default normal.dot. The file it leaves you when it exits is an absolutely perfect template: they do not get any more pure than this.
5. Make a copy of this file and give it any name you like. Let's call it **project.dot**. We now have two files that are known good and guaranteed perfect in the directory.
6. Re-start Word and use File>Open to open **Project.dot**. Look in the Title Bar of Word. Do you see “**Project.DOT**” displayed? You don't, huh? You've got “**Document2**”, haven't you? You opened the file out of Explorer by double-clicking, didn't you? Let that be a lesson to you as to the level of nit-picking detail with which you have to follow this procedure, or it won't work {big grin}. Throw that document away and go back and open **project.dot** using File>Open! Alternatively, you can **right-click** on Project.dot in Windows Explorer and select **Open** – that opens the template rather than creating a new document based on it.
7. Use File>Page Setup to set your correct paper size, layout, and margins. Work slowly and patiently. Every project document you create will inherit these settings, so get them right. It's a matter of personal preference, but I would not put headers and footers in just yet. Let's find out if this is going to recover your document before we add the nice-to-have's.
8. Now, go to Format>Style>Organizer and click the **Styles** tab. In the right-hand column of that dialog box you will see the styles for **Normal.dot** displayed (there should only be four at the moment). Disregard this and click the **Close** button below the right-hand-column. We do not want to do anything with **Normal.dot**. The **Close** button turns into an **Open File** button. Click it.
9. Navigate your way to the “best” of your sub-documents. Make very sure it is not the master document you are looking at. This is critical: we must be very sure that we are not about to open the master document. Open the subdocument's style list.
10. Copy all the styles into your new **project.dot** file **except** the ones whose names begin with “Heading”. Say “yes to all” if you get prompted to overwrite. Click Close to this dialog box when done. This is called an “organizer copy”, and it is the only 100 per cent safe way to copy styles from one document to another. The nine Heading styles are built-in to every Word document. You won't see them in the list until they are used in a document, but they are part of the document structure, so we do not want to add them. It is possible that one of them actually contains the corruption.

Now we need to fix the List Templates. Open the Format>Bullets and Numbering menu.

1. Go first to the Bulleted tab. Click in each of the offered samples and click the **Reset** button for each. Work your way through the whole list.
2. Repeat this operation on the Numbered, and Outline Numbered tabs. Again, click in each of the offered samples and click the **Reset** button for each. Work your way through the whole list on both tabs.
3. Click Cancel to leave this dialog box. What we have done here is to set all the numbering list templates back to their factory defaults. In Word 97 and above, bullets and numbering are the same thing. A corruption in this mechanism is the most likely source of your master document woes.
4. Save and close the template. Exit Word and re-start it. We do this to force a write back to your registry. For some unbelievable reason (I know the reason, but it's just too silly to mention...) Word stores its bullets and numbering specifications in your user registry. This is why your numbering breaks whenever someone logged in as a different user opens your documents, even on the same computer. This is the subject of another FAQ ...
5. We now have a new, dependable template, in a known good condition, with the same style settings as your master document had. The only possible differences can come if you ignored Word's warning that all the subdocuments of a master document must have the same template attached. If they didn't, Word should have imposed it. If this causes a problem, you can sort it out after you fix the formatting at the end of the procedure. If you try now, you will probably break your new template.

## 6. Creating a base document

Next we create a new, blank document from the new template you just created.

Do it this way:

1. Select File>New and choose **Project.dot** as your template.
2. You will be rewarded with a new blank document. Let's call this file “**New Document**” hereafter.
3. Go to Tools>Options>Save and ensure that you have **Fast Saves** turned OFF. Check “**Always make backup**” is ON at the same time.
4. Go to Tools>Track Changes>Highlight Changes and ensure that “**Track changes while editing**” is turned off.
5. Give **New Document** a name (must be a new name, do **not** save over an existing document) and save it. You must not write over the existing documents: you may have to repeat one or more of these procedures more than once; and you will need your originals when you've finished.
6. Make sure you save to a directory where you have plenty of space: this thing is going to get quite large. Just a word about floppies (diskettes to those of us from IBM...). We do not save Word documents to removable storage. Not **ever**! If you need a document on a removable disk, close Word and copy the file using Explorer. A floppy is simply not large enough or reliable enough to contain a Word document. This has to do with the “streaming” format Word uses to write files to disk. If you use floppies, you break documents. So don't.

## 7. Copying text

Now follow the following procedure (exactly) for each subdocument:

1. Open each document or sub-document, one at a time.
2. Make sure the paragraph marks are displayed. Use Tools>Options>View to display them if needed: it is essential that you can see the paragraph marks.
3. Remove all the Section Breaks from the document. Search for **^b** and replace with nothing. Note the “b” must be lowercase and the caret “^” is required. Most document corruptions are contained in Section Breaks: you must remove them.
4. Remove all your direct formatting: Select all of the text with **Ctrl + A**, then remove all paragraph formatting with **Ctrl + Q** and all character formatting with **Ctrl + spacebar**. Your document may start to look quite peculiar at this stage: do not worry about it, and don't try to fix it now.
5. Select all the text from the top of the document down to **but not including** the final paragraph mark and copy it to the clipboard. The bottom paragraph mark in a document hides the default Section Break, which is probably what contains the problem. If you copy the last paragraph mark, you will copy your problem.

6. Close the subdocument **without saving**. You do not want to save the changes you have just made back to your original. Close the document, and when prompted to save, say **no**.
7. Open **New Document** and paste the text you have copied into its correct position (normally, at the end).
8. Save **New Document**, and do not attempt **any** fixing at this stage. You save between each operation because you may encounter a serious corruption in one of the files you open. If you do, Word will crash without warning, and you will lose everything you haven't saved.
9. Repeat these steps until you have copied all the text from all the components into **New Document**.

## 8. Repairing formatting

Do not attempt any fixing of any of the text until you have all the text assembled in **New Document**. If your PC is correctly configured, Word 97/2000 will easily handle a document up to about 1,000 pages, so do not worry: it will slow down a bit but it won't break.

Now you begin the fixing process. Do it in this order or you will end up chasing your tail:

1. Replace the Section Breaks using Insert>Break. You still have your originals so you can search for them to see where they should go. But remember, the Section Breaks probably contain the problem: if copy any Section Breaks by mistake, you probably get to start over from the creation of **New Document** {grin}.
2. When replacing the Section Breaks, you should probably leave out most of the breaks you find in the original master document. In a master document, each subdocument is surrounded by two extra “encapsulating” Section Breaks that are needed only in the master document: they should not be used in New Document. The fewer Section Breaks you end up with, the more reliable your result will be.
3. Now unify the styles. First go through and re-name/re-apply misnamed styles. Often where unskilled authors have attacked a document, you will find that styles have been applied at random, and that there are several versions of each style. For example: You may find a “text” style, a “Text” style, a “body-text” style, a “Body-Text” style and a “body text” style. Careful inspection of the document will show that they are all actually the same thing, created by different authors, in different source documents.
4. Before you can sort out your styles, you need to pick **one** name (doesn't matter which one, but your life will be easier of you pick the Word built-in style name) and use Find/Replace to apply that to the paragraphs that have the other named styles. When using Find/Replace, make sure you do not have any text in the **Find What** or **Replace With** boxes. Click the **More** button to reveal the **Search Options**, and on the **Format** button choose the **Style** option.
5. You must now delete the duplicate styles from both the document and its template (otherwise in a months time, you will have random styles again...) I suggest you use Organizer to do this: it is easier to see what you are doing, and it is the only way of deleting styles reliably from the template. Note that Word will not permit you to delete built-in styles (which is why I suggested that you use those...).
6. Now you need to correct the formatting of each style. You use Format>Style to alter the properties of each style so that the document is correctly formatted. Some hints:
   a) Make sure you check the “Add to template” box for each style, or your changes will not be saved.
   b) Make sure **none** of your styles are “Based On” Normal style. Correct any that are. Base your headings on the heading level above and your text styles on Body Text. I normally leave Normal Style orphaned so that wherever it is used in the document I can tell that I am looking at text that has not been formatted yet.
   c) Make sure that all the automatic formatting options remain turned off. Make sure that you also disable Fast Saves and check Always Make Backup. Make sure you disable **Automatically Update** on all your styles. If your document contains numbering, you must also disable **Automatically Update Styles on Open** when you attach the template.

Now, you need to put the numbering right. Here's the procedure. This procedure fixes Heading Numbering that displays out-of-sequence numbering. It does NOT fix any other kind of bad numbering.

If you omit any of these steps or fail to perform any, the procedure may not work. So unless you *really are sure* that you do not need to perform a step, do it anyway :)

1. Go back to Tools>Track Changes>Accept or Reject Changes and click either **Accept All** or **Reject All**.

2. Click in the **first** paragraph in the document that has the Heading 1 style applied. If there is no such paragraph, create one.

3. Go to Format>Bullets and Numbering>Outline Numbered and click **none**.

4. Save and close the document. When you save the document with Fast Saves turned off, you cause Word to clean out of the document all the previous editing changes. With Fast Saves on, these remain in the text where they do embarrassing things like let your customer read all the previous versions of a contract, or store corruptions. Furthermore, normally when you make a change to a document, Word simply marks the changed bits as "superseded" and leaves them in the document, in case you want to undo the change. When you close the document, Word discards the "Undo List". In this case, the problem you are having would be saved one way or another: we want to force Word to clean the file up.

   If you skip the close at this point, chances are the whole procedure won't work :)

5. Re-open the document and click in the **first** Heading 1 paragraph in the document again. We do this operation always from the first Heading 1 paragraph because that is the beginning of the list. If we start with a lower-order paragraph like Heading 3, we run the risk that we might create two independent "lists" of numbers in the document, which is actually what our problem usually was in the first place.

6. Go to Format>Style>Modify>Format>Numbering> Outline Numbered and choose the appropriate numbering format. Notice that we came into the dialog a different way this time? Numbering is a hell of a lot more stable if it is applied as part of a style, and that's what we are doing this time. If you have correctly reset your registry, one of the formats shows the word "Heading" in its sample. This is a **critical** step: **only** the samples that show the word "Heading" will actually do Heading numbering. The other samples simply won't do it. You can force them to try if you customise them within an inch of their lives, but you run the risk of corrupting the document if you do. This risk approaches certainty unless you understand Lists, List Templates, List Galleries and Styles almost perfectly.

7. If it is possible to fix that document's numbering, you just have. Anything else wrong with the document is either not a numbering problem, or the document is beyond repair (which is not an unusual condition if you have been trying to fix it for a while).

8. You now need to scan through the document and make sure that every Heading is correct. A very quick way to do this is to go into **Outline View** and click the black number 7 button, so you only see the headings without the text. Bad numbering will then leap out at you.

9. You should also scroll through the document in Page Layout view after this. If users have made multiple attempts to fix the numbering, they may have resorted to manually typing heading numbers. You need to scan for these and replace them by applying the correct style to such paragraphs.

10. Now, take a backup copy of your document and put it in a different folder. This is your baseline that you can come back to if the document falls to bits again. If the document was very corrupt, the above procedure may not have completely fixed it (and short of saving to Word 2 format, there is no way to do so). In which case the corruption may recur.

11. Finally, you can customise the numbering format to your standard. Notice I said "**the** numbering format." If you apply a **different** numbering format to the headings, you may find yourself back at Step 1 again {grin}.

# Why Master Documents corrupt

Article contributed by **John McGhie**

**Update:** Word MVP **Steve Hudson** has an article describing how to use master documents safely **here**.

The complete explanation would be a book in itself. For now, it is enough to know that a Word document is a great big “list” of objects. An object can be anything you can put in a Word document. Each of these objects has many, many “properties” that determine how it appears and how it behaves.

The properties are all contained in several giant “tables” inside the file. The connection between any given object (say, a paragraph) and its properties is made with an amazingly complex lattice-work of “pointers”. These pointers are large binary numbers that cause Word to look at an exact byte location in the file to see what shape, size, or colour this object should be. Most objects have more than one pointer. Some pointers go to “collections” of properties (for example, a “List Template” that describes all the formatting for a numbered list) and some go simply to a single entry (for example the “language” that is just a single name).

Whenever we experience a “Word document corruption”, what has actually happened is that the pointer, or the entry in the table it points to, has become corrupted. The information found there is either nonsense, or it does not apply to the object in question. For example, a paragraph is trying to inherit page margins: a paragraph cannot have page margins, so Word gets terribly confused.

All these property tables are stored in Section Breaks. A Section Break is not just a “page break”, it is a binary container that stores several hundred properties in multiple tables. The largest Section Break is the “Default” Section Break. You will never see one. The default Section Break hides in the very last paragraph mark of a document. Because it is absolutely essential to the document (without it, the file is just a stream of bytes, not a document) Word maintains the contents itself and hides it from you and me.

The reason that Master Documents cause so much trouble is that you are asking Word to merge together many hundreds of different settings, some of which conflict, some of which apply only to one or a few paragraphs. A typical master document may contain 20 sub-documents. This means there are 21 “default” Section Breaks, each containing potentially-conflicting properties. Each subdocument also can contain multiple “user” Section Breaks. These may or may not override or conflict with the settings in one or more of the default Section Breaks.

If a property is specified, does it apply to this document? Some of this document? Several of these documents? And is the document that stores it open? Is it “active”? Read-only or editable? The number of possibilities rapidly expands, geometrically, until the structure simply becomes too complex. Word loses track of what it is trying to do. And takes a guess. The guess overwrites something: and Bingo! You lose your master document.

When we say you “lose” your master document, this “loss” can take many forms. You wouldn't be reading this at all if you had not so far experienced one of the lesser forms. You can still read “some” of your text, right? Trust me, it can get worse! The ultimate master document corruption results in some or all of the text paragraphs disappearing. Once this happens, there is no way to get them back: they are no longer in the file. Which can be very disconcerting if the corruption happened several weeks ago, and because you were not looking at that part of the document, you didn't find out about it until you came to print the whole thing, by which time you had long since over-written your backup!

A master document has only two possible states: Corrupt, or just about to be corrupt. And **that** is why we say that the only possible fix to a master document is “don't use it!”

For information on how to recover a Master Document, please see the article **How to recover Master Documents** .

Send all feedback to **word@mvps.org** Top - **Ctrl+Home** Previous - **Alt+left arrow** Page Updated:

# I have a “Name” column which I want to split into “FirstName”, “LastName” – how can I do it?

Article contributed by Dave Rado

Terms of use Disclaimer

Word's sorting capability is fairly rudimentary, especially for those migrating to it from WordPerfect (though it's surprising how many people don't realize Word can sort paragraphs, not just tables – or maybe not so surprising, given where the item is in the menus! The ability to sort on word 2 in field 3 would certainly be very useful (in Excel as well). I believe there has been a lot of demand for this capability to be added, so maybe it's on the way ...

But there are various things you can do in the meantime. One is, write a macro. Another is to use Find and Replace to separate the text into fields (using tabs as the field separator). For instance, I often use the following trick if I'm emailed a spreadsheet with a “Name” column which I want to split into “FirstName”, “LastName”: The following assumes the column is in Excel, but of course it could just as easily be in a Word table.

**1.** Select the column in Excel and paste it into a blank Word document.

**2.** Select the column in Word (press Alt + Left mouse click), then press Ctrl+Spacebar to remove manual character formatting, and select **Table + Convert Table to Text**.

**3.** Use Find and Replace to replace any spaces with single ones:

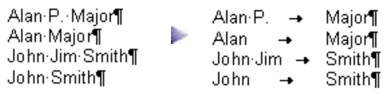
In the Find what box type: **^w**
(or select the "Special" button and choose "White space")
In the Replace with box type **[space]** (i.e. press the spacebar)
Click **Replace All**

**4.** Next, you want to replace the spaces preceding the last names with tabs, in order to be able to convert back to a table separated by tabs.

To allow for the possibility that, somewhere in the list, there might be some middle names or initials, select the “Use wildcards” check box in the Find and Replace dialog, and:

In the Find what box type: **(<*) ([! ]@)^13**
In the Replace with box type: **\1^t\2^p**
Click **Replace All**

That will give you the following result (with non-printing characters displayed):



```
Alan·P.·Major¶           Alan·P.  →  Major¶
Alan·Major¶              Alan     →  Major¶
John·Jim·Smith¶          John·Jim →  Smith¶
John·Smith¶              John     →  Smith¶
```

For more on using wildcard searches see: Finding and replacing characters using wildcards.

Alternatively, if your original list was in the format:

Major, Alan P.

Then you **don't** need to use wildcards in order to insert the tabs in the right place

In the Find what box type: **,[space]** (type a comma then press the spacebar)
In the Replace with box type **^t**
Click **Replace All**

**5.** Select the text (**Ctrl+A**) and select **Table + Convert Text to Table**.

You should find that the Word dialog suggests 2 columns and that text should be separated at tabs. If not that you don't have any rogue text or empty paragraphs selected at the end of the document.

If there **were** any middle names or initials in the list, I fix these later by creating a MiddleName column in Excel and putting these orphans into it.

**6.** Type “FirstName” and “LastName” in the two cells in the top row of your new table.

Copy your Word table, Alt+Tab into Excel, insert an additional column to the right of the “Name” column, and paste.

Sounds convoluted and I guess it is, but it doesn't take long – I've fixed 1,000 record mail merge Data Sources in minutes using this method.

# How to add pop-up lists to any Word document, so you can click your way through changes in seconds

*Or how to use the AutoTextList field*

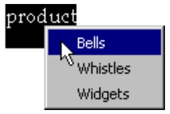
Article contributed by Bill Coan

Terms of use Disclaimer

## Background

Do you re-use some of your documents over and over again, making slight changes just before you print, fax, or email it each time? Do you, for example, send the same basic letter to each new customer, but edit the letter each time so that it refers to the specific product purchased by that customer?

Starting with Word 97, there's an easy way to add a pop-up list of choices to any Word document. This new feature lets you point at a word or phrase and simply right-click the mouse to switch to some other word or phrase.



For years Word has been able to do something similar with drop-down form fields, but many users chafe at the limitations imposed by form fields. For example, form fields require that documents be protected for forms, which means you have to sacrifice spellchecking, drawing tools, and text formatting. That's quite a penalty to pay, just so you can choose an item from a list!

In Word 97 and above, you can sidestep this penalty by relying on a new feature called the **AutoTextList** field. Before I show you how easy it is to use this new field, I need to provide some background about AutoText entries and paragraph styles, for users who haven't explored these items.

**Important!** If you're already familiar with AutoText entries and paragraph styles, or if you're not interested in “looking under the hood,” feel free to skip over these background items and jump right into the procedure that follows.

### Background Item #1
AutoText entries are snippets of text and/or graphics that can be added to your document by choosing the desired entry from the AutoText toolbar. (To display this toolbar, right-click any other toolbar and choose AutoText.)

### Background Item #2
Word's default global template, normal.dot, ships with numerous AutoText entries but you can create additional entries and store them in normal.dot or in any other template desired.

### Background Item #3
Paragraph styles are collections of paragraph properties that can be applied to a paragraph by clicking in that paragraph and then choosing the desired style from the dropdown list of styles on the Formatting toolbar. You can create new paragraph styles and store them in normal.dot or in any other template or document desired.

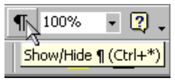
### Background Item #4 (the kahuna)
What few users realize is that Word organizes its AutoText entries by paragraph style. For example, if you click in a paragraph that has been styled as “Product Name,” Word's AutoText toolbar will automatically display only those AutoText entries that have been designated as Product Names (assuming there are any). The AutoTextList field has this same ability, except that it pops up the list of Product Names when you point at the field in your document and click the right mouse button.

## OK, the procedure

Ready to create your first pop-up list? Don't be intimidated by the number of steps. Each step involves just one or two mouse clicks and all 13 can be completed in less than five minutes. Besides, you only have to do this procedure once for each list and then you can use each list over and over again as needed.

Before you begin, you may want to print out this procedure to guide you along the way.

### Create a pop-up list of products

1. On the Standard toolbar, click the tool to display paragraph marks if not already displayed.


2. Position the cursor in an empty paragraph.
3. From the dropdown list of styles on the Formatting toolbar, choose the **Normal** style if not already chosen.
4. Type your list of products, pressing **<Enter>** after each product name.
5. With your mouse, select the entire list of products.
6. Click in the Styles box on the Formatting toolbar, type **ProductStyle** and press **<Enter>**, to create the new style:

(Alternatively you can go to Format | Style | New, but that's more long-winded).
7. Highlight each individual product name (being very careful **not** to highlight the paragraph mark after the name) and press **Alt+F3** and hit **OK**.
8. Delete the list of products and again choose the **Normal** paragraph style if not already chosen.
9. Press **Ctrl+F9** to insert a pair of field braces. **{ }**
10. Type the following expression between the field braces **exactly as shown**:

    **{** AutoTextList "product" \s "ProductStyle" \t "Right-click to select product" **}**

11. With the cursor still between the braces, press **F9** to update the field (If this fails to hide the field code, press **Alt+F9** to hide the code.)
12. With your mouse, select the new field, being careful **not** to select the paragraph mark (¶) that follows the field.
13. Press **Alt+F3** and hit **OK**.

### Insert a product list in your document

1. Click where you want the list to appear.
2. Start typing “product”; if Word offers to complete the typing for you, press Enter or Tab to accept the autocompletion. If Word doesn't offer to complete the typing for you, select “product” and press **F3**.

### Pop up the product list

1. Point at “product” and click the Right mouse button.
2. Choose the desired product.

### How it works

The secret to the AutoTextList field is in the field code, repeated here for ease of reference:

**{** AutoTextList "product" \s "ProductStyle" \t "Right-click to select product" **}**

Here's what each element of the code means:

| | |
|---|---|
| **AutoTextList** | This is the field type. This particular type of field creates a pop-up list. |
| **"product"** | This is the field's default value. After the field is created, this value will be replaced when you right-click the field and select a different value. |
| **\s "ProductStyle"** | This tells the field that you want the pop-up list to display only those AutoText entries that were formatted with the ProductStyle. |
| **\t "Right-click to select product"** | This tells the field to display a specific tooltip when the mouse pauses over the field. |

## Additional tips for advanced users

- If you rely on different templates for different types of documents, make sure you're editing a template when you complete the procedure described above. At Step **7** and against Step **13**, instead of pressing **Alt+F3**, choose **Insert |AutoText | AutoText....** and then set the **LookIn** list to your specific template and click **Add**.
- Don't delete the paragraph style created in Step 6. If you do, your AutoTextList field will display **all** of your AutoText entries, not just the ones of the designated style.
- In Step **7**, if you include the paragraph mark that follows the product name, the resulting AutoText entry will be inserted into your document with a paragraph mark and the entry will be formatted with the Style created in Step **6**. For most users, this is not desirable, but if this is what you want, by all means go ahead and include the paragraph mark.
- To make it easier to see the lists embedded in a document, set Field Codes to be shaded Always. To do this, choose Tools|Options|View|FieldCodes|Always.
- To navigate from field to field with the keyboard press F11. To navigate backward from field to field, press Shift+F11.
- To popup the shortcut menu when the cursor is on an AutoTextList field, press Shift+F10.
- If you don't like the "Create Autotext..." command that appears at the bottom of the popup menu, Choose Tools | Customize | Toolbars | Shortcut Menus | Text | AutoTextListField and drag the command from the menu to delete it, then click OK.

Send all feedback to **word@mvps.org** Top - **Ctrl+Home** Previous - **Alt+left arrow** Page Updated:

# Creating a macro with no programming experience using the recorder

Article contributed by Bill Coan

Terms of use · Disclaimer

Word's macro recorder can help you acquaint yourself with macros and with Office 97's vba programming language.

Let's assume that you create several documents from scratch on a daily basis. After composing the text, you press Ctrl+Home to position the cursor at the start of the document. Then you click the Center tool on the Formatting toolbar to center the first paragraph of your document. Eventually, it occurs to you that you should be able to record a macro for these last actions. That is, a macro that will automatically position the cursor at the start of the document and center the first paragraph.

Technically, this scenario may call for a solution involving paragraph styles. But for now let's create a macro solution and let's use the macro recorder to create it.

Proceed as follows:

1. Choose Macro|Record New Macro on the Tools menu (or simply double-click the REC button in the Status bar.)
2. Enter a macro name (sorry, no spaces or punctuation allowed).
3. Click the Assign Macro To Keyboard button.
4. Press Ctrl+F8, then click Assign and click Close. Word will display the Stop Recording toolbar, which lets you know that Word is recording your actions and will continue recording them until you click the Stop Recording button on this toolbar. (Don't click it yet!)
5. Press Ctrl+Home to position your cursor at the start of your document.
6. Click the Center tool on the Formatting toolbar.
7. Click the Stop Recording tool on the Stop Recording toolbar.
8. Open some other document.
9. Position the cursor in the middle of the document (doesn't matter where).
10. Press Ctrl+F8.
11. Smile with satisfaction as Word moves the cursor to the start of the document and centers the first paragraph.

So far, so good. Now to see what the macro looks like under the hood:

1. Choose Macro|/Macros on the Tools menu.
2. Select the macro that you just created.
3. Click Edit.

Word will open up the vba editor and display something like the following macro:

```
Sub MyFirstRecordedMacro()
'
' MyFirstRecordedMacro Macro
' Macro recorded 10/31/98 by Elgin
'
Selection.HomeKey Unit:=wdStory
Selection.ParagraphFormat.Alignment = wdAlignParagraphCenter

End Sub
```

4. To learn more about the code, position the cursor somewhere in Selection.HomeKey and press F1 to view a related help topic.

5. Close the help window and repeat Step 4 with the cursor in other locations.
6. On the vba editor File menu, choose Close and Return to Microsoft Word.

If you have the patience to try these steps, you'll be well on your way. The rest is just details. There are lots of them but you can always search the vba help system and return to the newsgroups for additional information.

# Running a macro automatically when Word starts or quits

Article contributed by Dave Rado

## To run a macro when Word starts

In a Global template, if you write a macro called **AutoExec**, it will automatically run when you launch Word.

## To run a macro when Word quits

In a Global template, if you write a macro called **AutoExit**, it will automatically run when you quit Word.

**See also** the article: Writing application event procedures.

# Running a macro automatically when a document is created, opened or closed

Article contributed by Dave Rado

## Using Document events

Open your template, press **Alt+F11**, and in  the Project window of the VBA environment, double-click on  “Microsoft Word Objects”, then on “ThisDocument”. On the toolbar you'll see two list boxes. If you pull down the one on the left, and change it from “(General)” to “Document”, a procedure called **Document_New()** will be created. If you then pull down the list box on the right, you'll see three events to choose from: **Close**, **New** and **Open**. You can select Close or Open from the list to insert a **Document_Close()** or **Document_Open()** procedure (or you can forget about the list boxes and just type, once you know the syntax).

A **Document_New()** procedure will run when a document based on that template is created; a **Document_Open()** procedure will run whenever a document based on that template is opened; and **Document_Close()** will run a document based on that template is closed.

Note that these procedures **cannot** be made global – they will only be fired when documents *based on the  template* are created or opened or closed.

## Using Auto macros

Another way of achieving the same objective is to create a Module (Insert + Module), and write a macro called **AutoNew()**, **AutoOpen()**, or **AutoClose()**. If stored in any template other than Normal.dot, these will behave in the same way as Document events; i.e. they will be fired when documents attached to the template are created, opened or closed.

However, if stored in Normal.dot, they will act globally – in other words, they will be fired when *any* document is created, opened, or closed. This is in contrast with a Document_Open procedure stored in Normal.dot, which will only execute when documents based on Normal. dot are opened.

Unfortunately, AutoNew, AutoOpen and AutoClose macros stored in an Addin (a .dot file stored in Word's Startup directory) will **not** behave globally. In fact there is no point in storing AutoNew, AutoOpen or AutoClose macros in an Addin, because you would (or should) never base a document on an Addin.

## Using Application Events

If you want a macro to be fired whenever *any* document is opened, regardless of which template the document is attached to, the simplest way, as discussed above is to write an AutoOpen macro and store it in Normal.dot. However there are problems associated with storing macros in Normal.dot, so if you want to avoid that route, the answer is to use Application Events. Application Events stored in global Addins *do* behave globally. And rather confusingly, some application events relate to documents.

In Word 97, you can use the DocumentChange event of the Application object to simulate global Auto macros; and in Word 2000, you can use the DocumentOpen, NewDocument and DocumentBeforeClose events of the application object. Storing these in an Addin works just like storing Auto macros in Normal.dot (i.e. they're global).

See the article: **How to create global event procedures similar to AutoOpen, AutoNew and AutoClose, without using Normal.dot** for more details.

# Getting to grips with VBA basics in 15 minutes

Article contributed by Bill Coan

Terms of use · Disclaimer

I can't turn you into a VBA expert but I can suggest a way to explore VBA that you may find helpful. Below, I've listed 22 steps that can be completed in approximately 15 minutes, assuming someone is kind enough to read them to you as you sit at your keyboard. If you have to read them by yourself and turn your attention alternately to the keyboard and back to the steps, then you may need a half hour or longer to complete the steps. Either way, the steps should give you a feel for what it's like to program in Word.

Before starting, launch Word, then press **Alt+F11** to launch the VBA editor, and then maximize the VBA editor window.

Ready? Let's start:

**1.** In the VBA editor, choose Options on the Tools menu and make sure the following checkboxes are checked:
Auto List Members
Auto Quick Info

**2.** Press F7 to view a code window, if not already displayed.

Then type “Sub Test” and hit the enter key. The VBA editor will create a subroutine for you that looks as follows:

```
Sub Test()

End Sub
```

Notice that the cursor is already flashing inside the subroutine, ready for you to type a command

**3.** Click in the code window and type the following, making special note of the dot at the end of the expression:

```
ActiveDocument.
```

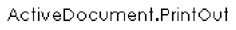
**4.** If you want to work on the active document (and who doesn't) then you have to start this way. (Think Jesus: No one gets to the father except through me!) When you type the "." at the end of the expression a list will pop up. This list is the most amazing guide you can imagine. Each item on the list is either a method or a property of the ActiveDocument or else it's an object unto itself that belongs to the ActiveDocument. For now let's deal with methods. Toward the end of this message we'll deal with properties. At the very end of the message, we'll deal with other objects besides the currently active document.

A method is something you can DO to the ActiveDocument, like print it out. The way to think is this: “I'm trying to do something to the active document as a whole, namely, print it. If I'm patient and scroll through this list, I'll almost certainly find a method that will do this for me.” When you first think this way, the word “method” will stick in your throat like a fishbone. Later it will feel more like burnt toast and later still it will feel like candy when you were young. Indeed, in this case, if you're patient, you will scroll far enough to encounter a method called PrintOut. Eureka! You know it's a method (as opposed to a property) because it has an icon next to it that looks like a green brick flying through the air and at this stage the whole concept of method makes you feel like throwing a brick. Nice mnemonic, eh?

Let's give it a try. Since you've scrolled down and selected PrintOut, you can press Tab to accept it or you can double-click it right there in the list. Either way, your statement now looks as follows

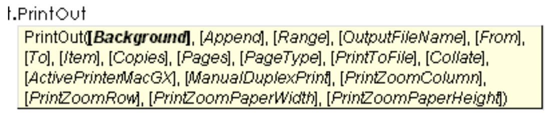
```
ActiveDocument.PrintOut
```

5. So far so good. Most people easily get this far. But now what?
6. One possibility is that you're done. After all, you've specified an object (the active document) and you've selected something that you want to do to it (print it out). Indeed, if you're willing to let Word print out the document in whatever way it chooses, then you **are** done.
7. Another possibility is that you want to control how the print job will be carried out. In this case, you must provide some “arguments” (another fishbone, at first). Not to worry. To enter one or more arguments, all you do is type a space after the method and let the editor help you enter them. Arguments are equivalent to the choices you make in the Print dialog box. So go ahead, type a space after PrintOut, so your screen looks like this (the pipe character “|” represents your flashing insertion point after you've typed a space):



```
ActiveDocument.PrintOut
```

8. But wait! When you type the space, the VBA editor suddenly gets very helpful again, showing you something like the following:



```
t.PrintOut
PrintOut([Background], [Append], [Range], [OutputFileName], [From],
[To], [Item], [Copies], [Pages], [PageType], [PrintToFile], [Collate],
[ActivePrinterMacGX], [ManualDuplexPrint], [PrintZoomColumn],
[PrintZoomRow], [PrintZoomPaperWidth], [PrintZoomPaperHeight])
```

9. Aha! Aha! These are the arguments for the PrintOut method. Most of them are immediately recognizable if you've ever paid attention to the Print dialog box. If you're wondering about one or more of them, simply press F1 and you'll call up a help topic that tells you all about each one of them.
10. Now that you can see all the arguments, you can enter values for as many of them as you desire.

    One way to do this is to type the name of the argument and then a value, using “:=” to connect them and a comma to separate one argument/value from the next, like so:

    ```
    ActiveDocument.PrintOut Background:=False, Copies:= 2
    ```

    Another, boneheaded (in my opinion) way to do this is to enter values for ALL of the arguments, in which case you don't have to type the names of the arguments but you do have to account for ALL of them, as follows:

    ```
    ActiveDocument.PrintOut False, , , , , 2, , , , , ,
    ```

11. Let's go with the named-argument approach:

    ```
    ActiveDocument.PrintOut Background:=False, Copies:= 2
    ```

12. That wasn't so painful, was it? Now press **F5** to run the subroutine. Or return to Word and choose **Tools|Macro|Macros...|Test|Run**. (Pressing F5 is easier!)

    A quick review before we plunge on to properties. You think: “I'm trying to do something to the active document as a whole, namely, print it. So I start by typing “ActiveDocument.” and a list pops up. I scroll through the list and select the PrintOut method. Then I type a space and enter some arguments. In this case, I want background printing off and I want two copies, so I type the names of those arguments and values for each of them. I connect each argument name to its value by using “:=” because I'm part of the cognoscenti.”

    **Take a big breather here because now it's time to explore properties instead of methods . . .**

13. Let's go back to our original assumption, namely, that you want to work on the active document as a whole. In this case, though, let's assume you want to change one of the properties of the document, rather than hit it with a brick.
14. Once again, click in a code window and type the following, making special note of the dot at the end of the expression:

    ```
    ActiveDocument.
    ```

**15.** Remember, if you want to work on the active document (and who doesn't) then you have to start this way. (No one gets to the father except through me!) When you type the “.” at the end of the expression a list will pop up. This list is the most amazing guide you can imagine. Each item on the list is either a method or a property of the ActiveDocument or else it's an object unto itself that belongs to the ActiveDocument. For now let's deal with properties.

A property is a single characteristic. One of the properties of a document is its password. The way to think is this: “I'm trying to change a property of the active document as a whole, namely, its password. If I'm patient and scroll through this list, I'll almost certainly find a password property.” Indeed, in this case, if you're patient, you will scroll far enough to encounter a property called Password. Eureka! You know it's a property (as opposed to a method) because it has an icon next to it that looks like a finger pointing at a piece of information. Pretty useless mnemonic, eh?

Let's give it a try. Since you've scrolled down and selected Password, you can press Tab to accept it or you can double-click it right there in the list. Either way, your statement now looks as follows:

```
ActiveDocument.Password
```

**16.** So far so good. Many people easily get this far. But now what?

**17.** In this case, the next step is to specify a value for the property. To do this, all you do is type a space and an equals sign after the name of the property, so your screen looks like this (the pipe character “|” represents your flashing insertion point after you've typed a space):

```
ActiveDocument.Password = |
```

**18.** Since the VBA editor has no idea what value you want to use for a password, it can't offer any suggestions. Instead, you simply have to come up with an idea on your own and type it in, perhaps as follows:

```
ActiveDocument.Password = "billcoan"
```

That's it! That's it! Now press **F5** to run the subroutine. Or return to Word and choose **Tools|Macro|Macros...|Test|Run**. (Pressing F5 is easier!)

When this statement runs, it will assign “billcoan” to be the password for the currently active document. You might wonder how you were supposed to know that the password had to be enclosed in quotation marks. Well, experience ought to be worth something, oughtn't it? In any case, if you had any question, all you would have had to do is position the cursor anywhere in the name of the property (“Password”) and press F1. This would display a help topic that tells you that a password requires a string, which is to say, a bunch of characters inside some quotation marks.

**Take a big breather here because now it's time to explore objects other than the currently active document . . .**

Let's face it, the currently active document, as a whole, can hold our attention for only so long. After all, you can carry out only so many methods on it (printout, save, saveas, etc., etc.) and you can change only so many of its properties (password, grammar checked, spelling checked, etc., etc.).

But what about working on a particular part of a document, such as the first paragraph all by itself, or on a collection of parts, such as all the paragraphs? Here lies opportunity! After all, documents aren't just objects unto themselves; they're composed of hundreds of other, smaller objects. And you can work on each of those objects individually or as parts of collections.

Let's dig deeper and find out how.

**19.** The good news is that you can “reach” any part of a document that you want to work on by starting out as though you were going to work on the document itself. In other words, you can start as you almost always start. That is, once again click in a code window and type the following, making special note of the dot at the end of the expression:

```
ActiveDocument.
```

**20.** If you want to work on a part of the active document then you have to start this way. (Remember: No one gets to the father except through me!) When you type the "." at the end of the expression a list will pop up. This list is the most amazing guide you can imagine. Each item on the list is either a method or a property of the ActiveDocument or else it's an object unto itself that belongs to the ActiveDocument. For now let's deal with objects that belong to the ActiveDocument.

An object is something that you can "work on" by applying methods to it or by changing its properties. Word documents contain lots of different types of objects. When multiple objects of the same type exist (or can exist) in the same document, they are treated as "collections." For example, a word document has, or can have, multiple paragraphs. Each paragraph is an object. All of the paragraph objects, together, form the "paragraphs collection."

The easiest document objects (and collections) to think about are paragraphs, words, and characters. The way to think is this: "I'm trying to work on an object that belongs to the active document, namely a paragraph. Since a document can contain more than one paragraph, I'll have to locate the collection of paragraphs and then specify the specific paragraph that I want to work on. If I'm patient and scroll through this list, I'll almost certainly find the collection."

When you first think this way, the words "object" and "collection" will stick in your throat like a fishbone. Later it will feel more like burnt toast and later still it will feel like love when you hit puberty. Indeed, in this case, if you're patient, you will scroll far enough to encounter a collection called Paragraphs. Eureka! You know it's not a method because it doesn't have an icon next to it that looks like a green brick flying through the air. You don't think of it as a property, either, but you quickly find out that the VBA editor *does* think of it as a property. OK, OK. So one of the "properties" of a Word document is that it contains a collection of paragraphs. Great. So "Paragraphs" is a collection of paragraph objects and "Paragraphs" is a property of a Word document. This is confusing, so quit worrying about it. Focus on the list! Find the item you want to work on!

Let's give it a try. Since you've scrolled down and selected Paragraphs, you can press Tab to accept it or you can double-click it right there in the list. Either way, your statement now looks as follows:

```
ActiveDocument.Paragraphs
```

**21.** So far so good. Most people easily get this far. But now what? A major wrinkle, that's what! But hold on, it's easy to deal with. Since "Paragraphs" is a collection, you have to tell the VBA editor which paragraph you're interested in. You do this with a number in parentheses. For example, (1) refers to the first paragraph. Let's assume you want to work on the first paragraph in the collection. Type until your statement looks like this:

```
ActiveDocument.Paragraphs(1).
```

**22.** Guess what? You've just "drilled down" from the ActiveDocument object to the Paragraphs collection to the first Paragraph. That is, you've "reached" or "specified" an object that you want to work on. From here on out, life is easy. Why? Because working on a paragraph object is just like working on a document object. As soon as you type the dot after the object, the VBA editor shows you a list of all the methods, properties, and objects (or collections of objects) that belong to *your* object. Simply select the method that you want to carry out on your object, or select the property that you want to change, or keep drilling down by selecting an object or collection that belongs to your object. That's all there is to it.

A possible 23rd step would be to repeat Steps 1 -22 but replace all occurrences of "ActiveDocument." with "Selection." This allows you to drill down from the Selection object and discover the various methods, properties, and collections associated with the Selection object.

A possible 24th step would be to repeat Steps 1 -22 but replace all occurrences of "ActiveDocument." with "Application." This allows you to drill down from the Word application object and discover the various methods, properties, and collections associated with that object.

# The art of defensive programming

*Or how to write code that will be easy to maintain*

Article contributed by Jonathan West

Terms of use · Disclaimer

The source code that you write has two quite separate purposes.

1. It is a set of instructions to the computer, telling it to perform a particular task.
2. It is a description of the task and how you have gone about executing the task, for yourself and/or other programmers.

This article is mainly about the second purpose, which a surprising number of people don't really think about.

If you can't understand a program, then you can't debug it. Even with code that you have written yourself, if you come back to it six months or a year later, you may find yourself wondering “Why on earth did I write that? What was it for?” It doesn't take long to forget the details of a program when you aren't working on it any more. Make life easier for yourself, and write programs as clearly as possible. Also, provide such defences as you can against the possibility that VBA might change between versions of Word.

The following example is based on a real question that came up in the newsgroups in 1999. The questioner wanted a macro that would do the following for a particular set of styles named “AGR1” to “AGR5”:-

1. Check to see if the style already exists in the document.
2. If it **does**, then do nothing.
3. If it **doesn't**, then copy it from a central template.

This is a fairly common kind of problem that Word VBA macros are excellent at solving. The following code was suggested in response to the question.

```
Dim StyleArray As Variant
Dim StyleExists As Boolean
Sub Numbering()
With ActiveDocument
.UpdateStylesOnOpen = False
End With
StyleArray = Array("AGR1", "AGR2", "AGR3", "AGR4", "AGR5")
StyleExists = False
For x = 0 To 4
For Each oStyle In ActiveDocument.Styles
If oStyle = StyleArray(x) Then
StyleExists = True
End If
Next oStyle
If StyleExists = False Then
Application.OrganizerCopy _
Source:="C:\Program Files\Microsoft Office\" & _
"Templates\Final Numbering TDS.dot", _
Destination:=ActiveDocument, Name:=StyleArray(x), _
Object:=wdOrganizerObjectStyles
End If
StyleExists = False
Next x
End Sub
```

Now, this code does the job, but there are lots of ways in which it could be made easier to read and understand, and less prone to future problems. Remember, source code isn't just there for the computer to execute. It is a message to yourself (or to the programmer who someday may have to come after you and maintain your code) as to what you were trying to achieve. If you make the code easy to read, then that helps you. Anyway, here is a set of improvements that could be made to the code.

1. There is no need for the Dim statements to be module-level declarations. They aren't needed in any other routine. Therefore they should come after the Sub Numbering() statement, so that there is no confusion with any other routine which might happen to use the same variable names. People tend to re-use favored and meaningful names, so this problem can be significant.
2. “Numbering” isn't a very meaningful name for the routine. It's always a good idea give a routine a name that clearly describes its purpose. Perhaps ImportAGRStyles would be better.
3. You should have the “Require Variable Declaration” checkbox set in the VBA Options dialog, so that there is an Option Explicit statement at the start of every module. That way, there is much less chance that VBA will interpret a typographical error in one of your variable assignments as the creation of an entirely new variable of type Variant. This being so, you would need Dim statements for x and oStyle, as Long & Style respectively. See also **Why variables should be declared properly**.
4. With–End With statements are a good idea, but there isn't really any need to use them if you are only putting one item in between. If you have two or more lines between them, then yes, great idea, use them whenever you can.
5. When updating the code at some point in the future, you might want to add another one or two items to StyleArray. It would be quite likely that you would forget (at least initially) to change the limit value of x, so that it reflects the new size of the array. It is much easier to let VBA to remember for you, by using UBound(StyleArray). That will adjust automatically to the number of items you include using the Array function.
6. A speedup trick. In the For Each oStyle loop, once you find a case where you set StyleExists = True, you don't need to go round the loop any more times. Therefore you can add an Exit For statement to drop you out of the loop.
7. Using If StyleExists = False Then is not very efficient. Better would be If Not StyleExists Then. In fact, for readability, better still would be to rename the variable StyleIsMissing and swap the True/False values round, so the line can now read If StyleIsMissing Then
8. You can reset the value of StyleExists (or StyleIsMissing, as we would be renaming it) just once at the start of the loop, rather than once outside and once at the end of the loop. Fewer lines of code means fewer places that bugs can creep in!
9. I don't trust default properties of objects. It would be too easy for MS to change them between versions of Word and break something in my code. Therefore I would prefer to use oStyle.NameLocal rather than just oStyle when comparing its value with StyleArray (x). We haven't been through enough versions of Word with VBA to find out whether this may be a common problem with MS, though a few things did change between Word 97 and Word 2000. I would simply prefer to minimize the risk, especially as I think it improves readability. Similarly, in the OrganizerCopy command, it would be better to use ActiveDocument.FullName instead of just ActiveDocument.
10. Indenting the code so that it follows the structure of the If–Then statements and For–Next loops makes it much easier to understand where the loops and branches are. Also, it is a good idea to indent continuations of lines (perhaps by double the normal amount), so that it is quite clear that they are not new instructions. Also, it is common to indent everything except for comments and the lines which begin and end a routine.
11. When a few lines of code together are needed for a specific task, keep them together, and then put a blank line below to show that you are starting on a new task. That way, the eye more clearly sees what your design is.

That's quite a few comments, they are longer than the code I've been discussing! However, with every line of code that you publish (by which I mean that will ever be seen by anyone other than yourself), you should be able to justify it as being about the best way it could be written. I have learned the need for this kind of coding discipline the hard way, by having to fix extremely nasty and obscure bugs. Also, I have learned that for longer running macros (though this particular macro should be over quickly) every speedup trick in the book is worth applying. The code with these changes would now look like this.

```
Option Explicit

Sub ImportAGRStyles()

    Dim StyleArray As Variant
    Dim StyleIsMissing As Boolean
    Dim x As Long
    Dim oStyle As Style

    ActiveDocument.UpdateStylesOnOpen = False
    StyleArray = Array("AGR1", "AGR2", "AGR3", "AGR4", "AGR5")

    For x = 0 To UBound(StyleArray)

        StyleIsMissing = True
        For Each oStyle In ActiveDocument.Styles
            If oStyle.NameLocal = StyleArray(x) Then
                StyleIsMissing = False
                Exit For
            End If
        Next oStyle

        If StyleIsMissing Then
            Application.OrganizerCopy _
                    Source:="C:\Program Files\Microsoft Office\" & _
                    "Templates\Final Numbering TDS.dot", _
                    Destination:=ActiveDocument.FullName, _
                    Name:=StyleArray(x), _
                    Object:=wdOrganizerObjectStyles
        End If

    Next x

End Sub
```

Note that both versions of the routine do exactly the same thing. Maybe the second is marginally faster, but not to an extent that really matters. However, the second is much safer because the layout is clearer, the variables are local, and default properties are not used. It is no harder to write like this, once you get into the habit. For a short macro like this, perhaps it doesn't matter so much, but once you start writing macros, you will probably think of more complex tasks that you can automate with them. Then the “defensive” approach can make a big difference to the length of time it takes to get things right.

There's one other thing to notice. I didn't put any comments in. For a routine that's as simple as this, if the code is written well, there shouldn't be any need for comments. At most, you might want a line or two at the start to describe the purpose of the routine. (You might also have other “standard” comments describing the author and revision level of the routine, but that's another issue altogether.) For a longer routine or a particularly obscure bit of code, it's probably a good idea to include a few extra comments at strategic points. These comments should aim to explain what you are trying to do, rather than how you are doing it. Once you know the what, the how should be clear from the code itself, and so doesn't need duplication.

## Recommended further reading.

If you want a much fuller treatment of this subject, I would very strongly recommend that you buy a copy of *Code Complete,* by Steve McConnell, published 1993 by Microsoft Press, ISBN 1-55615-484-4. The book was written before Word VBA existed, and even before Visual Basic existed, and so its examples are mainly in C, Pascal and some in BASIC. Despite this, it contains the best set of guidelines I have come across for writing good code in any language. It covers most of the points I have made in this article, in much greater depth than I possibly could here. The book is beautifully clear (just like code should be!), and you will have no difficulty following the examples, whichever language they are written in.

Send all feedback to **word@mvps.org** Top - **Ctrl+Home** Previous - **Alt+left arrow** Page Updated:

# Why variables should be declared properly

Article contributed by Dave Rado

Terms of use Disclaimer

Almost **all** variables should be dimensioned as whatever they are (Dim MyRange As Range, Dim MyString As String, etc.), for several reasons:

**1.** Otherwise you can sometimes get unexpected results

**2.** Your code will run much faster and use less memory

**3.** You'll then have access to intellisense – for instance, if you type MyRange, and then type a dot, a list of all properties and methods supported by the range object will pop up – **provided** that MyRange was Dim'd as a Range. You can then cursor to the one you want and press Return (or select it with the mouse) to insert it.

**4.** It makes debugging far easier, because the VBA Editor will then be able to "spot" errors it would otherwise miss.

To force yourself to explicitly dimension all variables, your code window should have the line "Option Explicit" at the top of it. Select Tools + Options; and tick the "Require variable declaration" checkbox. This will have the effect of inserting "Option Explicit" at the top of all new modules you create. Doing this will give you all of the above benefits plus one more:

**5.** It makes debugging far easier in another way, because if you accidentally misspell a variable name somewhere, this will be picked up as an undeclared variable if you compile your project (Debug menu).

**Important note:** In order to explicitly declare variables, each one **must** be declared separately. For example, the statement:

```
Dim Str1, Str2 As String
```

is declaring all except the final one as a Variant, **not** as a string! You **must** declare each variable explicitly, as in:

```
Dim Str1 As String, Str2 As String
```

Or:

```
Dim Str1As String
Dim Str1 As String
```

# How to cut out repetition and write much less code, by using subroutines and functions that take arguments

Article contributed by Dave Rado

Most of us write routines that do similar operations more than once. It makes your code much less cumbersome and much easier to follow if you hive off all such repetitive chunks of code into separate *subroutines* or *functions*.

The difference between a sub and a function is that a function can return a value. Within the function itself, you can treat the function name like a variable, and give it a value and then you can call the function and get that value. Here's a ridiculously simple (and not very useful!) example:

```
Function GetResult As Long
    GetResult = 2
End Function
```

---

```
Sub Test()
    MsgBox GetResult
    'Returns 2
End Sub
```

---

But suppose you want to do a calculation 2 + 2. You could use:

```
MyResult = 2 + 2.
```

But alternatively, you could use a function to do the operation. The advantage is: less repetitive code (not so much in this example, because it's so simple, but in general).

So you could have:

```
Function SumTwoValues(FirstNum As Long, SecondNum As Long) As Long

    'Any sub or function can have variables passed to it,
    'and these variables, which need to be declared as shown here, enclosed in brackets
    'are called arguments

    SumTwoValues = FirstNum + SecondNum

End Function
```

---

And you can call the function like this:

```
Sub MainMacro()
    Dim MyResult As Long
    MyResult = SumTwoValues(2, 2)
End Sub
```

---

That's exactly the same as MyResult = 2 + 2, but if the function contained more than just one line of code, and was called often, using a function like that would greatly reduce the amount of code needed and also make your code easier to follow.

Let's take a more complex example. Supposing you had a macro that needs to do many Find and Replace operations, one after another. By using a subroutine that takes arguments to do the Find and Replaces it means you only need to have a single line of code for each and Replace. Here's a simple example:

Let's suppose that all your Find & Replace operations are identical except for the find text and replacement text. Then you could have a sub or function that gets called, which looks something like this:

```
Sub DoFindReplace(FindText As String, ReplaceText As String)
```

```
With Selection.Find
    .ClearFormatting
    .Replacement.ClearFormatting

    .Text = FindText
    .Replacement.Text = ReplaceText

    .Forward = True
    .Wrap = wdFindContinue
    .Format = False
    .MatchCase = False
    .MatchWholeWord = False
    .MatchWildcards = False
    .MatchSoundsLike = False
    .MatchAllWordForms = False
    Do While .Execute
        'Keep going until nothing found
        .Execute Replace:=wdReplaceAll
    Loop
    'Free up some memory
    ActiveDocument.UndoClear
End With

End Sub
```

---

You can then call it like this:

```
Sub MainMacro()
    'Remove double spaces
    Call DoFindReplace("  ", " ")
    'Remove all double tabs
    Call DoFindReplace("^t^t", "^t")
    'Remove empty paras (unless they folow a table or start or finish a doc)
    Call DoFindReplace("^p^p", "^p")
    'etc etc
End Sub
```

---

So only one extra line is needed for each Find and Replace operation.

You can make it a bit more flexible by making any other parameters (such as .MatchCase) that might change from one find & replace operaration to the next into arguments that you can pass values to, instead of hard coding them.

For instance, if .MatchCase is always set to False in your macro, you can just hard code it as shown above; but if it's True for some and False for others then you could use:

```
Sub DoFindReplace(FindText As String, ReplaceText As String, _
  bMatchCase As Boolean)

    'rest of sub as before except
    .MatchCase = bMatchCase

End Sub
```

and you could call that like this:

```
Sub MainMacro()
    Call DoFindReplace("Ibm", "IBMI", True)
    Call DoFindReplace("laserjet", "LaserJet", False)
End Sub
```

---

With Subs and Functions that take arguments – as with Word's Methods (which are actually built-in functions) – you can choose whether or not to specify the variable names when you call them. In the examples given so far I haven't used the variable names in the calling statements; and in the first few examples, there was no real point, because it's obvious what's going on in them.

But in the last example, it's not so obvious what the third variable is (without looking at the function) – so it's better to specify the variable names in this case, which you do like this:

```
Sub MainMacro()
    Call DoFindReplace(FindText:="Ibm", ReplaceText:="IBM", bMatchCase:=True)
    Call DoFindReplace(FindText:="laserjet", ReplaceText:="LaserJet", bMatchCase:=False)
End Sub
```

That is much easier to follow, for anyone maintaining your code, than

```
  Call DoFindReplace("Ibm", "IBMI", True)
  Call DoFindReplace("laserjet", "LaserJet", False)
```

Sometimes you might want to specify an argument in a sub or function, but you might want it to be optional. Most built-in Word functions include some optional arguments (such as the Background argument of the Printout method, for instance).

To make an argument optional, you simply prefix it with the keyword **Optional** when you declare it, for example:

```
Sub DoFindOrReplace(FindText As String, Optional ReplaceText As String)
```

In this example, the procedure could check the value of the optional ReplaceText variable; and do a Find & Replace if it had a value, but a Find if it didn't.

With optional arguments, you can also specify what value the variable should have if no value is specified. In other words you can specify its default value. If you do that, it's not optional in quite the same sense as before; it's actually mandatory, but you just don't have to specify its value in your code. If you don't specify its value, it will use the default value. So for example, you could have:

```
Sub DoFindReplace(FindText As String, ReplaceText As String, _
  Optional bMatchCase As Boolean = False)
```

Then you'd only have to specify the value of the bMatchCase argument if you wanted it to be set to True. This can save a lot of typing!

## Related articles

**For some code examples which call subs or functions with arguments:**

**How to use a single VBA procedure to read or write both custom and built-in Document Properties**

**How to get the username of the current user**

**Getting names of available printers**

**How to find out, using VBA, how many replacements Word made during a Find & Replace All**

**Finding and replacing symbols**

**Other related tips:**

**The art of defensive programming**

**Why variables should be declared properly**

# When to use parentheses to enclose subroutine and function arguments

Article contributed by Jonathan West

The rules are confusing concerning the use of parentheses to enclose argument lists. I have even seen MS Knowledgebase articles that have got it wrong. The rules are as follows.

**1.** For the initial line of a Sub or Function, you use parentheses to enclose the arguments (if any), e.g.

```
Sub MySubroutine(a As Long, b As String)
```

or

```
Function MyFunction(a As Long, B As String) As Long
```

**2.** When calling a function, you use parentheses to enclose the arguments, like this.

```
x = MyFunction(a:=1, b:="abc")
```

or

```
x = MyFunction(1, "abc")
```

**3.** When calling a sub directly, you don't use parentheses, like this.

```
MySub a:=1, b:="abc"
```

or

```
MySub 1, "abc"
```

**4.** When calling a sub using the **Call** keyword, you **do** use parentheses (this made sense to somebody!)

```
Call MySub(a:=1, b:="abc")
```

or

```
Call MySub(1, "abc")
```

Note that it doesn't matter whether or not you use the Call keyword to call a subroutine. The effect on the program is identical whether or not you use Call (assuming you have the parentheses right). Most of the code on this site doesn't use Call, simply because it is fewer words to type.

**5.** When dealing with methods of an object, use the same rules, depending on whether you are obtaining a return value from the method (like a function), or just applying the method to the object without a return value (like a subroutine). For example, on the one hand:

```
Selection.GoTo What:=wdGoToPage, Name:="5"
```

But on the other hand (because you're now using a function which returns a value):

```
Dim MyRange As Range
Set MyRange = ActiveDocument.Range
Set MyRange = MyRange.GoTo(What:=wdGoToPage, Name:="5")
```

Terms of use Disclaimer

# Early vs. Late Binding

*Article contributed by Dave Rado*

There are two ways to use Automation (or OLE Automation) to programmatically control another application.

**Late binding** uses CreateObject to create and instance of the application object, which you can then control. For example, to create a new instance of Excel using late binding:

```
Dim oXL As Object
Set oXL = CreateObject("Excel.Application")
```

On the other hand, to manipulate an existing instance of Excel (if Excel is already open) you would use GetObject (regardless whether you're using early or late binding):

```
Dim oXL As Object
Set oXL = GetObject(, "Excel.Application")
```

To use **early binding**, you first need to set a reference in your project to the application you want to manipulate. In the VB Editor of any Office application, or in VB itself, you do this by selecting Tools + References, and selecting the application you want from the list (e.g. “Microsoft Excel 8.0 Object Library”).

To create a new instance of Excel using early binding:

```
Dim oXL As Excel.Application
Set oXL = New Excel.Application
```

In either case, incidentally, you can first try to get an existing instance of Excel, and if that returns an error, you can create a new instance in your error handler.

## *Advantages of Early Binding*

1. Your code will run considerably faster, because it can all be compiled up front. With late binding, the code relating to an application you declared as an object has to, in effect, be compiled as it runs.

2. Because your code can all be compiled up front, debugging is far easier – select Debug + Compile, and the compiler will be able to spot syntax errors which would have been missed had you used late binding.

3. You have full access in your project to intellisense (type a keyword and a dot to get a popup list of properties and methods supported by that keyword, select one to insert it; type a keyword and press F1 to launch the Help topic on that keyword).

4. You have full access to the application's object model via the Object Browser and VBA Help.

5. You have access to the application's built-in constants. For instance, if you are automating Word from Excel, you can use:

   ```
   Dim objWord As Word.Application
   Set objWord = New Word.Application

   With objWord
       .Visible = True
       .Activate
       .WindowState = wdWindowStateMaximize
       .Documents.Open ("c:\temp\temp.doc")
   End With
   ```

   Furthermore, when you type

   ```
   .WindowState =
   ```

   you'll get a pop-up list of the supported constants, and can simply pick “wdWindowStateMaximize” from the list.

If you used late binding, you would need to use:

```
.WindowState = 1
```

.. and you would need to know (by looking it up in Word's Object Browser) that the value of the constant “wdWindowStateMaximize” happens to be 1.

All this makes programming using early binding immeasurably easier than using late binding.

## *Advantages of Late Binding*

1. The main advantage is that code which uses late binding is more certain to be version-independent

   If you set a reference in a Word 97 project to “Microsoft Excel 8.0 Object Library”, then the project **will** run OK on a machine which has Office 2000 installed. Word 2000 changes the reference on the fly to the “Microsoft Excel 9.0 Object Library”.

   But as they famously say, YMMV. Problems have been found in certain circumstances. For instance, if you run a Word 97 project containing a reference to the Excel 8.0 object library on a machine with Office 2000 installed, it will run OK, but you may get the occasional “cannot open macro storage” error unless you save the project in Word 2000. If you do save it in Word 2000, the reference will change to the Excel 9.0 object library. So if you use early binding and support a mixed environment, it may be safest to create separate Word 97 and Word 2000 versions of your addins, despite the maintenance overhead.

2. The more references your project contains, the larger the file

size and the longer it takes to compile.

3. Some programming environments don't allow you to create references to another application.

## Summary

Personally, as someone who finds programming difficult at the best of times, I would never dream of using late binding – why make life harder for yourself than it has to be? But some programming geniuses prefer to use late binding, because of the peace of mind it gives them regarding version independence – or maybe some of them just enjoy the challenge! <g> But you pays your money and makes your choice ...

To those unfortunate souls using programming environments in which you **have** to use late binding, all I can say is: Look on the bright side – you could have ended up as an Assembly language programmer ...

# Getting help with calling Word's built-in dialogs using VBA (and why doing so can be much more useful than you'd think)

Article contributed by Jonathan West and Dave Rado

## Where to find Help on them

There are two Help topics in Word VBA Help that are required reading to get you started with built-in dialogs: “Displaying built-in Word dialog boxes” and “Built-in dialog box argument lists”. Unfortunately, in the latter article, Microsoft listed the arguments you can use but forgot to mention what the arguments mean or what values they can take!

Fortunately, the dialog box arguments are almost identical to the arguments of the commands of WordBasic, so if you know one, you can work out the other. Therefore, the WordBasic Help file is at present by far the best resource for programmers wanting to use the dialogs. It is an absolute must-have.

Even more fortunately, the WordBasic Help is now available on-line – **click here to download it**.

## Why use built-in dialogs?

There are four reasons to use them.

**1.** One obvious use is in order to display them to the user.

**2.** You can execute a built-in dialog without actually displaying it, thus allowing you to execute all of the relevant settings simultaneously; whereas with VBA, each statement executes one at a time. As a result, using the wdDialog method can sometimes make your code run much faster and use far less memory.

A good example is wdDialogFormatBordersAndShading. Using this, you can execute all your borders and shading arguments with one Execute statement; whereas using “native VBA”, you have to set many separate properties one at a time (top border, bottom border etc.). As a result, not only will your code run much faster if you execute the built-in dialog, but you are far less likely to get “Formatting too complex” errors while your code is running. Consider the following, for instance:

```
With Dialogs(wdDialogFormatBordersAndShading)
    .ApplyTo = 3
    .Shadow = 0
    .Shading = 0
    .Foreground = 0
    .Background = 0
    .LeftStyle = 0
    .RightStyle = 0
    .TopStyle = 0
    .BottomStyle = 0
    .HorizStyle = 0
    .VertStyle = 0
    .Execute
End With
```

All the above properties are executed in one go by the .Execute statement, whereas if you use “native VBA” as shown below, each line executes before the next line starts to run, and so the macro runs far more slowly and uses far more memory:

```
With Selection.Tables(1)
    With .Shading
        .Texture = wdTextureNone
        .ForegroundPatternColor = wdColorAutomatic
        .BackgroundPatternColor = wdColorAutomatic
    End With
    .Borders(wdBorderLeft).LineStyle = wdLineStyleNone
    .Borders(wdBorderRight).LineStyle = wdLineStyleNone
    .Borders(wdBorderTop).LineStyle = wdLineStyleNone
    .Borders(wdBorderBottom).LineStyle = wdLineStyleNone
    .Borders(wdBorderHorizontal).LineStyle = wdLineStyleNone
    .Borders(wdBorderVertical).LineStyle = wdLineStyleNone
    .Borders(wdBorderDiagonalDown).LineStyle = wdLineStyleNone
```

```
        .Borders(wdBorderDiagonalUp).LineStyle = wdLineStyleNone
        .Borders.Shadow = False
    End With
```

**3.** A third reason for using the wdDialogs is that in a few cases, it does much of the work for you that you would otherwise have to write a lot of code to do – for instance, if you want to change the page setup for the current selection, if you use wdDialogPageSetup, and set .ApplyPropsTo = 3, the section breaks are inserted for you. Doing the equivalent in “native VBA” is much more involved because you have to check whether there is already a section break before or after the selection, and whether the selection begins at the start of the doc and or ends at the end of the doc. Why write the code to do that in VBA when it's already been written in C++ by Microsoft? For instance, the following code snippet creates a landscape section at the selection (Chr$(34) is how you specify the inch (") symbol in WordBasic):

```
With Dialogs(wdDialogFilePageSetup)
    .ApplyPropsTo = 3
    .PageWidth = 11.69 & Chr$(34)
    .PageHeight = 8.27 & Chr$(34)
    .TopMargin = -1.7 & Chr$(34)
    .BottomMargin = 0.81 & Chr$(34)
    .LeftMargin = iLeftMargin & Chr$(34)
    .RightMargin = 0.95 & Chr$(34)
    .Orientation = wdOrientLandscape
    .DifferentFirstPage = False
    .HeaderDistance = 0.28 & Chr$(34)
    .FirstPage = 0
    .OtherPages = 0
    .Execute
End With
```

**4.** You can get and set information that would be difficult or, in some cases, impossible to get using “native VBA”, by looking (in code) at the settings of a built-in dialog – without having to display the dialog.

For some very useful examples, see:

**Finding and replacing symbols**

**How to change the behaviour of Word's document protection, so users’ formfields don't get reset when they unprotect and reprotect**

**How to get the column number of the selection (in a document containing snaking, or newspaper-style, columns)**

**How to retrieve Word's default Documents path or Pictures path setting**

**How to assign a Name to a FormField that doesn't already have a Name, using VBA**

**Changing the selected (current) printer in Word without changing the system default printer**

**How to set the default suggested filename to be displayed by the Save As dialog the first time a user saves a new document**

**How to safely update a document's styles from its template without using the Organizer (and how to make the Tools + Templates and Add-ins dialog safe)**

:

### “Gotchas”

Note that a fair number of the wdDialog arguments simply don't work. For example, the Context argument (the “Look in” setting) for the Autotext dialog has no effect, and there are many other arguments which don't work.

There are also well-known bugs with several of the wdDialogs. For example, the wdDialogFileOpen dialog won't allow the user to open more than one file at a time, although there is a workaround – see: **Calling FileOpen dialog in VBA does not allow opening of multiple files**.

But at least with the aid of the WordBasic Help file you **can** find out what the dialogs are *meant* to do!

# How to create a Userform

Article contributed by Doug Robbins

Terms of use Disclaimer

This example will step you through the process of creating a template that contains an autonew macro which, when you create a new document from the template, will cause a Userform to be displayed, into which you can enter some information that you want to appear in the document.

For example, when creating a letter or a fax; when you click on the OK button on the Userform, the information that has been entered into it will be inserted into bookmarks located in the document at the places that you want the information to appear.

The example shows how you would deal with two pieces of information that you want to put into the document. At pertinent places in the following, comments have been added indicating where you would modify the code to deal with more pieces of information.

| Step | Comments |
|---|---|
| **1.** Create a template. | See: Creating a Template – The Basics (Part I) and Creating a Template (Part II) |
| **2.** Create two bookmarks in your template named “Text1” and “Text2”. To create each bookmark, select a single space so that when you turn on the option to see the bookmarks under **Tools>Options>View**, the bookmarks appear as **[]** rather than **\|**. When you create bookmarks with a space selected like this, the text will be inserted inside the bookmark, rather than after it.<br><br>You can then use a cross reference to the text of the bookmark if you want the same information to appear in another location. | Add additional bookmarks for each piece of data) Use more descriptive names for the bookmarks if you like as it does make it easier to remember which is which and what goes where |
| **3.** Go to the Visual Basic Editor (Tools>Macro>Visual Basic Editor) | Alternatively, you can hold down the Alt key and press F11 |
| **4.** With the Template selected in the Project Explorer, select UserForm from the Insert menu | A UserForm will appear in the right hand pane and the Controls toolbox will appear in the left hand pane.<br><br>If there are no other Userforms in the Template, it will by default be UserForm1. |
| **5.** From the Controls toolbox, select the TextBox control – ab\| and then move the mouse cursor over to the UserForm, click and hold the Left mouse button and drag the cursor to make the TextBox the desired size. Repeat this step for a second TextBox or for as many as you require. | By default, the TextBoxes will be numbered TextBox1, TextBox2 ...TextBoxn.<br><br>You can change the names of the TextBoxes in the Properties window in the left hand pane. If you name them the same as the Bookmarks that you inserted into the template in Step 1, it makes it easier |

| | | |
|---|---|---|
| **6.** | Dimension the TextBoxes by selecting and dragging as required for the text that you intend to be inserted into them | |
| **7.** | Add any labels that you want to have on the form to assist the user by clicking on the Label control – A in the Controls Toolbox and inserting them in the same way as for the TextBoxes. | |
| **8.** | In the Controls Toolbox, click on the CommandButton control and insert one in the UserForm. | By default, this will be named CommandButton1 |
| **9.** | Right Click on the CommandButton in the UserForm and select the View Codeitem from the menu. | |
| **10.** | Enter the following code between the Private Sub CommandButton1_Click() and theEnd Sub<br>`With ActiveDocument`<br>`    .Bookmarks("Text1").Range _`<br>`    .InsertBefore TextBox1`<br>`    .Bookmarks("Text2").Range _`<br>`    .InsertBefore TextBox2`<br>`End With`<br><br>`UserForm1.Hide` | Repeat the lines beginning with .Bookmarks<br>for eachbookmark.<br>Similarly, you can insert the same information into a second bookmark by inserting a line referring to the range of that bookmark |
| **11.** | Close the Visual Basic Editor | |
| **12.** | From the Tools Menu select Macro, then Macros, then with the Template selected in the “Macros in:” box, type the name *autonew* in the “Macro name:” box and click the Create button. | |
| **13.** | Enter the following command in the module screen:<br>`UserForm1.Show` | |
| **14.** | Exit from the Visual Basic Editor and save your template. | |

When you create a new document from the template, the autonew macro opens the user form for the user to input data, then when the command button is clicked, the code associated with that button inserts each piece of information into the respective bookmark.

This has been restricted to just enough to get you started. There's lots more that you can do with UserForms. If you get stuck, post a question to the **microsoft.public.word.vba.userforms** newsgroup and someone will spring to your rescue.

# Useful WordBasic commands that have no VBA equivalent

Article contributed by Jonathan West

When Microsoft released Word 97, a new programming language VBA replaced the WordBasic language that had been available in earlier versions of Word. For most things, VBA is a much more powerful and flexible programming language than WordBasic, but there are a few very useful WordBasic commands which have no direct equivalents in VBA.

Fortunately, VBA includes the WordBasic object, which gives access to most of the old WordBasic commands.

## SortArray

This is perhaps the most useful of the commands left behind. It allows you to sort the elements of an array using a single line of code. At its simplest, you can use it on a one-dimensional array as follows.

```
Sub SortTest()
    Dim ss(2) As String
    Dim i As Long

    ss(0) = "orange"
    ss(1) = "apple"
    ss(2) = "banana"
    WordBasic.SortArray ss()

    For i = 0 To 2
        Debug.Print ss(i)
    Next i

End Sub
```

This sorts the array in ascending alphabetical order

However, you can also sort in descending order, and sort either dimension of a two-dimension array. The full list of the SortArray arguments is as follows

```
SortArray ArrayName[$]() [, Order] [, From] [, To] [, SortType] [, SortKey]
```

- ArrayName is the name of the array
- Order is 0 for ascending (by default), 1 for descending
- From is the first element to sort (0 by default)
- To is the last element to sort (by default the last element of the array)
- SortType determines whether you are sorting rows or columns. 0 (default) for rows, 1 for columns
- SortKey is applicable only to two-dimensional arrays, and indicates the row or column used as the sort key. It is 0 by default

Note that, unlike most VBA methods, you **don't** use named arguments with this command; thus you **can** have

```
WordBasic.SortArray MailingList$(), 1, 1, 20, 0, 1
```

but **not**

```
WordBasic.SortArray ArrayName:=MailingList$(), Order:=1, From:=1, To:=20, _
        SortType:=0, SortKey:=1
```

Also, you cannot miss out arguments if you want to use later ones, thus you **can** have

```
WordBasic.SortArray Test(), 0, 0, 2, 0, 1
```

but **not**

```
WordBasic.SortArray Test(), 0, , , , 1
```

There is one other limitation of the SortArray command. It will sort an array declared as such, but it will not sort an array that is contained in a Variant. If you create an array like this:

```
Dim vArray as Variant
vArray = Array("orange", "apple", "banana")
```

SortArray will not sort it.

(Also if you do not declare your array at all, it will be treated as a variant and will not be sorted).

## FileNameInfo$()

This is another very useful function for which there is no direct VBA equivalent. FileNameInfo allows you to get just the filename or a fully qualified pathname from a filename given to it. The nearest equivalent in VBA are the Name, FullName and Path properties of the Document object.

FileNameInfo is different in that you don't need to have the document open.

The syntax is

```
x = WordBasic.FilenameInfo$(Filename$, FileType)
```

where Filename is the name of the file, and FileType is a number which defines the part of the filename you want to return:

1 - the full pathname, e.g. C:\My Documents\My File.doc"
2 - the filename only, if the file is in the current folder, otherwise the full pathname
3 - the filename only
4 - the filename without the extension
5 - the path without the filename
6 - the UNC pathname

One case where FileNameInfo$ is very useful is to get the pathname of a file which has just been selected by the user in the FileOpen dialog. The following code returns the full pathname of a file selected by the user.

```
With Dialogs(wdDialogFileOpen)
    If .Display Then
        MsgBox WordBasic.FilenameInfo$(.Name, 1)
    Else
        MsgBox "No file selected"
    End If
End With
```

## DisableAutoMacros

If you are running a macro that opens (or creates) several files, the last thing you may want is for an AutoOpen (or AutoNew) macro to fire up each time. WordBasic has a means of preventing this, which VBA never copied.

```
WordBasic.DisableAutoMacros 1   'Disables auto macros
WordBasic.DisableAutoMacros 0   'Enables auto macros
```

This command is also very useful when launching an instance of Word from another application, or from VB, when you will generally not want any AutoExec macros to fire.

## ToolsBulletsNumbers

WordBasic allows you to remove all manually typed numbering from a selection using the old Word 2 command:

```
WordBasic.ToolsBulletsNumbers Replace:=0, Type:=1, Remove:=1
```

This is particularly useful for removing manually typed numbering from Headings in a document you have been emailed, prior to applying List Numbering. If you go into Outline View, set the Heading Level to the number of levels you need to remove the typed numbering from, and run the above line, it will just remove numbering from those Headings and will leave the body text alone. Or you can use the following macro to do the same thing:

```
Sub RemoveNumbersFromHeadings()
```

```
Dim ViewType As Long, ShowHeadingLevel As Long, MyRange As Range

Application.ScreenUpdating = False
'Set Range variable to current selection so it can be returned to at the end
Set MyRange = Selection.Range
'Set variable to current View so it can be returned to at the end
ViewType = ActiveWindow.View.Type
'Switch to Outline View
ActiveWindow.View.Type = wdOutlineView

'Checks the state (using the ID in case the toolbar has been customised) of _
all the ShowHeadings buttons on the Outline toolbar; if none are depressed, _
then it must be set to ShowAll. Stores result in a variable so that _
the same level can be returned to at the end of the macro

For ShowHeadingLevel = 71 To 77
    If CommandBars.FindControl(ID:=ShowHeadingLevel).State Then
        ShowHeadingLevel = (ShowHeadingLevel - 70)
        Exit For
    End If
Next ShowHeadingLevel
'if none of the heading level buttons depressed sets variable to 1
If ShowHeadingLevel = 78 Then ShowHeadingLevel = 1

ActiveWindow.View.ShowHeading 3
ActiveDocument.Select

WordBasic.ToolsBulletsNumbers Replace:=0, Type:=1, Remove:=1

If ShowHeadingLevel = 0 Then
    ActiveWindow.View.ShowAllHeadings
Else
    ActiveWindow.View.ShowHeading ShowHeadingLevel
End If

ActiveWindow.View.Type = ViewType
MyRange.Select

Set MyRange = Nothing

Application.ScreenUpdating = False

End Sub
```

## FileCopy/FileCopyA

The VBA FileCopy statement will not copy files that are open. However, the WordBasic equivalent will (this is what is known as progress!).

Unfortunately, the syntax of WordBasic equivalent is different in Word 97 and Word 2000!

The following works even if the file being copied is open:

```
If Left$(Application.Version, 1) = "8" Then
    'Word 97
    WordBasic.CopyFile FileName:="c:\OldDirectory\Temp.doc", _
    Directory:="C:\NewDirectory\Temp.doc"
Else
    'Word 2000 and above
    WordBasic.CopyFileA FileName:="c:\OldDirectory\Temp.doc", _
            Directory:="C:\NewDirectory\Temp.doc"
 End If
```

## FileProperties

If you want to intercept the FileProperties menu command, the only reliable way to do it is to use:

```
Sub FileProperties()
    'your code here
    WordBasic.FileProperties
End Sub
```

In fact, if you let Word create the code for you, using the method described **here**, the above code will be created.

## SendKeys

If you want to do a screen capture using VBA, (simulating the PrtScr key), you have to use:

```
WordBasic.SendKeys "%{1068}"
```

## Getting Help with WordBasic Commands

Microsoft has made the old WordBasic help file available online. for further information about WordBasic commands, you can download it from **here**..

# Working with Bookmarks in VBA

Article contributed by Ibby

## Types of Bookmarks

The most important thing you need to know when working with bookmarks in Word is that there are two “types” of bookmarks – “placeholder” bookmarks and “enclosing” bookmarks.

Before we proceed, and whenever you work with bookmarks, you should turn on display of bookmarks by going to **Tools | Options | View** and selecting “Bookmarks”. This makes it easier to see what's actually happening.

### (1) Placeholder Bookmarks

If you click somewhere in the document and insert a bookmark it will look like a beam **I** – this is a “placeholder” bookmark.

### (2) Enclosing Bookmarks

Now, if you select some text and insert a bookmark it will look like the selected text is enclosed in square brackets ie: **[**selected text**]** – this is an “enclosing” bookmark.

## Inserting and retrieving text from a Bookmark

There are several methods of inserting text at/into a bookmark. The method you use depends on whether you need to retrieve the text from the bookmark at a later time.

Lets look at the more obvious ways of inserting text at a bookmark.

```
ActiveDocument.Bookmarks("myBookmark").Range.Text = "Inserted Text"
```

If the bookmark is a placeholder bookmark, the inserted text will look like this:

**I** Inserted Text

If the bookmark is an enclosing bookmark, it will be **deleted**, and the inserted text will appear in it's place.

```
ActiveDocument.Bookmarks("myBookmark").Range.InsertBefore _
  "Inserted Text"

ActiveDocument.Bookmarks("myBookmark").Range.InsertAfter _
  "Inserted Text"
```

With both these methods, if the bookmark is a placeholder bookmark, the text will be inserted after the bookmark:

**I** Inserted Text

With enclosing bookmarks (even if the bookmark only encloses a space), the following occurs:

**InsertAfter** – **[** Original Text **]** Inserted Text
**InsertBefore** – **[** Inserted Text Original Text **]**

In order to retrieve the text in a bookmark, the bookmark needs to be
an enclosing bookmark. Then you can use the following to retrieve the text from the bookmark:

```
strBookmark = ActiveDocument.Bookmarks("myBookmark").Range.Text
```

You have already seen how to add text to an enclosing bookmark using the InsertBefore method above. But what if you want to insert text into a placeholder bookmark (making it an enclosing bookmark) so that you can retrieve the text from it at a later time ? And what if the bookmark is already an enclosing bookmark but you want to replace the text inside it ? There is no single command in VBA to achieve this. What you need to do is replace the bookmark with the inserted text (the bookmark is deleted), then re-create the bookmark around the inserted text. The following code is an example of how this is done:

```
Dim bmRange As Range

Set bmRange = ActiveDocument.Bookmarks("myBookmark").Range

bmRange.Text = "Inserted Text"

ActiveDocument.Bookmarks.Add _
   Name:="myBookmark", _
   Range:=bmRange
```

See also **Inserting text at a bookmark without deleting the bookmark**

# Determine the index number of the current paragraph, table, section ...

*Or of any object that has a Range property*

Article contributed by Dave Rado,
with acknowledgments to Andrew Gabb, Jonathan West and Ibby

Terms of use Disclaimer

Note that none of the following applies reliably if you are in Outline View; if you are, you need to change to some other view, such as Normal view, first.

## 1. In order to operate on the currently selected paragraph, table, section, etc.

The first questions is – why do you need to know the index number? If you want to know in order to operate on the currently selected paragraph, table or other object, you can simply use:

```
Selection.CollectionName(1)
```

For example:

```
Selection.Paragraphs(1).Range.Font.Bold = True

Selection.Sections(1).Headers(wdHeaderFooterPrimary).Range.Text = "Hello"

Selection.Tables(1).Borders.Enable = False
```

All the above operate on the currently selected object.

Similarly, if you were working with a range variable, you could use:

```
MyRange.Paragraphs(1).Range.Font.Bold = True
```

Etc.

However if you really **do** need to know the index number, use the following method:

## 2. Get the index number, by setting a range to the start of the document

The fastest and simplest way to get the index number, by far, is to set a range from the start of the document to the end of the first selected paragraph (or other object); and then use the Count property, as follows:

```
MsgBox ActiveDocument.Range(0, Selection.Paragraphs(1).Range.End).Paragraphs.Count

MsgBox ActiveDocument.Range(0, Selection.Sections(1).Range.End).Sections.Count

MsgBox ActiveDocument.Range(0, Selection.Tables(1).Range.End).Tables.Count
```

Again, you could also use this method to get the index number of a range rather than of a selection, as follows:

```
MsgBox ActiveDocument.Range(0, MyRange.Paragraphs(1).Range.End).Paragraphs.Count
```

Etc.

This method was suggested in the newsgroups by Andrew Gabb.

# How to customise the Control Toolbox in the VB Editor

*So that you don't have to set up all the properties for your controls every time you add one to a UserForm*

Article contributed by Dave Rado

Supposing you frequently need to add Multiline Text Boxes to your UserForms, with the EnterKeyBehavior property set to True. Or let's say you want to drag a “Next” button straight onto your userform and not have to change the text in it to say “Next”, and not have to change the dimensions of the button.

A timesaver you might find useful; having created your multi-line textbox or Next button, or whatever, and having set all its properties the way you want them, drag your control onto the Control Toolbox.

Then right click on the new control, and select “Customize”.

Change the Tooltip text to “Multiline Textbox” or “Next button”, or whatever.

You can then edit the button images on the Control Toolbox if you want, to make it easier to tell them apart (also under Customize, then click on Edit Picture).

In future you can save yourself time by dragging the new control straight onto your UserForms and the appropriate properties will be set straight away.

I use this for all my frequently used buttons such as an OK, Cancel, Next and Previous, as well as for various different types of text labels, text boxes, frames, and so on. As well as saving time, it also helps greatly in making all your userforms look consistent.

You can add additional tabs (or “pages”) to the toolbox – right click, New Page. For instance I have one for my buttons and one for my other controls.

Best of all, you can export your customised toolbox, one “page” at a time – right-click, Export Page; and other members of your workgroup can import the resulting .pag file(s) - right-click, Import Page. This helps greatly in achieving consistency throughout a group of developers. It also means to can take your customisations with you – if you buy a new PC, for example.

# Intercepting events like Save and Print

Article contributed by Dave Rado, Anna-Karin Bohman and Jonathan West

Terms of use · Disclaimer

## Intercepting commands

To intercept **any** Word command, you can:

1. Press **Alt+ F8** to bring up the Macros dialog and where it says “Macros in”, select “Word Commands”.
2. Find and select one of the commands you want to intercept – for instance, to intercept the Print commands you need to find FilePrint and FilePrintDefault. To intercept the Save commands you need to find FileSave, FileSaveAs and FileSaveAll
3. Where it says “Macros in”, select the template you want to store the macro in, and click “Create”.
4. The code needed to execute the command will be written for you; just add your own code.

In the case of the Save event, writing FileSave, FileSaveAs and FileSaveAll macros isn't enough, because they won't intercept the user closing an unsaved document and being asked if they want to save changes – but you can intercept that by writing a macro called **AutoClose**; or by writing a **Document_Close event** procedure in the “ThisDocument” code module.

**See also:**

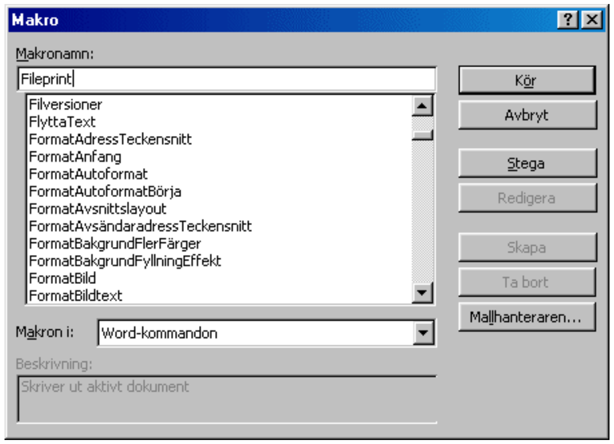
**Word commands, and their descriptions, default shortcuts and menu assignments**

**How to make the Paste Special dialog default to pasting Inline rather than Floating**

### If you are using a version of Word other than English

If you are **not** using an English version of Word, and if you create a macro using the name shown in the list of Word commands, only the description of what the macro does will be added to the new macro, not the necessary code. To get the necessary code, you have to create a macro using the English name for the command! But how do you find out the English name? You can get a full list of the English commands from **here**.

When you get the command's name right, the listbox at the very bottom will display the description of what the command does:


Makro

Makronamn: Fileprint

Filversioner
FlyttaText
FormatAdressTeckensnitt
FormatAnfang
FormatAutoformat
FormatAutoformatBörja
FormatAvsnittslayout
FormatAvsändaradressTeckensnitt
FormatBakgrundFlerFärger
FormatBakgrundFyllningEffekt
FormatBild
FormatBildtext

Makron i: Word-kommandon

Beskrivning: Skriver ut aktivt dokument

## Intercepting events (Word 2000 or later)

Intercepting a command isn't quite the same as intercepting events, but in most cases it's the best you can do. However, in Word 2000 or later, a number of new Application Events were made available in VBA.

Two Application Events you can use include **DocumentBeforeSave** and **DocumentBeforePrint**. Both of these, but especially the former, work better than trying to intercept the relevant commands.

If not familiar with writing application event procedures, see the article: **Writing application event procedures**.

A DocumentBeforePrint event procedure looks like this:

```
Private Sub oApp_DocumentBeforePrint(ByVal Doc As Document, _
 Cancel As Boolean)
    'Your code here
End Sub
```

If you want to prevent printing from occurring in certain circumstances, you can set the *Cancel* variable to True, e.g.:

```
Private Sub oApp_DocumentBeforePrint(ByVal Doc As Document, _
 Cancel As Boolean)
    Dim Result As Long
    Result = MsgBox("Have you checked the " & "printer for letterhead paper?", vbYesNo)
    If Result = vbNo Then Cancel = True
End Sub
```

A DocumentBeforeSave procedure looks like this:

```
Private Sub oApp_DocumentBeforeSave(ByVal Doc As Document, _
  SaveAsUI As Boolean, Cancel As Boolean)
    'Your code here
End Sub
```

Again, you can set Cancel = True if you want to cancel the save.

If you set the *SaveAsUI* variable to True, the **Save As** dialog box will be displayed.

Terms of use • Disclaimer

# Writing application event procedures

*Or how to intercept events affecting any open document, such as the user changing focus from one document to another*

Article contributed by Dave Rado, with acknowledgements to Bill Coan

You can respond to certain application-level events using Auto macros – see: **Running a macro automatically when Word starts or quits**. However, to respond to other application events you need to create an application object declared “WithEvents” – i.e. declared in such a way that it will respond to events.

## How to set up code that will respond to application events

If you open a template, press **Alt+F11**, and in the Project window of the VBA environment, double-click on “Microsoft Word Objects”, you'll see a built-in Class Module called “ThisDocument”. If you open this, you can use the list boxes on the toolbar to help you create document event procedures.

So it would make sense for there to be another built-in class module called “ThisApplication”; but Microsoft, in their wisdom, felt this would be too user-friendly, so you have to create your own ... Help is very unhelpful on the topic – there isn't even a definition in Help of what a Class *is*! – but if you're interested, Bill Coan covered the topic at **http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnword2k2/html/odc_wdappevnt.asp**.

In essence, though, a Class is simply a **container** object which allows you to create properties, which are objects in their own right, and then create properties and methods for those objects.

Note that the “big picture” view of what we're about to do is this:

- We want to respond to events.
- We need an application variable (declared with WithEvents) to receive the events.
- We need a class module to serve as a container for the application variable.

So here's how to create an application event procedure:

1. Create an Addin; that is, a .dot file stored in Word's startup directory so that it is **global**.
2. Open the Addin, and in the VB Editor, select **Insert + Class Module**.

   Rename the Class Module from Class1 to **ThisApplication**.

   **Note:** The Class Module and all the variables in this article can be called whatever you like; but I have tried to make all the names as meaningful as possible.

   I like “ThisApplication” because it shows that it is functionally analogous to the built-in “ThisDocument” class module. But note that your “ThisApplication” object is simply a **container**. It doesn't actually represent an application. It represents all the properties of the Class Object, whatever we define those properties to be.

3. Insert the following code in the Class Module:

```
Option Explicit

Public WithEvents oApp As Word.Application
```

This specifies that **oApp** is an object variable which will be used to respond to the events triggered by the ActiveX object Word.Application.

**By declaring it publicly, you have made oApp a *property* of the class object ThisApplication.**

At this point you are simply declaring what **type** of object oApp is (its type being a Word Application object), and declaring the fact it will respond to events. You are **not** actually *assigning* the oApp variable to the Word application yet – that comes later. So the **oApp** object doesn't actually exist yet (and no area of memory has been set aside for it yet).

The WithEvents keyword can only be used in Class Modules and can only refer to ActiveX –  i.e. OLE-compliant – objects.

4. Now select **Insert + Module**.

In the Module, insert the following code:

```
Option Explicit

Dim oAppClass As New ThisApplication
```

Where “ThisApplication” is the name of your Class Module. The statement creates an object variable **oAppClass** which references (is a pointer to) the class object ThisApplication which you created earlier.

Using the keyword “New” in the declaration creates a new *instance* of the class object – that is, it loads an instance of the class into memory and makes the variable oAppClass point to the newly created instance.

From now on, you can refer to this new instance of the class by using the **oAppClass** variable.

Now add the following code in the same Module:

```
Public Sub AutoExec()
   Set oAppClass.oApp = Word.Application
End Sub
```

The oApp object is a *property* of the class object oAppClass, because you declared it as a public variable in the Class Module in step **3.**

The above code creates an instance of the **oApp** object (loads it into memory), and makes it actually refer to the Word.Application object. In effect it makes the oApp object exist.

*To put it another way, you have now assigned the actual Active-X Word Application to the variable, whereas previously you had only declared what type of variable it was going to be.*

By calling your procedure **AutoExec**, you make it run automatically when your Addin loads – and your Addin will automatically load when Word loads.

5. Now run the AutoExec macro, by clicking in it and pressing **F5**, to initialise the oApp object.

6. Switch to your ThisApplication Class Module. You'll see two list boxes on the toolbar. If you pull down the one on the left, and change it from “(General)” to “oApp”, a procedure called **oApp_Quit()** will be created. If you then pull down the list box on the right, you'll see several other application events to choose from (and you can delete the oApp_Quit() procedure later, if you don't want it):

**a)** In Word 97, you only have the choice of **Quit** and **DocumentChange**.

**b)** In Word 2000, you have the choice of:

- Quit
- DocumentChange
- DocumentBeforeClose
- DocumentBeforePrint
- DocumentBeforeSave
- DocumentOpen
- NewDocument

- WindowActivate
- WindowBeforeDoubleClick
- WindowBeforeRightClick
- WindowDeactivate
- WindowSelectionChange

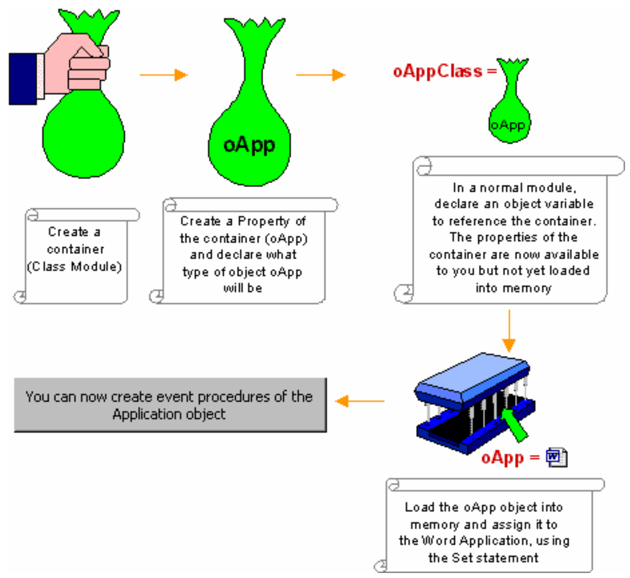
Note that (rather confusingly) these are all **Application** events (even though some of them refer to documents doing things) as distinct from Document events such as **Document_Close**. Because it is an **Application** event, for instance, a procedure called oApp_DocumentBeforeClose will fire when **any** document closes; whereas because it is a Document event, a Document_Close procedure will only fire when documents based on the template that the Document_Close procedure is stored in are closed.

**7.** Having inserted the event procedure you want using the listboxes, add your code, and you're done.

**Note:** If you want to monitor application events affecting documents based on a specific template, you can either store the event code in a global Addin as described above and test for which template is in use by using **ActiveDocument.AttachedTemplate**; or you can store the event code in the "ThisDocument" Class Module of the relevant template.

For an example of the latter, see: **How can I prevent users from editing the header of a document in Word 2000 or higher?**

## Summary of set-up procedure


oApp
oAppClass =
oApp
Create a container (Class Module)
Create a Property of the container (oApp) and declare what type of object oApp will be
In a normal module, declare an object variable to reference the container. The properties of the container are now available to you but not yet loaded into memory
oApp =
Load the oApp object into memory and assign it to the Word Application, using the Set statement
You can now create event procedures of the Application object

## oApp_Quit

An oApp_Quit event procedure works in exactly the same way as an AutoExit macro – it fires when Word quits. Unfortunately, both the AutoExit macro and the oApp_Quit procedure fire *after* any "Do you want to save changes" dialogs have appeared.

There is no event of the Application object analogous to an AutoExec macro, however.

## oApp_DocumentChange

The DocumentChange event triggers whenever you open/close/create a document and whenever you change focus from one document to another. If the event code is stored in an Addin, it will be global.

In Word 2000, there are separate events you can use for each of these –

oApp_NewDocument for when a new document is created, etc. But in Word 97, you can use oApp_DocumentChange to simulate them.

Even in Word 2000, simulating them sometimes works better than using the "proper" events.

For more details, see: **How to create global event procedures similar to AutoOpen, AutoNew and AutoClose, without using Normal.dot**.

## oApp_DocumentBeforeClose, oApp_DocumentOpen, oApp_NewDocument, oApp_ WindowActivate, oApp_WindowDeactivate

All of these combined – at least in theory – replace (and build on) the functionality of oApp_DocumentChange.

The oApp_DocumentBeforeClose, oApp_DocumentOpen, oApp_NewDocument event procedures are covered in the article: **How to create global event procedures similar to AutoOpen, AutoNew and AutoClose, without using Normal.dot**.

The oApp_ WindowActivate, oApp_WindowDeactivate events are self-explanatory. It's safest to use the list boxes on the toolbar to insert the event procedures, as the syntax is complicated.

## oApp_DocumentBeforePrint, oApp_DocumentBeforeSave

These are discussed in the article: **Intercepting events like Save and Print**.

## oApp_WindowBeforeDoubleClick, oApp_WindowBeforeRightClick, oApp_WindowSelectionChange

Again, these are self-explanatory, and it's safest to use the list boxes on the toolbar to insert the event procedures, to ensure that you get the syntax right.

Using the WindowBeforeDoubleClick and WindowBeforeRightClick events seem to have a significant performance hit on Word; but the WindowSelectionChange events seems to work well (subject, that is, to the code you put in it).

Terms of use Disclaimer

# How do I generate an index in Word?

Article contributed by John McGhie

The Microsoft Word Help suggests that you can automatically generate an index. Sorry, but you can't (the "result" looks like an index, but the reader can't use it). You **can** automatically mark index entries: however, the amount of work required to edit the result into a useable index is usually double the effort required to manually mark the index entries one-by-one.

Instead of automatically generating something that is not useable, the reader would far prefer you to express the document electronically and provide a free text search. A free text search serves the reader's needs far better than a badly-constructed index, and the search engines available these days are smart enough to look for what the reader “wanted” rather than what he or she “asked for”.

## Making an Index

An experienced technical writer wrote this article. As a technical writer, I produce long documents running to thousands of pages of technical material. Indexes are part of my game. I can't tell you how to produce one **automatically**, but I can tell you how to produce one **easily**!

Before 1990-ish, Indexing was a profession of its own; in addition to an Author and an Editor, a large book had an Indexer. Even today, if you are making a book such as a medical encyclopedia that is going to remain in print for many years, it is simply stupid **not** to use a professional indexer. Really good indexes are an even mix of science and art form, and the quality improvement a professional makes is well worth paying for. Of course, few of us these days work on publications that are going to last long enough to justify this effort. And even fewer of us have the time to produce such an index. If you do have the time, obtain a copy of “**Indexing, The Art of**” by G. Norman Knight (Allen & Unwin, ISBN 0-04-029002-6). Norman Knight is a former President of The Society of Indexers, and his book is simple and charming. Reading it, you will soon realize that indexing is not difficult; it simply takes attention to detail and patience.

### Planning the Job

Word has one of the nicest and most powerful index generators around built right in, so you have all the tools you are going to need. You need to allow a week per 500 pages to generate an index in a technical book. Technical publications are fairly “information dense”. Scholarly monographs and the like are usually quicker to index.

### Types of Index

In the old days (say, 1995 or thereabouts!) indexes were all produced by the “shoebox” method. They literally used a shoebox into which they inserted index cards: three-inch by five-inch cards upon which they wrote the index term and its page number. The Indexer would sit with a large pile of “galley proofs”, single-page images as they were returned from the typesetter, and go through each one line-by-line seeking and recording the index terms. At the finish, they typed the index out with its page numbers and sent it off to the typesetter for publication. There is a software tool specially built for indexing that emulates this process exactly. I tell you this simply because, in certain circumstances, this method is still the best today. ***If your document is going to be published from a different computer to the one it is being created on, and that machine cannot interpret Microsoft Word XE tags, and you do not know what the page numbers are yet because the other machine is going to do the pagination, then use the shoebox method!***

Word will do two forms of index: The **Concordance Index** and the **Mark-up Index**. It will also do something half-way in-between, using its “**Mark All**” command.

#### Mark-up Indexes

**A Mark-up index is the method I recommend.** It's quick, accurate, easy to understand, and easy to correct. With a little care in the planning, it normally results in a very useable index.

As the term implies, you produce a mark-up index by embedding mark-up "tags" in the Word document. Word automatically looks up the page numbers at Print time and generates and formats the index for you. Study the help topic "Create an index" and all its sub-topics. This is the way I recommend. It's the way that all good writers create an index these days. Mark by mark, page by page! It is explained in detail below.

### Concordance Indexes

I implore you not to waste your time with a Concordance Index for most publications. It results in a huge pile of rubbish that is of very little use to the reader. And it takes nearly as long to make as it does to generate an index properly. The Concordance Index is a hangover from the past when people were desperately hoping to produce an "automatic index" to reduce the labor. Every major word-processor will do them, and no professional writer or editor would, these days, permit one.

To make a Concordance index you make up a table of all the terms you want Word to find in one column, and the index entry you want to see for each term in the other. For more information, see "Create a concordance file" in the Word help file. But the end result is that you have every term indexed at EVERY place it occurs. Most of the mentions of a term in a book are simply passing references: what the reader wants to see in the index is only **one** page number; the one that contains the main topic for the term. If you send them on a wild goose chase to 20 other places first, they will think most unkindly of you.

The concordance mechanism does have its place: It can often be used to good effect in Reference Books such as Programming Reference Manuals, where each command or function is referred to only in a small section of the text, then rarely mentioned anywhere else in the book.

### For the truly adventurous...

Technical writers and other folk who publish seriously-huge documents in HTML may want to spend a little time learning about Concordance Indexes. In conjunction with VBA, a concordance index is a great way to automatically generate hyperlinks in your document. You tag every mention of each term with the concordance indexing mechanism, then use VBA to change the tags into hyperlink tags.

## Indexing Made Easy

Here are some worthwhile hints I can give you so you do not go mad during the process:

1. Print a copy of the book and go through it with a highlighter, marking the items you would like to see in the index. If you are not the subject-matter expert, get someone who is expert in the subject to do this for you (the process is massively easier if you understand the subject well). Mark only places where the reader will get information about each item. For example, if you want to include "installation procedure", you would mark "Follow the procedure below to install..." in Chapter 1, you would not mark "if you completed the installation procedure..." in Chapter 5. The first is what the reader would expect to see when he looks up 'Installation Procedure'. The second might cause the reader to come and look you up {grin}.
2. Make some design decisions before you start putting codes in the file. The most important are:
   - How many levels of entry are you going to allow? If it is more than three, I will personally come and shoot you! Such an Index is both unusable and unmaintainable {grin}.
   - Are you going to reverse the terms? "Indexing, the art of" or "The art of Indexing"? Normally do the former, but whichever you select, you must do it for every entry
   - How will you treat numbers? All as if they were spelled out; or all up the front above the "A"s? In technical books, do the second, but whichever you do, you must do it for every number.
   - Will you use "see" references to condense the index? My vote in modern times is: "No, don't bother".

     "See" references mean the reader finds the index entry, then has to go find another index entry before they can find the page. It annoys your reader, it doesn't save much paper, and these days paper is not very expensive.

- Will you "put the Table of Contents in the Index"? Debate rages in the more pedantic Indexing circles about this one.

  The pedants (sorry, "purists") say you should not include in the index terms that are contained in headings in the table of contents. I say: "Of course you should". Research shows that some people (about 35 pct) look in Tables of Contents, some people (about 60 pct) look in Indexes. Few readers these days have a clear picture of the conceptual difference between them, and each reader will secretly thank you if he can find what he wants in both places. I always include an index entry for every heading in the book. So shoot me!
- Sort order: Word-by-word or "letter-by-letter"? By default, Word does the former. Purists like the latter: I don't; I can never find anything in such an index, and most readers hate it. So shoot me again! To produce a letter-by-letter sort, you have to place the generated index in a two-column table (page numbers in one column, text in the other). Then copy the text column, remove the spaces from it with Find/ Replace, then shift that column to all upper-case and sort by it. Then remove the uppercase column and turn the table back into text.
- Avoid the classic hilarity of putting "the book" in the Index. If you are writing a book called "All About Word" you may get sued for a laughter-based injury if you include "Word" as a term in the index. But for your own amusement, have a look in the indexes (not "indices"!) of a few cheap-and-nasty technical manuals such as are often produced in-house as training manuals. You will be surprised how often you see this classic faux pas. And you may immediately become suspicious that you are looking at an automatically generated index!

**3.** Now run through and tag the entries you have highlighted, according to the instructions in the help topic "Mark index entries". Unfortunately, if you have made a few indexes, you will know how to do this, and if you haven't, your first attempt will contain errors. Sorry: I had to go through this too {grin}.

I will give you a hint that will save you a bit of time (quite a lot, actually...) **Do not** put in the subentries at this stage. By that I mean tag each item as a main term. If the entry does belong as a subentry, you will find that you can add the main term to the tag more simply on your second pass.

#### A Word About Tagging:

Word's index tags are both case-sensitive and "space-sensitive". "Installing" and "installing" are not the same thing: each will appear under its own heading. "Administration" and " Administration" are not the same thing: one will sort right at the top of the index. See? When you are debugging "entries out of sequence" you sometimes have to look extremely closely to ensure that the tags really do match exactly.

To enter an index tag in a heading, ensure that your headings are formatted by styles, and do not apply any formatting overrides to the heading. If you apply direct formatting to the headings that contain index tags, the direct formatting will be copied through to your Index.

A colon : and a semicolon ; are not the same thing! You use colons to divide the levels of sub-entry in your index tags. When you are in a hurry, it is too easy to type the un-shifted character (the semi-colon) instead of the shifted character (the full colon) in the tag. If you do, you will get some very weird errors in your generated index. There's no easy way to find these, but the semi-colon will appear in the index. If you have strange things happening (items that do not appear under their correct entries or sub-entries) try searching your generated index for semi-colons. If you find any, at least you know "what" is wrong: finding the tag that produced the problem is a real chore (it will not be on the page in the index...). Try this: Reveal your hidden text (so you can see your XE tags) then search for a semi-colon with the font format hidden text. If you find any, chances are they are in your bad index tags.

**4.** Now generate the index. Ignore the formatting at this stage; just print it. Leave it as a single column for ease of reference. If you have a big screen, you can open a second window into the document and look at the index that way (see the Window menu) but for most, it's easier to print the first result.

**5.** Now sit down with a colored pen or pencil (you can't see blue or black against black type...) and edit the index.

- Mark all the terms that should become sub-entries, and show the term they should be sub-entries of.
- Now run down it, and for each term, ask yourself "What else could the reader possibly call this?" Add an entry for each.
- Run down it again, and for each term, ask yourself "Is there anything else the reader would need to know about when looking this up?" Add a "See also" for each one you find.

6. Go through and edit the tags in the file to implement the changes you have identified.

   You can find index tags easily by using the Browse buttons on your vertical scroll bar (see “Browse to the next or previous page, table, or other item” in the help).

   In later versions of Word (2002 and above) you can use **Ctrl + G** to bring up the "**Go To**" dialog. Set "**Go to what**?" to "**Field**". Set the **Enter field name** box to "**XE**". Click **Next**, then **Close**. Your "Previous" and "Next" browse buttons (at the extreme bottom right corner of the Word window, under the vertical scroll bar) will now go to the next or previous index entry fields on each click, until you change to something else.

   If you use **Find**, or **Browse by Find**, you can specify **^d XE** as your **Find** string to find only index tags.

   If you know exactly what the text of the tag is, you can use **^d XE "*tag text string*"** to find exactly that tag. However, this requires you to work out exactly what the tag content will be, and that's not easy three levels down in an Index.

   So I prefer to use **Ctrl + G, Page Number** (from the index), then **Ctrl + F, ^d** (to find the next XE tag. Then keep hitting **Browse Next** to find the tag you want.

7. Now regenerate your index. (Click in it and press **F9**). You can now change it to double-column if you wish. You format an index by using Format>Style to change the styles Index 1 through Index 9. Each style controls the formatting of one level of entry.

## Page Number Conflation

Page number conflation is where only the first and last page numbers appear for a topic. In the index you see 88 - 95 instead of 88, 89, 90...

I am very tempted to say "don't bother"! Tag the first instance of each term. If your reader does not have the brains to see that the information on a topic continues for several pages, they should be kept away from your book in case they hurt themselves... However, if you absolutely must conflate, there are two ways of doing it:

- If you place the same index tag on each page of the topic, Word will automatically conflate the page numbers.
- If you bookmark the whole section, then place the name of the bookmark in the XE tag, Word will generate a conflated page reference for you.

## See! It isn't that hard

There! That's the way I do it. If you trust me and do it that way, you will find out why I do it that way. If you don't trust me and do it another way, you will find out why much sooner {grin}.

Send all feedback to **word@mvps.org** Top - **Ctrl+Home** Previous - **Alt+left arrow** Page Updated: