VBA aka Macros. Why? How?

You've probably heard of VBA or macros. You may have even seen snippets of VBA code on web sites, discussion boards and books. But what good is it? And if it's good, how to you use it?

That's what we're here to talk about.

Let me set the scene:

Suppose you've got a problem in PowerPoint that has a relatively simple solution. For example, The Client has suddenly decided that they hate the nice rounded corner rectangles you created for them.

They want square corners. Tomorrow.

No big deal, right? Select a rectangle, a couple more clicks, and boom! Square corners, voila!

But hello, wait! There are 10 of these presentations. Each has 50 slides. Each slide has dozens of rectangles. Rectangles with rounded corners that need fixing.

This is going to be a very long night. And some serious wrist pain before it's over.

So you ask around on the interwebs and people tell you how to remove the rounded corners. Which you already know how to do. Or tell you that there's no way to fix more of them than you can select at one time.

But then there's this one reply that looks like a bunch of almost-English words randomly strewn on the page by inebriated chimpanzees for all the sense it makes.

They say it's VBA and that it's just what you need.

OOOoooookaaaayyy. But what's a VBA (or macro – they may have called it that) and what do you DO with it now that you have it?

VBA means Visual Basic for Applications. It's a programming language that's built into PowerPoint. It's used to write macros, bits of computer code that make PowerPoint do things that you and your mouse could easily do on your own.

Big whoop, you say?

VERY big whoop. See, VBA can make PowerPoint do these things hundreds of times, automatically. WAY faster and more accurately than you can do it yourself. VBA never gets bored and YOU never get carpal tunnel syndrome. Now do we have a deal?

We have a deal. So how do I USE it?

Assume for now that you have some code and just want to use it. Later on, I'll show you some sample code you can try out. Oddly, the sample code turns all of the bazillion rounded-corner rectangles in a presentation into square-cornered rectangles. Mere coincidence? I think not.

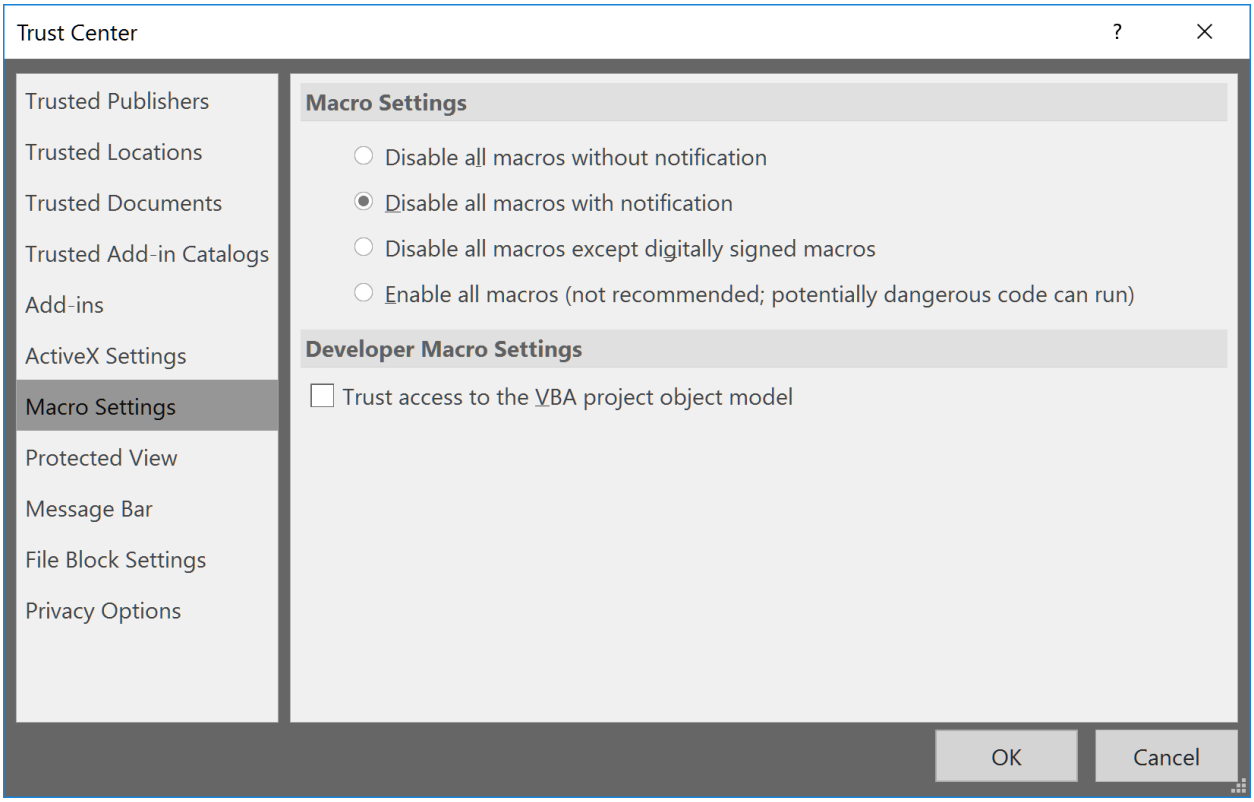
I won't explain how it works, but I'll show you how to make it work. For you.

First, some setup

First, we'll make sure that PowerPoint permits VBA to run in a safe way.

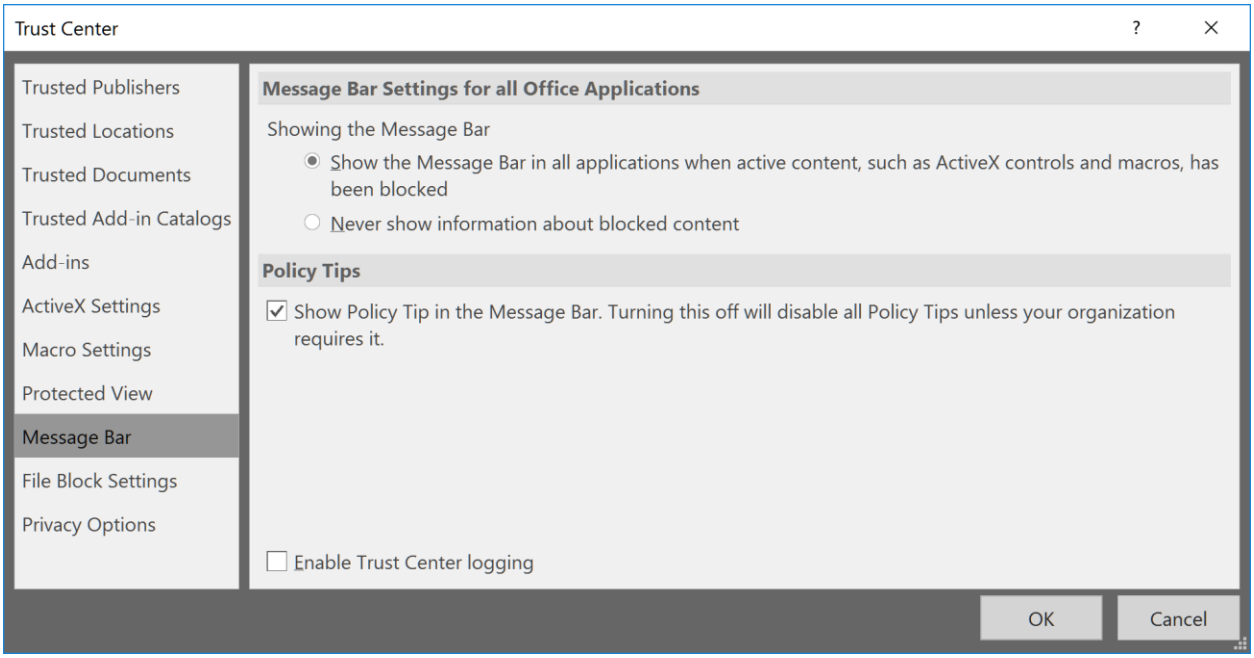
Start PowerPoint and open any presentation or start a new one.

Choose File | Options. In the left pane of the PowerPoint Options dialog box, click Trust Center. Click the Trust Center Settings… button. In the Trust Center dialog box that appears, click Macro Settings in the left pane.



Under Macro Settings in the right-hand pane, click Disable all macros with notification.

Next click Message Bar:

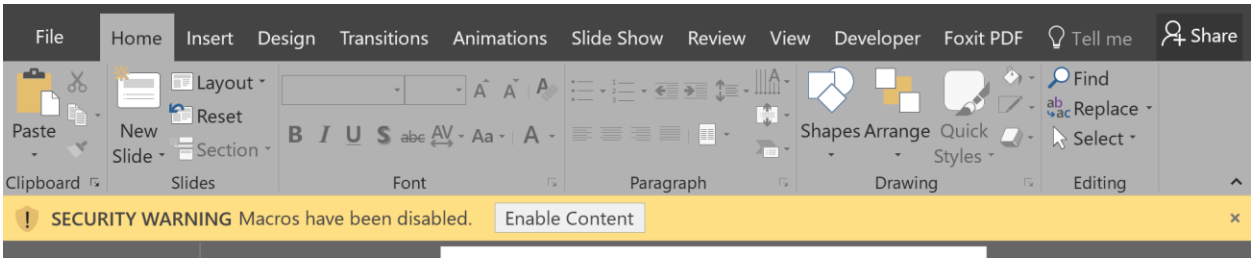


Click Show the Message Bar in all applications when active content, such as ActiveX controls and macros, has been blocked.

Click OK twice to return to PowerPoint.

You only need to make these settings once. They're now part of PowerPoint until you change them.

With these settings in place, PowerPoint will warn you when you open a file that includes macros and some other types of "risky" content:



When you see this warning, click Enable Content to open the file and use the macros or click the X at the right of the yellow Message Bar to dismiss the warning and open the file with the macros disabled. It's a good idea to disable macros if you don't know where the file came from or didn't expect it to contain macros.

Give your macros a home

Start a new blank presentation, choose File | Save As, choose any convenient location, set Save as type: to PowerPoint Macro-Enabled Presentation (*.pptm), name it My Macros and click Save.

You can actually include macros in any presentation file, but it's a good idea to keep them together in a separate presentation file.

If you add macros to a PPTX file, you'll have to save it as a PPTM (macros aren't allowed in regular PPTX files) and if you send the PPTM file to someone else, their security settings may set off warnings or prevent them from opening the presentation at all.

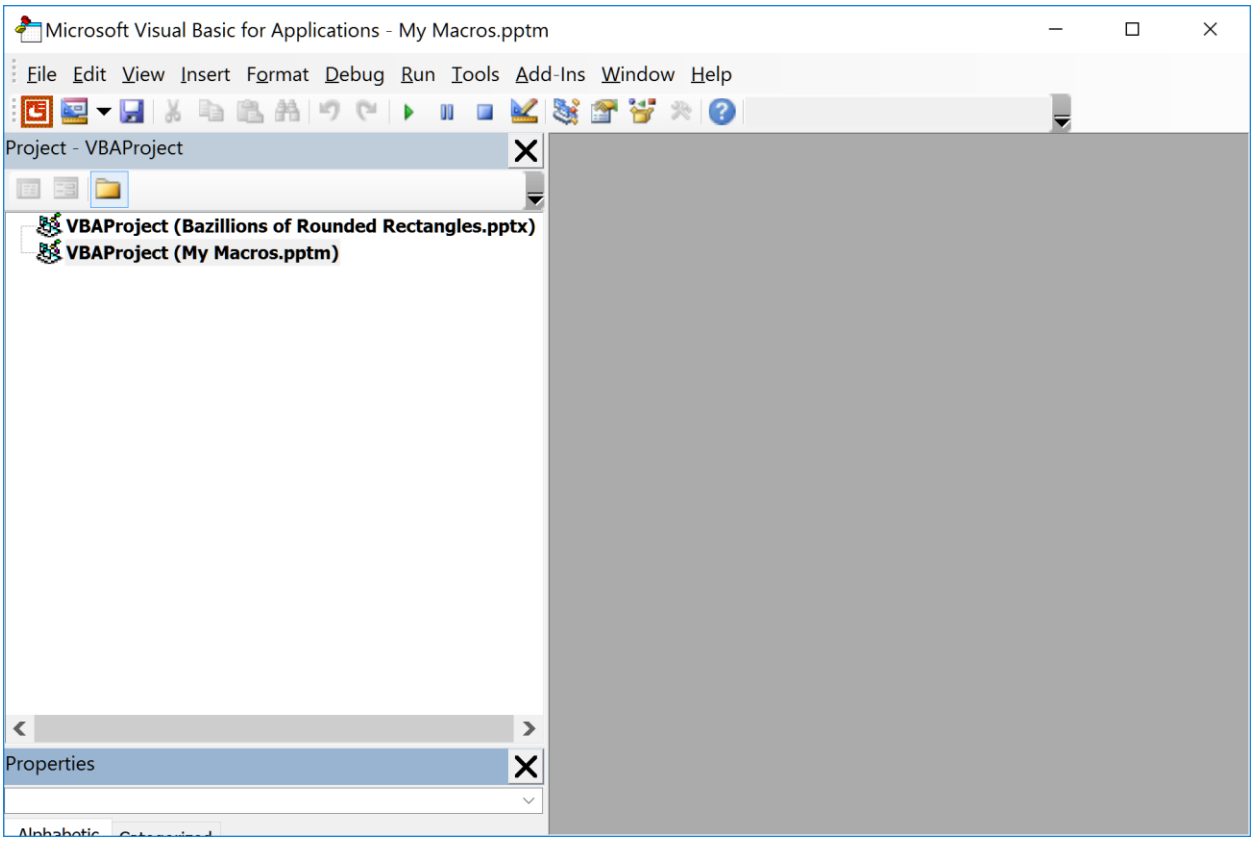
Nuff said? OK. So save your file as My Macros.PPTM.

Leave My Macros open and also open the presentation you want to run the macros ON. Mine's called Bazillions of Rounded Rectangles, because that's what started this whole exercise: files with lots of rounded rectangles that needed fixing.

Set up the VBA editor (aka IDE)

The VBA editor (also called the Integrated Development Environment or IDE to its friends) is where you'll store VBA code in PowerPoint. There's a little setup to do there too.

Open the IDE by pressing ALT + F11. It will look something like this:



If you don't see the Project pane, press CTRL + R. If the Properties pane is missing, press F4.

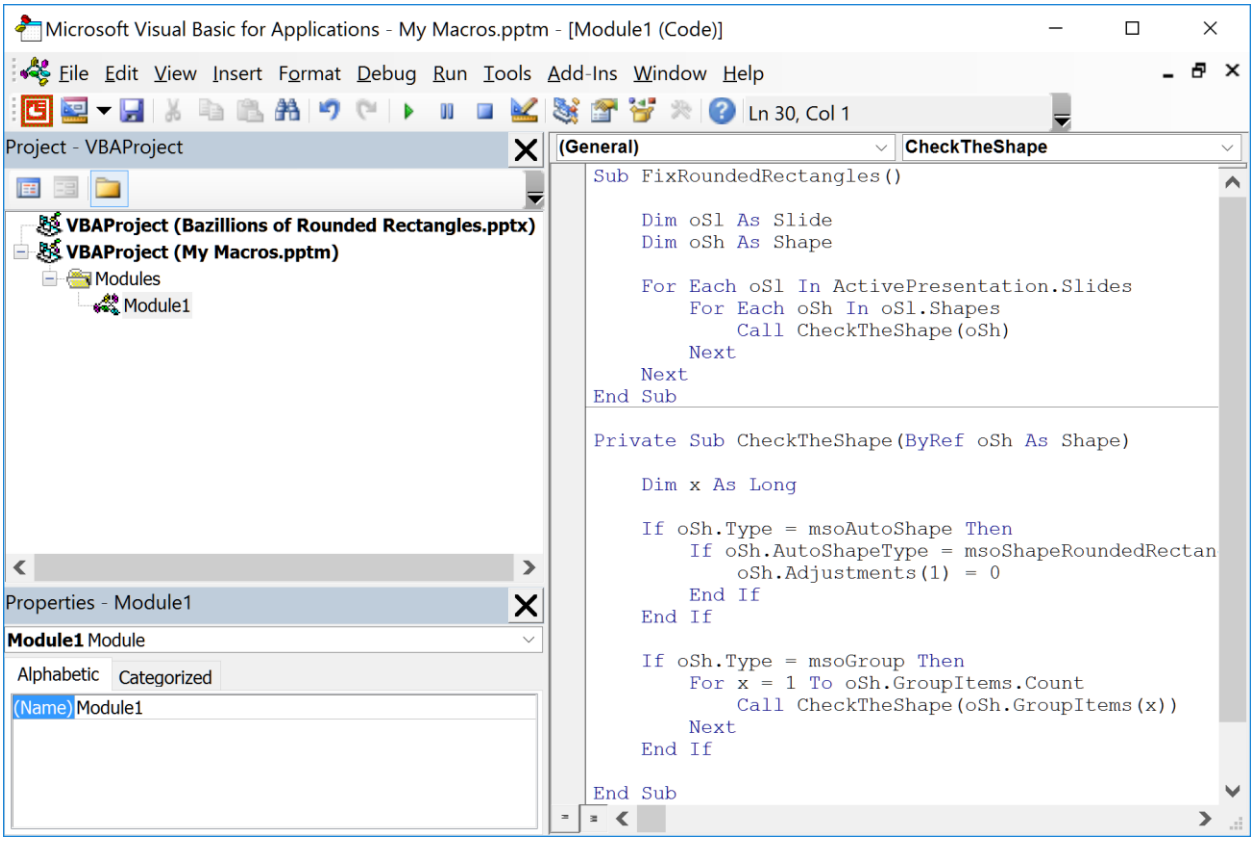
Notice that the Project pane shows you all open presentations, whether or not they contain macros.

Next, we'll add a new Module to our My Macros project. Modules are one of the places that VBA uses to hold code. Click My Macros in the Project pane to select it then Press ALT + IM (Insert Module) or choose Insert | Module from the IDE menu bar.

Click anywhere in the new module then copy and paste in this code:

```
Sub FixRoundedRectangles()

    Dim oSl As Slide
    Dim oSh As Shape

    For Each oSl In ActivePresentation.Slides
        For Each oSh In oSl.Shapes
            Call CheckTheShape(oSh)
        Next
    Next
End Sub

Private Sub CheckTheShape(ByRef oSh As Shape)

    Dim x As Long

    If oSh.Type = msoAutoShape Then
        If oSh.AutoShapeType = msoShapeRoundedRectangle Then
            oSh.Adjustments(1) = 0
        End If
    End If

    If oSh.Type = msoGroup Then
        For x = 1 To oSh.GroupItems.Count
            Call CheckTheShape(oSh.GroupItems(x))
        Next
    End If

End Sub
```

If you've maximized the module window, it should look like this:



Notice that VBA automatically named the new module Module1. Very original, yes? For files with only a few macros, there's no reason to change the name, but as you accumulate more useful macros in this one file, it'll be easier to use if you organize your macros into different modules and give the modules meaningful names.

So humor me. Click on Module1 in the Project window to select it. Then in the Properties window next to Name, change Module1 to FixRoundRectangles. The module name in the Project window changes to match. You can thank me later.

To make sure there are no serious syntax problems with the code, choose Debug |Compile from the menu bar. If there's a problem, you'll see a message explaining what VBA doesn't like about the code. Usually in a geeky, obtuse and sometimes incomprehensible way. Click OK and the problem line will be highlighted for you. Fix it as best you can and compile again until you get no error messages.

Rock It!

This is where the magic starts. And where the carpal tunnel problems stop.

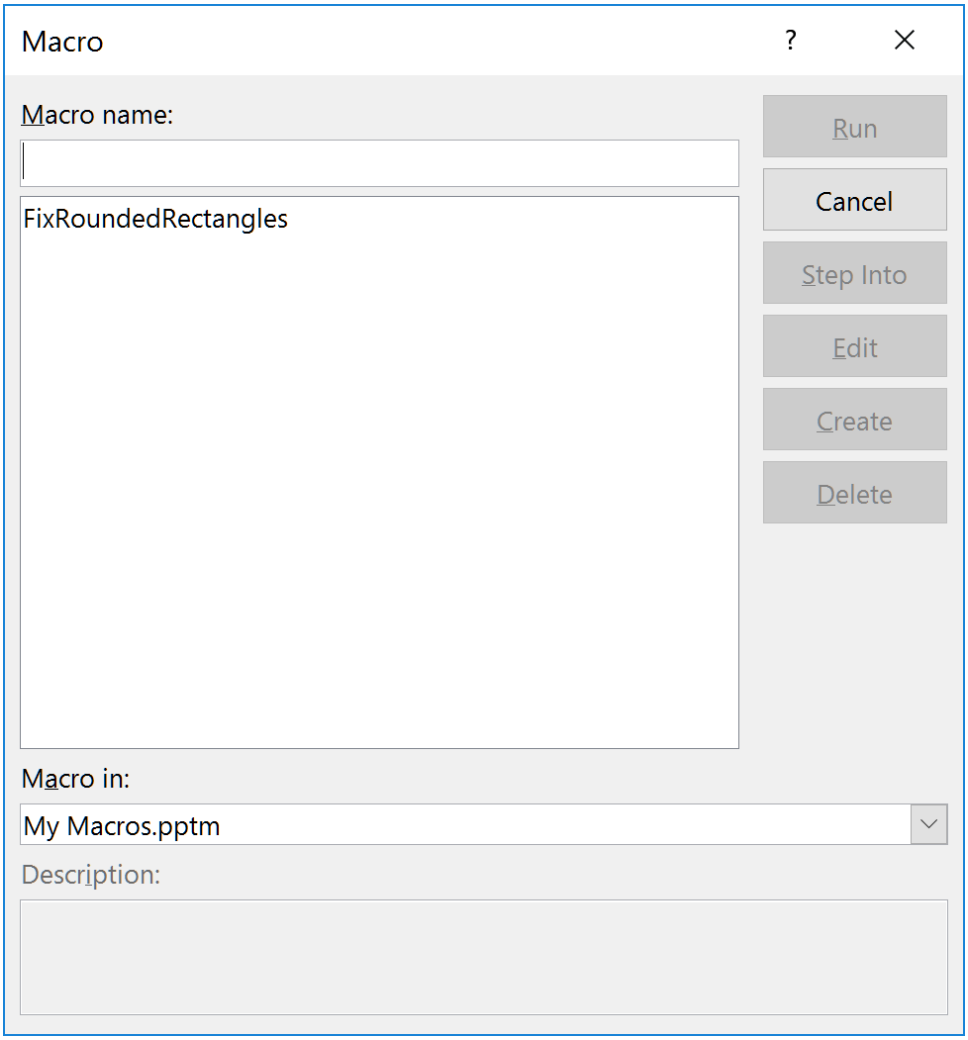
Leave the IDE open but switch to the presentation you want the macro to fix. It should be one with lots of rounded rectangles, of course. It's important to make this the active presentation, ie, the one that's currently visible and receiving keystrokes/mouseclicks before you go to the next step.

Which is: Press Alt + F11 to go back to the IDE. Click just below the Sub FixRoundedRectangles() line, then press F5 to run the code.

Switch back to your presentation that has the rounded rectangles. USED to have rounded rectangles. Because now every rounded rectangle is a normal square-cornered rectangle.

Want to see it happen again? Immediately press CTRL + Z to undo what VBA has done, then run your code again.

Hint: While you're looking at the presentation you want to run your code on, you can press ALT + F8 to open the Macro dialog box and run your macros directly rather than jumping to the IDE and back:



Click the name of the macro you want to run then click Run (or just double-click the name). If you don't see the macro listed, check the Macro in: dropdown list box. Choose either the name of your macros file or All open presentations.

Why stop with just one macro?

Now that you know how to add and run macros, you may find yourself collecting useful-looking ones when you find them on the web, in books, etc. Use your My Macros file as a kind of macro library. Whenever you want to run a saved macro, open My Macros and run the code.

When you see some useful-looking code, select it, copy it, open your macros file, go into the IDE, add a new module and paste in the code. Remember, ALT + F11 to open the IDE, ALT + IM to insert a new module, click inside the module and CTRL + V to paste in the code.

If the code turns red when you paste it into a module, the likely cause is unwanted linebreaks or other extra characters (invisible) copied from the web browser or other source. Delete everything between line breaks and get rid of spaces before indented items to correct this problem.

Want to learn more or find other useful macros?

Visit the PowerPoint FAQ at http://www.pptfaq.com and scroll down to the Programming section. There, you'll find lots of ready-to-run example code, VBA tutorials and links to many other VBA-related sites.

If you need help finding a macro or fixing one that's not working quite right, you can find helpful volunteers on the Microsoft Answers forum (https://answers.microsoft.com/en-us/msoffice/forum/msoffice_powerpoint) and other VBA-related sites.