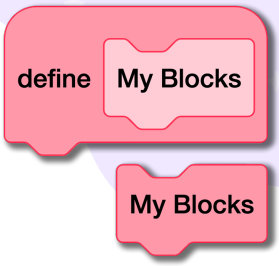


Make Your Own Custom My Blocks

Have you seen the “My Blocks” category in the Scratch blocks palette? When you click on it, no blocks are present until you create one. Let’s explore how to create a customized block and when and why you might want to use them.

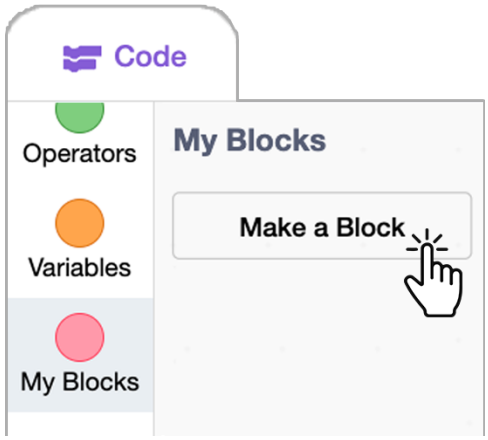
In this guide, you’ll find:

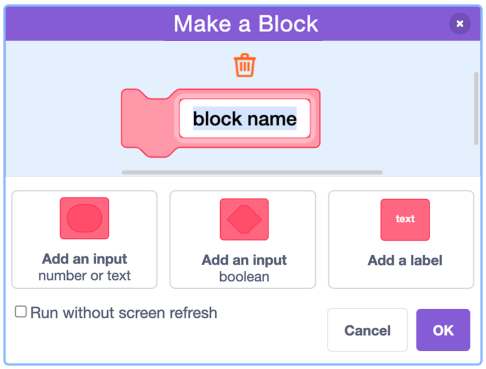
Creating a Basic My Block

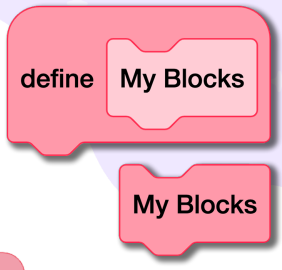
As you’re creating a Scratch program, say you’ve written a sequence of steps that performs a particular action. That action could be performing a complicated spin, playing a set of musical notes, drawing a shape with the pen tool, performing some complex math…or anything else. That stack of blocks is known as a “procedure” or a “routine.”

Most computer programming languages allow you to create a name for a procedure and then call for it to run at any point in the main program. In Scratch, we use “My Blocks.”

To create a block, click on the “**Make a block**” button under the “My Blocks” category in the blocks palette. You can give your block any name you want, but it is best if it is a **descriptive name** so you can recall later what the procedure does. For this basic block, once you’ve provided a name, simply click “OK” (no need to add or check anything else).



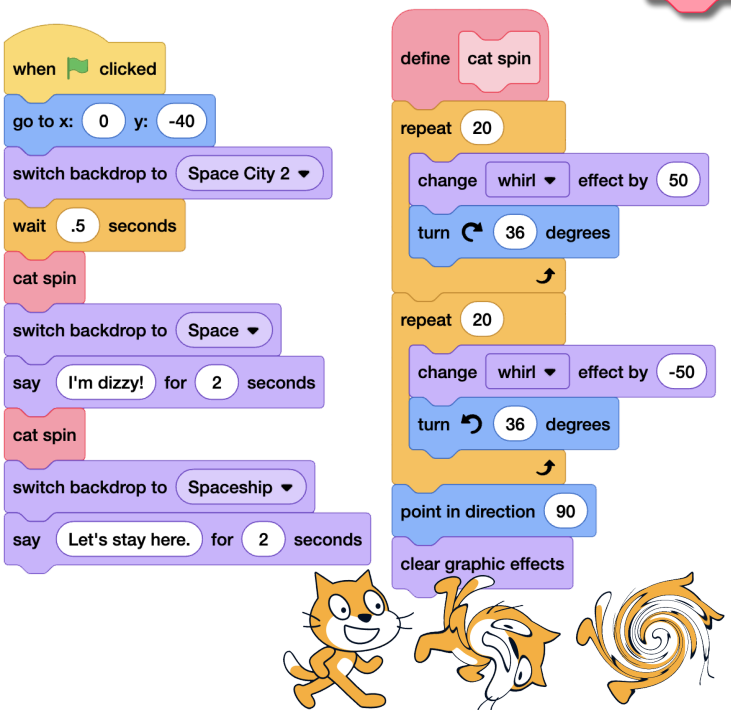




Once a new block is created, you'll see a new "**define**" event handler block appear on the scripts area. Place all the blocks in your procedure under this block. (In the example below, what do you think the "cat spin" procedure does?)

Now that you've defined what your custom block does, you can use the new custom block in your main program. The advantage of writing this procedure separately is that you can use one block to call for a set of steps to be run each time you need them. **Creating separate procedures using custom blocks makes the code faster to write and read, and easier or quicker to edit.**

A custom block isn't just for code that will repeat. If you have a long complex sequence that performs a particular action, you may want to define it as a custom block so you can more easily identify and edit the steps involved later.

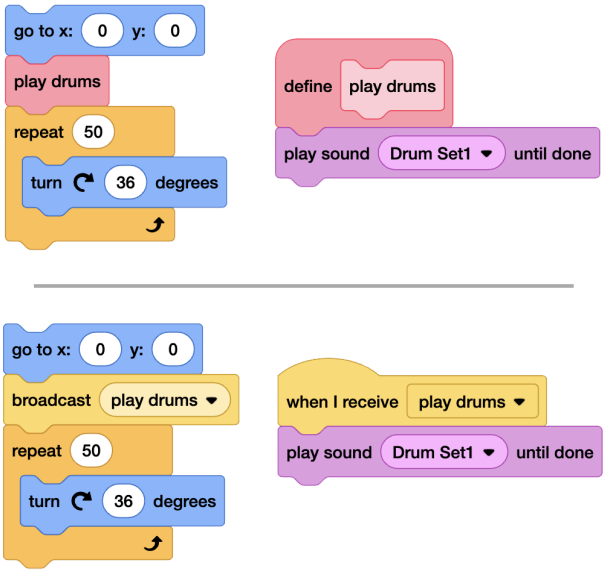


My Block Versus Broadcast

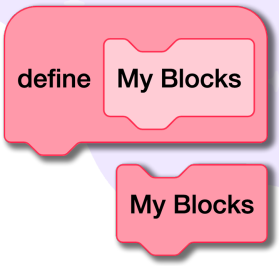
If you have used a broadcast block before, you have already practiced writing a procedure outside the main program. When the sprite or backdrop receives a broadcast, it can run a series of code that you have defined under the "when I receive" hat block. So what is the difference between using a broadcast and creating a custom block to hold a procedure?

When you use a custom My Block in your program, the program pauses and runs through all the steps under the "define" block before preceding.

When you use broadcast in your program, the program sends the message and then proceeds with the next steps in the program, so code sequences may run simultaneously.



Try these two code sequence pairings to see the difference between calling for a custom block and calling for a broadcast.

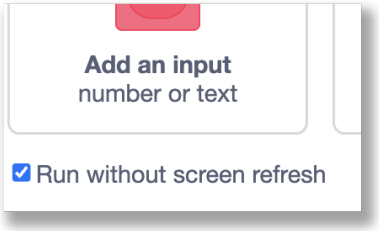


When choosing whether to use a custom My Block or a broadcast, think about whether you want to run through each step in the procedure before moving on (My Block), or want the procedure to run simultaneously as the main program moves ahead (broadcast).

Since the program pauses to perform all the code defined under a custom block before moving on, if you put a forever block under a custom My Block, it will never get to the next code blocks in the main program. It is best to use a broadcast for code that runs forever.

It is also important to note that **a custom block is specific to the sprite where it was defined**. If you want to use the same procedure for additional sprites, you'll have to copy the "define" event block and the procedure attached to it to each sprite. And unlike a broadcast block that can send messages globally between all sprites and backdrops, a custom My Block is local, usable only by the sprite it is defined on. The call for the custom block isn't received by any other sprites, even if their custom block has the same name.

"Run Without Screen Refresh"

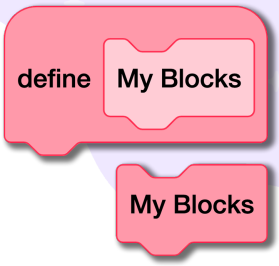


When you create a custom My Block, the option to "Run without screen refresh" can be checked on or off. What does it mean? When you are viewing a Scratch program run on the stage, you may not realize that the screen is constantly refreshing, redrawing the scene at a certain frame rate.

If the refresh is happening at the right speed, your eyes should perceive continuous motion and you can see all the intermediate steps as the program progresses. But if the sequence runs quickly between refreshes, your eyes will not be able to perceive the action and you'll only see the finished product. Sometimes we may want to see each step, and sometimes we may just want the result. (See https://scratch.mit.edu/projects/921473501 or https://scratch.mit.edu/projects/921541090 for examples of with or without screen refresh.)

When you check to "Run without screen refresh," Scratch will attempt to run the script as fast as possible (in one frame) and will not refresh the screen until it has finished executing all the steps in the procedure. If you've used turbo mode before in Scratch, it is a similar concept, but turbo mode applies to the entire program, while "Run without screen refresh" only applies to the procedure in your custom block.

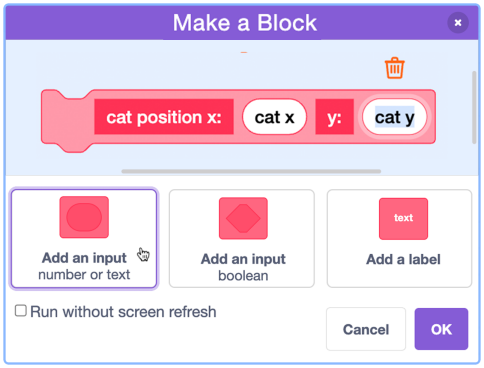
One important note is that **if any of the blocks in your custom block involve time, you should *not* use "Run without screen refresh."** Because the program needs to honor the time, trying to run the code without refreshing could cause a lag, screen freezes, or even program crashes.



My Block with Parameters

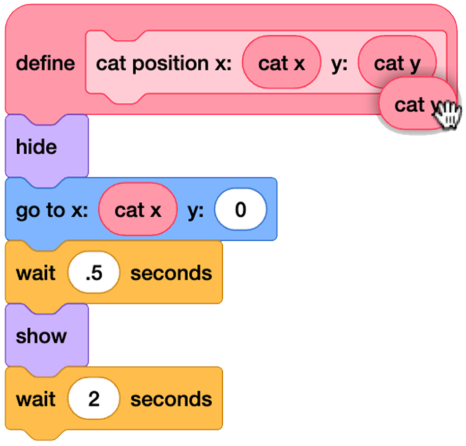
What if you want to perform the same procedure each time the custom block is called in the main program, but with a small modification each time, like the text shown or the coordinate used or a different value in one or more of those blocks? You can create a My Block with parameters!

When creating your custom block, you'll need to add an input. **With an input in place, the custom block will use the parameter (the data you'll provide in the input bubble) when you use the block. You'll also want to give your input a descriptive name.**



If you are using more than one input in your block, you can also add labels (descriptive text) between the inputs to help you remember what each input bubble is for when you use it in your main program.

As before, add the desired code sequence under the "define" block. Then, in place of static, unchanging information in your blocks, you can use the input. To get the input, click on the input label in the "define" block and drag it out to place it in a code block.



Now, when you use this custom block in the main program, you can see the blank input bubble where you can enter the parameter.

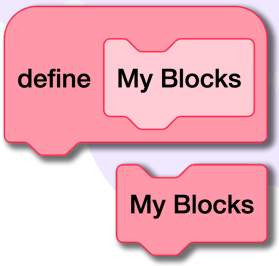
You are able to use the same code sequence and customize it over and over without rewriting the code stack each time.

(See https://scratch.mit.edu/projects/924066221 and https://scratch.mit.edu/projects/921462593 as examples.)

You can:

- Create multiple custom blocks, with or without inputs, to use throughout your program.
- Use a custom block inside another custom block. (Example: https://scratch.mit.edu/projects/921283811)
- Combine broadcast blocks and custom blocks, depending on your needs, to create a dynamic program! (Example: https://scratch.mit.edu/projects/921470324)

What will you create?



Tip: If you'd like to translate this guide, **click here to make a copy** of this Google doc.