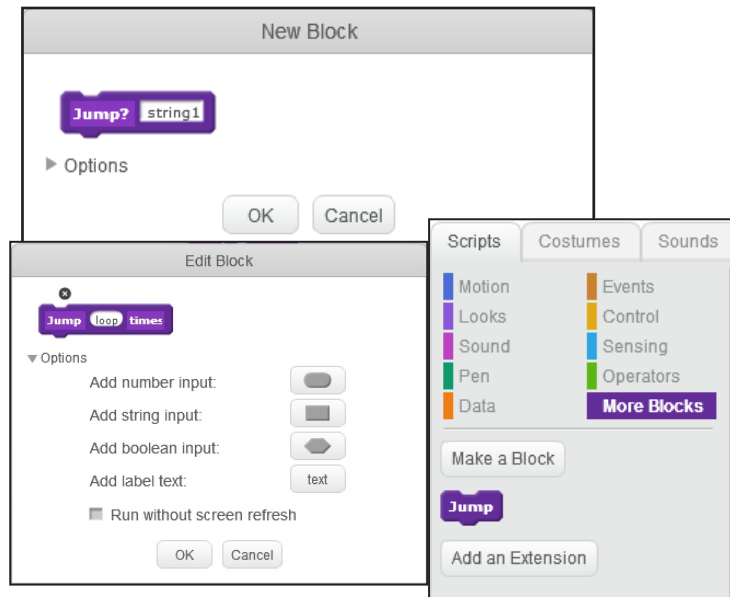


# Chapter 10



## Create Your Own Blocks with Scratchy

After completing this chapter you will be able to know create your own blocks and make the sprite:

- use the defined Jump block;
- jump according to the number inputted by the user;
- rotate in degrees back and forth;
- jump based on string input by the user; and
- jump using the boolean input from the user.

# Create Your Own Blocks

Congratulations. You have reached the last chapter of this book. You've created a lot of scripts, and we've discussed several ways to make them more efficient. In this chapter, you'll learn another: creating custom blocks. The best way to illustrate the power of custom blocks is with an example. Suppose you want a sprite to jump up and down. That's easy, right? You snap together a sequence of blocks, as shown in Figure 10-1. Now suppose you want several sprites to jump in your project or you frequently write scripts with jumping sprites. Every time, you'd need to go back and forth between categories in the block palette to drag out and snap together these three blocks. Or instead, you could create a custom block called **Jump** (see Figure 10-2) that represents and already includes these three blocks. Save this custom block in the backpack, and then every time you need a sprite to jump, you just snap this one custom block into your script (see Figure 10-3) instead of the three-block sequence.



Figure 10-1. Jump script



Figure 10-2. Custom Jump block

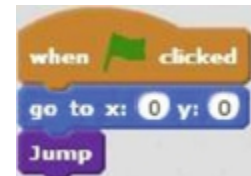


Figure 10-3. Script with custom block

Not only do custom blocks make creating scripts more efficient and faster, but they also make troubleshooting your scripts easier. Because the script is broken down into parts, you can more easily track down and isolate problem areas. Custom blocks are an example of what high-level languages call procedures or functions, which are subprograms you can use in other programs. Instead of creating a certain sequence of code every time you need it, you just create it one time by writing a procedure, then call the procedure repeatedly from many other programs. Next, I will show you how to create and use your own custom blocks. After that, you can try out some example scripts that show you different types of custom blocks in action.

## Make a Block

To make a custom block, you first create the block itself, and then you define the actions it will perform. To create the block, go to the **More Blocks** category of the block palette (see Figure 10-4) and click the **Make a Block** button (see Figure 10-5).

The **New Block** window opens (see Figure 10-6) where you can type a name for the custom block, such as **Jump**, and select any additional options that you require (see Figure 10-7). Click **Options** to view your choices. The choices are a number input field **Jump number**, a string input field **Jump string**, or a Boolean input field **Jump boolean**, as well as a descriptive label, as in **Jump number times**. For example, Figure 10-8 creates a custom block called **Jump** with a number input. The input is given a name, which you can use when instructing other blocks in the custom block's definition script how to use the input. By default, the number input is called **number**. You can always modify the name of the input, whether it is a number, string, or Boolean input. If you change your mind about needing an input option and label, you can delete it by selecting the specified field in the define block and clicking the small **x** in the black circle above the specified field that you want to delete. Once you're done, click the **OK** button to create the custom block.

Part 2 Scratch Programming

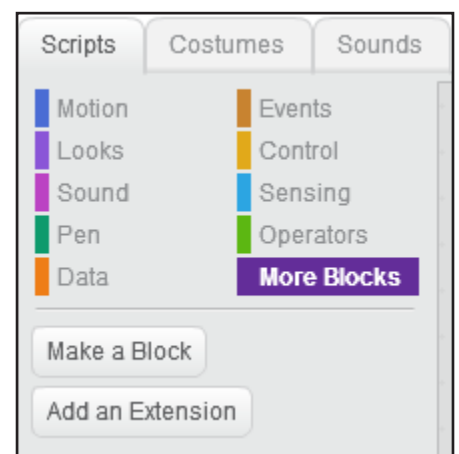


Figure 10-4. The More Blocks category

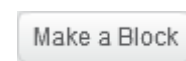


Figure 10-5. Click **Make a Block** to begin creating a custom block



Figure 10-6. The New Block window

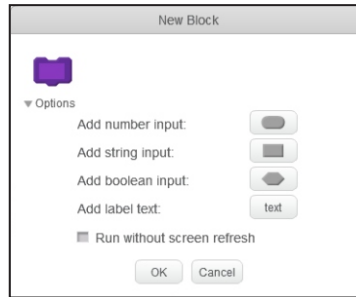


Figure 10-7. Options available for custom blocks

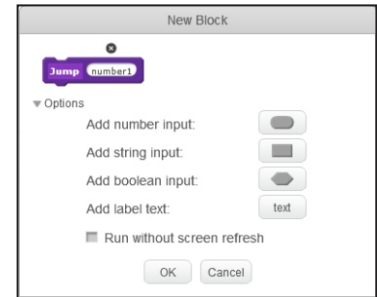


Figure 10-8. Creating a block named Jump with a number input

After you create a custom block, it shows up as a stack type block in the **More Blocks** category below the **Make a Block** button (see Figure 10-9 ). You use this custom block in your scripts. Because it's a stack type block, you can snap other blocks above and below it. In addition, Scratch creates a new hat block and places it in the scripts area (see Figure 10-10 ). You'll use this hat block to define the actions the custom block will perform.

Remember, making a custom block consists of two parts. First, you create the block, and then you define it in the scripts area. To define a custom block, snap code blocks to the hat block in the order you want the custom block to execute them when it's activated. For example, drag the three blocks for the jumping script and snap them below the **define Jump** block, as shown in Figure 10-11 . If you want to use this custom block in other projects, be sure to save your definition script in the backpack. To save your custom block in the backpack, just click below the scripts area and open the backpack by clicking the triangle-shaped icon. Drag and drop the definition script into the backpack. If you want to use this custom block in another project, just drag the definition script from the backpack and drop it in the scripts area. If you click the **More Blocks** category, you will see the custom block there.

Summarizing, you will have two scripts in the scripts area, the definition script (see Figure 10-11 ) that defines the custom block and the main script that uses the custom block (see Figure 10-3 ). If the custom block is defined with any input field, then these inputs are passed from the main script in to the definition script.

Let's create some example projects next.

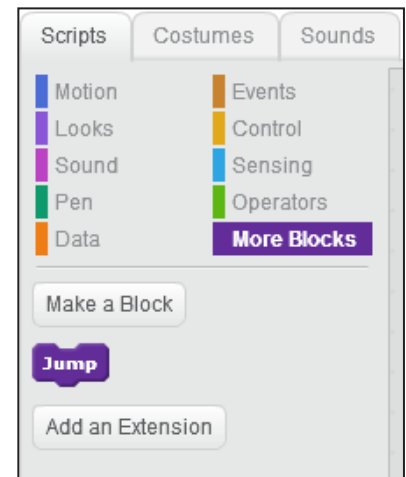


Figure 10-9. A new custom block in the block palette



Figure 10-10. The define block for the new custom block






Figure 10-11. Custom block defined

## Activities

In the following examples, you will create custom blocks, some with input options and some without, and then put them to work. Remember, you always need have the script that defines the block and the actual script(s) that use the custom block together in the same scripts area.

# Activity 10-1: Jump

In this activity, you'll create a single block of code that makes the sprite jump. First, as discussed in the previous section, go to the **More Blocks** category of the block palette and click **Make a Block** to create a new block called **Jump**. The  block appears in the scripts area.

Define the Jump block, by snapping the three blocks shown in Script 10-1 to the  block. The first of the three adds 30 to the current Y coordinate of the sprite to raise its position. The next block pauses the script for 0.5 seconds. The last block of code  subtracts 30 from the current Y coordinate of the sprite to lower its position. So, when you use the block in a script, it makes the sprite move up and then back down.

**Table 10-1** lists the blocks and describes the actions used in this activity.

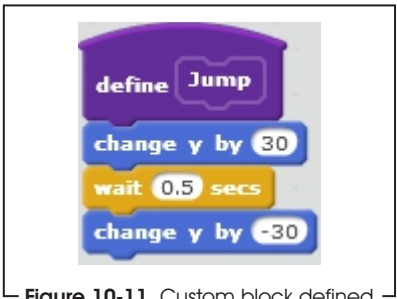








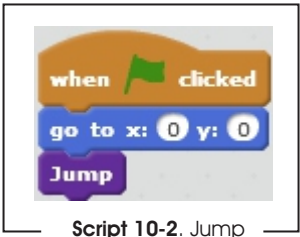
Figure 10-11. Custom block defined

Table 10-1. Code Blocks in Define Jump

Blocks	Actions
	The hat block that defines the block called Jump .
	Add 30 to the current Y coordinate of the sprite and move it to that new Y coordinate.
	The script waits 0.5 seconds. No actions are performed for 0.5 seconds.
	Subtract 30 from the current Y coordinate of the sprite and move it to that new Y coordinate.




Now that the new  block is defined, you are ready to use it in a script. Script 10-2 starts running when the user clicks the green flag. The next block of code moves the sprite to the center of the stage.

The last block of code makes the sprite move up and back down. Notice that creating your own blocks makes the script shorter. By creating the  block, you condensed three blocks of code to just one. **Table 10-2** lists the blocks and describes the actions used in this activity.



Script 10-2. Jump

Table 10-2. Code Blocks in Jump

Blocks	Actions
	Clicking the green flag activates the script. The green flag is the trigger to start the script running.
	Move the sprite to coordinates (X = 0, Y = 0), which is the middle of the stage.
	Execute the Jump block and make the sprite move up and back down.

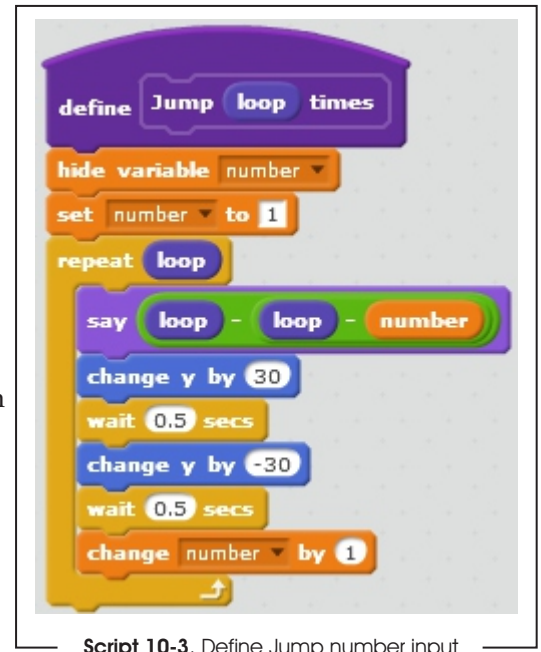
## Activity 10-2: Jump Number Input

This activity builds on the previous activity by creating a new block and adding a number input field to it. When run, this project asks the user how many times the sprite should jump. The sprite then jumps the requested number of times. Before defining the new block (Script 10-3), you need to create a new block called **Jump** and you need to add a number input (loop) and a text label (times) to it. You also need to create a variable called **number**. You don't need to create the custom block from scratch. You can modify the previous custom block from Activity 10-1. Just right-click the custom block in the **More Blocks** category and select **edit** to open the Edit Block window (see Figure 10-12) Select **Options**, click the number input icon to add the number input field. Click the **OK** button to finish.

After you have created this new block, you'll need to define it. The first block after the **define** block hides the reporter window of the variable called **number** from the stage. The next block assigns the value 1 to the **number** variable. Next, there is a **repeat** block, which you want to repeat the number of times specified in the main program and passed to the **Jump** block via its input. To specify this, drag the **loop** block from **define Jump loop times** into the input field of the **repeat** block. The **repeat** block then repeats the sequence of actions within it the amount of times that is passed into the **loop** block from the script that contains the custom block. The first block in the sequence is a combination block, which is created by embedding a number of blocks within each other.

This block creates a speech bubble and displays the result of this equation **loop - loop - number**. Take note that the inner subtraction operator **loop - number** block is executed first. Also note that through each repetition of the sequence of actions, the value of the number input **loop** stays the same, but the value of the **number** variable changes.

This will make the sprite count up. The next block adds 30 to the current Y coordinate of the sprite and moves the sprite to that new position. The next block pauses the script for 0.5 seconds. The next block subtracts 30 from the current Y coordinate of the sprite and moves the sprite to that new position. The last two blocks pause the script for 0.5 seconds and add 1 to the current value of the **number** variable. **Table 10-3** lists the blocks and describes the actions used in this activity.



### OUTPUT

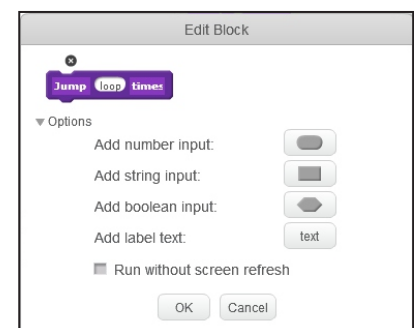
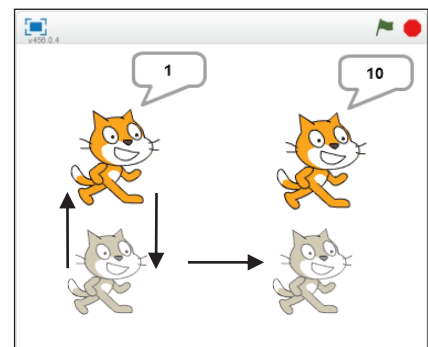


















Table 10-3. Code Blocks in Define Jump Number Input

Blocks	Actions
	The hat block that defines the block called Jump times .
	Hide the specified variable's ( number ) monitor from the stage area.
	Set the current value of the number variable to 1.
	Repeat the actions represented by the blocks within this block a certain number of times.
	The block that represents the inputted number.
	The sprite gets a speech bubble and displays the specified text.
	Subtract one value from another and report the result.
	The block that represents the inputted number.
	Subtract one value from another and report the result.
	The block that represents the inputted number.
	Hold and report the current value of the number variable.
	Add 30 to the current value of the Y coordinate and move the sprite to that new Y coordinate.
	The script waits 0.5 seconds. No actions are performed for 0.5 seconds.
	Subtract 30 to the current value of the Y coordinate and move the sprite to that new Y coordinate.
	The script waits 0.5 seconds. No actions are performed for 0.5 seconds.
	Add 1 to the current value of the number variable.



Script 10-4. Jump number input

Now that you have defined the new block, it's time to create a script that uses the new block. Script 10-4 starts running when the user clicks the green flag. The next block of code moves the sprite to the center of the stage. The next block creates a speech bubble that displays the text How many times do you want me to jump up and down? , opens a user input field, and waits for the user's input.



The next block creates a speech bubble for 2 seconds and displays OK, I will jump up and down , the value of the user input held in `answer`, and times. The last block calls up the custom block `Jump times` and passes the value in `answer` to it. The custom block holds the value from `answer` in `loop` and uses it to instruct the sprite to jump the number of times that the user requested. You know now how to create a custom block with a number input. Do you remember the backpack? You can drag and drop the `define Jump loop times` block's definition script into the backpack and use it in other projects. **Table 10-4** lists the blocks and describes the actions used in this activity.

OUTPUT

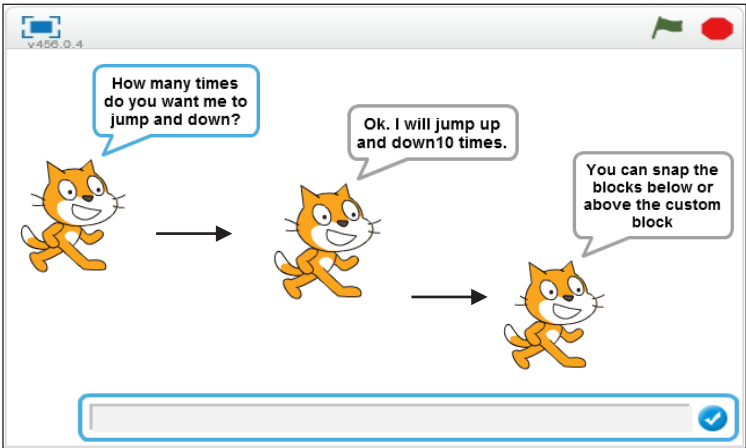










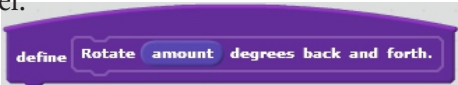






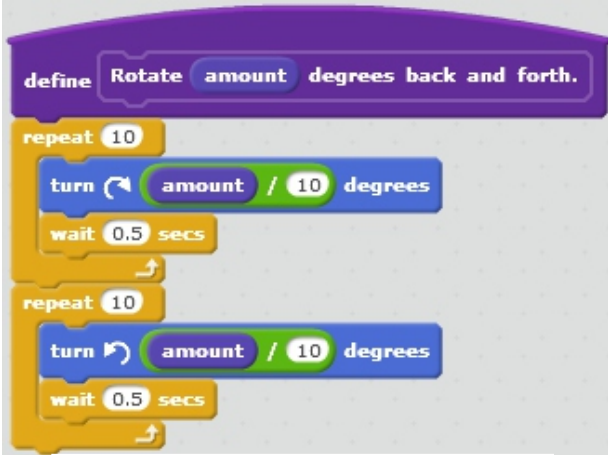
Table 10-4. Code Blocks in Jump Number Input

Blocks	Actions
	Clicking the green flag activates the script. The green flag is the trigger to start the script running.
	Move the sprite to coordinates (X = 0, Y = 0), which is the middle of the stage.
	The sprite gets a speech bubble that displays How many times do you want me to jump up and down? , opens a user input field, and waits for user input.
	The sprite gets a speech bubble that displays the specified text or value for 2 seconds.
	Join the two values that have been specified in the block and report the result.
	Join the two values that have been specified in the block and report the result.
	Hold and report the current user input value.
	Execute the Jump times block and make the sprite jump the specified number of times.
	Hold and report the current user input value.
	The sprite gets a speech bubble that displays You can snap blocks below or above the custom block for 2 seconds.

# Activity 10-3: Rotate Number Input

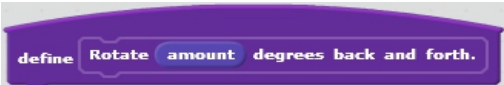










The cat has had enough jumping. Instead, take it for a spin with a custom block that makes the sprite rotate a number of degrees that is specified by the user. This is another example that shows how creating a custom block with a number input can reduce many blocks of code to just one custom block. As before, the first step is to create the new block; this time, call it **Rotate**, name the number input **amount**, and add **degrees back and forth.** as the label.

Underneath the  block (Script 10-5), snap two  blocks, one beneath another, setting each to repeat the sequence of actions within it 10 times. The first block within the first of these **repeat** blocks is a combination block that rotates the sprite clockwise a certain number of degrees, which is determined by the embedded operator block that divides  by 10. The last block within the first loop pauses the script for 0.5 seconds. The sequence of actions in the second  block is almost identical; the only difference is it rotates the sprite counterclockwise.  block is defined to rotate the sprite clockwise a certain number of degrees, and then counterclockwise back to its original position. **Table 10-5** lists the blocks and describes the actions used in this activity.



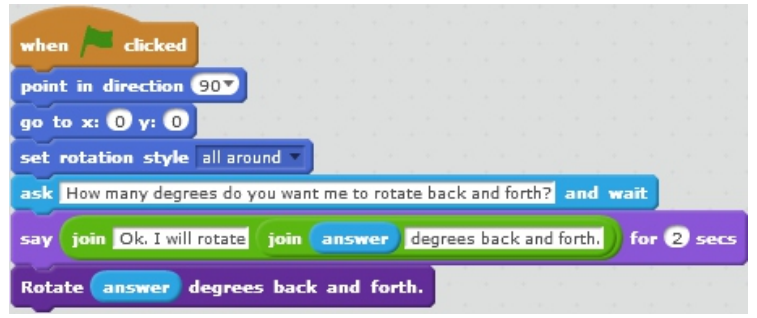
Script 10-5. Define Rotate number input

Table 10-5. Code Blocks in Define Rotate Number Input

Blocks	Actions
	The hat block that defines the block called Rotate degrees back and forth.
	Repeat the actions represented by the blocks within this block ten times.
	Turn the sprite clockwise a specified number of degrees.
	Divide one value by another and report the result.
	The block that represents the inputted amount.
	The script waits 0.5 seconds. No actions are performed for 0.5 seconds.
	Repeat the actions represented by the blocks within this block ten times.
	Turn the sprite counterclockwise a specified number of degrees.
	Divide one value by another and report the result.
	The block that represents the inputted amount.
	The script waits 0.5 seconds. No actions are performed for 0.5 seconds.



Let's create a script that uses this new custom block that makes the sprite rotate back and forth. Script 10-6 activates when the user clicks the green flag. The next three blocks of code, make the sprite face to the right, move the sprite to the center of the stage, and set its rotation style to **all around**. The next block creates a speech bubble that displays the text `How many degrees do you want me to rotate back and forth?`, opens a user input field, and waits for the user's input. The next block creates a speech bubble for 2 seconds and displays `OK, I will rotate`, the value of the user input, and the text `degrees back and forth.`. The last block calls the new custom block and passes it the value in `answer` to instruct the sprite to rotate back and forth the amount of degrees requested by the user. Table 10-6 lists the blocks and describes the actions used in this activity.



```
when green flag clicked
  point in direction 90
  go to x: 0 y: 0
  set rotation style all around
  ask How many degrees do you want me to rotate back and forth? and wait
  say join Ok. I will rotate join answer degrees back and forth. for 2 secs
  Rotate answer degrees back and forth.
```

Script 10-6. Rotate number input

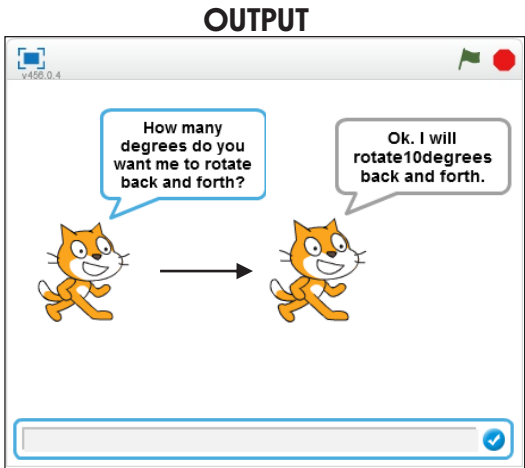






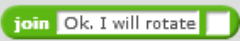






Table 10-6. Code Blocks in Rotate Number Input

Blocks	Actions
	Clicking the green flag activates the script. The green flag is the trigger to start the script running.
	Point the direction of the sprite to the right.
	Move the sprite to coordinates (X = 0, Y = 0), which is the middle of the stage.
	Set the rotation style of the sprite, so it can rotate all around its axis.
	The sprite gets a speech bubble that displays <code>How many degrees do you want me to rotate back and forth?</code> , opens a user input field, and waits for user input.
	The sprite gets a speech bubble that displays the specified text or value for 2 seconds.
	Join the two values that have been specified in the block and report the result.
	Join the two values that have been specified in the block and report the result.
	Hold and report the current user input value.
	Execute the <b>Rotate degrees back and forth</b> block and make the sprite rotate a specified number of degrees back and forth.
	Hold and report the current user input value.

# Activity 10-4: Jump String Input

Creating and using a custom block with a string input is just as easy as adding one with a number input. In this activity, the user is asked whether the sprite should jump. Depending whether the user answers, the sprite will jump or not. Start by creating the new custom block called **Jump?**. Add a string input by clicking the appropriate button in the Options area. You can keep the default name of string1 for the input (see Figure 10-13). After you click the OK button to create the custom block, the

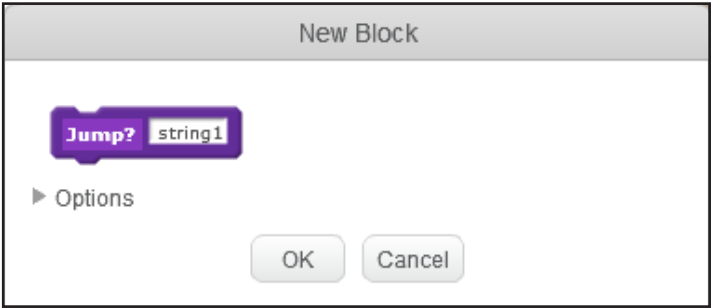


Figure 10-13. Jump string input

block should appear in the scripts area. Now snap the blocks shown in Script 10-7 to it to define your new block. The first block in the definition is an If/Then/Else conditional statement that first evaluates whether the user input is equal to the string yes. To set up this conditional statement, drag the string1 block, which will hold the user input passed into it.

From the define Jump? string1 block, drop in at the left field of the = yes block, and enter yes in the right field. Notice that there will still be a copy of string1 in the define Jump? string1 block. Finally, embed the whole thing in the if then else block. If the condition evaluates true, then Scratch executes the sequence of blocks in the Then section. The first block creates a speech bubble for 2 seconds and displays OK, I'll jump!. The next block adds 30 to the current Y coordinate of the sprite and moves the sprite to that position.

The third block pauses the script for 0.5 seconds. The last block in the sequence subtracts 30 from the current Y coordinate of the sprite and moves the sprite to that position. If the condition evaluates to false (the user input something other than yes), Scratch skips to the block in the Else section and creates a speech bubble for 2 seconds that displays OK, I won't jump!.

Table 10-7 lists the blocks and describes the actions used in this activity.





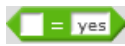
Script 10-7. Define Jump string input

## OUTPUT



Table 10-7. Code Blocks in Jump Define String Input

Blocks	Actions
	The hat block that defines the block called Jump? .
	Check if the condition is true. If the condition is true, execute the actions within it, before the word else . If the condition is false, execute the actions after the word else within the block.



If the value on the left side is equal to yes, then this condition is true; otherwise, the condition is false.

The block that represents the user input (string).

The sprite gets a speech bubble that displays OK, I'll jump! for 2 seconds.

Add 30 to the current value of the Y coordinate and move the sprite to that new Y coordinate.

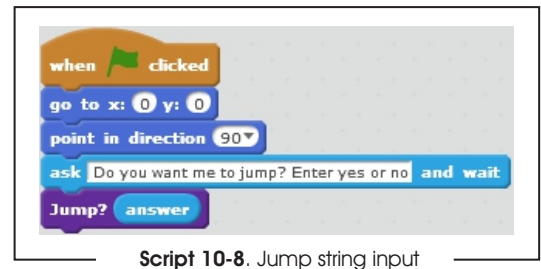
The script waits 0.5 seconds. No actions are performed for 0.5 seconds.

Subtract 30 to the current value of the Y coordinate and move the sprite to that new Y coordinate.

The sprite gets a speech bubble that displays OK, I won't jump! for 2 seconds.

With the custom block defined, you can use it in a script. Script 10-8 starts running when the user clicks the green flag. The next two blocks move the sprite to the middle of the stage and make it face to the right. The next block creates a speech bubble that displays the text Do you want me to jump? Enter yes or no , opens a user input field, and waits for the user's input. The next block calls the custom block by passing the user input held in `answer` to it.

**Table 10-8** lists the blocks and describes the actions used in this activity.



### OUTPUT



**Table 10-8. Code Blocks in Jump String Input**

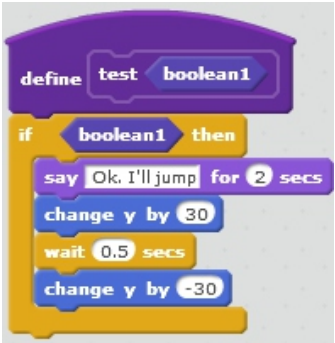
Blocks	Actions
	Clicking the green flag activates the script. The green flag is the trigger to start the script running.
	Move the sprite to coordinates (X = 0, Y = 0), which is the middle of the stage.
	Point the direction of the sprite to the right.
	The sprite gets a speech bubble that displays Do you want me to jump? Enter yes or no , opens a user input field, and waits for user input.
	Execute the Jump? block and make the sprite jump if the condition is true.
	Hold and report the current user input value.

# Activity 10-5: Jump Boolean Input

The last option to try is Boolean input for a custom block. Create a custom block called **test** and add a Boolean input to it. You can use the default name of `boolean1` for the value passed into the block (see Figure 10-14). Click **OK** to create the custom block and add the hat block `define test boolean1` to the scripts area. To define your new custom block, snap together the blocks of code shown in Script 10-9. The first block is an `if` block that receives the result of a condition that is passed into it from the main script. Currently, the result of this condition is represented by the `boolean1` block. The result is a value that is either true or false. Simply drag this block from the `define test boolean1` block and drop it in the If/Then conditional statement. If `boolean1` is true, then Scratch performs the sequence of actions within the repeat block. First, it creates a speech bubble that displays `OK, I'll jump.` for 2 seconds. The next blocks add 30 to the current Y coordinate of the sprite to move it to the new position, pause the script for 0.5 seconds, and subtract 30 from the current Y coordinate of the sprite to move it to that new position. If the condition is false, nothing happens.










Figure 10-14. Boolean input test



Script 10-9. Define Jump Boolean input

Table 10-9 lists the blocks and describes the actions used in this activity.

Table 10-9. Code Blocks in Define Jump Boolean Input

Blocks	Actions
	The hat block that defines the block called <code>test</code> .
	Check if the condition is true. If the condition is true, execute the actions within it. If the condition is false, skip to the next block.
	The block that represents the Boolean condition.
	The sprite gets a speech bubble that displays <code>OK, I'll jump.</code> for 2 seconds.
	Add 30 to the current value of the Y coordinate and move the sprite to that new Y coordinate.
	The script waits 0.5 seconds. No actions are performed for 0.5 seconds.
	Subtract 30 to the current value of the Y coordinate and move the sprite to that new Y coordinate.

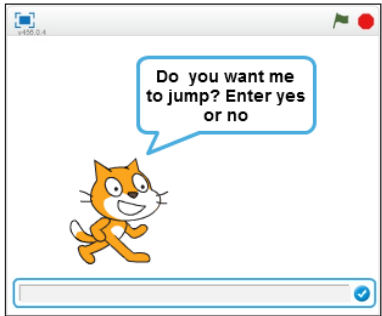
Let’s use the newly defined custom block in a script. Script 10-10 starts running when the user clicks the green flag. The next two blocks move the sprite to the middle of the stage and make the sprite face to the right. The next block creates a speech bubble that displays the text `Do you want me to jump? Enter yes or no`, opens a user input field, and waits for the user’s input. The next block passes the result of the condition (true or false) to the custom block. The custom block executes the actions defined for it if the condition is true. If the condition is false, nothing happens and the script ends. Save this custom block in the backpack if you want to use it later in other future projects.

**Table 10-10** lists the blocks and describes the actions used in this activity.




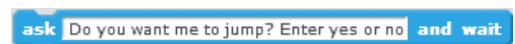





Script 10-10. Jump Boolean input

**OUTPUT**



**Table 10-10. Code Blocks in Jump Boolean Input**

Blocks	Actions
	Clicking the green flag activates the script. The green flag is the trigger to start the script running.
	Move the sprite to coordinates (X = 0, Y = 0), which is the middle of the stage.
	Point the direction of the sprite to the right.
	The sprite gets a speech bubble that displays <code>Do you want me to jump? Enter yes or no</code> , opens a user input field, and waits for user input.
	Execute the <code>test</code> block and test if the condition is true. If the condition is true, execute the actions defined in the custom block.
	If the value on the left side is equal to yes, then this condition is true; otherwise, the condition is false.
	Hold and report the current user input value.

**Summary**

Well done. You now know how to create custom blocks in Scratch. Custom blocks are handy, because you can save them in the backpack and reuse them in other projects. Not only do they streamline your scripts by condensing multiple blocks into a single block, but they also save you time by enabling you to quickly add frequently used sequences as a single block. After successfully running the scripts in this chapter’s examples, and understanding how they work, you can modify the scripts in the examples and create your own project. This chapter also marks the end of your Scratch lessons, but don’t let it stop you from learning. Keep practicing and creating your own projects. Whether you want to continue with other computer programming languages or your interests lie somewhere else, learning Scratch was worth it. Besides learning about how to program a computer, you learned a new way of thinking when it comes to problem solving. You learned how to break down a larger problem and solve it in small parts. You also learned how to solve problem sequentially.

Congratulations, and have fun creating Scratch projects.

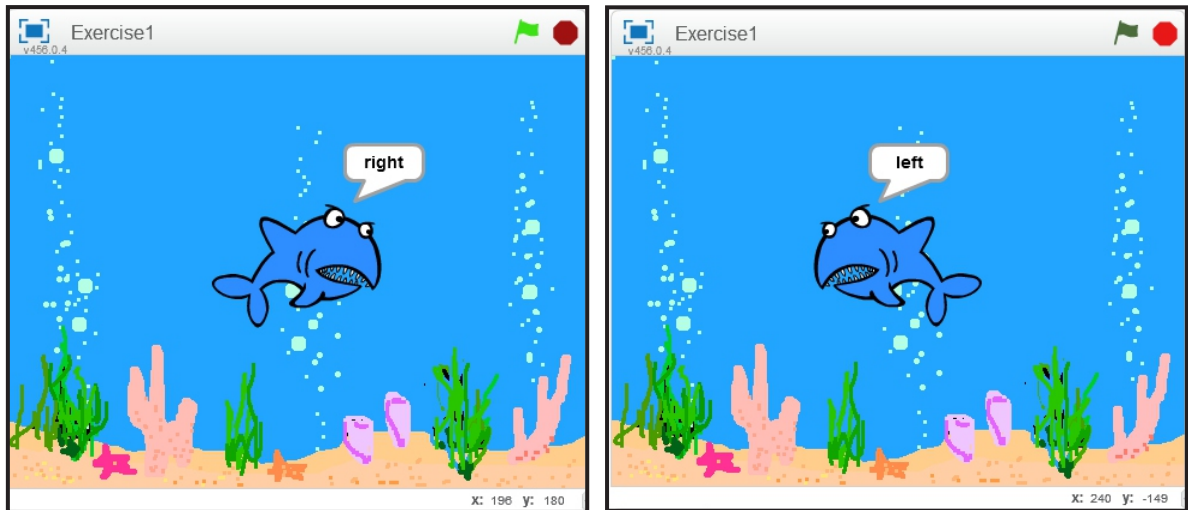


# Snap Script

a Short hands-on activity

1. Create a new block that makes the sprite face left and right. Create a script and use this new block.

## Output:



2. Create a new block that makes the sprite move along the stage and change costumes while moving. Create a script and use this new block.

## Output:

