Text Input

Let's use both the changeable and placeholder qualities of variables to make a program in which Scratch interacts with us. We've already used text output in Scratch with the `say [...]` blocks located in the `Looks` tab. However, we can also ask for text input as well. Many of the input-oriented blocks are located in the `Sensing` tab.

Why do you think that the `Sensing` tab contains blocks for input? How do you receive input from the world around you?

In the following activity, we are going to work with the following blocks:

![Blocks](image)

As you can see, the variable `answer` is changeable. It has a different value at different times, depending on its context—when it is being viewed and how it was updated.

What about the placeholder quality of variables? How can we leverage that in our programs? We can use `answer` as a placeholder for text input in our program. This way, the program will use whatever is entered for `answer` without knowing ahead of time what that is. This is better illustrated with an example. *Create the following program. Remember that blocks are color-coded according to their tabs.*

![Program](image)

Try It Out!

Load your Scratch program from *User Input and Interaction, Show Me Your State*, and make the following edits:

1. Drag the `ask [What's your name?] and wait` block to the Scripts pane. Click on it. Scratch will ask your name and wait for you to respond.
2. If you'd like to see your answer, check the box next to `answer` as is done in the image at the top of the page. The variable `answer` contains whatever you type when Scratch asks you a question using the `ask [...]` block.
3. Add another `ask [...]` block immediately after the “What's your name?” block.
4. Change the question text, and re-execute the program.
5. Now execute the program. What happens and why?
6. Drag the `answer` block to your script again so that it reads `say answer for 2 secs`

Does this do what you would expect to? Why or why not? What would you need to do to make it
work properly?

7. Personalize it in at least three other ways.

8. Provide documentation for your program (describe what it does) as the Instructions. Be sure to describe how your program is original.

When you are satisfied with your work, submit a link to your program or the program itself.