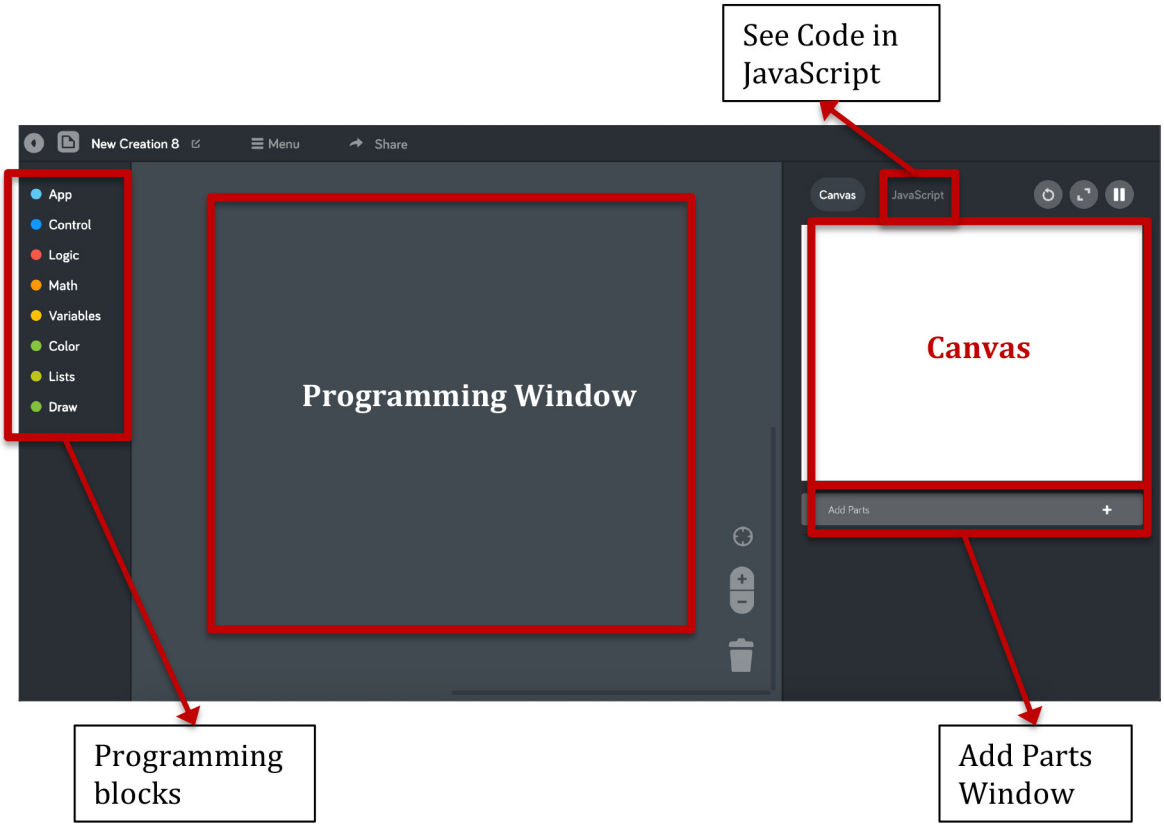
The Sound of CS

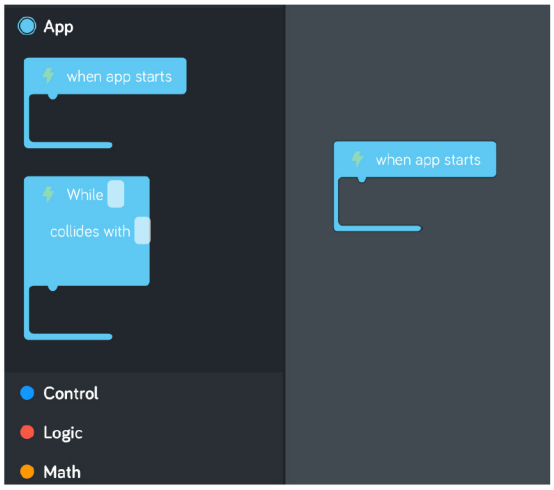
Now that students have had the opportunity to learn about what it is like to live a day in the life of Poppie Simmonds, a computer science student at the University of Birmingham, students will now learn how to code a story about the future computer scientist they know best, themself!

Begin by having students navigate to World.kano.me. Note that it is best if students use Google Chrome as their web browser. If you would like students to be able to save their work easily, or to avoid students losing their work, it is best to have students create an account using the sign-up option in the upper right-hand corner of the window. Otherwise, students can just begin by clicking the create button in the upper-center of the window.

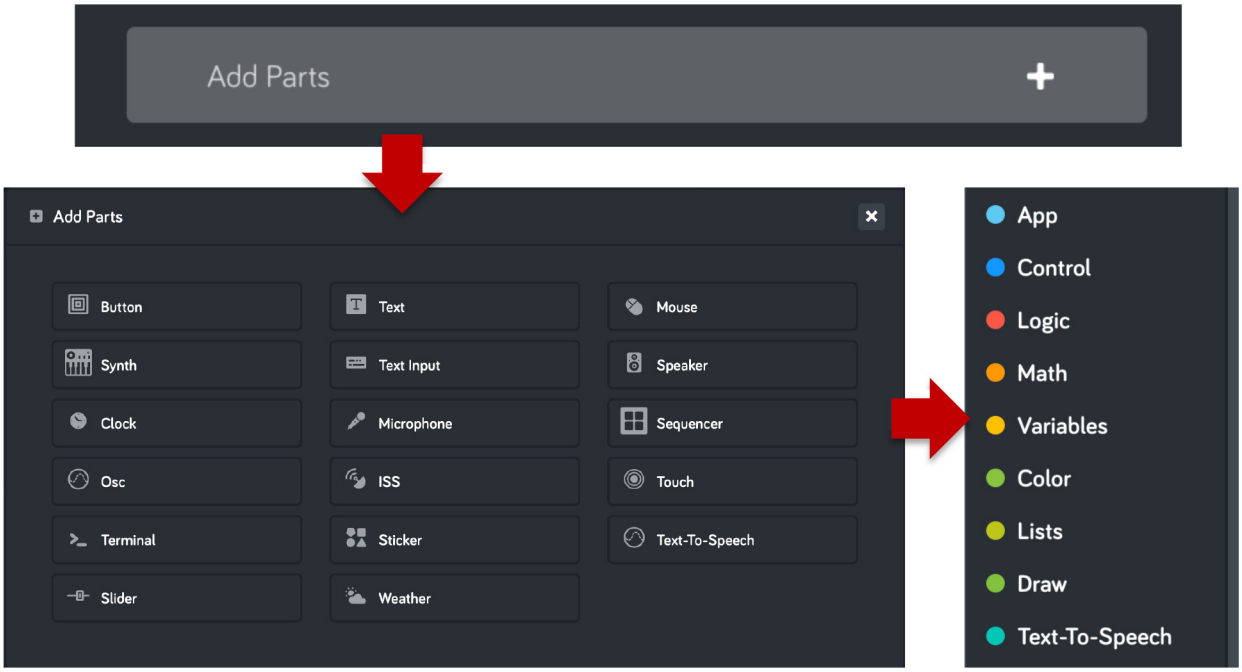
Within the create window, help students to get familiar with the different components of the window described here. On the left-hand side of the screen are all the different coding blocks that we will have access to when creating our stories. In the center of the screen is the window where we will drag all of our coding objects to create our code. On the right-hand side of the window is the canvas where objects that we have coded will appear, as well as the ability to add new objects to code and the ability to change our block coding to JavaScript to see what is happening in the background. Seeing our code in JavaScript will help students to see how their block-based coding correlates to real-world text-based programming.



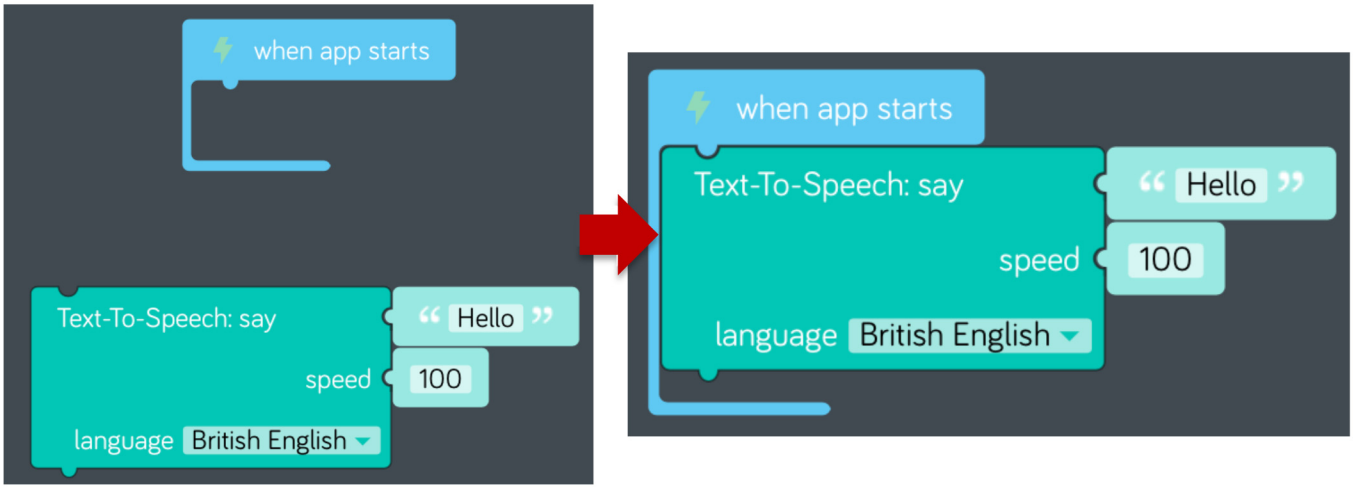
We will begin our programming by selecting the app category and dragging into our window a single when app starts block. This block signifies that the code we write will now be the first code to run when we begin playing our story.



Next, we will need to add an object to our project so that our story can be narrated. Under “Add Parts” on the right-hand side of the screen, select add “Text-To-Speech”.



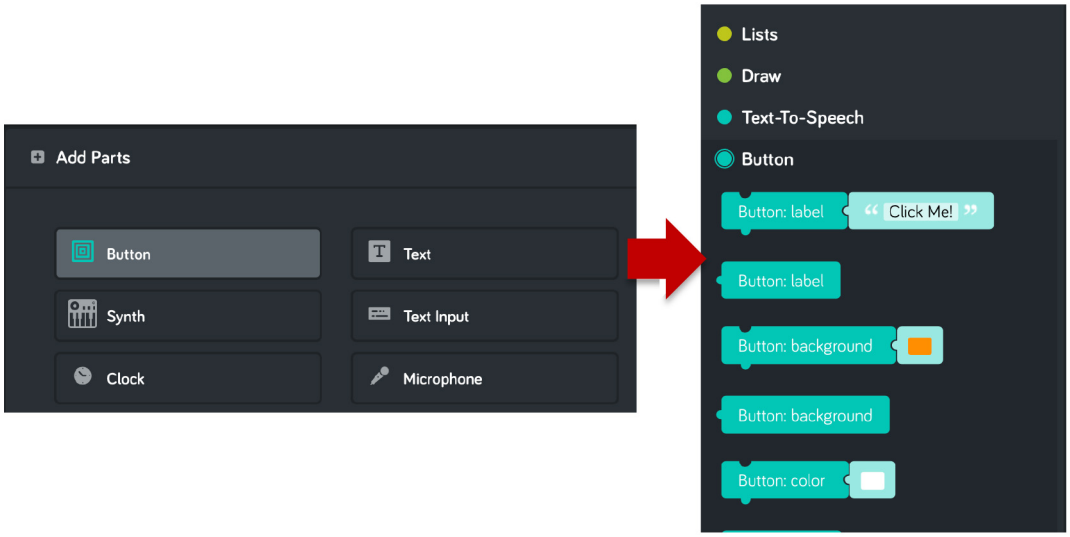
Students should notice that they now have more coding blocks on the left-hand side of their screen. Using those blocks, we will grab a **Text-To-Speech: say** block and add that block to our code. Note that it is not enough to just drag the block into the window. We will need to connect the block to our **when app starts** block. These connections form our code! Note that code reads from top to bottom so long as it is connected.



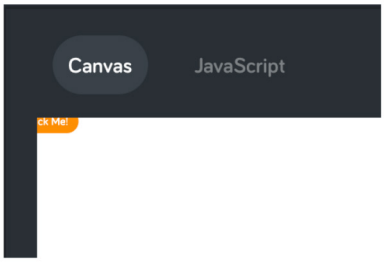
When we connect our block, we will hear it say “Hello” at speed 100 with a British English voice. Note that *say, speed, and language* are all different variables that we can alter to change how our story is told. For this example, we will only change the say variable, but students may feel free to change the other variables as they wish.

Now that we have our block, we can alter what our first block says. Sticking with our “Day in the Life” story, we know that almost everybody begins their day by waking up. Change the **Text-To-Speed: say** text to “Did you ever hear the story of young [student’s name]? A determined computer scientist, they began every day by waking from a long night’s rest.”

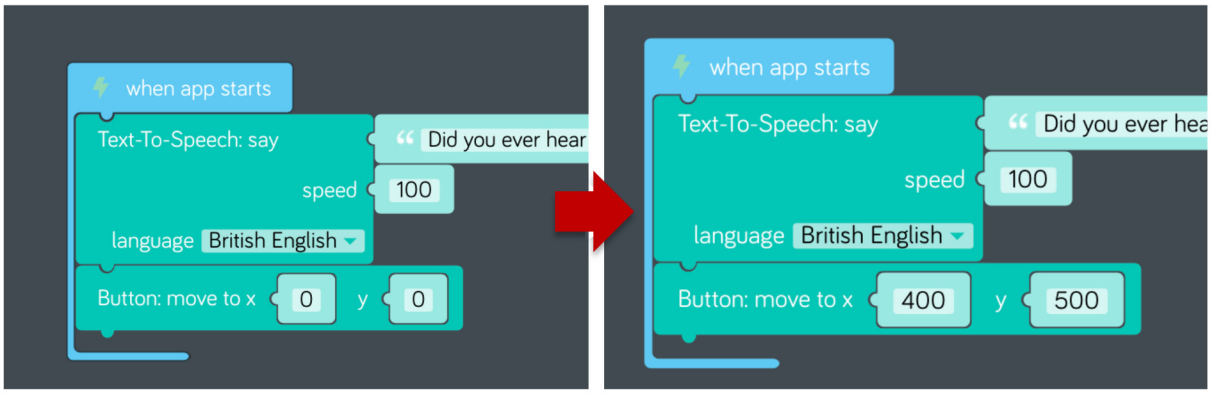
Now that we have a beginning to our story, we can continue in one of two ways: (1) we can drag in a new **Text-To-Speech: say** block and have the narrator simply continue through our story, or (2) we can add another object and allow our listener to continue the story at their own pace. For this example, we will go with the second option. Under the "Add Parts" menu, we can select to add a button to our canvas.



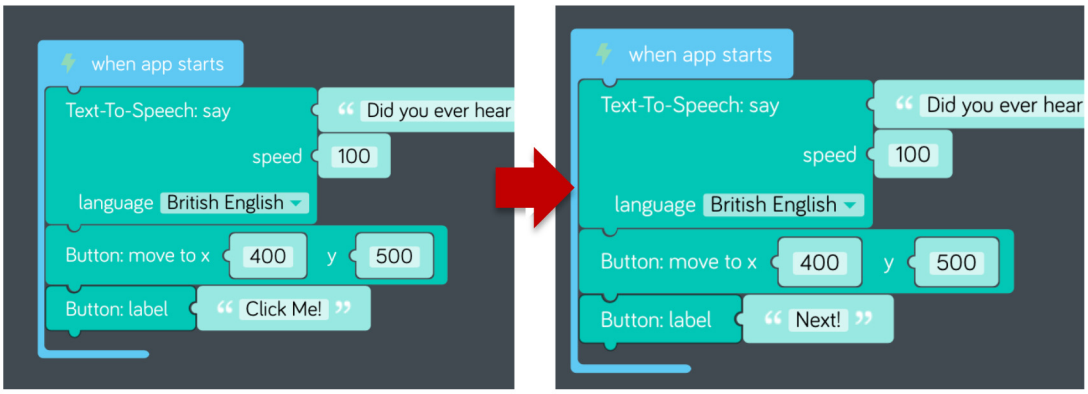
Notice that this adds more programming blocks to the left-hand side of the window just like before. We can also see on our canvas that we have an object in the upper left-hand corner. It is difficult to see, so let's move this button elsewhere before we make any changes.



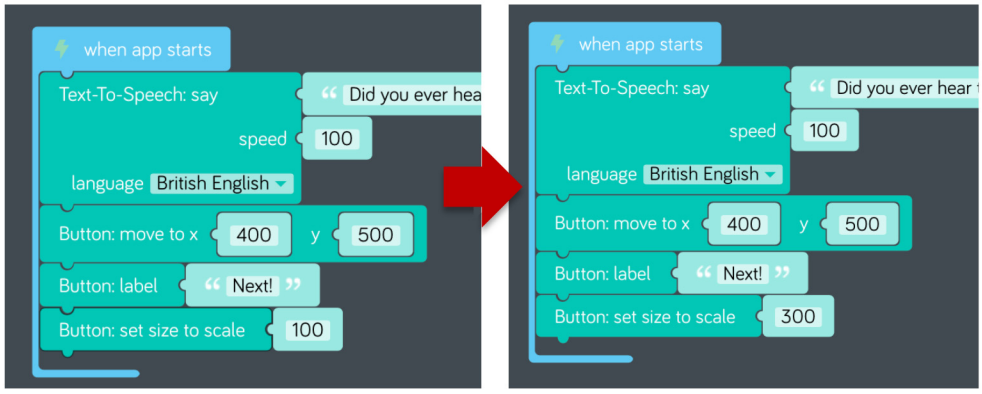
To move our button, we can use code. Under our button blocks, we can scroll down until we find **Button: move to x: 0 y: 0**. Notice how our button does not move when we add this new block. This is because the button is already at space (0, 0). We can move our button to the center of the window and the bottom by changing our x from 0 to 400 and our y from 0 to 500.



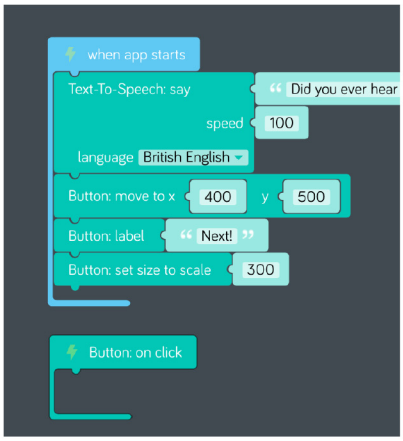
Next, we can change the label of our button from “Click Me!” to something a little more appropriate for our story. Using the coding blocks again, we can use a **Button: label** block to update the text to say “Next!”



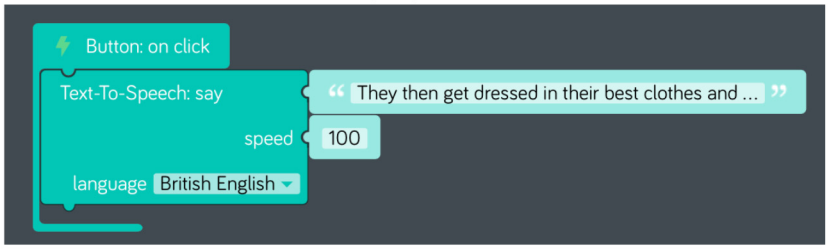
Now that our button more accurately reflects what we want to say, we notice that the button is really small. We can adjust the button size as well. Using a **Button: set size to scale** block, we can change our scale to 300 to make our button much bigger.



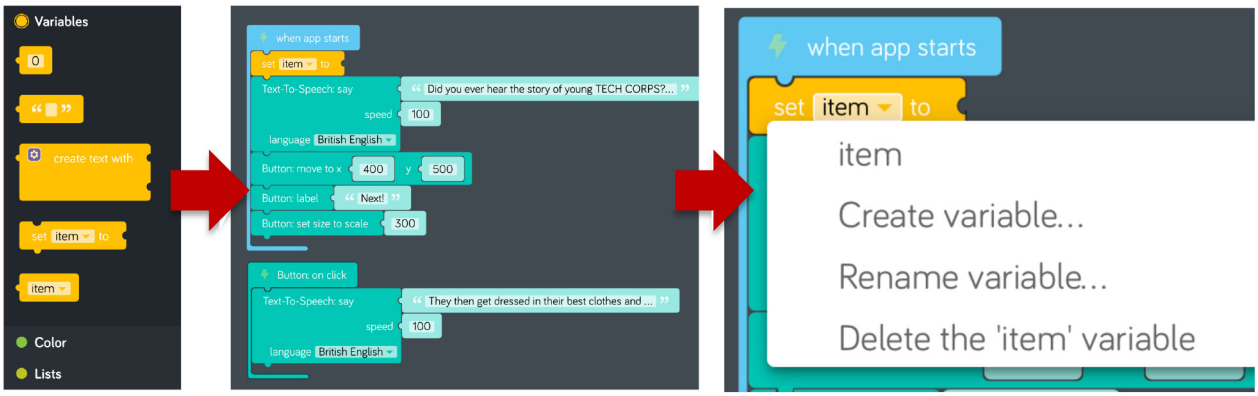
Our button is finally finished! Now all we need to do is code what happens when we click it. At the bottom of our button blocks, we have a **Button: on click** block. We can drag this block into our program as a standalone block. This block does not need to be connected to the rest of the program because it is dependent on an event occurring (the click of the button), rather than a continuation of the code we have already written.

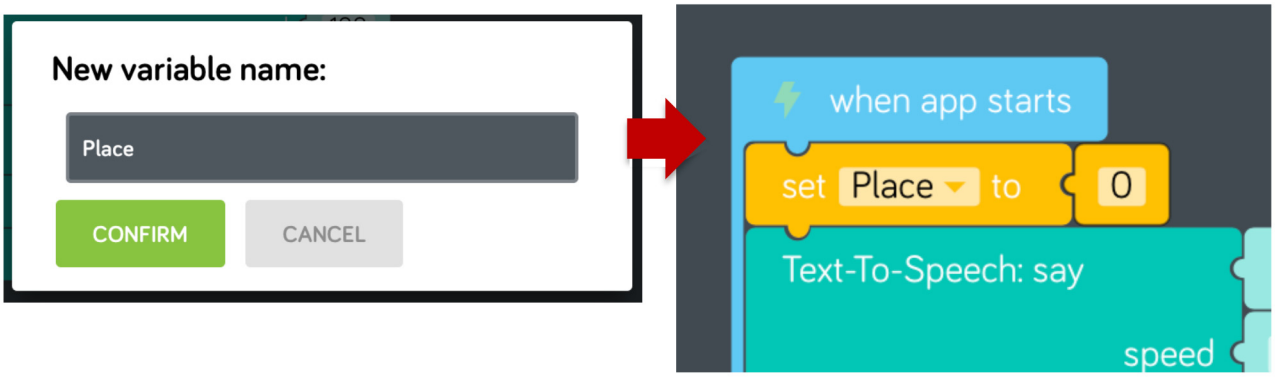


Now we can grab another **Text-To-Speech: say** block and write the second line of our story. Change the "Hello" text to say "They then get dressed in their best clothes and head to class." Have students click the "Restart" button and listen to their story. When the first line of narration ends, have students click the "Next!" button once to tell the next line. If students click the button multiple times, they may crash their browser and lose their work, or the line may repeat for each click of the button.

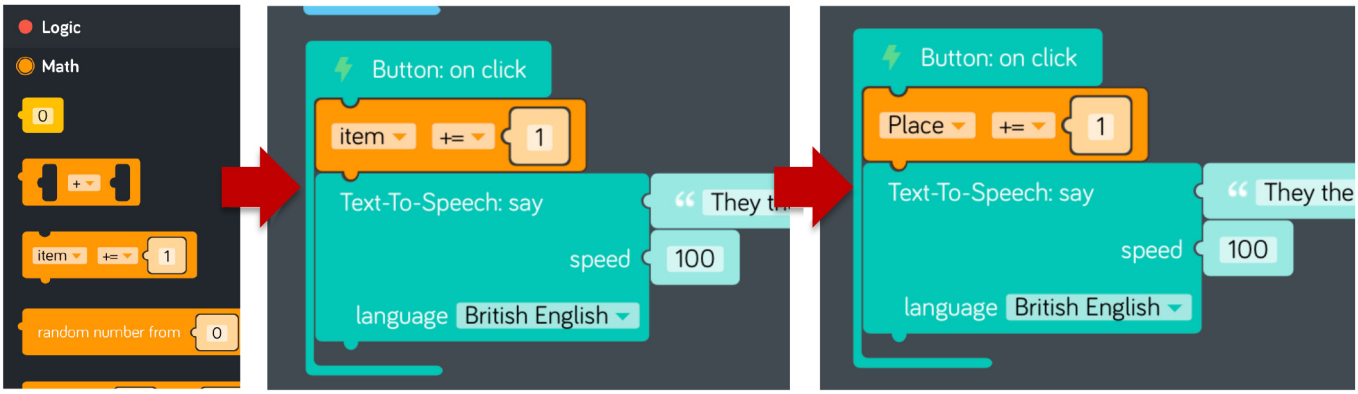


At this point, if we were to create another **Button: on click** block, our code would not know which block to use. In the case of the software, Kano will choose to use the coding block that is listed higher up, but we may not want this. In that case, we can fix this by adding what computer scientists call a variable. A variable is a placeholder for a value, very similar to the way it is in math. To create our variable, we can grab the **set item to** block under the "Variables" category and drag it into our code. We will want our variable to be initialized at the start of the program. Before our variable is ready to be used, we will want to give our variable a name and a starting value. By clicking the dropdown next to "item", we can select "Create variable…" and name our variable "Place". Finally, we can drag a 0 block from our "Variables" category and attach it to our **set Place to** block.

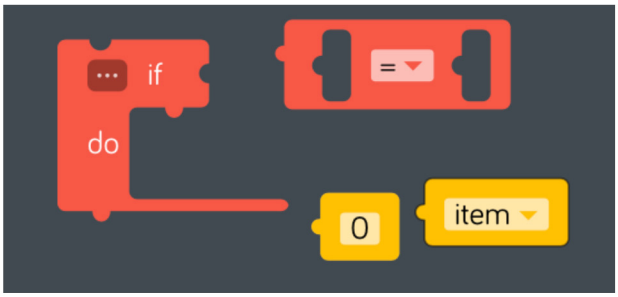




Now that we have this variable, we can use it to keep track of where we are in our story. We can increase the value of Place by 1 every time our button is clicked. This will ensure that our narrator reads each line only once and in the correct order. In our **Button: on click** function, we can drag an **item+=1** block from the "Math" section and change the variable from "item" to "Place".

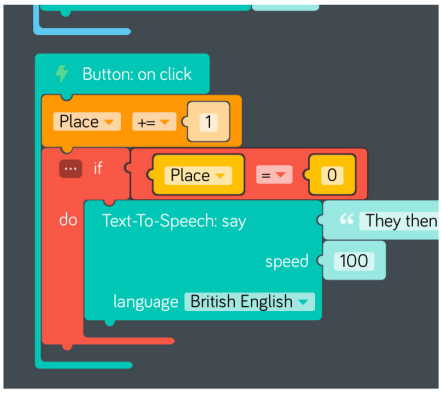


Finally, we can add a conditional to check what the value of "Place" is and read the appropriate line. To add our conditional, we will need many different blocks. We will need an **if do** block from the "Logic" category, a **blank = blank** block from the "Logic" category, an **item** block from the "Variables" category, and a **0** block from the "Math" or "Variables" category.

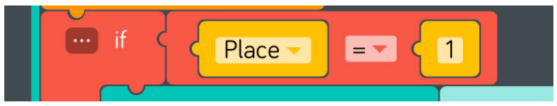


We will need to change our "item" block to "Place" by clicking the dropdown and selecting "Place". We will then insert our "Place" block into the blank space in the blank = blank block and insert our number 0 block into the other blank space. Note that in computer science, the variable "Place" and the number 0 could be swapped to either side and the program will still run the same.

To attach the remaining blocks, we will need to move our **Text-To-Speech: say** block inside our **if** block and then insert our **if** block back into the program where our **Text-To-Speech: say** block was. We can finish the conditional by inserting the **Place = 0** block where it says "if". The statement now reads **if Place = 0, do Text-To-Speech: say "They then get dressed in their best clothes and head to class."**

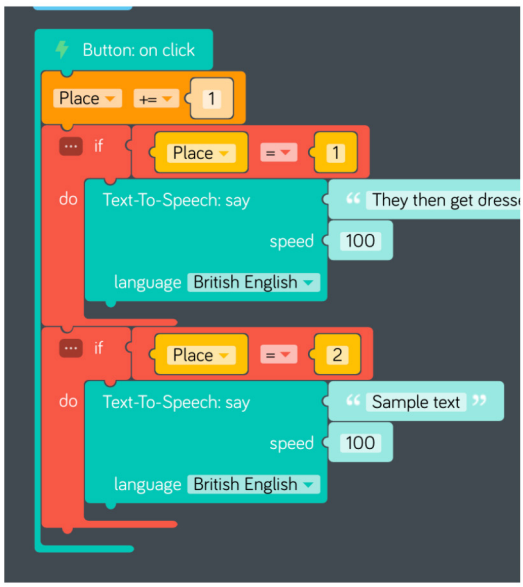


At this point, students may test their code. What do they notice when they click the button? It seems that the button is no longer working, but the value of "Place" increments to 1 when we click the button, and we are only telling the program to do something when the button is clicked and "Place" equals 0. We can fix this by changing the **0** value to **1**.



Have students test their code again. Everything should be working as expected now!

Challenge students to continue their story using the things that they have learned about coding. Students will need to add additional if statements to match the one they have already entered. They can do so by either dragging each of the individual blocks over, or they may simply right click the conditional, click duplicate, and then drag the code to be below the prior if statement. The code should look something like this:



Students may feel free to be creative with their stories. They can tell a factual account of their day, or they may tell a work of fiction as they dream of their perfect computer science career. Students may also feel free to explore Kano a little deeper on their own to illustrate their story on their canvas. They can change the background and draw shapes using the draw blocks, or they can add a sticker under the "Add Parts" option. Students may also choose to add text to allow users to read the story as it is narrated. Note that when a part is added, it shows up in the canvas regardless of whether the student has coded any blocks from that category. To remove an object from the canvas, students can simply click the X next to the item in the parts list.

This activity can be shortened or lengthened to account for differences in time. If you would like, you may also choose to have students present to the group on their story.