06 - More p5.js Variables

Learning About More p5.js Variables

WE'LL COVER THE FOLLOWING

Intro

In the previous chapter, we learned about the p5.js **frameCount** variable that provides us with a number that represents the number of times the **draw** function is called. There are bunch of other highly useful variables that we could be using in p5.js. We will be learning about a few more in this chapter.

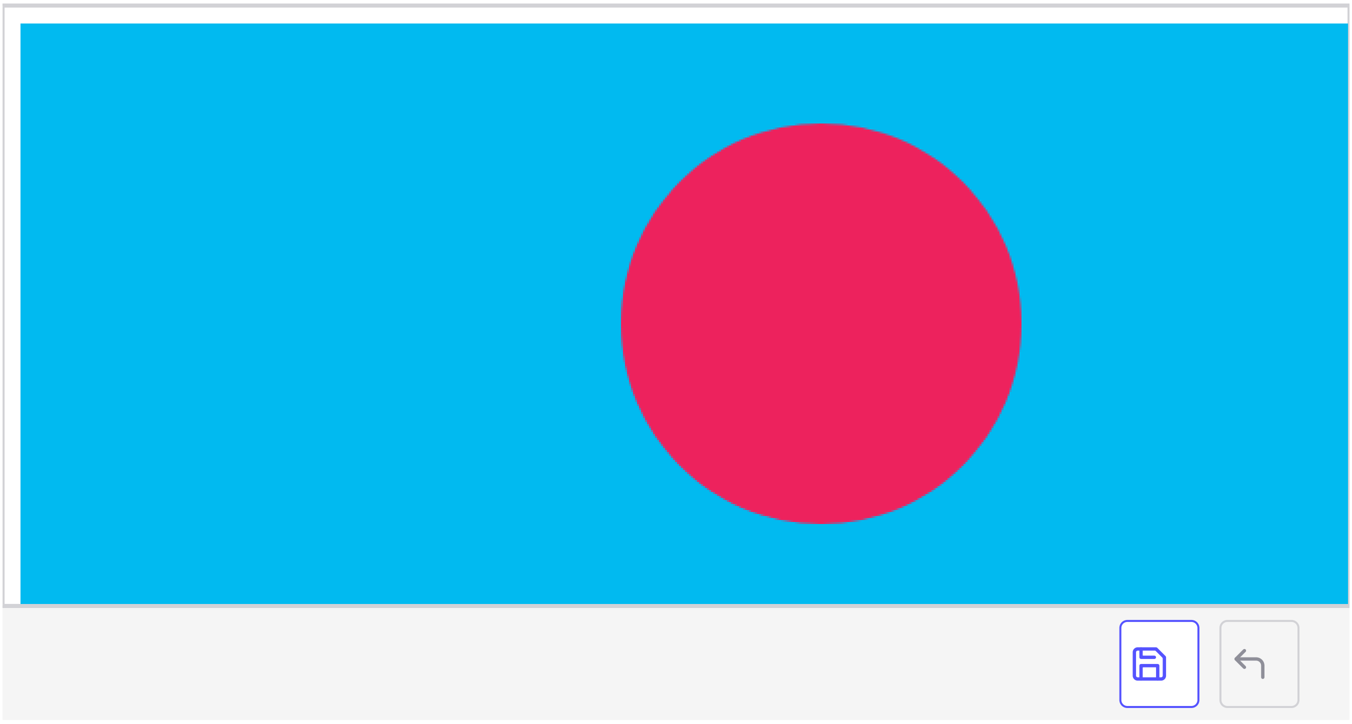
mouseIsPressed

mouseIsPressed is the first p5.js variable that we will see, that will allow us to add some interactivity to our programs. **mouseIsPressed** is a p5.js variable that assumes the value **true** when the mouse is clicked on the canvas area and **false** for every other time. Let's alter our example from the previous chapter to quickly see how we can use this variable.

Output

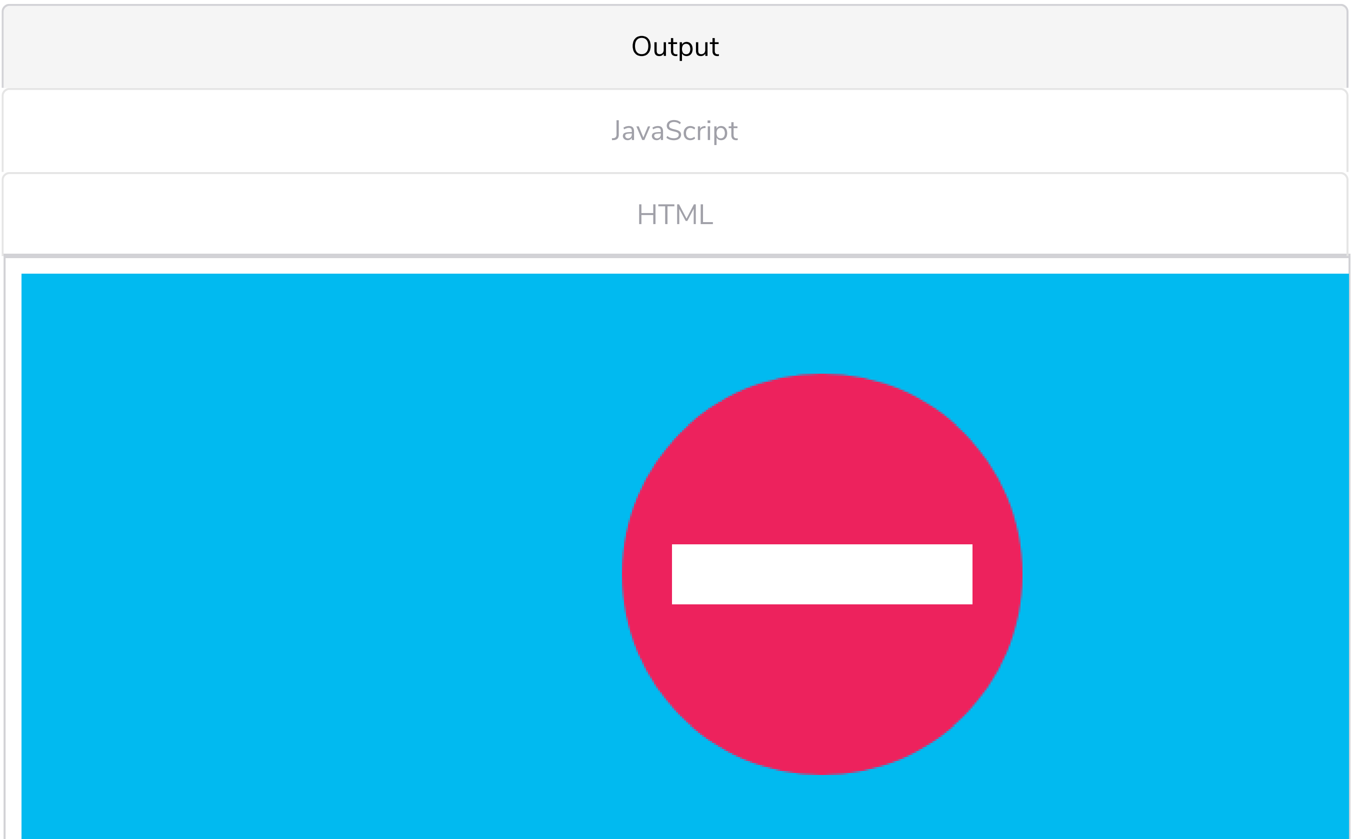
JavaScript

HTML



Clicking on the canvas area will now display the rectangle inside the circle. By using the **mouseIsPressed** p5.js variable, we made the display of the rectangle conditional to the mouse click.

Toggling the state of something based on mouse click might be a more involving example so let's see how to tackle that as well. Say we would like to change the background color for our sketch every time we click the mouse button. We will make it so that it will toggle in between two colors.



In this example, we are creating a global variable called **toggle** that would store a **boolean** value. Then we make this **boolean** value to change to the opposite of what it was with each mouse click by using the exclamation mark operator. When used in front of a **boolean** value, *exclamation* mark would simply invert that value, meaning it would make a **true** a **false** and vice versa.

You might notice that the **mouseIsPressed** variable doesn't seem to work great in capturing our clicks. This is because **draw** function is being called numerous times in a second which makes it hard to detect mouse clicks using a conditional. Later, we will see a better way of detecting mouse clicks using p5.js **Events**.

mouseX & mouseY

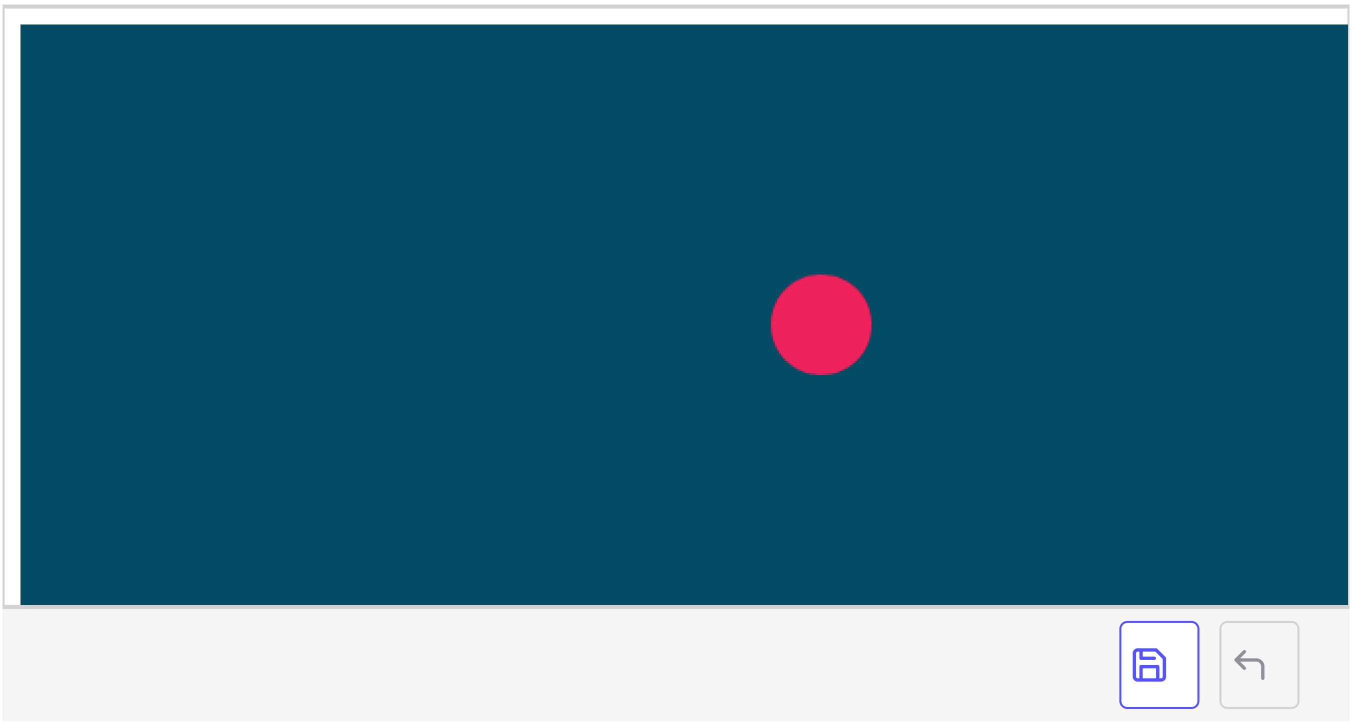
p5.js variable **mouseX** holds the current horizontal position of the mouse and **mouseY** holds the current vertical position. This sounds simple enough, but it has the potential to enable a great deal of user interaction in our programs and hence is incredibly useful variables. If we are to provide these values as x and y coordinates of a shape, we would essentially be moving that shape as we move our cursor on the screen.

Let's try this with a simplified version of our previous program. Here is a version of it with just a circle being drawn in the middle of the screen.

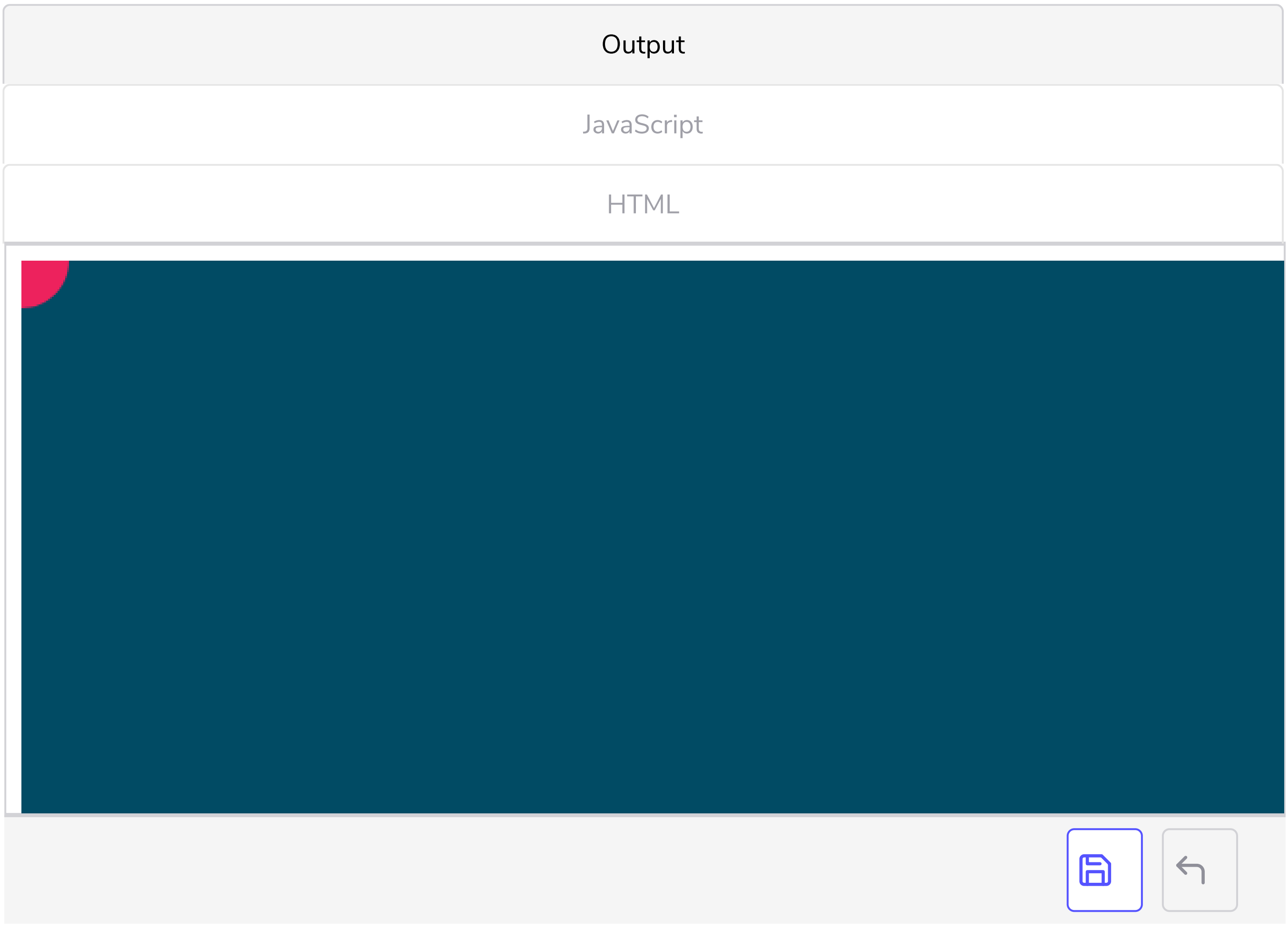
Output

JavaScript

HTML



Now let’s use **mouseX** and **mouseY** variables for x and y values.

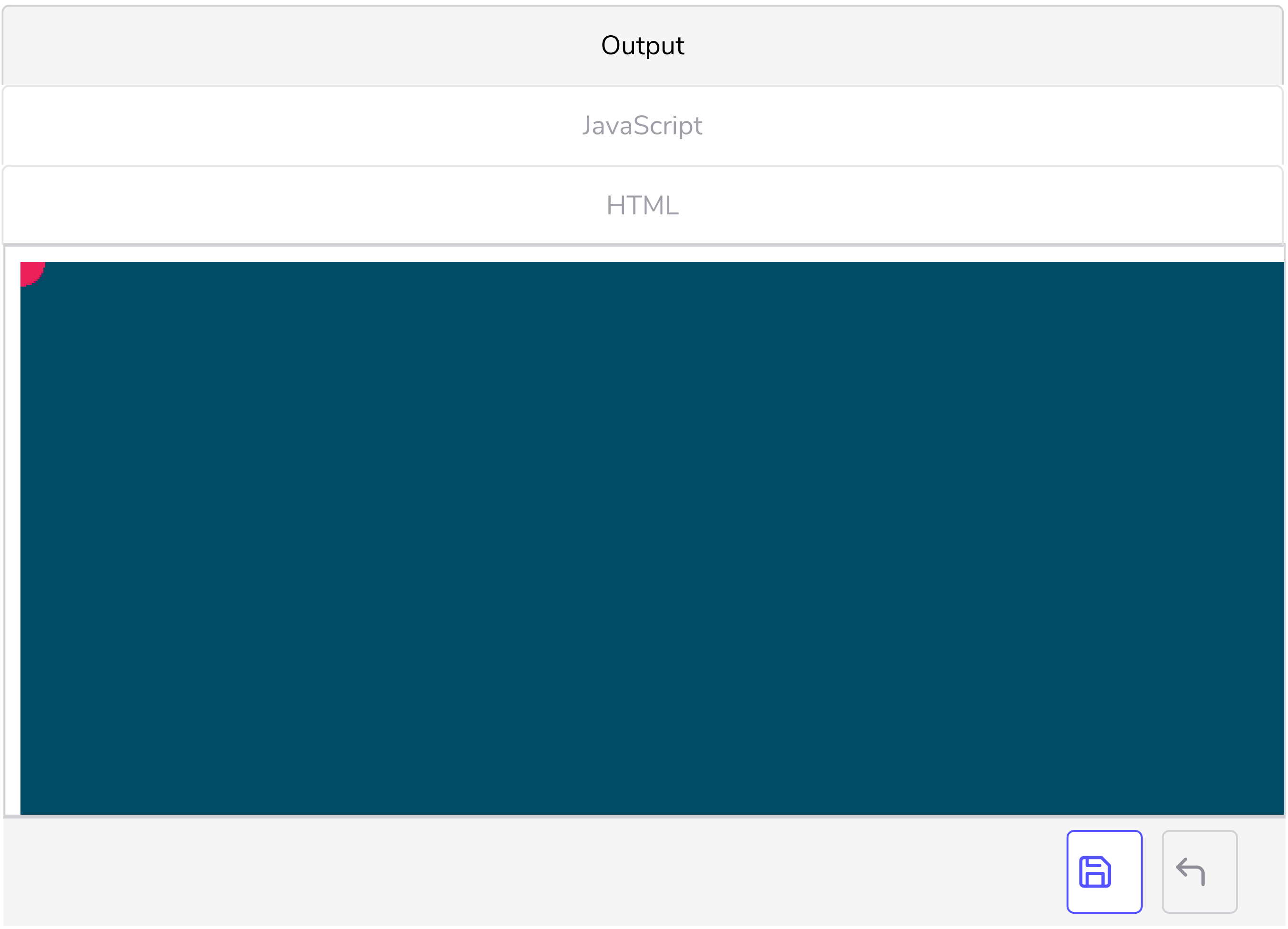


Try moving your mouse on the canvas. Isn’t this amazing? By using two built-in variables, we made our otherwise static sketch into something that a user can interact with.

Did you ever wonder why we are setting the **background** function inside the **draw** function? We seem to only need to set this value once, so you might have assumed it should go to the **setup** function.

background function inside the **draw** function allows us to override everything that was drawn in the previous frame with a solid color. Without that declaration, at the beginning of the frame, you would notice that drawings from the previous frame persist on the screen. But for certain use cases, this might be exactly what you might be going for.

Here is the example from before with a smaller circle size, lower opacity for the shape color and the **background** being declared only once in the **setup** function.



Summary

In this chapter, we learned about a couple of more p5.js built-in variables that would specifically help us in creating programs that are interactive; programs that can respond to the user action.

We learned about the p5.js **mouseIsPressed** variable that assumes a **true** value whenever the mouse is clicked. But we also learned that this variable might not be the best way to handle user input. We will later see the concept of Events in p5.js which is much better in handling user input.

We also saw **mouseX** and **mouseY ** variables and how they can be used to animate objects based on the mouse cursor position which allows us to add a great deal of interactivity to our programs in an easy manner.

Practice

Build a script that would draw a rectangle to the screen at every mouse click, at the position of the mouse cursor.

Output

JavaScript

HTML

Console

Clear