# 03 - Colors in p5.js

Understanding the Color Setting Functions in p5.js

WE'LL COVER THE FOLLOWING


- Color Functions in p5.js
- Summary
- Practice


## Color Functions in p5.js #

Now that we can draw shapes in p5.js let's look at how to control the color in our sketches. We are already assigning a light gray color to the background by passing the values `220, 220, 220` to the **background** function.

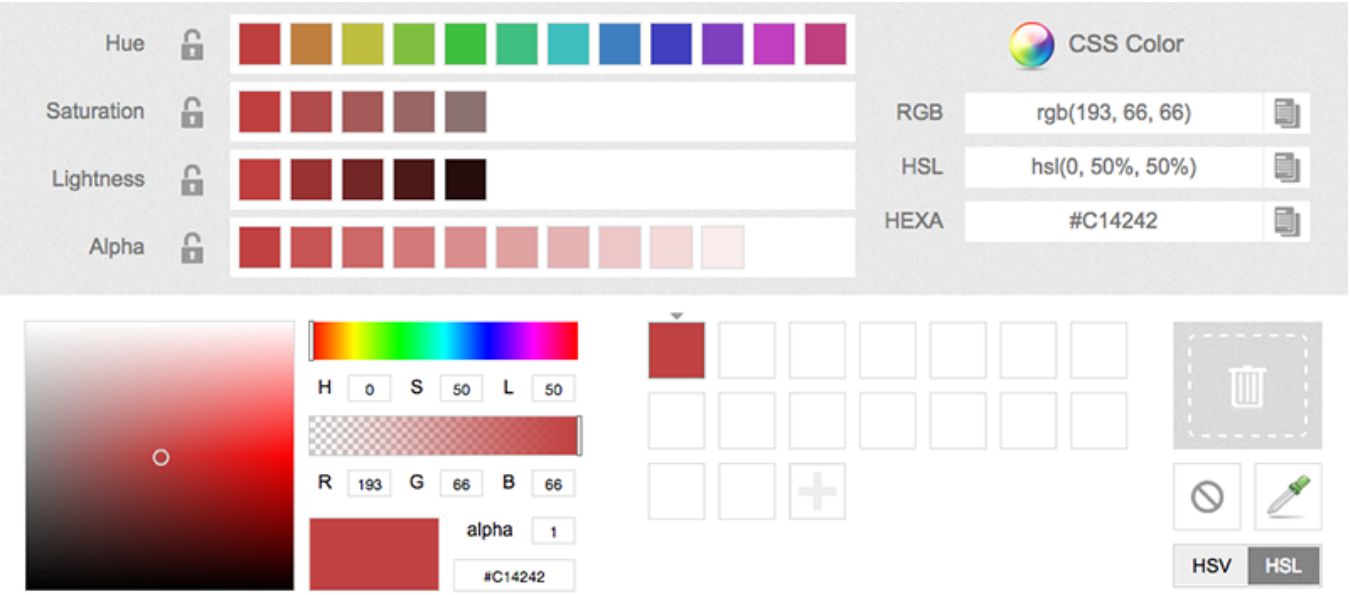
p5.js by default uses the *RGB* color values where R stands for red, G stands for green and B stands for blue. This means that we will *usually* need to pass these three color components to a color accepting function to set the desired color. Each of these color components can have a value in between 0 and 255. This means that if we are to pass `0, 0, 0` to the **background** function, we will end up getting a black color for the background and if we are to pass `255, 255, 255`, we will get a white color. p5.js being a helpful library allows us to pass a single value when we want all these three values to be equal. Which means that instead of passing `0, 0, 0`; we can also just pass a single 0.

Whenever we have equal amounts of these three color components, the resulting color will be a white, black or a shade of grey. So passing a single value to a color setting function is useful if we wanted to have a grayscale color. But if we want hue in our color then we need to pass all these three values to be able to specify the amount that we want for each component. 255 is the maximum value that a color component can accept; so if we are to pass `255, 0, 0` as a color to the **background** function we will get a pure red color.

If we pass `0, 255, 0`, then we will get a pure green color, and so on.

RGB color model is an additive model, which means that adding these colors together in their full intensity will result in white compared to paint colors which are subtractive, where adding them all together will result in a dark-brownish color. Finding the exact color that you want by tinkering with these values can be a bit hard if you are not too familiar working with additive RGB colors. If that's the case you can use an online color picker service to help you with finding the desired color. An online search for the term 'color picker' will result in numerous results that you can use to identify the RGB components for the desired color. Here is an example service from Firefox.

Color picker tool




Using a service like this you can make note of the RGB values that correspond to the color that you choose and make use of those values inside p5.js.

We can actually pass a fourth argument to a color setting function. This fourth argument, called the alpha component of the color, controls the opacity of the color and again accepts values from 0 to 255. 0 would result in full transparency and 255 would result in full opaqueness.

So we can pass a single value, three values or four values to a color setting function. I don't want to overwhelm you with too much information, but we can pass only two arguments as well. If we are to do so, we would be setting a grayscale color and an alpha component for that grayscale color.

If this abundance of options seems overwhelming, remember that they are

there for our convenience. p5.js could have restricted the color functions to

only work with four inputs which would have covered all the cases but would have been time-consuming to provide additional data when we only wanted something like opaque white which happens more often than not. It seems like developers of p5.js built their functions smart enough so that they would result in different output based on different number of arguments.

Knowing how the colors work in p5.js is great, but we can only change the color of the background so far. To be able to change the color of the shapes, we will have to make use of a couple of more functions.

The first function that we should know of is **fill**. **fill** allow us to set the fill color of the shapes. Fill color is the color that fills inside the shapes and if you are wondering what other color control there is for shapes, there is also the stroke color that defines the color of the outline of a shape. The default color for the fill and the stroke is white and black respectively. All the shapes except for line have both a fill and a stroke color.

We can set the fill color of the shapes by calling the **fill** function and passing color arguments to this function as discussed earlier. What **fill** function will do is that, it would set the active color to be the chosen color until we set the color to something else by using another fill function.

**stroke** function works in a similar manner. We pass it color arguments and it sets the color of the stroke for all the shapes until the next stroke function. A **fill ** or a **stroke** function that comes after a prior one would overwrite the settings of that prior.

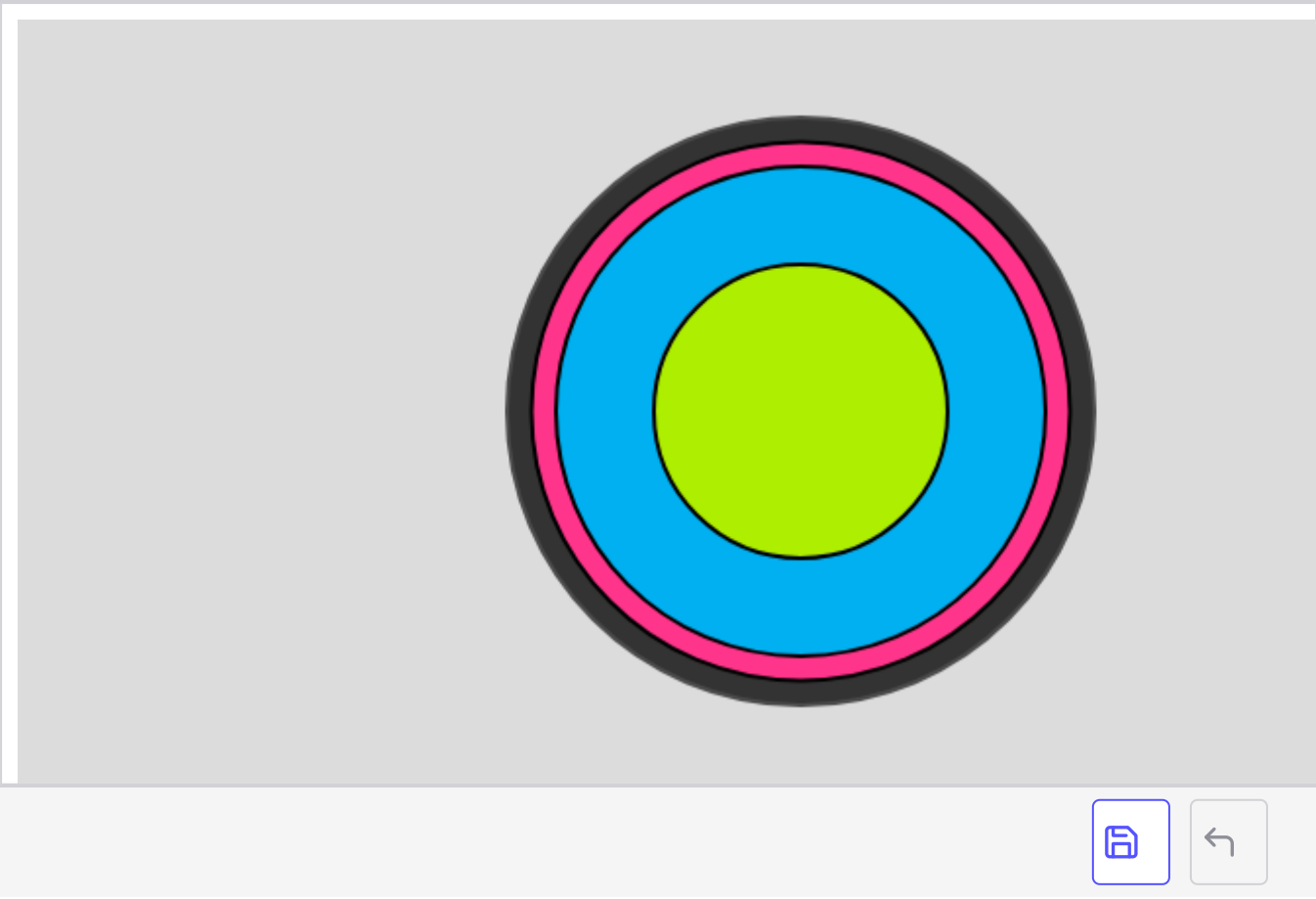
At this point, one other useful function to know could be **strokeWeight** which allows us to set the thickness of an outline.

Here is a small sketch that makes use of some of the functions we learned about in this chapter.

Output

JavaScript

HTML



Notice how we are using the fill function before the shape that I want to set the color for. And we keep using it to be able to switch the color for different ellipses.

Two more functions that are worth mentioning are **noFill** and **noStroke** functions. As their name implies, when called these functions will respectively get rid of the fill and the stroke of shapes. These functions are called without any arguments.

```
noFill();
noStroke();
```

## Summary #

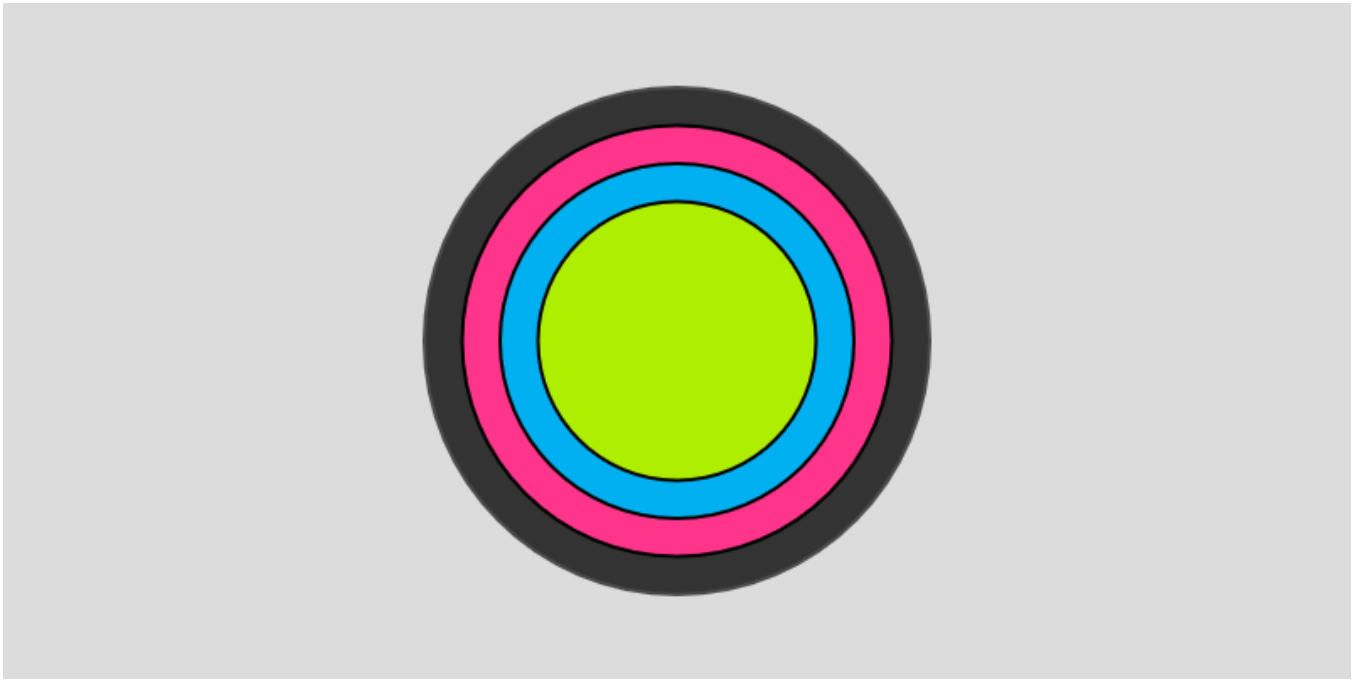
In this chapter, we haven't seen any new JavaScript functionality or new programming structures. We just looked at some operating principles of the p5.js library and some specific functions that comes with it. In particular we learned about how some of the color setting functions work in p5.js, such as **fill**, **stroke** and **strokeWeight**. We also learned about other functions that are related to fill and stroke operations such as **noStroke** and **noFill** . Another

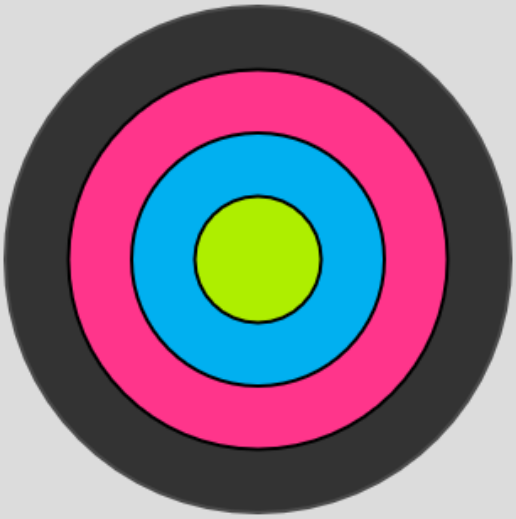
thing we learned about is the RGB color model.

Even though this chapter didn't really advance our programming language, I think one point is very valuable to make. You might be thinking to yourself that you are not into creative coding and won't need this p5.js specific information after this course, having learned to code. But these operating principles such as using additive RGB values, or concepts such as fill and stroke are so commonly used that even though what we are learning could seem very specific to p5.js, they are general principles or concepts that are utilized by lots of other drawing libraries or programs. Understanding them will serve us well in our journey of learning how to program.

## Practice #

Build the above script in such a way that one variable would control the size of all the circles (meaning changing that variable should change the size of all the circles) and another one should control the radius difference for all the circles.





Output

JavaScript

HTML

Console

Clear