+++ TEACHING GUIDE +++
MODULE 6: SHAPES AND COLORS

Agenda:

1. Other shapes in p5.js (20 min)
2. How color works on a computer (10 min)
3. RGB color (15 min)
4. Stroke and Fill (15 min)

Other shapes in p5.js

Last session you learned how to make ellipses and place them around the drawing canvas. The kind of shapes you can make with p5.js is not limited to ovals and circles. The p5.js language has many built-in shapes for you to draw with. Let's take a look at some of them.

First go to the p5.js web editor by typing editor.p5js.org into your Internet browser and log in to your account, so you can experiment with these new shapes.

rect()

[*The biggest difference between ellipse() and rect() is that, by default, the X-Y component indicates the center of an ellipse, but it indicates the top left corner of a rectangle. It would be worth demonstrating the difference on a grid or graph paper if the learners need more help with math/coordinate system review.*]

The function called rect() allows you to draw a rectangle (or a square). It works very similarly to the ellipse() function.

After the line for background(220); type rect(100,100,50,50); and press the pink Play button at the top. You should see a small white square appear in the canvas.

Just the like ellipse() function, the rect() function takes 4 numbers to display.

```
rect(100,100,50,50);
rect(x   ,   y,   w,   h);
```

The first two numbers tell the program where you want to draw the rectangle. These numbers are the X-Y location of the *top-left corner* of the rectangle. (This is different from the ellipse() function which uses the X and Y location for the *center* of the ellipse.)

The next two numbers tell the program how many pixels wide and how many pixels high you want to make the rectangle.

Add two more lines of rect() to see how the shape works in p5.js. For example:

```
rect(30,20,300,50);
rect(200,150,100,200);
```

line()

The next function we will learn in p5.js is the line. The line also takes 4 numbers (also called parameters) to display. In this case, the 4 numbers are the start and end points of the line. So to draw a line, we need to tell the program two X-Y locations like this:

```
line(x1,y1,x2,y2);
```

Try adding line(50,300,150,200); to your program.

This will draw a line between the two coordinates (50,300) and (150,200).

Add a few more calls of line() in your program to get comfortable with entering 2 X-Y coordinates.

triangle()

[*For the less mathematically-inclined, triangles will be one of the more difficult shapes to draw the way you want because they require listing 3 sets of X-Y coordinates. The order of the coordinates does not matter.*]

The last new shape we'll learn today is the triangle. The triangle() function is similar to the line() function in that you need to tell it the X-Y locations of the corners of the triangle. For a triangle, there are three corners, so you'll need 3 separate X-Y locations:

```
triangle(x1,y1,x2,y2,x3,y3);
```

Add this to your program:

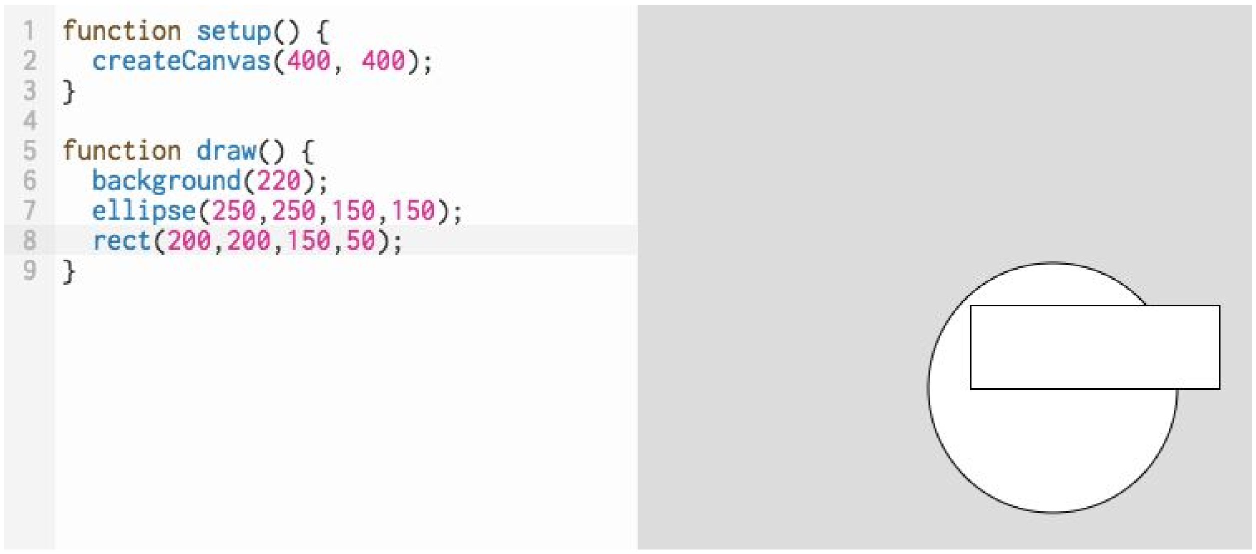
triangle(100,300,150,300,125,350);

Add another triangle to your program using different X-Y coordinates.

Ordering your lines of code

[*When demonstrating the concept of drawing order to the learners, write two (or more) shapes that will overlap into your code. Have the learners guess which will be on top and which will be partially hidden. Then reverse the order of the shapes in your code to show the alternative. Show a few examples if learners don't grasp this right away.*]

By now, you've probably noticed that the shapes you draw can overlap or be on top of each other. It's important to realize that your program will draw the shapes in the same order each time. The program runs from top to bottom. So if you have an ellipse on line 7 of your code, and a rectangle on line 8 of your code, the program will first draw the ellipse and then draw the rectangle. If those two shapes are in the same area of your canvas, the rectangle will appear to be on top of the ellipse, because it was drawn after the ellipse.
It's almost as if the program is just stacking the shapes on top of each other as it draws.



```
1  function setup() {
2    createCanvas(400, 400);
3  }
4
5  function draw() {
6    background(220);
7    ellipse(250,250,150,150);
8    rect(200,200,150,50);
9  }
```

Notice the rectangle is on top of the ellipse because it was drawn second in the program.

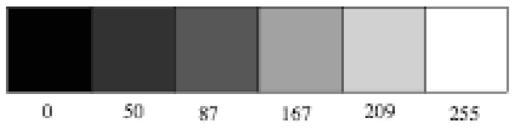
How color works on a computer

So far, every shape you've made has been white. And the background has always been grey. Now it's time for a big, creative step forward: drawing in color!

First, it important to understand how color works in the digital world. On a computer, color is defined by a range of numbers. Instead of saying, "I want this circle to be purple," you have to give the computer specific numbers to make the shape appear in the color you want.

The range of numbers you can use to tell the computer what to do is from 0 to 255. Let's look at how a computer defines the color black and the color white.

The tell the computer to display black, you give it the number 0. To tell the computer to display white, you give it the number 255. Every number between 0 and 255--like 50, 87, 167, and so on--is a shade of grey. As the numbers increase in size, the color grey gets lighter and lighter, until it's completely white at 255.



To understand this better, let's try changing the background color. To do this, you simply enter a new number between 0 and 255 into the line of code that says background(220);

You'll notice that 220 is fairly close to 255, so the background of the canvas is a light grey color. As you lower that number, you'll see the background become darker. Try replacing 220 with a smaller number like 75. Press the "Play" button to see how the drawing looks now.

Try a few more numbers to see how smaller numbers are darker, and higher numbers are lighter.

RGB color

[*The idea of additive color is probably new to most learners. In most cases (painting or drawing with markers/crayons) students have used subtractive color: add colors together makes the overall color darker. Computers display color by mixing light, which follows the additive color system. The more color you add, the brighter the overall color becomes. So mixing the maximum (255) of red, green, and blue results in white being displayed. Mixing very low amounts of red, green, and blue will create very dark (or even black if you choose 0 for each color) colors. Grasping this general concept is important, but not imperative. A learner can always just use a color picker tool to get the R G B values they desire.*

With the colorpicker.com tool, demonstrate that they can change the hue with the horizontal band of colors to the right, then they can choose the lightness/darkness and saturation of that color in the big main square on the left.]

It's nice to draw in black and white and every shade of grey in between. But it's much more fun to use color! Computers have a special way of displaying color. They use something called the "RGB Color Model". RGB stands for Red, Green, and Blue. The computer uses different combinations of those three primary colors to display any color you can imagine.

Just like with the black, white, and grey colors, you use numbers between 0 and 255 to tell the computer how much of that color you want to use. To learn how to display different colors, we'll experiment with the background again.

To make the background red, try this:

```
background(255,0,0);
```

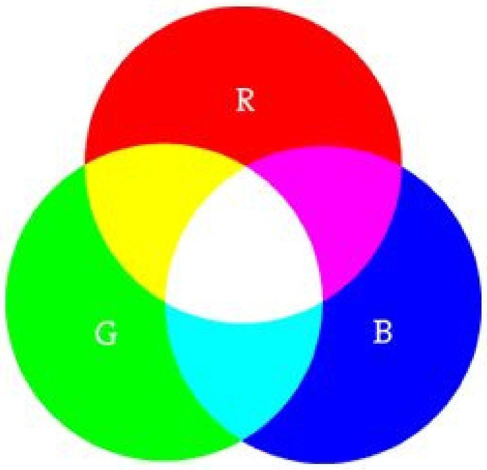
Notice, that to make a color besides black, white or grey, you need three numbers. The first number is how much red you want, the second is how much green, and the third is how much blue. So to make a green background,

```
background(0,255,0);
```

You want zero red displayed, all the green (255) displayed, and zero blue display. How would you display a blue background?

If you want other colors to display, you simply mix Red, Green and Blue values until you get the color you want. Here are some standard mixes:

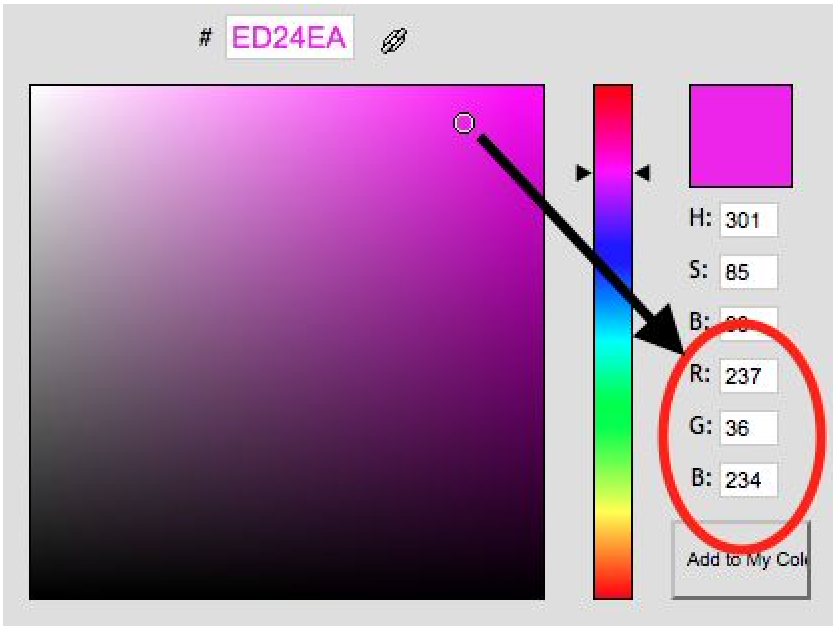
- Red + Green = Yellow
- Red + Blue = Purple
- Green + Blue = Cyan (blue-green)
- Red + Green + Blue = White
- No colors = Black



The main thing to remember is that higher values (closer to 255) will result in brighter colors, and lower values (closer to 0) will result in darker colors.

You can experiment with different combinations to make colors you like. There are also good tools on the internet to help you find the RGB color you want.

Visit http://www.colorpicker.com/ and use your mouse to click on a color you like. On the right side you will see text that displays the R, G, and B values of the color you clicked on. You can then add those colors to your program by typing them in.



For example, to make a nice pink background, include these numbers in your background() code:

```
background(237,36,234);
```

Stroke and fill

[*Stroke and fill will likely be strange terms for something the learners already know: an outline color and the inside color of a shape. Once they get these terms the main thing to highlight is the order which you type color commands matters. And that you must write a new line of stroke or fill every time you want to change the color.*

The other new concept of the section is noStroke() and noFill(). These functions (like all commands in javascript) are case-sensitive, so they must type it exactly as written. This is the time to review how the Shift key works for learners less familiar with the keyboard.]

You've just learned how to set the background color to whatever you want: whether that's grey or your favorite shade of blue. It's time to introduce a few more tools so you can change the color of any shape you want to draw. Those tools are stroke() and fill().

stroke()

Stroke is the outline around your shape. By default, the stroke color is black. Notice that any of the shapes you draw had a thin black outline around them. Changing the color of the stroke is much liking changing the background color. You add a line of code with and R, G, and B value:

```
stroke(120,0,200);
```

This will make all the outlines (and lines) you draw purple. You can choose any color you like by adjusting the 3 RGB values of stroke().

fill()

Fill is what you call the color inside the shape you draw. So far they have all had a white fill. Now you can use the fill() function to change that color, just like you changed the background and stroke.

```
fill(220,220,100);
```

Adding this line of code will make the inside of a shape a light yellow color.

A note on colors and drawing order

Sometimes you might be tired of making every shape have a color. Luckily, p5.js has two functions that allow you to draw a shape without a fill or without a stroke.

Those two functions are called noFill(); and noStroke();

If you don't want the shape to have a stroke color, simply enter noStroke(); on the line before you draw the shape. The same is true for noFill();. Type that before the shape if you don't want it to have any color.

The important point here is that you must type your color instructions--fill(), stroke(), noFill(), noStroke()--*before* you type your instructions for drawing a shape.

It's as if you have to choose the color of pencil or color of paint you want to use before you draw something. You can't tell the computer to draw something and then later tell it what color you wanted.

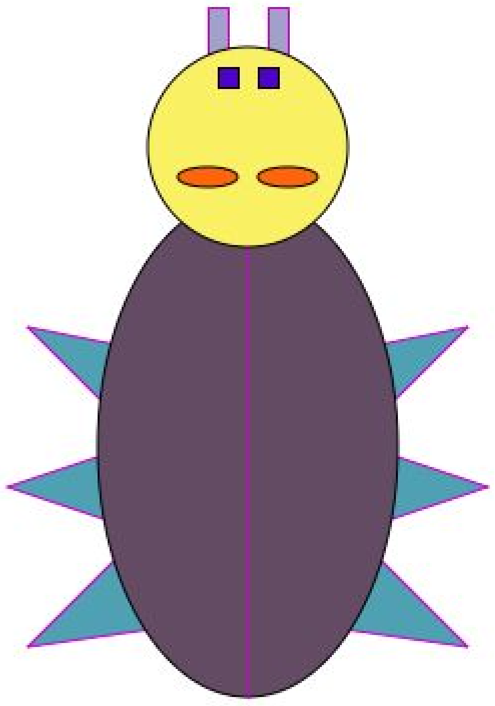
The other thing to remember is that once you've chosen a fill color, like fill(100,20,120);, every shape you draw after you type that fill() command will have that color. If you want a new shape to have a different fill or different stroke color, you must type new instructions for the program before that new shape.

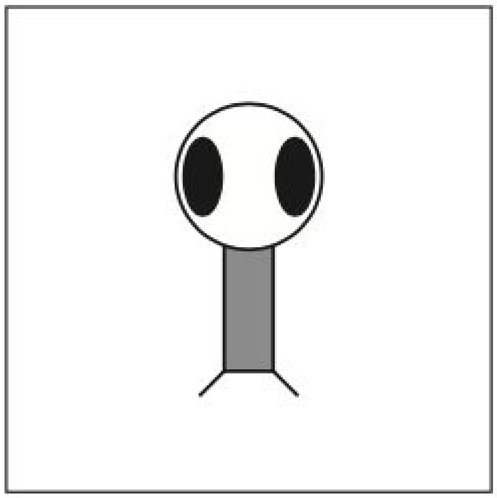
Exercise: Create a character

By now you have all the tools you need to create your own unique characters in code. For this exercise you'll be designing a colorful character using circles, rectangles, triangles and lines. Be sure to use a range of colors to make your character come to life.

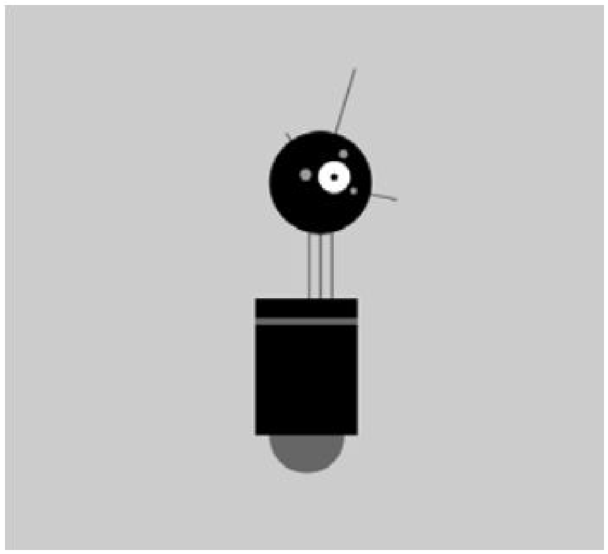
A few ideas to get you started: you can create an insect/beetle, a robot, or a person. Give your character legs, arms, a body, and a unique face.

Once you are happy with the character you've created, save your sketch by giving your sketch a name (like "Exercise 6" or "My Character") and pressing the "Save" button near the top of the webpage screen.





Credit: Daniel Shiffman



Credit: Casey Reas & Ben Fry