**Lesson 4** fill() and stroke() functions

**How can we change the color of shapes in p5.js?**

| **Overview** | |
| --- | --- |
| In this lesson students will learn how to change the color of basic shapes by using the stroke() and fill() functions. They will create their robot drawing from Lesson 1 using these new function calls. | |
| **Lesson Objectives** | |
| Students will be able to   * Change shape color using fill() and stroke() functions * Adjust outline of shapes using the strokeWeight() function * Use pseudocode to plan a p5 sketch | |
| **Suggested Duration** | |
| Two Periods (90 minutes) | |
| **Blueprint Foundations Student Outcomes (**https://blueprint.cs4all.nyc/outcomes/) | |
| Abstraction  Analyze | **Give examples** of specific patterns in something I can see, do or touch. |
| Abstraction  Prototype | **Describe** different things I tried in order to achieve a goal. |
| Algorithms  Analyze | **Describe how** instructions can have different outputs depending on inputs. |
| Programming  Analyze | **Experiment** with the commands of a programming language. |
| **Vocabulary** | |
| * **Stroke**: A line or a border around shapes * **Syntax**: The spelling and grammar rules of a programming language, like JavaScript * **Pseudocode**: A detailed outline of what a computer program will do, without using JavaScript syntax | |
| **Planning Notes** | |
| * Students will need to refer to their robot drawings from Lesson 1. For students who were absent or lost their drawing, print out extra copies of the [worksheet](https://drive.google.com/file/d/1fpkP_ULAhbsTFQAOq5CDDDtwx6f_wZR7/view?usp=sharing). * Materials: Have colored pencils available for students to color in their drawings | |
| **Resources** | |
| * Video tutorial: [1.4 Color](https://youtu.be/riiJTF5-N7c) | [Code](https://editor.p5js.org/codingtrain/sketches/rJ9MQSwvm) * [p5.js Color Tutorial](https://p5js.org/learn/color.html) * Video: [RGB color model](https://www.khanacademy.org/partner-content/pixar/color/color-101/v/color-2) * [Color Picker Tool](https://htmlcolorcodes.com/color-picker/) | |
| **Assessments** | |
| * Assess the **Independent Practice**. Check for the ability to:   + Use correct syntax for the fill() function   + Call fill() and shape functions in the correct order * Assess the **Student Activity**. Check for the ability to:   + Write clear and concise pseudocode   + Use RGB values to create specific colors   + Use correct syntax for the stroke() and strokeWeight() functions | |

| **Do Now** |
| --- |
| * Show students the [code below](https://editor.p5js.org/mparker/sketches/vtKD5ERWh). Ask them to infer:   + What does the fill() function do?   + What do each of the three parameters of the fill() function do? |
| **Discussion: fill() and stroke()** |
| The fill() function:   * Have students share their thoughts from the Do Now. Explain that fill() controls the color of shapes by taking in three parameters with a range of 0 to 255: A red value, a green value, and a blue value, known as **RGB values**. 255 means use **all** of that color, and 0 means use **none** of that color, which is why fill(0,0,0) makes shapes black.      * Ask students to guess what would happen if you changed one of the fill() functions from the Do Now to be fill(255,255,255). Then try it out to reveal that this actually makes a shape white. Show the images below to explain that color mixing rules with RGB values are based on what happens when you mix *light* together, not paint.      * Return to the code from the Do Now and comment out all of the fill() commands except for the first one. Explain that even though you used only one fill() function, it will make every shape that comes after it that same color. You may use the following analogy with your students:   + “When you write fill(0,0,0), imagine you’re picking up a black marker. Every shape that you make after that will be black, unless you pick up a different marker *first*. To make the next circle red, I’m going to pick up a red marker by writing fill(255,0,0) *before* the next ellipse function.”      * **Independent Practice**   + Have students duplicate the code from the Do Now. Using the above image of intersecting ellipses as a reference, ask them to follow the prompts below:  1. Make the first ellipse yellow. 2. Make the ellipse circle pink. 3. Make the third ellipse gray. 4. Make the last ellipse a color of your choice!   Grayscale and the background() function:   * After students have shared their responses, explain that there is an easier way to make shapes gray. If students use only one parameter from 0 to 255, the color made by fill() will be black, white, or a shade of gray. * Tell students that there is another function in this sketch that also accepts values from 0 to 255, and see if they can figure out what it is. * Once a student has identified the background() function, remind them that the background defaults to gray: background(220). Using their knowledge of fill(), have students infer how they could make the background red.   The stroke() function:   * Ask students if they notice anything else about the shapes from the Do Now. To give them a hint, you may ask them to look closely at an ellipse with a bright color and ask if that is the *only* color used to draw that shape. Students may notice there is a black outline around each shape. Explain that in p5, that outline is called a *stroke*. * Explain strokes in p5.js using the following language:   + By default, all shapes have a black stroke that is 1 pixel wide.   + To change the *color* of this stroke, we use the stroke() function *before* the shape is called.   + The stroke() function takes the same parameters as fill, so we can put in one number from 0 to 255 to make a gray outline, or use three numbers for each RGB value to make any color we like.   + To change the *thickness* of the stroke to more than 1 pixel, we need to use a different function, strokeWeight()--lowercase “s” capital “W”. If we write strokeWeight(5), that would draw a shape with an outline that is 5 pixels wide. |
| **Teacher Demo: Code Along using fill() and stroke()** |
| * Demo 1: Tell students that you will practice using these functions together. Have volunteers name the type of shape, fill, stroke, and stroke weight. Open up a new sketch and demonstrate how to use each function, and in what order.   + Note: This is a good time to show students how to use a [color picker tool](https://htmlcolorcodes.com/color-picker/) as a resource for colors that are tricky to create. In the tool linked above, make sure they use the values marked RGB, and not others like HSL or CMYK (which are different methods used for color that they won’t be using in class) * Demo 2: This time, name the shape, fill, stroke and stroke weight yourself, and get student volunteers to call out the code while you type. Make students specify where in the code functions should be called. Stress that fill(), stroke(), and strokeWeight() need to be called *before* the shape you want them to affect. * Demo 3 (Option A): Ask students what parameter they could give to strokeWeight() to *remove* the stroke. Type out strokeWeight(0) to show students one way to create shapes without strokes. * Demo 3 (Option B): Introduce students to two more functions: noStroke() and noFill(). Ask students to predict what each function does based on the name. After using these functions in your code, ask students if they notice something different about these functions. Point out that unlike fill() and stroke(), these functions *don’t have parameters*. Many functions require parameters, but some don’t. |
| **Student Activity: Using Pseudocode to draw a Robot in p5** |
| * Have students take out their robot drawings from Lesson 1. Pass out new [worksheets](https://drive.google.com/file/d/1fpkP_ULAhbsTFQAOq5CDDDtwx6f_wZR7/view?usp=sharing) to students who need them. * Give students a few minutes to color in their drawings using colored pencils. Encourage them to use at least three different colors and two different stroke weights.   Introducing Pseudocode   * Before students jump into the web editor, it is important for them to create a plan for what their code will look like using **pseudocode**. This is a common practice not just for students, but for professional programmers. * In Lesson 1, students were informally introduced to pseudocode when they wrote down the instructions for a given shape on the other side of their worksheets:      * Explain that pseudocode is a detailed outline of what students would like their program to do, **without** using JavaScript syntax. You can use the [example below](https://docs.google.com/document/d/1eMYz7sBGKfKC5UhyZQ2yKa4a8VpbvUU37d2DPaEizaw/edit?usp=sharing) to model how you would write the pseudocode for each visual element in the robot drawing (note that the black dots represent anchor points for x and y coordinates, not black ellipses):        * Now students should write their own pseudocode. They can add to the shape instructions they’ve already written on page 2 of the worksheet, or in their design journals. * Once students have shown you their robot design and pseudocode, they may begin coding in p5. Remind students to add comments to label the different elements in their drawing. * Note: Students may not have sufficient time to code before the end of the period. If this happens, allow students another period to code their robot drawings. Here is a completed [example](https://editor.p5js.org/mparker/sketches/g8PRwFEKt): |
| **Wrap Up** |
| * Before students leave, make sure they save their sketches and ask them to share their sketches with you so you can assess their progress. |
| **Extensions** |
| N/A |