Chapter 1 : Color

Exercise P.1.2.1

Step 1:
This exercise will focus on observing color palettes through interpolation. In the setup function we create our canvas and set the dimensions to 800 x 800. We set the color mode to HSB , we have no stroke and we also have a color shaker.

```
function setup() {
  createCanvas(800, 800);
  colorMode(HSB);
  noStroke();
  shakeColors();
}
```

Step 2:
Now we define the variables we will use.

```
 */
'use strict';

var tileCountX = 2;
var tileCountY = 10;

var colorsLeft = []
var colorsRight = [];
var colors = [];

var interpolateShortest = true;
```

We let tileCountX equal to 2 and tileCountY equal to 10. We let all the other variables equal to an empty array.

Step 3:
We begin this step in the draw function. Here the number of color gradients (tileCountX) and the number of rows (tileCountY) are determined by the position of the mouse. The map method is used to apply a function on every element in an array.

```
function draw() {
  tileCountX = int(map(mouseX, 0, width, 2, 100));
  tileCountY = int(map(mouseY, 0, height, 2, 10));
```

We then use a nested for loop to draw the grid row by row. The colors for the left and right columns are set in the arrays we created at the start colorsLeft and colorsRight.

```
for (var gridY = 0; gridY < tileCountY; gridY++) {
  var col1 = colorsLeft[gridY];
  var col2 = colorsRight[gridY];
```

The colors in between are calculated using lerpColor. This function performs the interpolation between the color values. We use the parameter 'amount' to specify the position between the start and end color.

```
for (var gridX = 0; gridX < tileCountX; gridX++) {
  var amount = map(gridX, 0, tileCountX - 1, 0, 1);

  if (interpolateShortest) {
    // switch to rgb
    colorMode(RGB);
    interCol = lerpColor(col1, col2, amount);
    // switch back
    colorMode(HSB);
  } else {
    interCol = lerpColor(col1, col2, amount);
  }

  fill(interCol);
```

Step 4:

We create a function called shakeColors. This function is used when the mouse is pressed and gives the canvas a new palette of colors to work with when the mouse is clicked.

```
function shakeColors() {
  for (var i = 0; i < tileCountY; i++) {
    colorsLeft[i] = color(random(0, 60), random(0, 100), 100);
    colorsRight[i] = color(random(160, 190), 100, random(0, 100));
  }
}

function mouseReleased() {
  shakeColors();
}
```