PUI Assignment 9 - Worksheet

Aaron Lee

p5.js Library Tutorial

In this tutorial, we will be adding a basic interactive background to a webpage using the p5.js library.

p5.js is a library where you can create coded visuals on a canvas. We will be using it create a simple background that draws a circle which follows our mouse cursor around the webpage.

You will need to download the files from the repo below to follow along with this worksheet.
https://github.com/thelittlefloor/PUI-Aaron-Lee/tree/main/assignment_9/p5_tutorial

The folder includes the complete p5.js library files which will run from your local drive, the template file which you will use to follow along with this worksheet, and the finished code in the 'final' folder, in case you get stuck.

In the 'base' folder, open 'index.html' and 'sketch.js'. The html file is a sample webpage onto which we will be adding our interactive background, open the html file to see the basic webpage. The 'sketch.js' file is where we will be coding our background.

Go to 'sketch.js'
First some basics. The sketch file needs 2 baseline functions to run. The setup() function, and the draw() function. The setup function runs once at the beginning of a session, followed by a continuous looping of the draw function.

Let's create our canvas, in the setup function, call the createCanvas() function and pass 2 parameters which provides the width and height of the canvas you want in pixels. We will use 400 by 400 pixels for now.

```
function setup() {
    createCanvas(400, 400);
}
```

Refreshing the webpage will not show you anything, thats because the canvas is empty. Lets add a background color to the canvas by calling the background() function in the draw function. Pass it 3 parameters to signify the RGB value you want, for now lets go with, (150, 150, 150).

```
function draw() {
    background(150, 150, 150);
}
```

Refreshing the webpage should show you the 400 by 400 pixel canvas.

Now lets figure out the position of the circle we will draw on the canvas. Its quite simple, the current position of the mouse subtracted by the current position of the circle will create a vector for which to add to the current circle position. Multiply that vector by a constant lower than 1 in order to make the circle ease into the position of the mouse.

target - current = d
current = current + (d x ease_constant)

Simply do this for both the x and y coordinate to get the position of the circle on the canvas. We will create global x and y coordinate variables by defining them outside of the setup and draw function.

```
let x = 0;
let y = 0;
var ease = 0.1;

function setup() {
    createCanvas(400, 400);
}

function draw() {
    background(150, 150, 150);

    var targetX = mouseX;
    var dx = targetX - x;
    x = x + dx*ease;

    var targetY = mouseY;
    var dy = targetY - y;
    y = y + dy*ease;
}
```

Refreshing the webpage will not show any changes. Thats because we did not use the x and y coordinates to draw our circle yet!

Add the fill() function with RGB parameters to indicate the color you want the circle to be filled with. Then draw the circle with the ellipse() function, give the circle parameters (x, y, 60, 60) the x and y are the coordinates of the circle, and the 2 subsequent integers are the width and height of the circle.

```
function draw() {
    background(150, 150, 150);

    var targetX = mouseX;
    var dx = targetX - x;
    x = x + dx*ease;

    var targetY = mouseY;
    var dy = targetY - y;
    y = y + dy*ease;

    fill(200, 200, 200);
    ellipse(x, y, 60, 60);
}
```

Now refresh the webpage and move your mouse over the canvas. The circle should follow the mouse around.

But we want the canvas to be the background of the webpage, not just an element place on the bottom like it is now. Fortunately, we can access the CSS of the canvas right in the sketch file.

First, lets store the canvas inside a variable. Create a global canvas variable and set it to the createCanvas() function. Instead of an arbitrary pixel size, we can set the canvas size to the size of the window.

```
var canvas;
let x = 0;
let y = 0;
var ease = 0.1;

function setup() {
    canvas = createCanvas(windowWidth, windowHeight);
}
```

Refresh to see that the canvas now instantiates with the size of the window

Now we can use the canvas variable to change the CSS of the canvas. Lets place it in absolute position at the top left corner of the window. But we also must adjust the z-index, or the canvas will cover up all the webpage content.

```
function setup() {
    canvas = createCanvas(windowWidth, windowHeight);
    canvas.position(0,0);
    canvas.style('z-index', '-1');
}
```

Now, refresh. You will see the canvas has properly fit the entire window and is place as a background to the webpage content.

All done? Not quite!
Notice that the canvas fits the window size when first uploaded, but does not resize itself to fit the window size when the window is resized. We can remedy this by creating an event so that the canvas resizes itself when the user resizes the window.

```
function windowResized(){
    resizeCanvas(windowWidth, windowHeight);
}
```

Add the code above as another function in the sketch file.

Now refresh, and thats it! You now have an interactive background for your webpage!