Objects, Classes and Arrays - Week 02

Using Arrays

Arrays and array management forms a really useful skill when programming with objects. When declaring an array use an appropriate array name. Normally you would use a plural

```
let shapes = [];
```

There are a number of methods to add elements to an array. The push() method add an element to the end of an array whereas the the unshift() method inserts the new element into the beginning of the array

```
shapes.push("Apples");
// ["Apples"]
shapes.push("Oranges");
// ["Apples","Oranges"]
shapes.unshift("Bananas");
// ["Bananas", "Apples", "Oranges"]
```

You can also target a particular item in the array by using the index value. If the index does not exist then it will add spaces between it what exists.

```
shapes[2] = "Peaches"
// ["Bananas", "Apples", "Peaches"]
shapes[4] = "Lemons"
// ["Bananas", "Apples", "Peaches", , "Lemons"]
```

Making You first Class

There are a few steps you need to follow for this work. Create a new File with the name of your class beginning with a capital letter eg. SquareDonut.js.
Link the class to your html document.

```
 <script type="text/javascript" src="SquareDonut.js"></script>
```

At this stage you should check in your browser to see whether the class has loaded. Look at the sources tab.
Now you are ready to create your class and then make some objects!!!!! Yippeee.
In the SquareDonut.js let us first define the class and setup the constructor.

```
class SquareDonut{
    constructor(){
        this.width = 100;
        this.height =200;
        this.posX =350;
        this.posY=350;
    }
}
```

These properties will be set using arguments and parameters later but for know let make each object the same. To instansiate (or make) a new object with use the new keyword in the app.js script

```
let temp = new SquareDonut()
console.log(temp);
// {width:100, height:200, posX:350, posY:350}
```

As you can see the object is made up of properties and values. These values can be accessed using the dot notation

```
console.log(temp.height)
console.log(temp.posY)
// 200
// 350
```

Now we are going to pass arguments to the class and then assign those arguments to parameters. The parameters are set up in the constructor as follows

```
class SquareDonut{
    constructor(_posX, _posY){
        this.width = 100;
        this.height =200;
        this.posX = _posX;
        this.posY= _posY;
    }
}
```

So now when we instansiate a new object we need to pass it two arguments

```
let temp = new SquareDonut(220,320)
console.log(temp)
// {width:100, height:200, posX:220, posY:320}
```

Now we are going to change the constructor so that the width and height of each object are random values

```
class SquareDonut{
    constructor(_posX, _posY){
        this.width = random(40,100);
        this.height =random(40,100);
        this.posX = _posX;
        this.posY= _posY;
    }
}
```

Generating an Array of Objects in a Grid

So finally we need to generate and array of 100 objects. THese objects will have a position on a grid. Each objects width and height will be different

```
let shapes=[];
let gridCount =10;
let gridSpace = 800/gridCount

function setup(){
    createCanvas(800,800);
    rectMode(CENTER);
    angleMode(DEGREES)

    for(let y=0; y<gridCount; y++){
        for(let x=0; x<gridCount; x++){
            shapes.push(new SquareDonut(x*gridSpace, y*gridSpace))
        }
    }
}
```

This will generate an array objects that look like the following

```
[SquareDonut {width: 99.84309571442938, height: 25.051098861396248, posX: 0, posY: 0}, Square
```

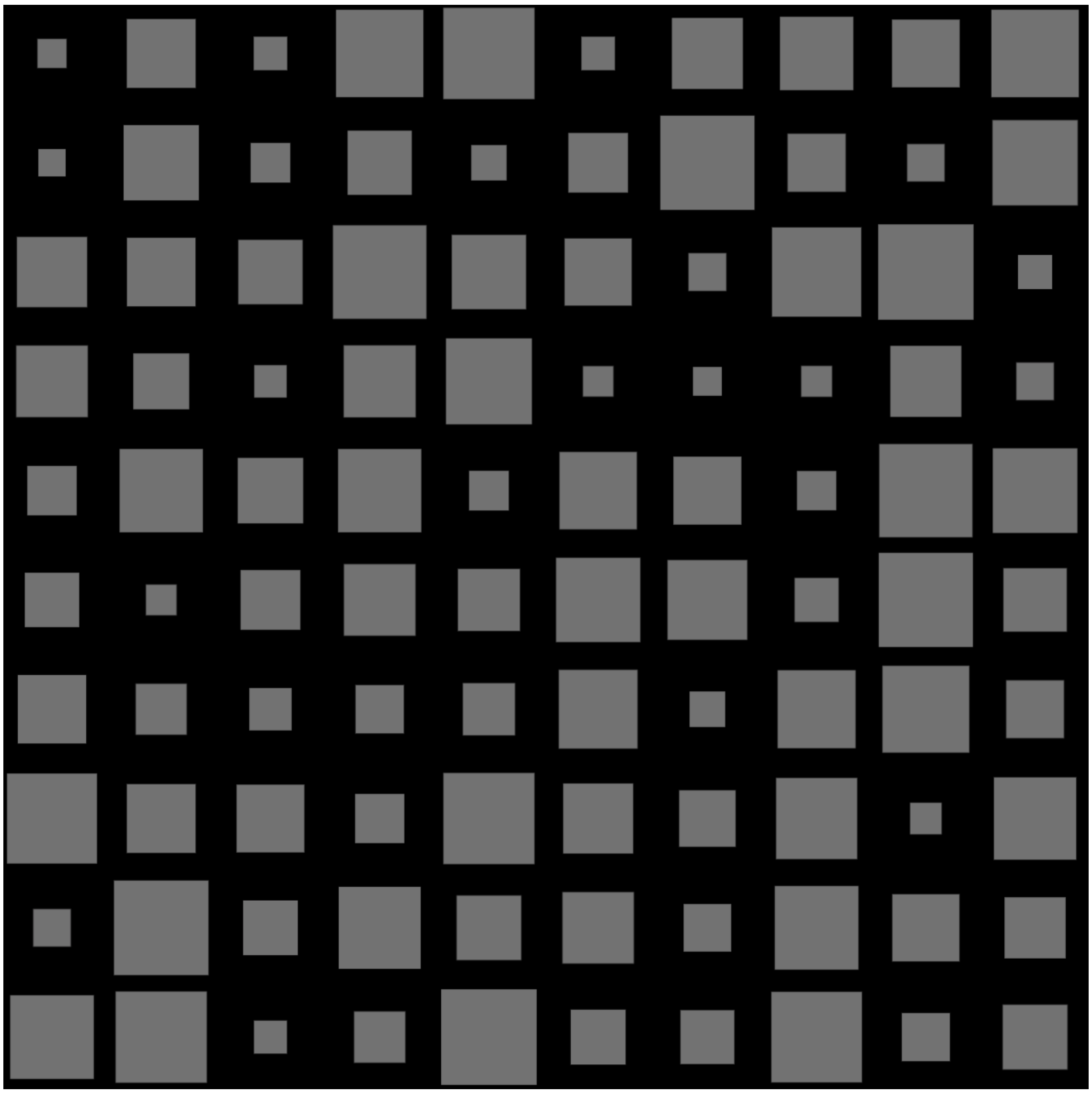
Drawing Your Objects on Screen

Now we are well on our way to create a visual output of this array. Lets start by creating a method in our class that each object will have

```
class SquareDonut{
    constructor(_posX, _posY){
        this.width = random(40,100);
        this.height =random(40,100);
        this.posX = _posX;
        this.posY= _posY;
    }

    render(){
        fill(0);
        noStroke();
        push();
        translate(this.posX,this.posY);
        rect(0,0,this.width, this.height);
        pop();
    }
}
```

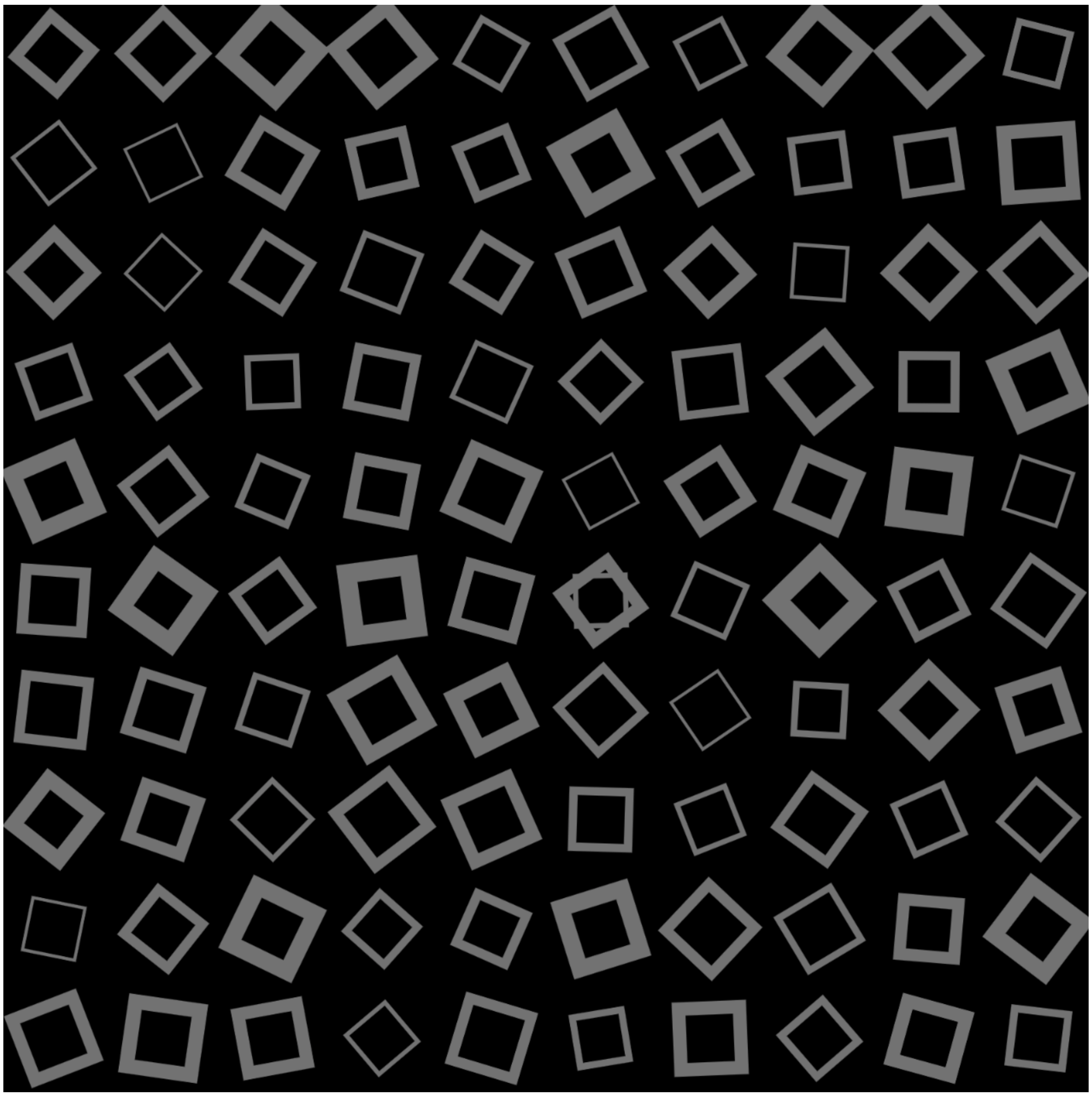
Now in our draw loop in the app.js file we need to call that method on each item in our array

```
function draw(){
    background(0);

    translate(gridSpace / 2, gridSpace / 2)
    shapes.forEach(shape => {
        shape.render()
    });
}
```



Learning More Advanced Shape Methods

Rectangles and Ellipses only let you get so far in P5. Learning to draw shapes with vertexes and shapes with holes removed is where it gets interesting.



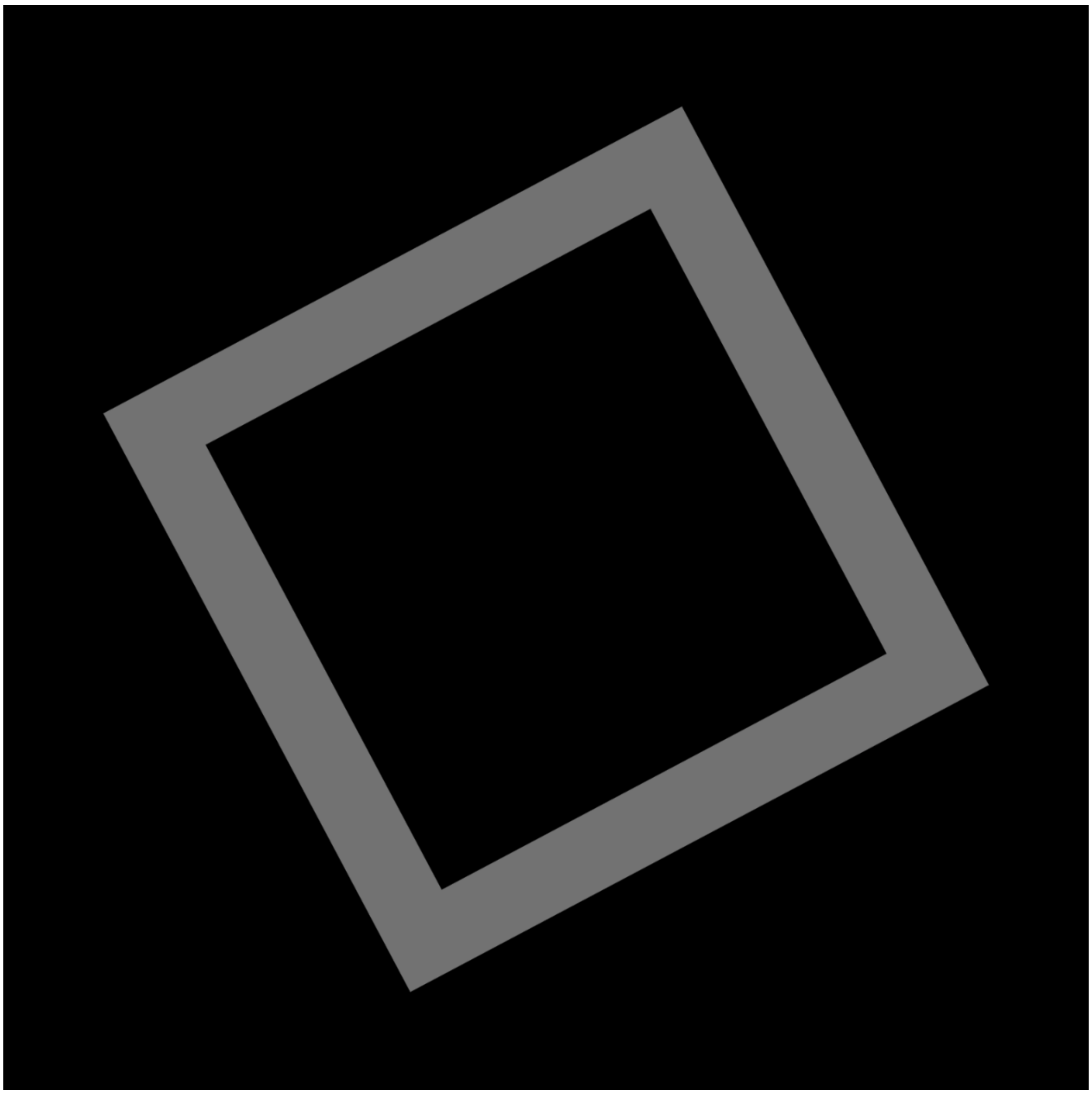
The Square Donut

Lets start with a simple SquareDonut (which is a square with a smaller square removed on the inside). To draw a square using vertices we do the following (imagine (0,0) is in the middle of the square)

```
beginShape();
vertex(-this.width / 2, -this.width / 2);
vertex(this.width / 2, -this.width / 2);
vertex(this.width / 2, this.width / 2);
vertex(-this.width / 2, this.width / 2);
endShape();
```

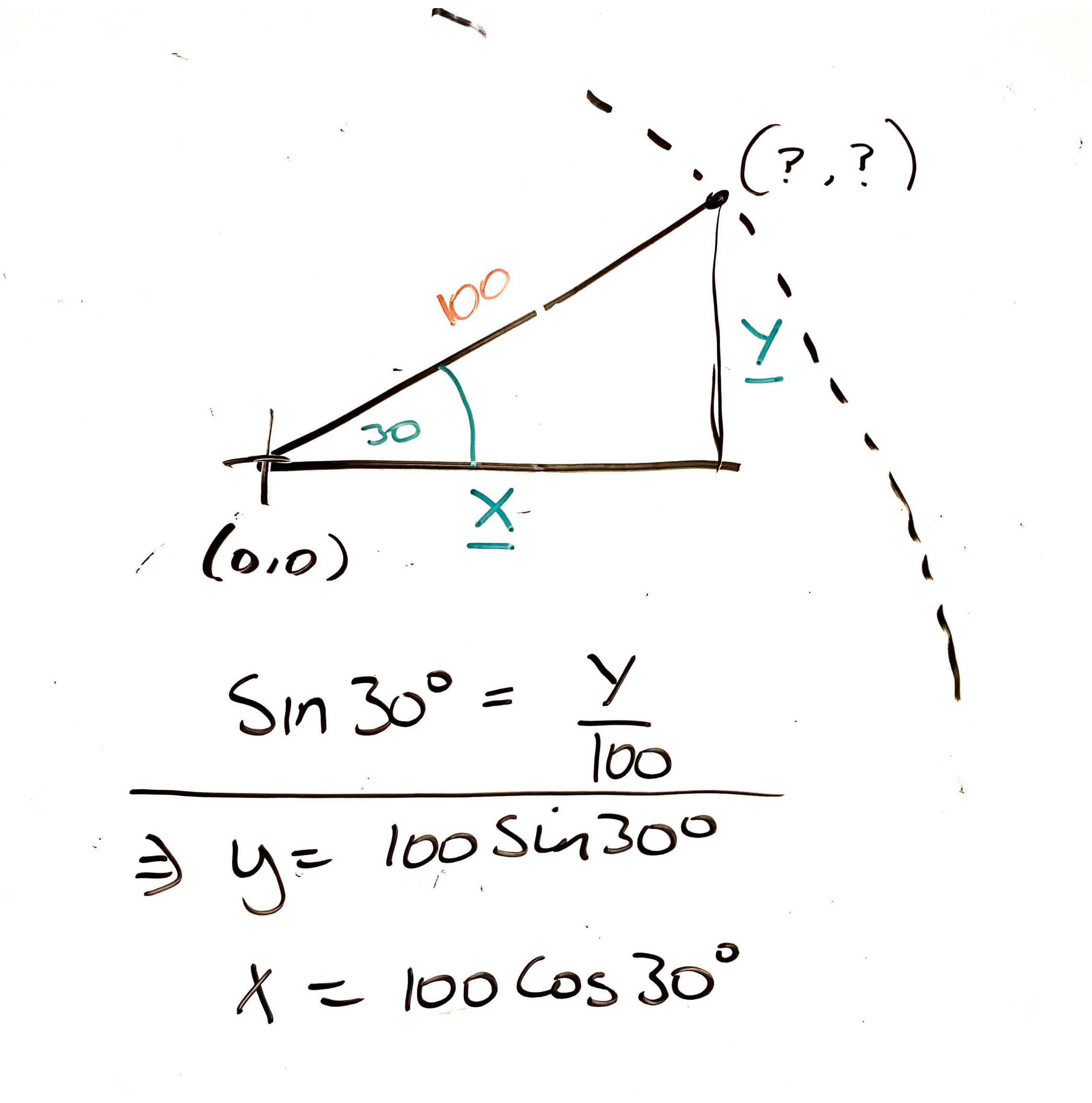
The reason we are using this.width/2 is because we are drawing from the center. Now we need to draw a smaller box but the vertices need to be drawn in the opposite direction of the outer shape

```
beginShape();
vertex(-this.width / 2, -this.width / 2);
vertex(this.width / 2, -this.width / 2);
vertex(this.width / 2, this.width / 2);
vertex(-this.width / 2, this.width / 2);

    beginContour()
        vertex(-this.widht/4, -this.widht/4);
        vertex(this.widht/4, -this.widht/4);
        vertex(this.widht/4, this.widht/4);
        vertex(-this.widht/4, this.widht/4);
    endCountour()

endShape();
```



The Polygon Donut

The Polygon requires a lttle more Maths using the Sine and Cosine functions. A polygon is many up of a number sides and the angle between each point is 360 degrees / number of Sides.
To Calculate any point on a circle when you know the angle is really simple. Using the diagram below the point on the circle could be described as (radius*Cos(Angle), radius*Sin(angle)).



If we know the radius and the angle changes then we can work out the points of a polygon. We just need to know how many side we have and we can therefore calculate the angleStep size.

```
let numberOfSides = 6
let angleStep = 360 / numberOfSides

 beginShape();
    for (let a = 0; a < 360; a = a + angleStep) {
        let posX = (this.width / 2) * cos(a);
        let posY = (this.width / 2) * sin(a);
        vertex(posX, posY)
    }
endShape()
```

And its the same to generate a polygon donut. Remember to go in the opposite direction for the inside shape

```
let numberOfSides = 6
let angleStep = 360 / numberOfSides

 beginShape();
    for (let a = 0; a < 360; a = a + angleStep) {
        let posX = (this.width / 2) * cos(a);
        let posY = (this.width / 2) * sin(a);
        vertex(posX, posY)
    }
    beginContour()
    for (let a = 360; a > 0; a = a - angleStep) {
        let posX = (this.width / 2) * cos(a);
        let posY = (this.width / 2) * sin(a);
        vertex(posX, posY)
    }
    endCountour
endShape()
```