# NM2207

# Session 10 challenge

# Overview of what we will do today:

* Create hundred circles
* Use arrays and for loops for graphical animation

**Preparatory instructions**

Copy in the code from soManyCircles3 and make it so that there are 100 circles instead of just 5.

If you want to make things from scratch, you can try out Part 1 and 2 below. Otherwise, skip directly to Part 3.

**Part 1 Warm up**

1. In the global scope:
   1. Instead of one disk, create 100 in an array using a for loop and paper.circle.

Hint: create a variable called circleList to index the individual disks in the array. This is similar to circleList in soManyCircles3.

1. Create a function called circleMover which takes in no arguments and returns no values. Inside this function,
   1. Everything you did for one disk, you will now do for each in your array.

* Use a for loop that increments your array index each time through the loop uptil the number 99 (so that there are 100 disks in all).

1. Inside the for loop
   1. Keep creating circles and adding them into your variable that stores the disks.
   2. All the circles originate at the center of the paper.
   3. Each circle has a different random xrate and a different random yrate, which we will use to move the disk at a random speed in x and y directions.
   4. Initialize the disk colors to a random hue.
2. Call the circleMover() function when the page is loaded: i.e., exactly once.

**Part 2 setInterval**

1. Rip, mash, and modify the circleMover function so that
   1. It can work with a list called circleList, and with a list that has 100 instead of 5 circles.
   2. Its bounce edges are adjusted to the new paper width and height
2. Initialize a variable called timer to a function call to the setInterval method. This method should call the circleMover function every 30 milliseconds.

**Part 3 Tracking mouse states**

1. Next, we want to change a feature of the graphical objects depending on if and where the mouse is pushed. Disks that are within a certain distance of a pressed mouse will turn white, otherwise they show their own color.
2. To do that, use Raphael to add an event listener to the paper.
   1. That is: we can’t add an eventListener directly to the paper. We need to add it either to centerDiv
   2. Or we need to add it to a transparent, filled rectangle on top of everything on the paper in order to catch mouse activity.
   3. If you choose to go with b, then graphical objects must be filled to catch mouse activity! Use the ‘fill-opacity’ attribute.
3. Add the event listeners to your filled transparent rectangle that you can use to track the mouse state. Remember that you need to get the ‘node’ attribute of Raphael graphical objects in order to use the addEventListener method.
4. Create a “state” variable (object) named mouseState with three properties (‘pushed’, ’x’, and ‘y’) to keep track of the position and of whether or not the mouse is currently pushed.
   1. Update the x and y of the mouseState variable in the mousemove event listener. The default value of mouseState.pushed should be 0. Test your mouseState code before proceeding by printing info to the console!

**Part 4 Using mouse states to update a lot of shapes**

1. In the mousedown event listener, use the distance function to check all the disks for their distance to mouseState.x and mouseState.y. If the distance is less than 100, change their fill to white.
2. Remember to update the state in the mouseup and mousedown event listeners so that we can keep track of when mousedown event ends! Revise your pizzaHog codealong to remember why we did this.