1 Draggable Point

```
console.log("point js file has been called");
class Point{
// class Point x,y,r, stroke, fill, over, canvas
constructor(x,y,r, stroke, fill, over){
    //basic position, size and colours
    this.x = x;
    this.y = y;
    this.r = r;
    this.stroke = stroke;
    this.fill = fill;
    this.over = over;
    //set true if mouse inside point circle
    this.inBounds = false;
    //cointinually registered mouse position
    this.xMouse = 0;
    this.yMouse = 0;
    //listeners
    canvas.addEventListener('mousedown', this.mDown.bind(this));
    canvas.addEventListener('mousemove', this.mMove.bind(this));
    canvas.addEventListener('mouseup', this.mUp.bind(this));
}
mDown(e){
    // if the mouse is pressed (goes down) and the mouse is inside the point circle,
    // set the this object as taken
    if(this.inBounds){
        Point.taken = this;
    }
}
mMove(e){
    // event registered every time the mouse moves
    // object variables updated with current mouse position
    this.xMouse = e.offsetX;
    this.yMouse = e.offsetY;
    //update boundary boolean
    this.inBounds = this.boundsCheck(this.xMouse, this.yMouse, this.x, this.y, this.r);
}
mUp(e){
    //when mouse goes up set taken point as nothing
    //hence deselect this point
    Point.taken = "";
}
/**
 * called from animation loop
```

```
 */
update(){
// make x,y coordinates of the point the same as the mouse position
// if the point has been taken
    if(Point.taken == this){
        this.x=this.xMouse;
        this.y=this.yMouse;
    }
    this.draw();
}
draw(){
    // change fill state if mouse is over or the point is selected
    if(this.inBounds || Point.taken == this){
    ctx.fillStyle= this.over;
    }else{
        ctx.fillStyle= this.fill;
    }
    ctx.strokeStyle = this.stroke;
    ctx.lineWidth = 2;
    ctx.beginPath()
    ctx.arc(this.x,this.y, this.r, 0, 2*Math.PI);
    ctx.fill();
    ctx.stroke();
}
/**
 * Pythagoras distance check
 * @param  x,y,positions of mouse and of point circle and radius of point circle
    (number)
 * @return boolean
 */
boundsCheck(x_1, y_1, x_2, y_2, r){
        var d = Math.sqrt( Math.pow(x_2 - x_1, 2) + Math.pow(y_2 - y_1, 2) );
        if(d<r){
            return true;
        }else{
            return false;
        }
}
/**
 * Make x, y coordinates of point available outside of object
 * @return number
 */
getX(){
    return this.x;
}
getY(){
    return this.y;
```