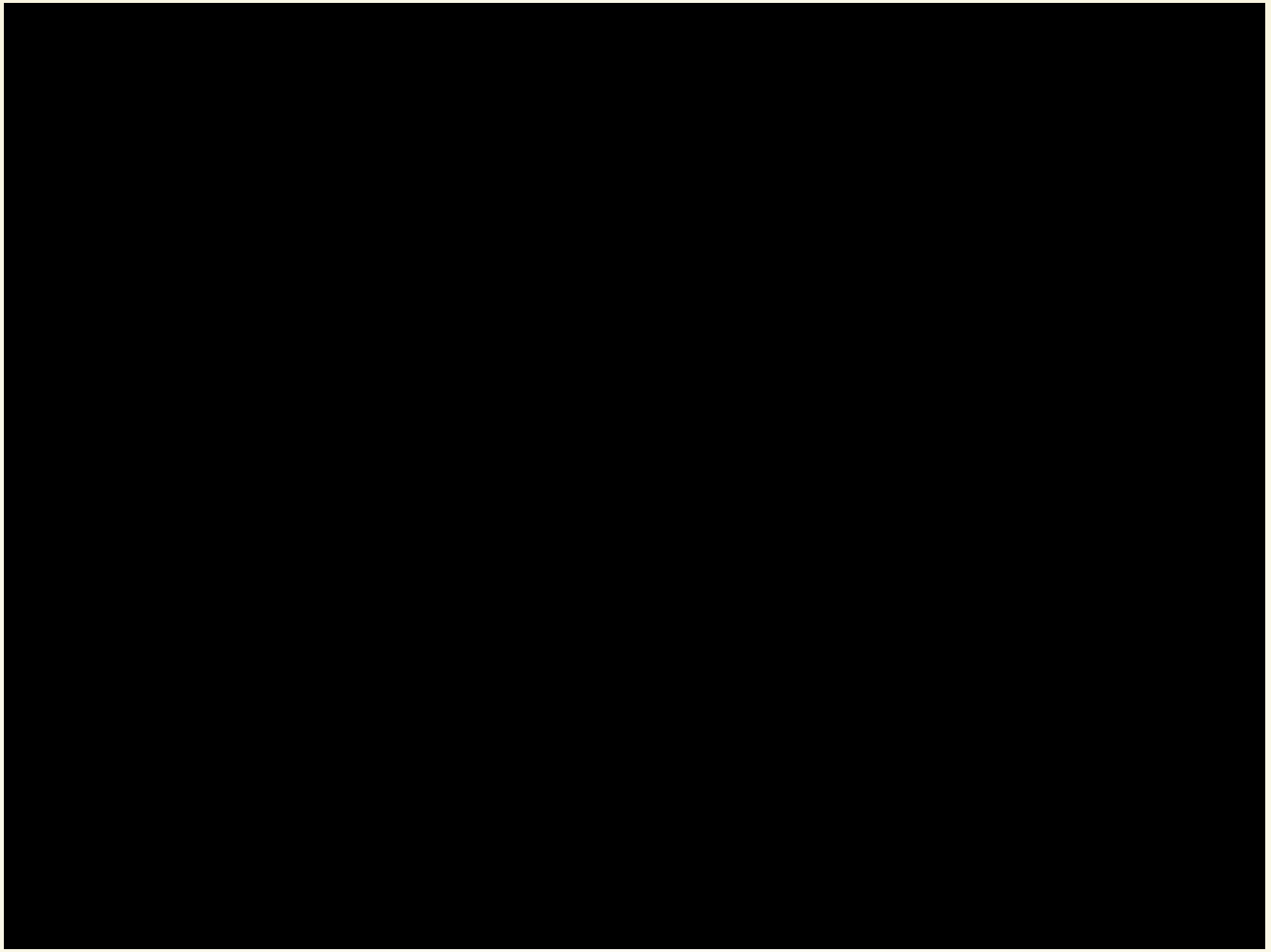
-->

GET MOUSE EVENTS

John R. Williams and *Abel Sanchez*



CALLBACK FUNCTIONS

Callback functions are used when you need to be notified when some event has occured. Often this will be when some other function has finished executing. The code below show a simple example of "attaching" a function to a reference that is being called. In this case 'callMe' is a reference. If the reference points to a function it will be called.

MOUSE EVENTS

We need to capture the mouse events. The browser gives us an easy way to do this. Every time the mouse is clicked the browser calls 'onmousedown'. We can attach our own function called, say 'trackMouse' to this event, so that it gets fired whenever the mouse is clicked. There are many "mouse events" that can be tracked and another we'll use is 'onmousemove'. However, this one produces a stream of events at a rate of around 60 per second so we'll be careful about using it.

Here is the code.

```
var x;
var y;
window.onload = Run;
function Run(){
    document.onmousedown = trackMouse;
}
function trackMouse(e) {
    x = e.pageX;
    y = e.pageY;
    alert('Mouse = ('+x+', '+y+')');
}

```


your turn now

In class Assignment 1.
Here is code to track your mouse as it moves on the canvas.

```
<html>
<head>
<script type="text/javascript">
//  This tracks Mouse Events
        var context;
        window.onload=getCanvas;
        function getCanvas(){
                context = document.getElementById("myCanvas").getContext("2d");
                document.onmousemove = drawCircles;
    }
        function drawCircles(e) {
                var posx = e.pageX;
                var posy = e.pageY;
                DrawCircle(posx,posy, 3);
        }
        function DrawCircle(x, y, radius){
                context.fillStyle = "rgba(255, 0, 0, 1)";
                context.beginPath();
                context.arc(x, y, radius, 0, Math.PI*2, true);
                context.closePath();
                context.fill();
        }
</script>
</head>
<body>
<canvas id="myCanvas" width="900" height="300" >
</canvas>
</body>
</html>
```

Change the code as follows:
Make a button that onclick="SetCreateParticle()" - see code below. Now makeParticle is called when we do 'onmousedown' on the canvas. When we 'onmouseup' we call 'letGo' and draw all the particles. You need to turn 'onmousemove' off.

```
var particles = [];
function Particle(center,radius){
  return {center:center, radius:radius};
}
function makeParticle(e){
  var radius = 10;
  particles.push(Particle(Vector(e.pageX,e.pageY),radius));
}
function letGo(){
  drawAll();
}
function drawAll(){
  var n = particles.length;
  for(i=0;i<n;i++){
    //?????  call the function to draw a circle
  }
 }
function SetCreateParticle(){
  document.onmousedown = makeParticle;
  document.onmouseup   = letGo;
}
```

In class assignment 2. - Dragging Particles across Canvas
Change the code to add another button that
onclick='SetDragParticle'.
Add a global variable pickedParticle.
so that we can pick up one of the Particles and by using 'onmousemove' drag it to a new position. You will need to write a function to "pick" the particle (see pickParticle below) and a function that redraws it as you drag it with the mouse and another 'onmouseup' to fix the particle in that new position.

Now you should be able to draw something like this - maybe better than this !!!

THE END