

```

1
2 console.log("js file has been called");
3
4
5
6
7 // Now this line will be the same size on the page
8
9 canvas = document.querySelector('#myCanvas');
10 var ctx = canvas.getContext('2d');
11 var width = 800;
12 var height = 600;
13 canvas.width = width;
14 canvas.height = height;
15
16 console.log(width);
17 console.log(height)
18 // rgb(0,0,0) rgb(153,153,153) rgb(255,255,255)
19 // rgb(204,0,0) rgb(255,204,51) rgb(51,51,255)
20 // rgb(255,102,102) rgb(255,255,153) rgb(0,153,204)
21 // draw rectangle
22 ctx.fillStyle='rgb(0,153,204)';
23 ctx.strokeStyle='rgb(0,0,0)';
24 ctx.lineWidth=10;
25 ctx.beginPath();
26 ctx.rect(10,10,100,100);
27 ctx.stroke();
28 ctx.fill();
29
30 // draw circle
31 ctx.fillStyle='rgb(255,204,51)';
32 ctx.strokeStyle='rgb(51,51,255)';
33 ctx.lineWidth=10;
34 ctx.beginPath();
35 ctx.arc(200,60, 50, 0, 2*Math.PI);
36 ctx.stroke();
37 ctx.fill();
38
39
40 // add text
41 ctx.fillStyle="rgb(0,0,255)";
42 var myFont= "bold 30px monospace";
43 ctx.font=myFont;
44 ctx.fillText("Hello World", 300,50);
45 /*
46 var BoxImg = new Image(); // Create new img element
47 BoxImg.src = 'image_test.png'; // Set source path

```

```

48 ctx.drawImage(BoxImg, 600,10,100,100);
49 */
50
51 // draw line
52 ctx.strokeStyle="rgb(255,0,0)";
53 ctx.lineWidth=0.5;
54 ctx.beginPath();
55 ctx.moveTo(0, 200);
56 ctx.lineTo(750,200);
57 ctx.stroke();
58
59 //draw polyline with closure
60 ctx.strokeStyle="rgb(255,102,102)";
61 ctx.fillStyle="rgb(255,255,153)";
62 ctx.lineWidth=10;
63 ctx.beginPath();
64 ctx.moveTo(0, 250);
65 ctx.lineTo(500,250);
66 ctx.lineTo(700,300);
67 ctx.lineTo(400,300);
68 ctx.closePath();
69 ctx.stroke();
70 ctx.fill();
71
72 // draw shape with a gradient
73 var my_gradient=ctx.createLinearGradient(10,350,10,550);
74 my_gradient.addColorStop(0,"rgb(255,102,102)");
75 my_gradient.addColorStop(0.5,"rgb(255,255,153)");
76 my_gradient.addColorStop(1,"rgb(0,153,204)");
77 ctx.fillStyle=my_gradient;
78 ctx.beginPath()
79 ctx.rect(10,350, 200,200);
80 ctx.fill();
81 ctx.stroke();
82
83 // quadratic curves (bezier)
84 ctx.strokeStyle="rgb(255,0,0)";
85 ctx.beginPath();
86 ctx.moveTo(300,400);
87 ctx.lineWidth=10;
88 ctx.quadraticCurveTo(500, 550, 700, 400);
89 ctx.stroke();
90
91
92
93 // draw circle
94 ctx.fillStyle='rgb(255,204,51)';

```

```
95 ctx.strokeStyle='rgb(51,51,255)';
96 ctx.lineWidth=10;
97 ctx.beginPath();
98 ctx.arc(width/2,height/2, 50, 0, 2*Math.PI);
99 ctx.stroke();
100 ctx.fill();
```

Listing 1: Python example

1 Functions

```
1 /**
2  * Draw a rectangle
3  *
4  * @param {number} x corner x
5  * @param {number} y corner y
6  * @param {number} w width
7  * @param {number} h height
8  * @param {string} fillcolour rgb string
9  * @param {string} strokecolour rgb string.
10 * @param {number} strokewidth x coordinate of second point.
11 * @return {null}
12 */
13 function drawRect(x,y,w,h, fillcolour, strokecolour, strokewidth){
14     ctx.fillStyle = fillcolour;
15     ctx.strokeStyle = strokecolour;
16     ctx.lineWidth = strokewidth;
17     ctx.beginPath()
18     ctx.rect(x,y,w,h)
19     ctx.fill();
20     ctx.stroke();
21 }
22 // call the function to make a rectangle
23 drawRect(700,100,250, 450, "rgb(240, 100, 80)", "rgb(0, 100, 80)", 3)
24 /**
25  * Draw a circle
26  *
27  * @param {number} x corner x
28  * @param {number} y corner y
29  * @param {number} r radius
30  * @param {string} fillcolour rgb string
31  * @param {string} strokecolour rgb string.
32  * @param {number} strokewidth x coordinate of second point.
33  * @return {null}
34  */
35 function drawCircle(x,y,r, fillcolour, strokecolour, strokewidth){
36     ctx.fillStyle = fillcolour;
37     ctx.strokeStyle = strokecolour;
38     ctx.lineWidth = strokewidth;
39     ctx.beginPath()
40     ctx.arc(x,y,r, 0, 2*Math.PI)
41     ctx.fill();
42     ctx.stroke();
43 }
44 drawCircle(700,500,50, "rgb(0, 255, 80)", "rgb(0, 100, 255)", 8)
45 /**
```

```

46 * Draw a white line between two points
47 *
48 * @param {number} x_1 x coordinate of first point.
49 * @param {number} y_1 y coordinate of first point.
50 * @param {number} x_2 x coordinate of second point.
51 * @param {number} y_2 y coordinate of second point.
52 * @return {null}
53 */
54 function draw_line(x_1, y_1, x_2,y_2){
55     ctx.strokeStyle="rgb(255,255,255)";
56     ctx.lineWidth=0.25;
57     ctx.beginPath();
58     ctx.moveTo(x_1, y_1);
59     ctx.lineTo(x_2,y_2);
60     ctx.stroke();
61 }
62 draw_line(0,400, 600,100)
63 //use the drawline method to make a grid
64 /**
65 * Draw a grid line between two points
66 *
67 * @param {number} n width and height of each grid square
68 * @return {null}
69 */
70 function draw_grid(n){
71     var grid_interval = n;
72     for(var i=0; i< width/grid_interval; i++){
73         draw_line(i*grid_interval,0,i*grid_interval,height);
74     }
75     for(var i=0; i< height/grid_interval; i++){
76         draw_line(0,i*grid_interval,width,i*grid_interval);
77     }
78 }
79 // call the function and draw the grid
80 draw_grid(50);
81 /**
82 * Draw a white line between two points
83 *
84 * @param {number} x_1 x coordinate of first point.
85 * @param {number} y_1 y coordinate of first point.
86 * @param {number} x_2 x coordinate of second point.
87 * @param {number} y_2 y coordinate of second point.
88 * @return {null}
89 */
90 function text_box(x,y,w,h, bCol, tCol, message){
91     ctx.fillStyle=bCol;
92     ctx.strokeStyle='rgb(255,255,255)';

```

```

93     ctx.lineWidth=1;
94     //create and fill-draw the rectangle
95     ctx.beginPath();
96     ctx.rect(x,y,w,h);
97     ctx.fill();
98     ctx.stroke();
99     // reset the context for the text color
100    ctx.fillStyle=tCol;
101    var myFont= "bold 25px monospace";
102    // position and draw text in middle of box
103    ctx.font=myFont;
104    ctx.textBaseline = 'middle';
105    ctx.textAlign = "center";
106    var output = message;
107    ctx.fillText(output, x+w/2,y+h/2);
108 }
109 // create one text box
110 text_box(0,0,300,50, "rgb(100,200,0)", "rgb(255,255,255)", "Little Text");
111 // create a set using an array and a loop
112 box_list = ["hello", "goodbye", "see you"]
113 box_height = 50;
114 for(var i =0 ; i<box_list.length; i++){
115 text_box(300,200+i*box_height,300,box_height, "rgb(0,0,100)", "rgb(255,255,255)",
        box_list[i]);
116 }
117
118 /**
119  * Draw a rectangle with rounded edges
120  *
121  * @param {number} x_1 x coordinate of first point.
122  * @param {number} y_1 y coordinate of first point.
123  * @param {number} x_2 x coordinate of second point.
124  * @param {number} y_2 y coordinate of second point.
125  * @return {null}
126  */
127 function rounded_rectangle(x,y,w,h, bCol = "rgb(0,0,255)") {
128     console.log("function called")
129     ctx.fillStyle=bCol;
130     ctx.lineWidth=1;
131     // corner radius cannot be more than half the height
132     var rad = 100;
133     if(rad > h/2){
134         rad = h/2;
135     }
136     ctx.beginPath();
137     // draw in order the 4 quater circles of the rounded rectangle edges
138     // straight lines will automatically connect them

```

```
139     ctx.arc(x+rad,y+rad, rad, Math.PI,3*Math.PI/2 );
140     ctx.arc(x+w-rad,y+rad, rad, 3*Math.PI/2,0 );
141     ctx.arc(x+w-rad,y+h-rad, rad,0,Math.PI/2 );
142     ctx.arc(x+rad,y+h-rad, rad,Math.PI/2,Math.PI );
143     ctx.closePath();
144     ctx.fill();
145     ctx.stroke();
146 }
147 rounded_rectangle(50,100,200,50);
```

Listing 2: Python example

2 Draggable Point

```
1 console.log("point js file has been called");
2 class Point{
3 // class Point x,y,r, stroke, fill, over, canvas
4 constructor(x,y,r, stroke, fill, over){
5     //basic position, size and colours
6     this.x = x;
7     this.y = y;
8     this.r = r;
9     this.stroke = stroke;
10    this.fill = fill;
11    this.over = over;
12    //set true if mouse inside point circle
13    this.inBounds = false;
14    //continually registered mouse position
15    this.xMouse = 0;
16    this.yMouse = 0;
17    //listeners
18    canvas.addEventListener('mousedown', this.mDown.bind(this));
19    canvas.addEventListener('mousemove', this.mMove.bind(this));
20    canvas.addEventListener('mouseup', this.mUp.bind(this));
21 }
22 mDown(e){
23     // if the mouse is pressed (goes down) and the mouse is inside the point
    circle,
24     // set the this object as taken
25     if(this.inBounds){
26         Point.taken = this;
27     }
28 }
29 mMove(e){
30     // event registered every time the mouse moves
31     // object variables updated with current mouse position
32     this.xMouse = e.offsetX;
33     this.yMouse = e.offsetY;
34     //update boundary boolean
35     this.inBounds = this.boundsCheck(this.xMouse, this.yMouse, this.x, this.y,
    this.r);
36 }
37 mUp(e){
38     //when mouse goes up set taken point as nothing
39     //hence deselect this point
40     Point.taken = "";
41 }
42 /**
43  * called from animation loop
```



```

44  */
45  update(){
46  // make x,y coordinates of the point the same as the mouse position
47  // if the point has been taken
48      if(Point.taken == this){
49          this.x=this.xMouse;
50          this.y=this.yMouse;
51      }
52      this.draw();
53  }
54  draw(){
55      // change fill state if mouse is over or the point is selected
56      if(this.inBounds || Point.taken == this){
57          ctx.fillStyle= this.over;
58      }else{
59          ctx.fillStyle= this.fill;
60      }
61      ctx.strokeStyle = this.stroke;
62      ctx.lineWidth = 2;
63      ctx.beginPath()
64      ctx.arc(this.x,this.y, this.r, 0, 2*Math.PI);
65      ctx.fill();
66      ctx.stroke();
67  }
68  /**
69   * Pythagoras distance check
70   * @param x,y,positions of mouse and of point circle and radius of point circle
71   * (number)
72   * @return boolean
73   */
74  boundsCheck(x_1, y_1, x_2, y_2, r){
75      var d = Math.sqrt( Math.pow(x_2 - x_1, 2) + Math.pow(y_2 - y_1, 2) );
76      if(d<r){
77          return true;
78      }else{
79          return false;
80      }
81  }
82  /**
83   * Make x, y coordinates of point available outside of object
84   * @return number
85   */
86  getX(){
87      return this.x;
88  }
89  getY(){
90      return this.y;

```

```
90 }  
91 }  
92 // static variable available to all Point objects  
93 // the same for all Point objects  
94 // means only one Point can be selected and moveable  
95 Point.taken="";
```

Listing 3: Python example