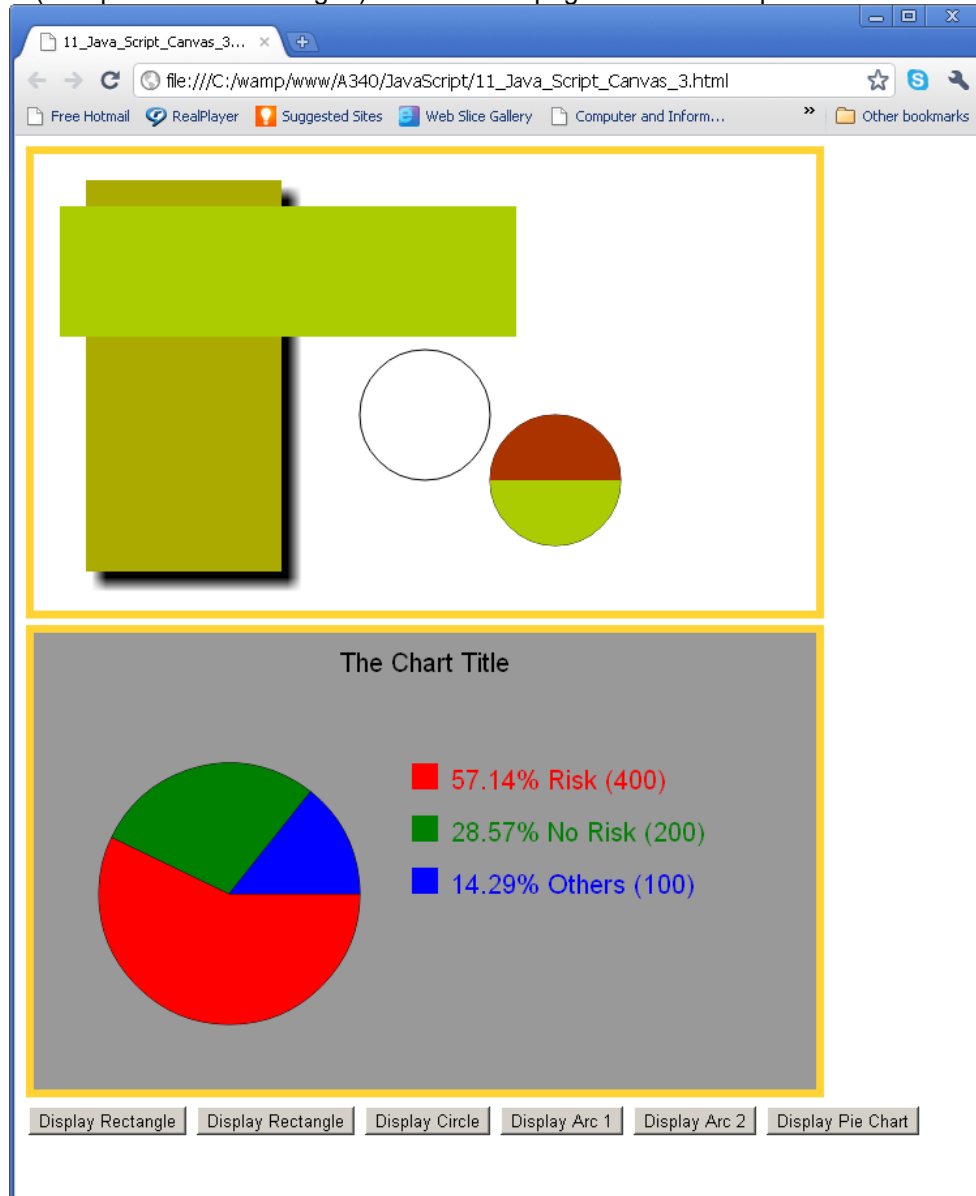


Laboratory Session #8

LAB GOALS

- Having some fun with HTML5 Canvas Graphics and JavaScript

Using your text editor (Notepad++ / TextWrangler) create a web page similar to the picture shown below.



Step 1: Start with a simple HTML web site similar to figure below, just make sure you put in a `<script>` tag in it:

```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript">
</script>
</head>
<body>
</body>
</html>
```

Step 2: Place the following documentation lines in the HTML file, right below the `<!DOCTYPE html>`. Make sure these are placed in the HTML documentation tag.

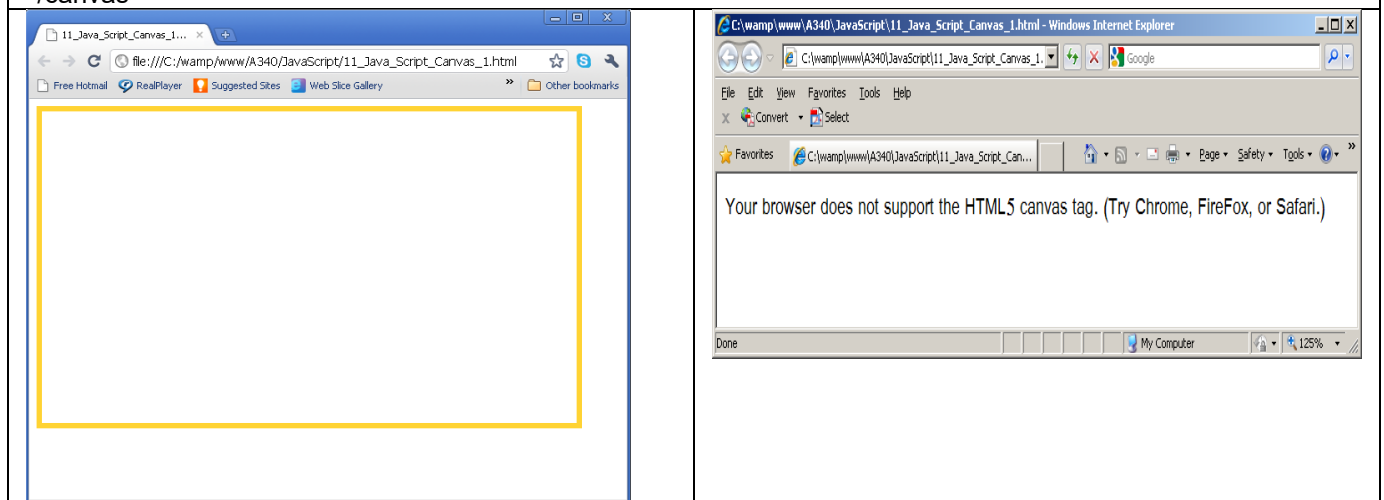
Your name
CSCI-490 Client Server Web Programming
Lab: 8 (JavaScript, HTML 5 Canvas, DOM manipulation, Event handling)
Semester, Year

Step 3: Set the title of the web page to "HTML5, Canvas Graphics, JavaScript". Then set the keywords for the page to JavaScript, Web Programming, Canvas Graphics, and Event Handling. Finally make sure the character set for the web page is set to UTF-8.

Step 4: In this lab we would like to have some fun with HTML5 Canvas Graphics. It should be noted that this feature of HTML 5 may not be fully implemented in all browser (notably some versions of Internet Explorer). Therefore for this lab, you want to stay with Google Chrome or another browser that supports Canvas.

Step 5: To start, place the following code inside the body of your HTML document. Save your file and render it using both Chrome and IE.

```
<canvas id="myCanvas1"
      width="600"
      height="350"
      style="border:6px solid #FFd333; background:#FFFFFF">
  Your browser does not support the HTML5 canvas tag. (Try Chrome, FireFox, or Safari.)
</canvas>
```



Step 6: Next we want to place a rectangle on our canvas. We will do this in two parts:

- Create a function called "displayRectangle()
- Create a button which can be used to call the displayRectangle()

To complete part a, copy the code shown in the right, to the <script> section of your document.

Note: To draw 2d objects such as lines, circles, rectangles, etc. we need to create a 2d context. For 3d objects we can use webgl context.

Examples:

<http://www.babylonjs.com/>

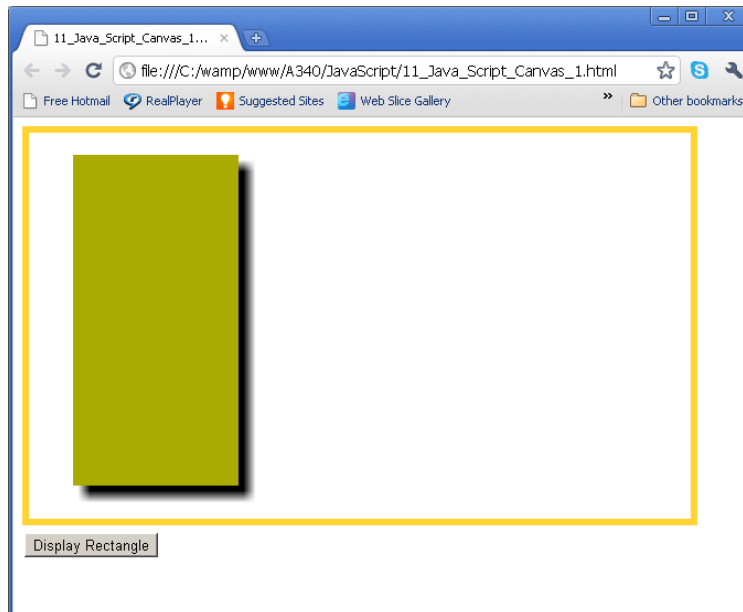
```
function displayRectangle(x,y, width, height, theColor, theCanvas, theShadow)
{
    var myCanvas = document.getElementById(theCanvas);
    var ctx = myCanvas.getContext("2d");
    ctx.fillStyle = theColor;

    if (theShadow == true)
    {
        ctx.shadowBlur=10;
        ctx.shadowOffsetX = 10;
        ctx.shadowOffsetY = 10;
        ctx.shadowColor = 'black';
    }
    else
    {
        ctx.shadowBlur=0;
        ctx.shadowOffsetX = 0;
        ctx.shadowOffsetY = 0;
    }
    ctx.fillRect(x,y, width, height);
}
```

To complete part b, copy the tag shown in the right to the <body> section of your document, below the <canvas> tag.

```
<button onclick="displayRectangle(40,20,150,300, '#AAAA00', 'myCanvas1', true);" >
    Display Rectangle
</button>
```

Clicking the button will create the following. You can learn more about each of the canvas functions used above in the w3Schools.com web site.



Step 7: Next we want to place another rectangle on our canvas. Note that all we need to do is to add another button to the body of the document, which calls our displayRectangle() function and passes the right parameters to that function. Place the following button in your document and render it.

```
<button onclick="displayRectangle(20,40,350,100, '#AACC00', 'myCanvas1', false);" >
    Display Rectangle
</button>
```

Step 8: Creating Arcs and Circles. Canvas provides us with the ability to draw arcs and circles. The built-in arc() function requires the center point of the arc, followed by the radius, starting angle and ending angle. Note that the angles are provided in radians, not degrees. (e.g., from 0 radians to 2PI)

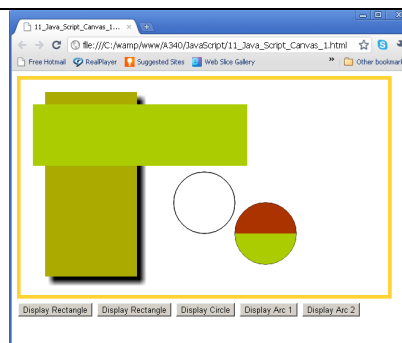
```
//-----
function displayCircle(centerX,centerY, radius, theColor, theCanvas)
{
    var myCanvas = document.getElementById(theCanvas);
    var ctx = myCanvas.getContext("2d");
    ctx.fillStyle = theColor;
    ctx.beginPath();           // Start a new starting point for a path (drawing)
    ctx.arc(centerX, centerY,radius, 0, 2*Math.PI);
    ctx.stroke();              // make the path visible (draw it) on the canvas
    //ctx.fill();              // commented out to prevent fill
}
//-----
function displayArc(centerX,centerY, radius, startAngle, endAngle, direction, theColor, theCanvas)
{
    var myCanvas = document.getElementById(theCanvas);
    var ctx = myCanvas.getContext("2d");
    ctx.fillStyle = theColor;
    ctx.beginPath();
    ctx.arc(centerX, centerY,radius,startAngle,endAngle);
    ctx.stroke();
    ctx.fill();
}
```

The above functions have been parameterized so that we can easily create circles and arcs without having to type several lines of code. Enter the above under your <script> tag and then put the following buttons in your body tag, and try rendering your page again. Click on the button to see what happens.

```
<button onclick="displayCircle(300, 200, 50, '#AACC00', 'myCanvas1');" > Display Circle </button>

<button onclick="displayArc(400, 250, 50, 0*Math.PI, 1*Math.PI, 'clockwise', '#AACC00', 'myCanvas1');" > Display
Arc 1</button>

<button onclick="displayArc(400, 250, 50,1*Math.PI, 2*Math.PI, 'clockwise', '#AA3300', 'myCanvas1');" > Display
Arc 2 </button>
```



Step 9: Let's get creative. How about using what we have learned and put together a function for creating PIE charts?

```
function displayPieChart(theTitle, centerX, centerY, radius, myData, myLabel, theColors, theCanvas)
{
```

```
    // Any ideas how you would do this?
```

```
    // Start this on your own and if you have questions and need pointers see the hint or let me know.
```

```
}
```

```
<body>
```

```
<canvas id="myCanvas2"
```

```
    width="600"
```

```
    height="350"
```

```
    style="border:6px solid #FFd333; background:#999999">
```

```
    Your browser does not support the HTML5 canvas tag. (Try Chrome, FireFox, or Safari.)
```

```
</canvas>
```

```
<br />
```

```
<script>
```

```
    var myData = [400, 200, 100];
```

```
    // create an array to hold the data
```

```
    var myLabels = new Array('Risk','No Risk', 'Others');
```

```
    // create an array for the labels
```

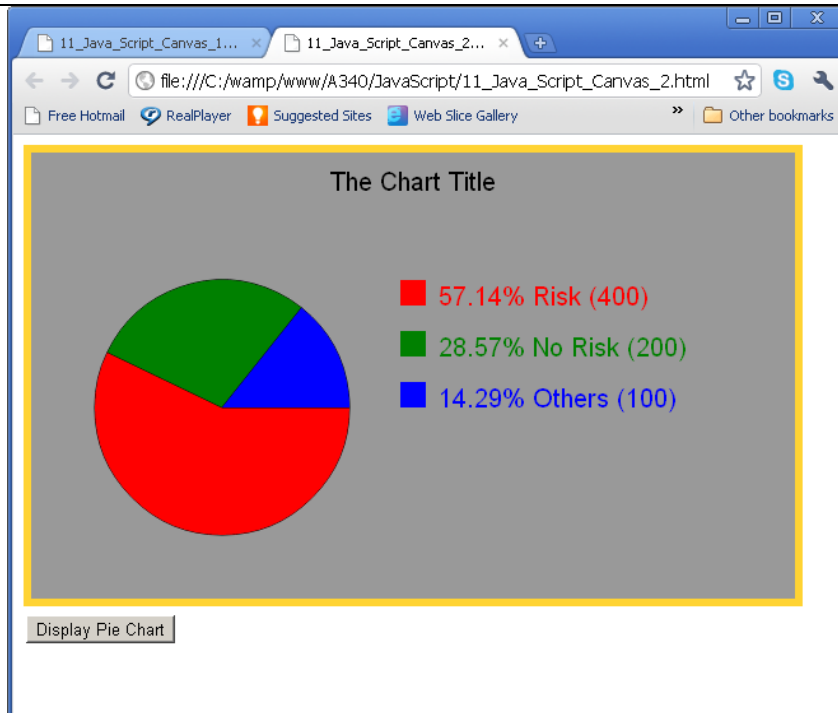
```
    var myColors = new Array('red', 'green', 'blue');
```

```
    // array for the colors
```

```
</script>
```

```
    <button onclick="displayPieChart('The Chart Title', 150, 200, 100, myData, myLabels, myColors, 'myCanvas2');" >
    Display Pie Chart </button>
```

```
</body>
```



Hint:

```
function displayPieChart(theTitle, centerX, centerY, radius, myData, myLabel, theColors, theCanvas)
{
    // Calculate the sum of all slices in the data array (400+200+100 = 700)
    // simply add up the numbers for each slice 400+200+100 and store it in sumArray. Note that this info is in myData array.

    var sumArray = 0;
    for(var i=0; i< myData.length; i++)
    {
        sumArray += myData[i];
    }

    // Calculate the slices in radians
    // The first slice would be 400/700 * (2PI). Notice the entire circle is 2PI radians, therefore
    // the first slice would be 4/7 of that circle. The second slice would be 2/7 of the circle and
    // the last slice will be 1/7. The slices() array below should hold each slice in radians.

    var slices = Array();
    for(var i=0; i< myData.length; i++)
    {
        slices[i] = myData[i]/ sumArray * 2 * Math.PI;
        //document.write(slices[i] + "Radians <br />");
    }

    // Calculate the slices as percentages (to be used by the chart legend)
    // Note this is very similar to what we did for radians, except instead of using radians we are
    // using percentages. So, 400/700 * (100) = 57%.
    // The percentages() array below should hold each slice in percentages.

    var percentages = Array();
    for(var i=0; i< myData.length; i++)
    {
        percentages[i] = myData[i]/ sumArray * 100;
        //document.write(percentages[i] + "%, ");
    }
    var direction = "clockwise";

    //-----
    // Setup the canvas and the context
    //-----
    var myCanvas = document.getElementById(theCanvas);
    var ctx = myCanvas.getContext("2d");
    ctx.color = 'blue';
    ctx.lineWidth = 1;

    //-----
    // Display the Chart Title
    //-----
    ctx.font = '20px Arial';

    // Calculate the Center of the canvas and position the title properly
    var titleX = (myCanvas.width / 2) - (ctx.measureText(theTitle).width / 2);
    ctx.fillText(theTitle, titleX, 30);
}
```

```

//-----
// Display the Pie Chart
//-----
var startAngle = 0;
for (var i =0; i < slices.length; i++)
{
    endAngle  = startAngle + slices[i];
    ctx.fillStyle = theColors[i];
    ctx.beginPath();
    // Finally, draw the bounding lines that creates each pie in the pie chart
    ctx.moveTo(centerX, centerY);           // go to center
    ctx.arc(centerX, centerY, radius, startAngle, endAngle, false); // draw arc
    ctx.lineTo(centerX, centerY);           // back to center
    ctx.stroke();
    ctx.fill();
    startAngle = endAngle;
}
//-----
// Display the Legend
for (var i =0; i < slices.length; i++)
{
    ctx.fillStyle = theColors[i];

    var x = centerX + radius * 1.4 ;
    var y = centerY - radius + i * 40;
    var width = 20;
    var height = 20;

    ctx.fillRect(x,y, width, height);

    ctx.font ='20px Arial';
    ctx.fillText(percentages[i].toFixed(2) + "% " +
        myLabel[i] +
        " (" + myData[i] + ")",
        x+30, y+20);
}
}

```