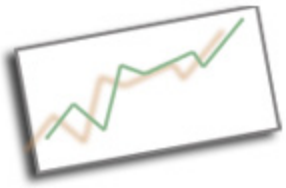
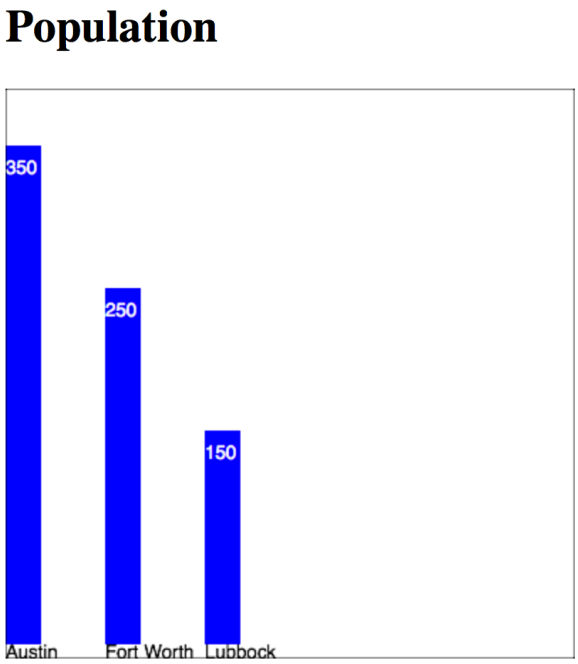


CodeActually
Dr. Cindy Royal
Texas State University
School of Journalism and Mass Communication

HTML5 Canvas

So far, we've been working with divs. But HTML5 has offered another, more efficient way to draw shapes using the <canvas> tag. This is one way that many data visualization tools use to draw charts, like Chart.js. It is supported by most modern browsers. Other tools, like D3, use the SVG format, which is currently not fully supported by all browsers.



Let's work with our similar charting data from above. The <canvas> itself is where the chart will be drawn. Make a page with a div to hold the canvas. Work with these dimensions for now. We can adjust later. For now, I made the width 900 to accommodate the largest bar we need to draw.

```
<!DOCTYPE html>
<html>
<head>

<script>

</script>
```

```
</head>

<body>
<div>
<h1>Population</h1>
<canvas id="canvas" height="400" width="400"></canvas>
</div>


</body>
</html>
```

That is the basic page setup. Now we need to write the script to create the shapes on the page.

In the script tag, include this code:

```
<script>
var canvas = document.getElementById("canvas");
var context = canvas.getContext("2d");
context.fillStyle = "blue";
context.fillRect(0,0, 300, 25);
</script>
```

This will draw a filled, blue rectangle, starting at the 0,0 (top, left) coordinate on the canvas. We set the context variable to establish the canvas as two dimensional. Then we set the fillStyle and the rectangle dimension for the context variable. Notice it starts at the top, so we will have to make adjustments as we did above to start at the bottom.

We can add some other bars below with different dimension and placement on the page. Notice how we can establish the exact location on the canvas, which will help us when we decide to make the chart vertically.

```
<script>
var canvas = document.getElementById("canvas");
var context = canvas.getContext("2d");
context.fillStyle = "blue";

context.fillRect(0,0, 300, 25);
context.fillRect(0,40, 200, 25);
context.fillRect(0,80, 100, 25);

</script>
```

Now, let's add some text to the chart. We have to establish a new fillStyle and a font for the text. We put this after the code to draw the rectangles, so it will create the text next. Position the numbers in the code toward the end of the bar. Put the names right under the bar.

```
<script>
var canvas = document.getElementById("canvas");
var context = canvas.getContext("2d");
context.fillStyle = "blue";

context.fillRect(0,0, 842, 25);
context.fillRect(0,40, 777, 25);
context.fillRect(0,80, 236, 25);

// this section puts the numbers at the end of the bar
context.fillStyle = "white";
context.font = "12pt Helvetica";

context.fillText('842', 800 , 20);
context.fillText('777', 700 , 60);
context.fillText('236', 200 , 100);

//this section puts the city names under the bars
context.fillStyle = "black";

context.fillText('Austin', 0, 40);
context.fillText('Fort Worth', 0, 90);
context.fillText('Lubbock', 0, 140);

</script>
```

That's it. Now you have created a chart using the <canvas> tag. We can manipulate the script to make it vertical with this code replacing the code above. We can also read from JSON data and put that into the city names of dimensions. Let's use some smaller dimensions for demonstration for the chart so it will fit in our 400 px height canvas.

You will have to do a little math to get the rectangles to start drawing in the right place and to finish just before the end of the canvas. You need to leave a little room for the text at the bottom and be able to add the numbers for the bars at the top.

Let's let JavaScript do the math. I put the data I am using in a JSON section as we did in the previous exercises. I made a variable for the space between the columns and I made a variable for the width of the bars. Then I went in and changed all the positioning, dimension and text to use these variables. I used math the determine where the rectangles would start drawing (canvas height minus 10 minus the desired height) and where to place the numbers at the top of the bar (canvas height plus ten minus height of bar). Feel free to work with these variables and dimensions.

```
<script>

var cities = [
{name: "Austin", population: 350},
{name: "Fort Worth", population: 250},
{name: "Lubbock", population: 150}
];

var spacebetween = 70;
var barwidth = 25;
var canvas = document.getElementById("canvas");
var context = canvas.getContext("2d");
context.fillStyle = "blue";

context.fillRect(0, (canvas.height-10)-
cities[0].population, barwidth, cities[0].population);
context.fillRect(spacebetween, (canvas.height-10)-
cities[1].population, barwidth, cities[1].population);
context.fillRect(spacebetween*2, (canvas.height-10)-
cities[2].population, barwidth, cities[2].population);

context.fillStyle = "white";
context.font = "10pt Helvetica";

context.fillText(cities[0].population, 0 ,
(canvas.height+10)- cities[0].population);
context.fillText(cities[1].population, spacebetween ,
(canvas.height+10)- cities[1].population);
context.fillText(cities[2].population, spacebetween*2 ,
(canvas.height+10)- cities[2].population);

context.fillStyle = "black";
context.fillText(cities[0].name, 0, canvas.height);
context.fillText(cities[1].name, spacebetween,
canvas.height);
```

```
    context.fillText(cities[2].name, spacebetween*2,
    canvas.height);

</script>
```

One last thing. Let's put a box around the canvas. Add this at the bottom of the JavaScript before you close the script tag. You might have to make a few adjustments to get it to look just right, reduce the width of the canvas, change the amount of space you leave at the bottom for text.

```
    context.strokeStyle = "black";
    context.strokeRect(0, 0, canvas.width, canvas.height);
```

Circles

Canvas has the ability to create arcs. Replace the script for our canvas with the following to see how it creates a circle.

```
    <script>
    var canvas = document.getElementById("canvas");
    var context = canvas.getContext("2d");

    context.fillStyle = "blue";
    context.beginPath();
    context.arc(50, 50, 50, 0, Math.PI*2, true);
    context.fill();

    context.fillStyle = "red";
    context.beginPath();
    context.arc(50, 50, 40, 0, Math.PI*2, true);
    context.fill();
    </script>
```

You will see two circles, one inside the other. This is generally what tools like Chart.js will use to make pie charts. This just demonstrates how it works. The arc feature does not create wedges. For now, just understand how the arc function uses the five parameters:

1st and 2nd parameter are the location of the circle's center on the canvas.
3rd parameter is the radius of the circle.
4th parameter is the starting angle.
5th parameter is the ending angle.

6th optional parameter is for counter clockwise or clockwise drawing of the circle.

You can create a half circle by changing the ending angle to Math.PI. Play around with these parameters to see what happens.

Now you have a general understanding of the canvas and should be able to work with any programs or tools that use this element!

Be sure to reference the code libraries for working examples of each of these exercises.