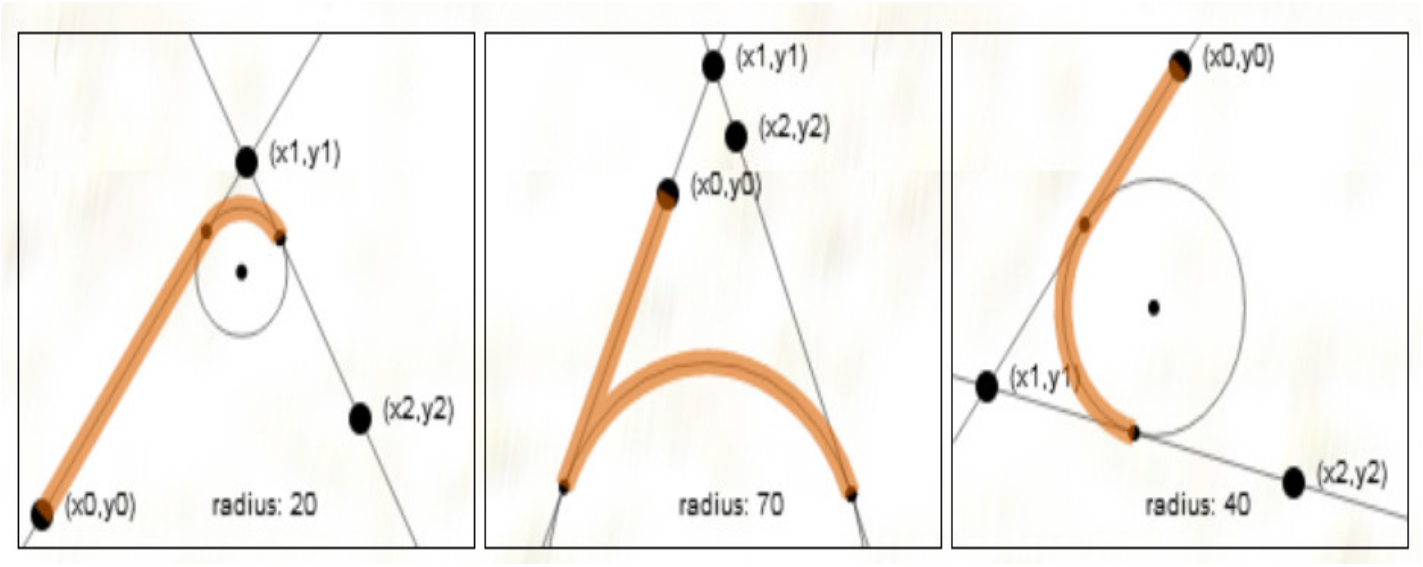
Drawing rounded rectangles: the `arcTo(x1, y1, x2, y2, radius)` method

INTRODUCTION

There is another method called `ctx.arcTo(x1, y1, x2, y2, radius)`, which is a bit complex to use, but very practical for drawing rounded rectangles.

In fact, the `arcTo(...)` method draws an arc of a circle depending on some tangents. Let's look at these pictures for a better understanding (original picture from http://www.dbp-consulting.com/tutorials/canvas/CanvasArcTo.html):



TYPICAL USE

```
ctx.moveTo(x0, y0);
ctx.arcTo(x1, y1, x2, y2, radius);
```

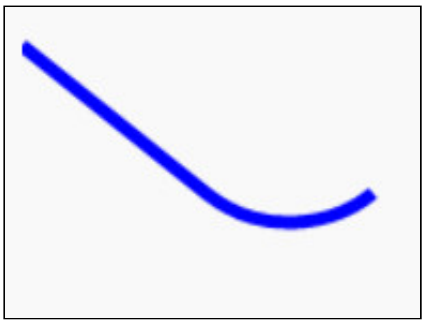
This method can be confusing. It was defined mainly for drawing rounded shapes like

rounded rectangles. We used an excerpt here from the excellent tutorial on the `arcTo(...)` method available at http://www.dbp-consulting.com/tutorials/canvas/CanvasArcTo.html.

It works like this:

1. Draw an imaginary line through `(x0,y0)` and `(x1,y1)`, draw another imaginary line through `(x1,y1)` and `(x2,y2)`,
2. Take an imaginary circle of radius `r`, and slide it up between the two lines until it just touches both lines. The two points at which the circle touches the lines are called the tangent points.
3. `arcTo(x1, y1, x2, y2, r)` will draw a line from the current point `(x0,y0)` to the first tangent point on the line from `(x0,y0)` to `(x1,y1)`,
4. It will also draw an arc from that tangent point to the other tangent point on the line from `(x1,y1)` to `(x2,y2)` along the circumference of the circle.
5. Finally, it adds the tangent point where the arc ends up, on the line from `(x1,y1)` to `(x2,y2)` to the path as the new current point on the path.

EXAMPLE 1: SIMPLE USE



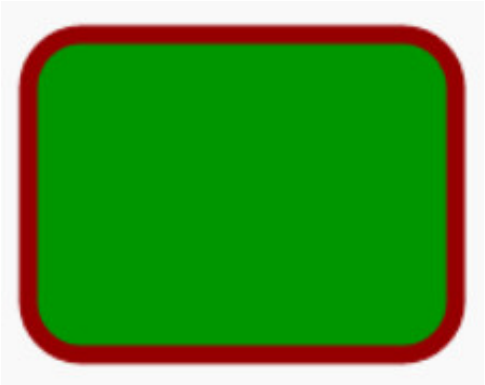
Try this interactive example: http://jsbin.com/bocomu/1/edit

```
context.beginPath();
context.moveTo(0, 20);
context.arcTo(100, 100, 200, 20, 50);
context.lineWidth = 5;
context.strokeStyle = "#0000ff";
```

```
    context.stroke();
```

EXAMPLE 2: DRAW A ROUNDED RECTANGLE

Try this interactive example: http://jsbin.com/kuqalu/1/edit



Source code:

```
varroundedRect=function(ctx,x,y,width,height,radius,fill,stroke)
{
     ctx.beginPath();
   // draw top and top right corner
   ctx.moveTo(x+radius,y);
   ctx.arcTo(x+width,y,x+width,y+radius,radius);
   // draw right side and bottom right corner
   ctx.arcTo(x+width,y+height,x+width-
radius,y+height,radius);
   // draw bottom and bottom left corner
   ctx.arcTo(x,y+height,x,y+height-radius,radius);
   // draw left and top left corner
   ctx.arcTo(x,y,x+radius,y,radius);
   if(fill) {
      ctx.fill();
   }
   if(stroke){
      ctx.stroke();
   }
}
```

```
22.
    varcanvas = document.getElementById('myCanvas');
    var ctx    = canvas.getContext('2d');
    ctx.strokeStyle = 'rgb(150,0,0)';
    ctx.fillStyle   = 'rgb(0,150,0)';
    ctx.lineWidth   = 7;
    roundedRect(ctx, 15, 15, 160, 120, 20, true, true);
```

In this example, each call to `ctx.arcTo(...)` draws a side plus a corner. This makes us suspect that the `arcTo()` method has been designed primarily for drawing rounded rectangles...

EXAMPLE 3 COMPARISON BETWEEN LINETO AND ARCTO

This example at JS Bin is the same as the previous one, except that we added at the end of the roundedRect function the same lines of code that draw the rounded rectangle, but using lineTo instead of arcTo. Just take a look!

JS Bin example