Styling lines

Several context properties can be used to set the thickness of the shape outlines, the way line end caps are drawn, etc.

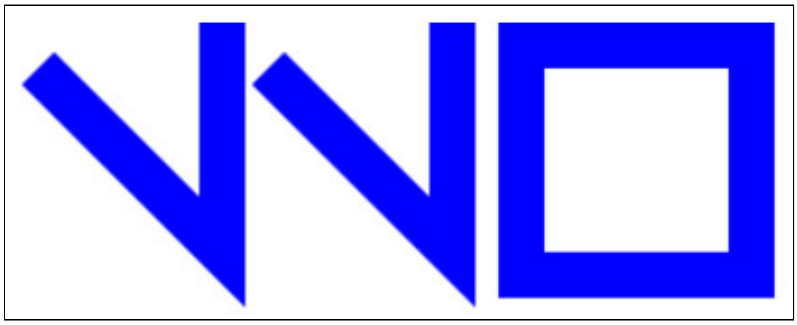
They apply to all shapes that are drawn in path mode (lines, curves, arcs) and some apply also to rectangles.

LINE STYLE: CHANGE THE LINE THICKNESS

We have seen this before. This is done by changing the value (in pixels) of the `lineWidth`property of the context:

```
ctx.lineWidth = 10; // set the thickness of every shape drawn in stroke/wireframe mode to 10 pixels
```

Here is a complete example where we draw with a lineWidth of 20 pixels. You can play with the complete interactive example here: http://jsbin.com/dacuco/2/edit



Source code:

```
<!DOCTYPE html>
<html>
 <head>
 <title>A simple example of lineWidth property use</title>
 </head>
 <body>
   <canvas id="myCanvas" width="500">
      Your browser does not support the canvas tag.
   </canvas>
```

```
<script>
 var canvas = document.getElementById('myCanvas');
 var ctx = canvas.getContext('2d');

 // first path
ctx.moveTo(20, 20);
ctx.lineTo(100, 100);
ctx.lineTo(100, 0);


 // second part of the path
ctx.moveTo(120, 20);
ctx.lineTo(200, 100);
ctx.lineTo(200, 0);

 // indicate stroke color + draw first part of the path
ctx.strokeStyle = "#0000FF";
 // Current line thickness is 20 pixels
ctx.lineWidth = 20;
ctx.stroke();       // draws the whole path at once

 // Draws a rectangle in immediate mode
ctx.strokeRect(230, 10, 100, 100);
</script>

</body>
</html>
```

LINE STYLE: CHANGING THE END CAPS FOR A LINE

The `lineCap` property of the context indicates the way line end caps are rendered. Possible values are `butt` (default), `round`, `square` (from top to bottom in the next illustration) . Note that a value of "round" or "square" makes the lines slightly longer than the default value "butt".

Try the next example interactively: http://jsbin.com/yaliya/2/edit

	ctx.lineCap="butt" (default value)
	ctx.lineCap="square"
	ctx.lineCap="round"

Source code:

```
<!DOCTYPE html>
<html>
 <head>
   <title>A simple example of lineCap property use</title>
 </head>
 <body>

 <canvas id="myCanvas" width="500">
    Your browser does not support the canvas tag.</canvas>
<script>
 var canvas = document.getElementById('myCanvas');
 var ctx = canvas.getContext('2d');
```

```
 // first path
ctx.moveTo(20, 20);
ctx.lineTo(100, 100);
ctx.lineTo(100, 30);

 // second part of the path
ctx.moveTo(120, 20);
ctx.lineTo(200, 100);
ctx.lineTo(200, 30);

 // indicate stroke color + draw first part of the path
ctx.strokeStyle = "#0000FF";
 // Current line thickness is 20 pixels
ctx.lineWidth = 20;

// Try different values : butt, square, round
ctx.lineCap = "round";

ctx.stroke();
 // Draws a rectangle
ctx.strokeRect(230, 10, 100, 100);
</script>

 </body>
</html>
```

Note that in this example, the rectangle is not affected. It has no line ends visible - all its sides meet. However, the next property we're going to look at will have an effect on rectangles!

LINE STYLE: SETTING THE TYPE OF CORNER WHEN TWO LINES MEET

The `lineJoin` property of the context indicates the way corners are rendered, when two lines meet. Possible values are `miter` (the default) for creating sharp corners, `round`, or`bevel` for "cut corners".

Try the next example interactively: http://jsbin.com/dozida/2/edit

	ctx.lineJoin="miter" (default value)
	ctx.lineJoin="bevel"
	ctx.lineJoin="round"

Source code:

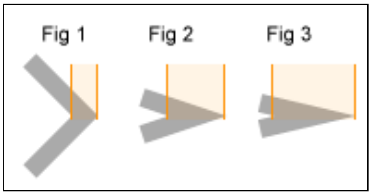
```
<!DOCTYPE html>
<html>
 <head>
 <title>A simple example of lineJoin property use</title>
 </head>
 <body>

 <canvas id="myCanvas" width="500">Your browser does not support the canvas tag.</canvas>

 <script>
 var canvas = document.getElementById('myCanvas');
 var ctx = canvas.getContext('2d');
// first path
```

```
    ctx.moveTo(20, 20);
    ctx.lineTo(100, 100);
    ctx.lineTo(100, 30);

    // second part of the path
    ctx.moveTo(120, 20);
    ctx.lineTo(200, 100);
    ctx.lineTo(200, 30);

    // indicate stroke color + draw first part of the path
    ctx.strokeStyle = "#0000FF";
    // Current line thickness is 20 pixels
    ctx.lineWidth = 20;
    // Try different values : miter(default), bevel, round
    ctx.lineJoin = "round";
    ctx.stroke();
    // Draws a rectangle
    ctx.strokeRect(230, 10, 100, 100);
    </script>

    </body>
    </html>
```

LINE STYLE: SPECIFIC CASE OF LINEJOIN="MITER", THE MITERLIMIT PROPERTY, A WAY TO AVOID LOOOOOOONG CORNERS!

The `miterLimit` property value corresponds to the maximum miter length: the distance between the inner corner and the outer corner where two lines meet. When the angle of a corner between two lines gets smaller, the miter length grows and can become too long.

In order to avoid this situation, we can set the `miterLimit` property of the context to a threshold value. If the miter length exceeds the `miterLimit` value, then the corner will be rendered as if the `lineJoin` property had been set to "bevel" and the corner will be "cut".



You can try an interactive example here: http://jsbin.com/bokusa/3/edit

In the example, try different values for the `miterLimit` property. You'll see that the way the corners are rendered changes at values around 2 and 3.

Source code:

```
<!DOCTYPE html>
<html>
 <head>
 <title>A simple example of miterLimit property use</title>
 </head>
 <body>

 <canvas id="myCanvas" width="500">Your browser does not support the canvas tag.</canvas>

 <script>
 var canvas = document.getElementById('myCanvas');
 var ctx = canvas.getContext('2d');
// first path
ctx.moveTo(20, 20);
ctx.lineTo(100, 100);
ctx.lineTo(100, 30);

// second part of the path
ctx.moveTo(120, 20);
ctx.lineTo(200, 100);
ctx.lineTo(200, 30);

// indicate stroke color + draw first part of the path
ctx.strokeStyle = "#0000FF";
// Current line thickness is 20 pixels
ctx.lineWidth = 20;
// Try different values : miter(default), bevel, round
ctx.lineJoin = "miter";
// try to change this value, try 2, 3, 4, 5 et...
ctx.miterLimit = 1;

ctx.stroke();

// Draws a rectangle
ctx.strokeRect(230, 10, 100, 100);
 </script>
```

```
</body>
</html>
```

KNOWLEDGE CHECK 3.5.7

Which context property defines the shape of line extremities?

- lineCap
- lineJoin
- lineWidth