

10강. 캔버스 [1] : 캔버스 구성 및 기본 드로잉

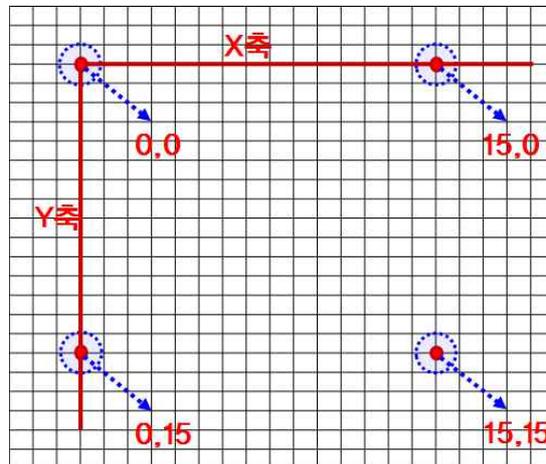
1. 기본 내용

1.1 캔버스

- canvas 요소는 웹 페이지에서 자바스크립트를 통해 즉시 그림을 그리는데 사용되며 단순한 그림 표현을 넘어 여러 효과와 함께 텍스트 및 애니메이션 표현이 가능하다.
- 그림을 그리기 위해서는 <canvas>를 사용해서 그림 영역을 지정하고, 자바스크립트를 사용해서 실제 그림을 그린다.

1.2 캔버스 좌표 시스템

- 캔버스의 좌표 시스템은 2D 컨텍스트로, 왼쪽 상단 모서리에 있는 평면 직교 표면을 (0,0)으로 시작해서 오른쪽으로 X 좌표 값이, 아래쪽으로 Y 좌표 값이 증가하는 구조이다.



1.3 그림 영역 지정

- canvas 요소에 가로 및 세로 크기를 지정하지 않으면, 브라우저는 자동으로 가로 300 픽셀, 세로 150픽셀 크기의 캔버스를 생성한다. width 속성과 height 속성을 이용하여 캔버스의 크기를 변경할 수 있다.

예제

```

<!DOCTYPE html> <html> <head> </head>
<body>
  <canvas id="canvas" width="700" height="400"
  style="border:solid 1px #ff0000">
    이 브라우저는canvas를 지원하지 않습니다.</canvas>
</body></html>

```



- 캔버스의 크기와 드로잉 표면의 크기
 - ▶ <canvas> 요소의 width/height 속성 사용 하여 변경하면 캔버스 크기를 드로잉 표면의 크기로 자동 변경한다.
 - ▶ CSS를 사용해서 캔버스 크기를 지정하는 경우는 드로잉 표면의 크기는 변경할 수 없기 때문에 캔버스와 드로잉 표면의 불일치로 인하여 예기치 않은 결과를 발생시킬 수 있음에 주의해야 한다.

<pre><style> canvas { width: 400px; height: 200px } </style></pre>	<pre><canvas id="myCanvas"> 이 브라우저는 캔버스를 지원하지 않습니다. </canvas></pre>
캔버스의 크기: 400×200	드로잉 표면의 크기: 300×150

- 브라우저 화면의 크기를 기준으로 캔버스 크기를 변경하는 경우는 윈도우 객체의 innerWidth 및 innerHeight 값을 이용한다.

```
var canvas = document.getElementById("myCanvas");
...
canvas.width = window.innerWidth;
canvas.height = window.innerHeight;
```

1.4 캔버스 컨텍스트

- 컨텍스트는 모든 그래픽 능력을 제공한다. 캔버스는 컨텍스트를 위한 컨테이너로서의 역할만 하고 실제로 캔버스에 그리는 등의 기능들은 모두 컨텍스트를 통하여 처리한다.

```
//캔버스 객체 생성
var 변수1 = document.getElementById('캔버스아이디');

//그리기 컨텍스트 생성
var 변수2 = 변수1.getContext('2d');

var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');

var 변수 = document.getElementById('캔버스아이디').getContext('2d');
var context = document.getElementById('myCanvas').getContext('2d');
```

1.5 캔버스 상태의 저장 및 복원

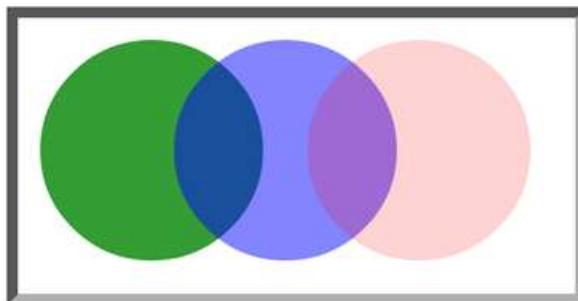
- 드로잉 상태를 저장하고 복원하기 위한 context.save() context.restore() 메소드를 제공한다.
- 드로잉 상태는 현재의 변환 행렬, 현재의 클리핑 영역으로 구성되어 있다.

예제

```

<!DOCTYPE html><html><head>
<script>
function save_restore() {
var canvas = document.getElementById("myCanvas");
var context = canvas.getContext("2d");
/ * 드로잉 상태를 3 개 저장 * /
    var colors = new Array ( "red", "blue", "green");
    var alphas = new Array (0.2, 0.5, 0.8);
    for (var i = 0; i < 3; i++) {
        context.fillStyle = colors[i];
        context.globalAlpha = alphas[i];
        context.save();
    }
/ * 저장된 드로잉 상태를 복원하여 원 3개를 그린다 * /
    for (var i = 0; i < 3; i++) {
        context.restore();
        context.beginPath();
        context.arc((i+1) * 120, 120, 100, 0, Math.PI * 2, false);
        context.fill();
    }
}
</script>
</head>
<body onload="save_restore();">
<canvas id="myCanvas" width="500" height="250" style="border: 10px inset #aaa">
캔버스에 사각형 그리기 연습
</canvas></body>
</html>

```



1.6 드로잉 작업을 위한 기본 형태

- 브라우저에 그림을 그리기 위해서는 캔버스 요소를 사용하여 그림을 그리고자 하는 영역을 정의하고 실제 그림을 그리는 것은 자바 스크립트를 사용하여 그린다.

- canvas요소에 폭과 높이를 지정하고, 자바스크립트에서 사용할 아이디 값을 지정한다.

```
<body onload="함수명();"
  <canvas id="myCanvas" width="300" height="200">
    이 브라우저는 HTML5의 canvas 요소를 지원하지 않습니다.
  </canvas>
</body>
</html>
```

- canvas 요소의 아이디를 문서의 getElementById('아이디') 메서드를 호출하여 canvas 객체를 생성하고 getContext('2d') 메서드를 호출하여 그리기 컨텍스트를 생성한다.

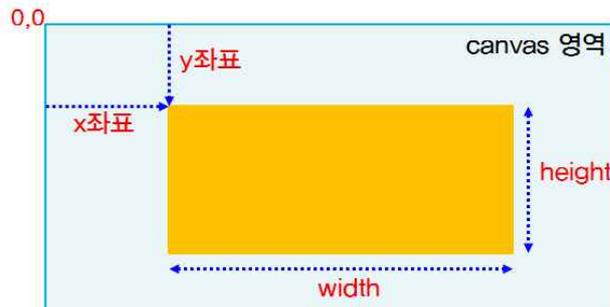
```
<!DOCTYPE html>
<html>
<head>
  <script>
    function 함수명() {
      var canvas = document.getElementById('myCanvas');
      var context = canvas.getContext('2d');
      ... 그림 작업 ...
    }
  </script>
</head>
```

예제	
<pre><!DOCTYPE html> <html> <head> <style> canvas { margin: 10px; padding 10px; border: 10px inset #aaa } </style> <script> function drawText () { var canvas = document.getElementById("myCanvas"); var context = canvas.getContext("2d"); context.font = "24pt 굴림체"; context.fillStyle = "maroon"; context.fillText("그림을 그리시다", canvas.width/2 - 130, canvas.height/2 + 15); } </script> </head> <body onload="drawText();" > <canvas id="myCanvas" width="300" height="200"> 이 브라우저는 canvas 요소를 지원하지 않습니다. </canvas> </body> </html></pre>	

2. 사각형 그리기

2.1 사각형 그리기

- 시작점(x , y)을 제공하고, 다른 2개는 사각형의 폭(W) 및 높이(H)를 제공한다.



2.2 사각형 그리기 관련 메서드

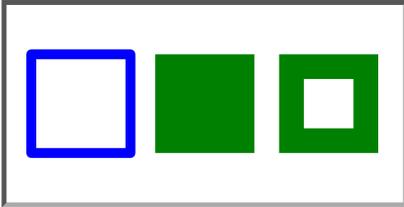
- context.fillRect (x, y, w, h) 메서드는 색이 채워진 사각형 영역을 그린다. 현재의 채우기 스타일 fillStyle 속성을 사용하여 지정한 사각형 영역의 캔버스에 채워진 사각형을 그린다.
- context.strokeRect (x, y, w, h) 메서드는 테두리만 있는 사각형 영역을 그린다.
- context.clearRect (x, y, w, h) 메서드는 지정한 사각형 영역을 지운다. 현재 클리핑 영역과 교차하는 지정한 사각형 영역에 있는 모든 픽셀을 투명한 검은색으로 채운다.

예제 사각형 그리기

```

<!DOCTYPE html><html><head>
  <script>
    function rect() {
      var canvas = document.getElementById("myCanvas");
      var context = canvas.getContext("2d");
      context.lineJoin = "round";
      context.lineWidth = 20;
      context.strokeStyle = "blue";
      context.fillStyle = "green";
      context.strokeRect(50, 100, 200, 200);
      context.fillRect(300, 100, 200, 200);
      context.fillRect(550, 100, 200, 200);
      context.clearRect(600, 150, 100, 100);
    }
  </script>
</head>
<body onload="rect();">
<canvas id="myCanvas" width="800" height="400" style="border: 10px inset #aaa">

```



```

캔버스에 사각형 그리기 연습
</canvas>
</body>
</html>

```

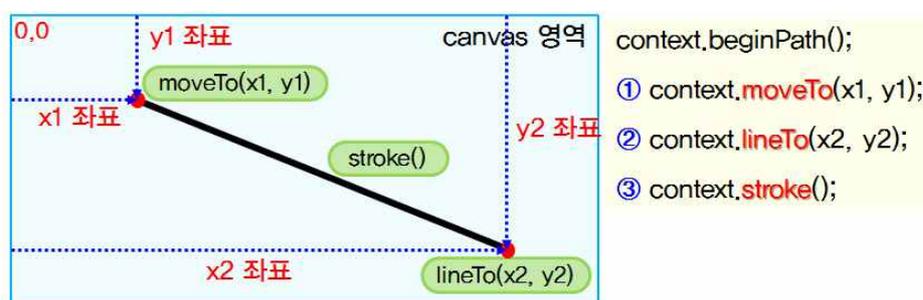
3. 선 그리기

3.1 패스와 서브패스

- 캔버스에서는 각 도형들을 이루는 선들의 집합을 패스(Path)라고 하고 각각의 선을 서브 패스라고 한다.
- 컨텍스트의 메서드를 이용하여 선이나 도형을 그리고자 할때는 beginPath() 함수를 호출하여 패스를 초기화 한다. 그리고 다양한 메서드를 사용하여 패스를 지정하고 선이나 도형을 그린다. 마지막으로 지정한 패스를 닫고 선이나 도형을 출력해야 한다.
- beginPath() 메서드는 현재 패스를 초기화한다. 이전까지 그렸던 패스를 모두 초기화하고 새로운 패스를 그린다는 의미이다.
- closePath() 메서드는 현재 패스를 닫는다. 패스 그리는 것을 종료한다는 의미이다.
- lineTo(x, y) 메서드는 직선을 연결한다. 이전 위치에서의 점과 현재 서브패스에 주어진 점을 추가하여 선을 그린다.
- moveTo(x, y) 메서드는 주어진 점으로 시작하는 새로운 서브패스를 만든다.
- stroke() 메서드는 현재 패스 또는 주어진 패스의 서브패스를 현재의 선 스타일로 그린다.
- fill() 메서드는 현재 패스 또는 주어진 패스의 서브패스를 현재의 채우기 스타일로 채운다.
- rect(x, y, w, h) 메서드는 사각형을 그린다.

3.2 선 그리기

- 시작점을 지정하기 위해서는 moveTo() 메서드를 사용한다. 그리고 다음 위치까지의 점을 지정하여 선을 그리기 위해서는 lineTo() 메서드를 사용한다. 그리고 stroke() 메서드를 사용해서 선을 출력한다.

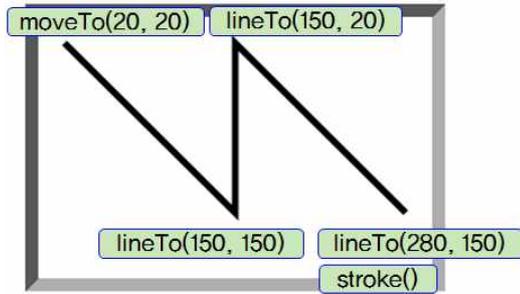


예제 선그리기

```

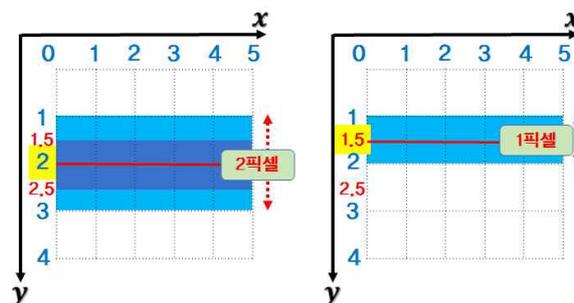
<!DOCTYPE html> <html> <head>
  <script>
    function Line() {
      var canvas = document.getElementById("myCanvas");
      var context = canvas.getContext("2d");
      context.beginPath();
      context.moveTo(20, 20);
      context.lineTo(150, 150);
      context.lineTo(150, 20);
      context.lineTo(280, 150);
      context.stroke();
    }
  </script>
</head>
<body onload="Line();">
<canvas id="myCanvas" width="800" height="400" style="border: 10px inset #aaa">
  캔버스에 사각형 그리기 연습</canvas>
</body></html>

```



3.3 선의 경계와 픽셀 경계

- 일반적으로 픽셀을 다루는 컴퓨터 그래픽 시스템에서는 픽셀의 위치를 정수 단위로 처리하기 때문에 픽셀의 경계 또한 정수 단위로 이루어진다. 그리고 캔버스 컨텍스트에서는 선을 그릴 때 두께를 지정할 수 있기 때문에 선의 중간에서 상하 / 좌우 0.5 픽셀만큼씩 그리려고 한다. 따라서 왼쪽 그림과 같이 1.5에서 상하 0.5 픽셀 만큼 확장하고, 2.5에서 상하 0.5픽셀 만큼 확장하기 때문에 결국에는 2픽셀 크기만큼 선이 그려진다. 그러나 오른쪽 그림에서는 상하 0.5 픽셀만큼씩 확장하면 정수 단위의 픽셀만큼씩 확장하면 정수단위의 픽셀 경계가 되기 때문에 1픽셀 만큼만 그려지는 것을 볼 수 있다. 정확한 픽셀 두께만큼의 선을 그리려면 픽셀 사이의 실수 값 위치를 지정해야 함을 잊지 말자.



3.4 다각형 그리기

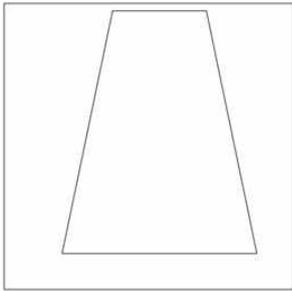
- 선 그리기 메서드에서 closePath() 메서드를 사용하면 다각형을 그릴 수 있다.
- closePath()메서드는 선이 마지막으로 종료된 지점과 최초의 지점을 자동으로 연결한다.

예제 선그리기

```

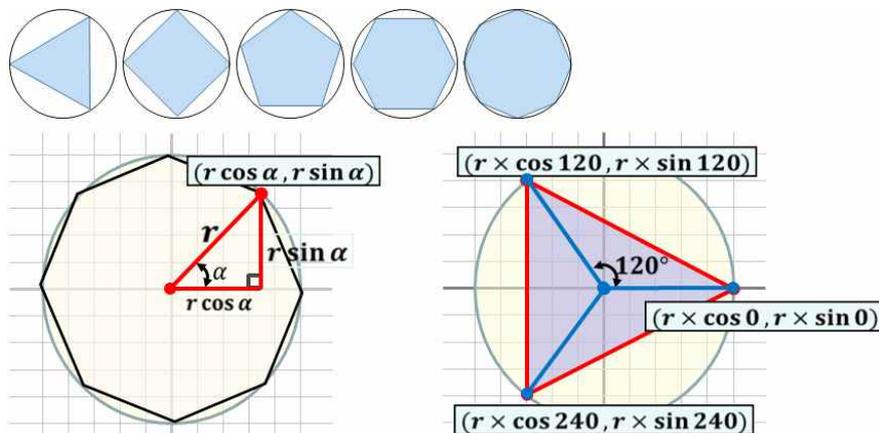
<!DOCTYPE html> <html> <head>
  <script>
function polygon() {
  var canvas = document.getElementById("myCanvas");
  var context = canvas.getContext("2d");
  context.beginPath();
  context.moveTo(150, 10);
  context.lineTo(80, 350);
  context.lineTo(350, 350);
  context.lineTo(280, 10);
  context.closePath();
  context.stroke();
}
  </script>
</head>
<body onload="polygon();" >
  <canvas id="myCanvas" width="600" height="400" style="border: 10px inset #aaa">
  캔버스에 그리기 연습
  </canvas>
</body>
</html>

```



3.4.1 정다각형을 그리는 알고리즘

- 정다각형은 모든 각도와 모든 측면이 동일하다. 또한 모든 꼭지점은 원 안에 위치한다는 것을 확인 할 수 있다.



- 입력: 좌표(x, y), 반지름(R), 면의 수(N), 시작각도(startAngle)

- 원의 반지름(R)과 면의 수(N)를 계산한다. (3각형은 3면, 4각형은 4면, 5각형은 5면)

```
if ( N < 3) return;
```

- 원의 중심으로부터 정다각형의 각 측면에 의한 각도($360/N$) 계산한다.

```
var degree = (Math.PI*2)/N;
context.save();
context.translate(x,y);
context.rotate(startAngle);
```

- 첫 번째 꼭지점의 위치를 (R, 0)으로 지정

```
context.moveTo(R, 0);
```

- 면의 수(N)만큼 루프를 통해서 꼭지점이 위치하는 각도 계산한다. 각 꼭지점들끼리의 드로잉을 수행한다.

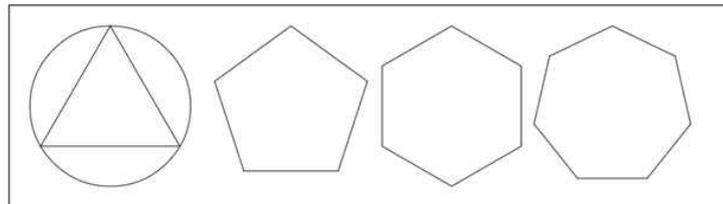
```
for (var i=0; i<N; i++) context.lineTo(R*Math.cos(degree*i),
    R*Math.sin(degree*i));
context.closePath();
context.restore();
```

```
polygon (context, 125, 125, 100, 3, -Math.PI/2);
```

```
polygon (context, 350, 125, 100, 5, -Math.PI/2);
```

```
polygon (context, 550, 125, 100, 6, -Math.PI/2);
```

```
polygon (context, 750, 125, 100, 7, -Math.PI/2);
```



3.5 점선그리기

- `context.setLineDash(segments)`는 현재 점선의 패턴을 설정한다. `segments` 값은 점선의 패턴으로 선이 그려지는 부분과 그려지지 않는 부분이 반복되는 배열이 된다.
- `segment = context.getLineDash()`는 현재 점선의 패턴을 반환한다.
- `context.lineDashOffset [=value]` 점선 모양의 패턴과 동일한 단위에서의 위상 오프셋을 반환한다. 선의 모양의 점선 패턴으로 변경한다.
- `setLineDash()`에서 `segments` 인자의 사용 방법은 다음 그림과 같다.

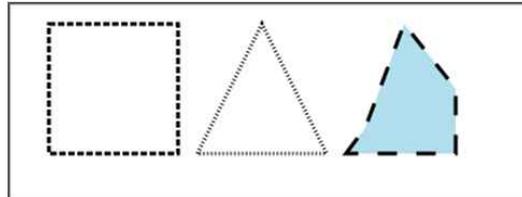
- `context.setLineDash([5])` 
 - 기본 설정. 선과 공백의 크기를 모두 5픽셀로 설정
- `context.setLineDash([1, 2])` 
 - 선의 크기는 1픽셀이고 공백의 크기는 2픽셀로 설정
- `context.setLineDash([2, 3])` 
 - 선의 크기는 2픽셀이고 공백의 크기는 3픽셀로 설정
- `context.setLineDash([5, 5, 2, 2])` 
 - 첫 번째 선의 크기는 5픽셀이고 공백의 크기는 5픽셀
 - 두 번째 선의 크기는 2픽셀이고 공백의 크기는 2픽셀

예제 점선그리기

```

<!DOCTYPE html><html><head>
<script>
  function LineDash() {
    var canvas = document.getElementById("myCanvas");
    var context = canvas.getContext("2d");
    // 사각형 그리기
    context.beginPath();
    context.setLineDash([5,2]);
    context.rect(30,15,100,100);
    context.stroke();
    // 삼각형 그리기
    context.beginPath();
    context.setLineDash([1,2]);
    context.moveTo(145,115);
    context.lineTo(195,15);
    context.lineTo(245,115);
    context.closePath();
    context.stroke();
    // 다각형 그리기
    context.beginPath();
    context.fillStyle = 'lightblue';
    context.setLineDash([15]);
    context.moveTo(260,115);
    context.lineTo(275,95);
    context.lineTo(305,15);
    context.lineTo(345,65);
    context.lineTo(345,115);
    context.closePath();
  }

```



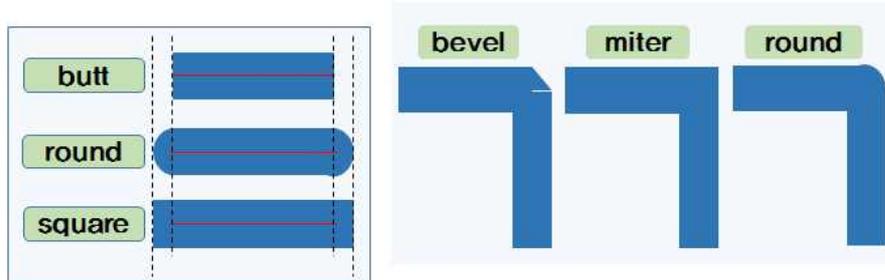
```

        context.fill();
        context.stroke();
    }
    </script>
</head>
<body onload="LineDash();">
<canvas id="myCanvas" width="400" height="150" style="border: 10px inset #aaa">
캔버스에 점선 그리기 연습
</canvas>
</body>
</html>

```

3.6 선 스타일 지정

- 선의 색상을 지정하고자 할 때는 strokeStyle 속성을 사용한다.
 - ▶ context.strokeStyle [=value]
- 선의 두께를 지정 할때는 lineWidth 속성을 사용한다.
 - ▶ context.lineWidth [=value]
- 선의 연결 부분의 스타일 지정할 때는 lineJoin 속성을 사용한다.
 - ▶ context.lineJoin [= bevel | round | miter(기본)]
- 선의 끝 부분의 스타일을 지정할 때는 lineCap 속성을 사용한다.
 - ▶ context.lineCap [= butt(기본) | round | square]



예제 선 스타일 지정

```

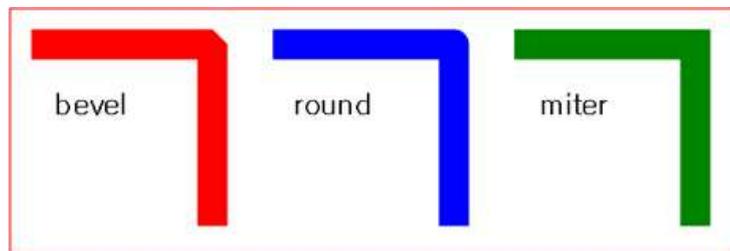
<!DOCTYPE html><html><head>
<script>
function LineJoin() {
    var canvas = document.getElementById("myCanvas");
    var context = canvas.getContext("2d");
    var lineStart = 50, lineEnd = 200, yStart = 50;
    context.lineWidth = "25";
    // bevel corner.
    context.beginPath();

```

```

context.strokeStyle = "Red";
context.lineJoin = "bevel"; //선의 연결 부분 스타일을 지정.
context.moveTo(lineStart, yStart);
context.lineTo(lineEnd, yStart);
context.lineTo(lineEnd, yStart + 150);
context.stroke();
// round corner.
context.beginPath();
context.strokeStyle = "blue";
context.lineJoin = "round"; //선의 연결 부분 스타일 지정
context.moveTo(lineStart + 200, yStart);
context.lineTo(lineEnd + 200, yStart);
context.lineTo(lineEnd + 200, yStart + 150);
context.stroke();
// miter.
context.beginPath();
context.strokeStyle = "green";
context.lineJoin = "miter"; //선의 연결 부분 스타일을 지정
context.moveTo(lineStart + 400, yStart);
context.lineTo(lineEnd + 400, yStart);
context.lineTo(lineEnd + 400, yStart + 150);
context.stroke();
}
</script>
</head>
<body onload="LineJoin();">
<canvas id="myCanvas" width="650" height="250" style="border: 10px inset #aaa">
캔버스 연습하기</canvas>
</body>
</html>

```



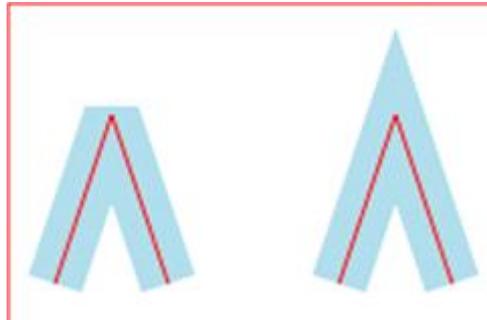
- 선의 연결 부분의 스타일이 miter인 경우 miter 길이의 비율을 지정하고자 할 때는 miterLimit 속성을 사용한다.
 - ▶ 선의 miter 연결 부분의 한계 비율 = (miter 길이) / (선 두께의 1/2 크기)
 - ▶ context.miterLimit [= value]

예제 **miterLimit** 속성

```

<!DOCTYPE html><html><head></head>
<body><canvas id="myCanvas" width="200" height="200" style="border: 10px inset
#aaa">캔버스 연습하기</canvas>
<script>
    var canvas = document.getElementById('myCanvas');
    var context = canvas.getContext('2d');
    // 첫 번째 연결선을 그린다.
    context.lineWidth = 20;
    context.miterLimit = 3.0;
    context.beginPath();
    context.strokeStyle = "lightblue";
    context.moveTo(30, 140);
    context.lineTo(50, 80);
    context.lineTo(70, 140);
    context.stroke();
    // 보조선을 선 중간에 그린다
    context.beginPath();
    context.lineWidth = 1;
    context.strokeStyle = "red";
    context.moveTo(30, 140);
    context.lineTo(50, 80);
    context.lineTo(70, 140);
    context.stroke();
    //두 번째 연결선을 그린다.
    context.lineWidth = 20;
    context.miterLimit = 10.0;
    context.beginPath();
    context.strokeStyle = "lightblue";
    context.moveTo(130, 140);
    context.lineTo(150, 80);
    context.lineTo(170, 140);
    context.stroke();
    // 보조선을 선 중간에 그린다
    context.beginPath();
    context.lineWidth = 1;
    context.strokeStyle = "red";
    context.moveTo(130, 140);
    context.lineTo(150, 80);
    context.lineTo(170, 140);

```



```
        context.stroke();
    </script>
</body>
</html>
```