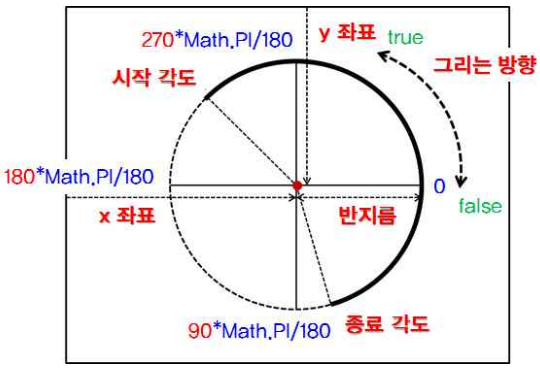
# 11강. 캔버스 [2] : 드로잉 확장

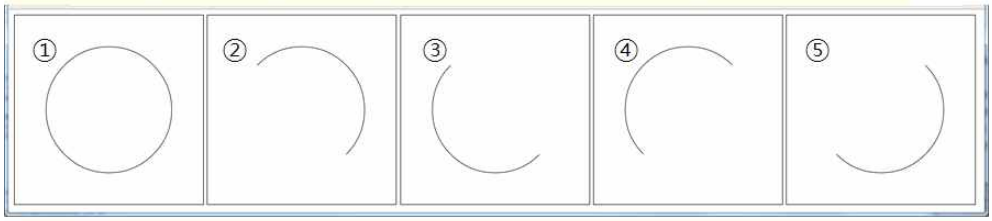
## 1. 원 그리기

### 1.1 원/원호 그리기 - arc() 메서드

- arc() 메서드에서는 시작 좌표(x, y), 반지름(radius), 시작각도(startAngle), 종료각도(endAngle), 그리는 방향(anticlockwise)을 지정해야 한다.
- 시작각도와 종료각도는 브라우저에서 원주를 따라 그려지는 호에 대한 각도로 라디안을 사용한다. 따라서 각도에 Math.PI/180을 곱해서 사용한다. 시계방향으로 그리기 위해서는 false값을 지정하고 시계 반대방향으로 그리기 위해서는 true값을 지정해야 한다.




```
context.beginPath();
context.arc(x, y, 반지름, 시작각도, 종료각도, 그리는 방향);
context.stroke();
```

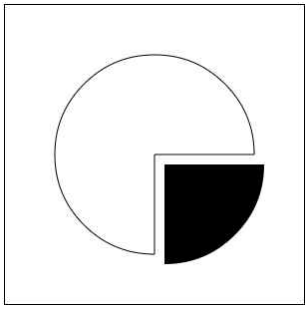



```
① context.arc(150, 150, 100, 0*Math.PI/180, 360*Math.PI/180, false);
② context.arc(150, 150, 100, 225*Math.PI/180, 45*Math.PI/180, false);
③ context.arc(150, 150, 100, 225*Math.PI/180, 45*Math.PI/180, true);
④ context.arc(150, 150, 100, 135*Math.PI/180, 315*Math.PI/180, false);
⑤ context.arc(150, 150, 100, 135*Math.PI/180, 315*Math.PI/180, true);
```

### 1.2 부채꼴 그리기

- arc() 메서드를 moveTo() 메서드와 함께 사용하면 부채꼴을 만들 수 있다. moveTo()메서드로 시작 지점을 지정하고, arc() 메서드로 호를 그린 후에 closePath() 메서드로 패스를 닫으면, 자동으로 시작지점과 호의 양끝점이 연결되어 부채꼴의 모양이 완성된다.

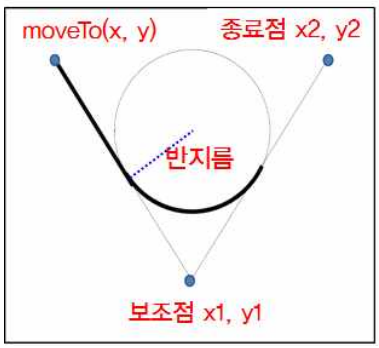
**예제**



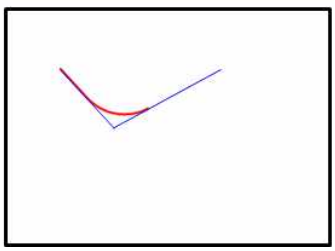
```
<!DOCTYPE html><html><head>
  <script type="text/javascript">
    function arcTo()
    {
      var canvas = document.getElementById('canvas');
      context = canvas.getContext('2d');
        context.beginPath();
        context.moveTo(150, 150);
        context.arc(150, 150, 100, 0, 90*Math.PI/180, true);
        context.closePath();
        context.stroke();
        context.beginPath();
        context.moveTo(160, 160);
        context.arc(160, 160, 100, 0, 90*Math.PI/180, false);
        context.closePath();
        context.fill();
    }
  </script>
</head>
<body onload="arcTo();">
  <canvas id="canvas" width="300" height="300" style="border:solid 1px #000000">
    canvas 사용하기</canvas>
</body>
</html>
```

## 1.3 직선과 접하는 원호 그리기 – arcTo() 메서드

- 직선과 접하는 원호를 그리기 위해서는 moveTo() 메서드와 arcTo( (x1, y1, x2, y2, 반지름)메서드를 사용한다.

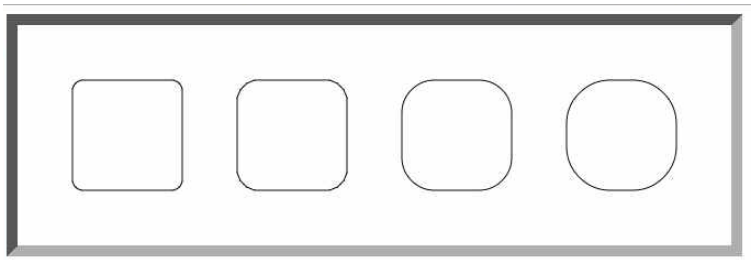



```
context.moveTo(50, 50);
context.arcTo(100, 100, 200, 50, 50);
context.stroke();
```



**예제 - 모서리가 둥근 사각형 그리기**

```
<!DOCTYPE html><html lang="Kor">
<head>
<title>캔버스 모서리가 둥근 사각형 연습</title>
<script>
function roundRect(context, x, y, width, height, radius) {
  if(width < 1) return;
  context.beginPath();
  context.moveTo(x + radius, y);
  context.arcTo((x+width), y, (x+width), (y+height), radius);
  context.arcTo((x+width), (y+height), x, (y+height), radius);
  context.arcTo(x, (y+height), x, y, radius);
  context.arcTo(x, y, (x+radius), y, radius);
  context.stroke();
}
function draw() {
  var canvas = document.getElementById("myCanvas");
  var context = canvas.getContext("2d");
  roundRect(context, 50, 50, 100, 100, 10);
  roundRect(context, 200, 50, 100, 100, 20);
  roundRect(context, 350, 50, 100, 100, 30);
  roundRect(context, 500, 50, 100, 100, 40);
}
</script>
</head>
<body onload="draw();">
<canvas id="myCanvas" width="650" height="200" style="border: 10px inset #aaa">
캔버스 연습</canvas>
</body>
</html>
```



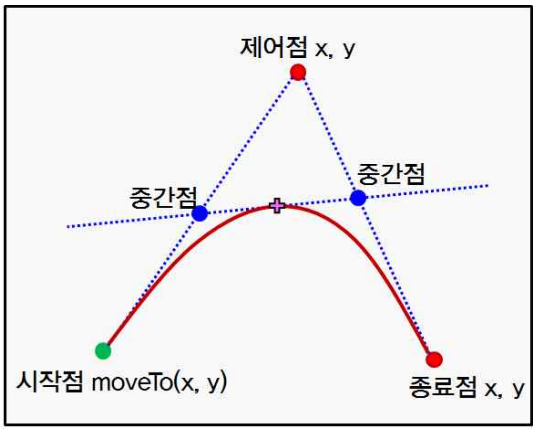
# 2 베지에 곡선

## 2.1 베지에 곡선

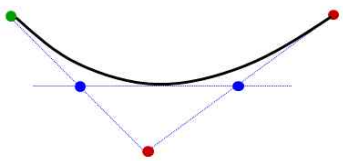
- 베지에 곡선(bezier curve)은 n개의 점으로부터 얻어지는 (n-1)차 곡선을 의미한다.

## 2.2 베지에 곡선

- 2차 곡선과 3차(다항) 곡선으로 이루어져 있다. 2차 베지에 곡선은 2개의 기준점(시작점과 종료점)과 한 개의 제어점을 필요로 한다. 시작점은 moveTo() 메서드를 사용하여 지정한다.
- quadraticCurveTo(제어점x, 제어점y, 종료점x, 종료점y)



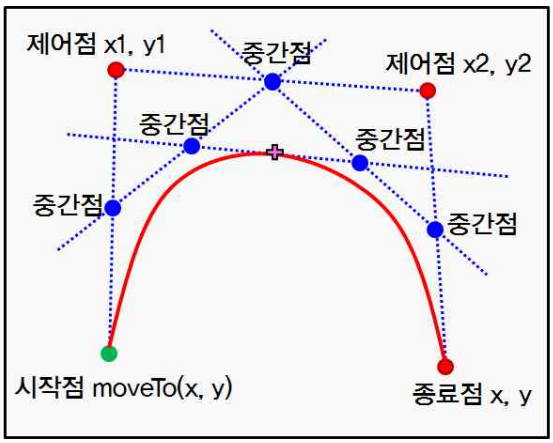

```
context.moveTo(50, 50);
context.quadraticCurveTo(200, 200, 400, 50);
context.stroke();
```



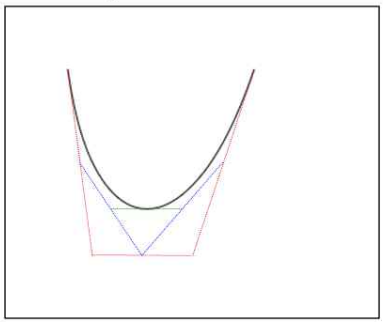
14

## 2.3 3차원 베지에 곡선

- 2개의 기준점(시작점, 종료점)과 2개의 제어점으로 정의된다. 4개의 점에서 중간 점 3개를 구하고, 중간 점 3개에서 중간 점을 2개를 다시 구한다. 마지막으로 중간점 2개에서 마지막 중간 점을 구하는 식이다.
- bezierCurveTo(제어점x1, 제어점y1, 제어점x2, 제어점y2, 종료점x, 종료점y)

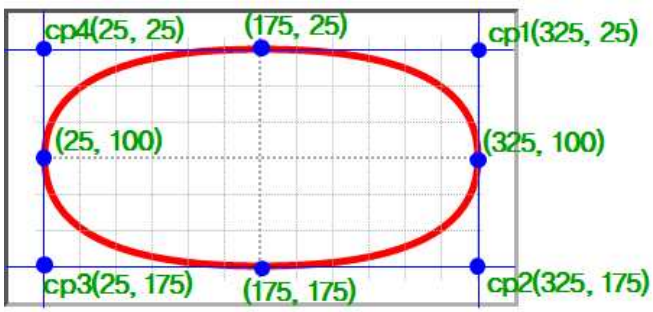



```
context.moveTo(50, 50);
context. bezierCurveTo(70, 200, 150, 200, 200, 50);
context.stroke();
```



**예제 – 2차 베지에 곡선**

```
<!DOCTYPE html><html lang="Kor"><head>
<title>캔버스 - 투명도 연습</title>
<script>
function draw() {
  var canvas = document.getElementById("myCanvas");
  var context = canvas.getContext("2d");
  context.lineWidth = 5.0;
  context.strokeStyle = "red"
  context.beginPath();
  context.moveTo(175, 25); //시작점을 중앙 상단으로 옮긴다
  context.quadraticCurveTo(325, 25, 325, 100);
  context.quadraticCurveTo(325, 175, 175, 175);
  context.quadraticCurveTo(25, 175, 25, 100);
  context.quadraticCurveTo(25, 25, 175, 25);
  context.stroke();
}
</script>
</head>
<body onload="draw();">
<canvas id="myCanvas" width="350" height="200" style="border: 10px inset #aaa">
캔버스 연습</canvas>
</body>
</html>
```




## 3. 스타일 지정

### 3.1 채우기 스타일 지정

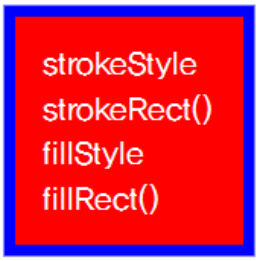
#### 3.1.1 fillStyle 속성

- 선을 그릴 때는 stroke() 메서드를 사용하고 색상을 지정할 때는 strokeStyle 속성을 사용한다. 색으로 채워져 있는 도형을 그릴 때는 fill() 메서드를 사용하고 채우기 색을 지

정할 때는 fillStyle 속성을 사용한다.

**예제**

```
<!DOCTYPE html><html>
<head>
  <script type="text/javascript">
    function arcTo()
    {
    context.strokeStyle = "blue";
    context.strokeRect(30, 30, 150, 150);
    context.fillStyle = "red";
    context.fillRect(30, 30, 150, 150);
    context.fillStyle = "red";
    context.fillRect(330, 30, 150, 150);
    context.strokeStyle = "blue";
    context.strokeRect(330,30, 150, 150);
    }
  </script>
</head>
<body onload="arcTo();">
  <canvas id="canvas" width="650" height="400" style="border:solid 1px #000000">
    canvas 사용하기  </canvas>
</body>
</html>
```





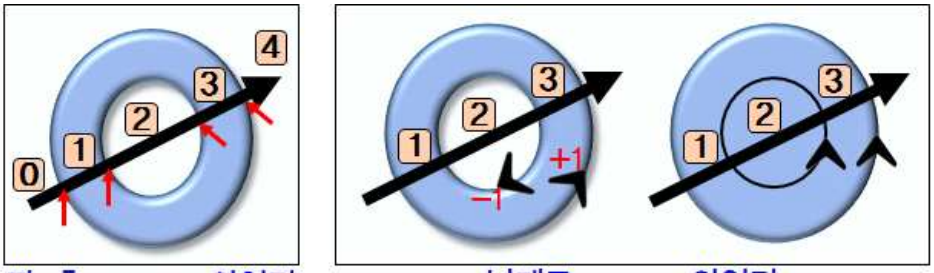
3.1.2 globalAlpha 속성

- 채우기 스타일을 이용할 때 도형을 채울 때 투명도를 지정할 수 있다. 투명도는 0~1.0 사이의 값을 지정한다.

## 3.2 와인딩 규칙

- 패스를 그릴 때마다 브라우저는 캔버스에 있는 지점이 곡선 내부에 있는지의 여부를 결정해야 한다. 패스가 교차하거나 중첩되어 있는 경우에는 명확하지 않기 때문이다. 캔버스에서는 현재 패스의 내부를 칠하는 방법에 따라서 다르게 동작한다.
- 패스 내부에 있는 점과 외부에 있는 점을 판별하는 방법으로 넌제로 와인딩 규칙과 짝-홀 와인딩 규칙이 있다.
- 짝-홀 와인딩(even-odd winding)은 선을 통과할 때마다 교차 횟수를 누적한다. 누적된 교차 횟수가 짝수라면 패스 외부라고 판단하고 채우지 않는다. 그러나 교차 횟수가 홀수이면 패스 내부라고 판단하고 지정한 스타일로 채우게 된다.
- 넌제로 와인딩(non-zero winding)은 가장 일반적으로 사용되는 방법으로 패스의 각 부분에 대한 드로잉을 하는 방향에 의존하는 방법이다. 어떤 한 지점이 곡선 내부에 있

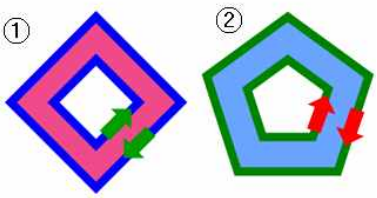
는지를 확인하려면, 해당 지점을 통해서 가상의 선을 그린다. 그런 다음 그 지점에 도달할 때까지 곡선을 교차하는 횟수를 계산한다. 모든 시계 방향 회전의 경우에는 1씩 감소시키고, 모든 시계 반대 방향 회전의 경우에는 1씩 증가시킨다. 마지막 계산된 카운터가 0이 아니면 해당 영역은 패스 안에 존재한다고 판단하여 해당 영역의 내부를 채우고, 마지막 카운터가 0이라면 해당 영역은 패스 밖에 존재한다고 판단한다.



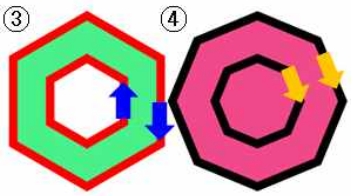

### 3.2.1 polygon() 알고리즘의 개선_와인딩 규칙 적용

```
function polygon(context, x, y, radius, sides, startAngle, anticlockwise) {
  if (sides < 3) return;
  var degree = (Math.PI*2)/sides;
  degree = anticlockwise ? -degree : degree; //각도의 방향을 반대로 계산
  context.save();
  context.translate(x,y);
  context.rotate(startAngle);
  context.moveTo(radius,0);
  for (var i = 1; i < sides; i++) {
    context.lineTo(radius*Math.cos(degree*i),radius*Math.sin(degree*i));
  }
  context.closePath();
  context.restore();
}
```

```
context.beginPath();
context.strokeStyle = "blue"
polygon(context,125, 125, 100, 4, -Math.PI/2, false);
polygon(context,125, 125, 50, 4, -Math.PI/2, true);
context.fillStyle="rgba(227,11,93,0.75)";
context.fill();
context.stroke();
```

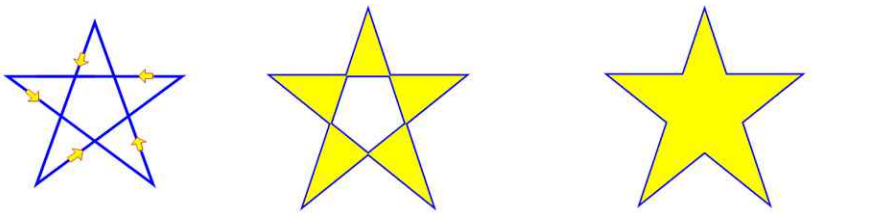



```
② polygon(context,350, 125, 100, 5, -Math.PI/2, false);
   polygon(context,350, 125, 50, 5, -Math.PI/2, true);

③ polygon(context,550, 125, 100, 6, -Math.PI/2, false);
   polygon(context,550, 125, 50, 6, -Math.PI/2, true);

④ polygon(context,750, 125, 100, 8, -Math.PI/2, false);
   polygon(context,750, 125, 50, 8, -Math.PI/2, false);
```



3.2.2 msFillRule 속성

- IE 11 이상에서 와인딩 방식 지정 가능하다.




## 3.3 그라데이션 스타일 지정

- 그라데이션 지정형식

```
context.beginPath();

var 변수 = context.createLinearGradient( x1, y1, x2, y2 );
                  createRadialGradient( x1,y1,r1,x2,y2,r2 )
변수.addColorStop(시작점_오프셋, 색상);
[ 변수.addColorStop(중간점_오프셋, 색상); ]*
변수.addColorStop(끝점_오프셋, 색상);

context.fillStyle = 변수;
```

0.0 … 0.x … 1.0
시작점 중간점 끝점

- createLinearGradient() 메서드 또는 createRadialGradient() 메서드를 사용하여 그라데이션을 지정을 위한 객체를 생성한 후에는 그라데이션에 따라 색상을 어떻게 배분하는지를 정의하는 addColorStop() 메서드를 사용하여 두 좌표간 또는 두 가상의 원 사이의 변환점 색상을 지정해야 한다.
  - addColorStop(오프셋, 색상)

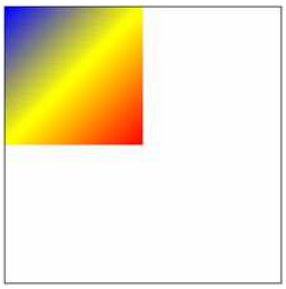
3.3.1 선형 그라데이션

- 선형 그라데이션을 지정하기 위해서는 createLinearGradient() 메서드에 시작좌표(x0, y0)와 종료 좌표(x1, y1)를 지정함으로써, 시작 좌표와 종료 좌표 간의 위치에 따라서 색상 변화가 있는 그라데이션 효과를 만들어 낼 수 있다.
  - createLinearGradient( x1, y1, x2, y2 )

**예제**

```
<!DOCTYPE html><html><head>
<script>
function draw() {
var canvas = document.getElementById("myCanvas");
var context = canvas.getContext("2d");
context.beginPath();
```

```
var gradient = context.createLinearGradient(0, 0, 150, 150);
gradient.addColorStop(0, 'blue');
gradient.addColorStop(0.5, 'yellow');
gradient.addColorStop(1, 'red');
context.fillStyle = gradient;
context.fillRect(0, 0, 150,150);
</script>
</head>
<body onload="draw();">
<canvas id="myCanvas" width="660" height="240" style="border: 10px inset #aaa">
캔버스 연습</canvas>
</body></html>
```

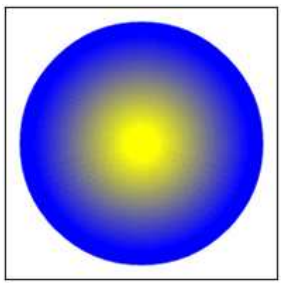


### 3.3.2 방사형 그라데이션

- 방사형 그라데이션을 지정하기 위해서는 createRadialGradient() 메서드에 원의 중심 좌표(x0, y0), 원의 반지름(r0), 또 다른 원의 중심 좌표(x1, y1), 또 다른 원의 반지름(r1)을 지정함으로써, 두 개의 가상 원이 생성되고 그 두 개의 가상의 원 사이의 위치에 따라서 색상 변화가 있는 그라데이션으로 효과를 만들어 낸다.
  - createRadialGradient( x1, y1, r1, x2, y2, r2 )

**예제**

```
<!DOCTYPE html><html><head>
    <script>
        function draw() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.beginPath();
        //방사형 그라데이션 객체를 만든다.
        var gradient = context.createRadialGradient( 100, 100, 10, 100, 100, 90 );
        gradient.addColorStop(0, "yellow");
        gradient.addColorStop(1, "blue");
        context.fillStyle = gradient;
        context.arc(100, 100, 90, 0, 360*Math.PI/180, true);
        context.fill();
    </script>
</head>
<body onload="draw();">
<canvas id="myCanvas" width="620" height="210" style="border: 10px inset #aaa">
캔버스 연습</canvas>
</body></html>
```

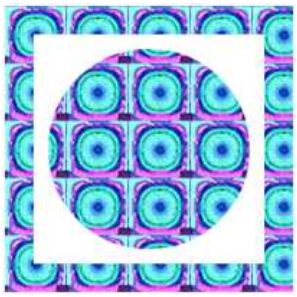


## 3.4 패턴 스타일 지정

- 패턴은 불투명한 CanvasPattern 인터페이스를 구현하는 객체로 표현되며 색상 및 그라데이션과 함께 패턴으로 도형과 텍스트를 그리고 내부를 채울수 있다. 패턴 객체는 createPattern() 메서드를 통해서 만들 수 있다.
  - context.createPattern( 이미지, 반복 형식 )

**예제**

```
<!DOCTYPE html><html><head>
  <script type="text/javascript">
    function arcTo()
    {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        var img = new Image();              //이미지 객체를 만든다.
        img.src = "pattern.png";  //이미지 객체에 이미지를 지정한다.
        img.onload = function () {
        context.beginPath ();
        var pattern = context. createPattern (img, ""); //이미지로 패턴 객체 생성.
        context.fillStyle = pattern;  //패턴 객체를 채우기 스타일로 지정
        context.arc(100, 100, 70, 0, 2 * Math.PI, false); //원을 그린다.
        context.fill (); //원 내부를 채운다.
        context.strokeStyle=pattern;
        context.lineWidth = 20;
        context.strokeRect( 10, 10, 180, 180);
        }
    }
  </script>
</head>
<body onload="arcTo();">
  <canvas id="myCanvas" width="200" height="200" >canvas 사용하기  </canvas>
</body>
</html>
```



## 3.5 그림자 스타일 지정

- 어떤 도형이나 텍스트, 또는 이미지 등에 입체감을 주기 위해서는 그림자를 지정해야 한다. 그림자를 지정하면, 도형이나 텍스트 등이 캔버스 위에 마치 떠 있는 느낌이 들도록 시각적 효과를 주기 때문에 이용되는 기능이다.

```
context.shadowOffsetX = 10;  // 그림자의 X좌표 지정
context.shadowOffsetY = 10;  // 그림자의 Y좌표 지정
context.shadowColor = 'green'; //그림자의 색상 지정
context.shadowBlur = 1; // 그림자의 흐림 정도 지정
context.fillStyle="rgba(220, 11, 93, 0.8)"
context.fillRect(120, 120, 100, 100);
```

**예제 – 그림자 스타일 지정**

```
<!DOCTYPE html><html><head>
  <script type="text/javascript">
    function shadow()
    {
      var canvas = document.getElementById('canvas');
      context = canvas.getContext('2d');

      context.fillStyle = 'rgba(10, 200, 100, 0.8)';
      context.beginPath();
      context.shadowOffsetX = -5;
      context.shadowOffsetY = -10;
      context.shadowColor = 'red';
      context.shadowBlur = 5;
      context.fillRect(50,50,100,250);

      context.beginPath();
      context.shadowOffsetX = 10;
      context.shadowOffsetY = 40;
      context.shadowColor = 'blue';
      context.shadowBlur = 10;
      context.fillRect(250,50,100,250);

      context.beginPath();
      context.shadowOffsetX = 40;
      context.shadowOffsetY = 10;
      context.shadowColor = 'green';
      context.shadowBlur = 30;
      context.fillRect(450,50,100,250);
    }
  </script>
</head>
<body onload="shadow();">
  <canvas id="canvas" width="700" height="400" style="border:solid 1px #000000">
```

```
    canvas 사용하기
  </canvas>
</body>
</html>
```