생능출판

HTML 표준화 문서를 기반으로 하는 지침서

속이 깊은 HTML5 & CSS3

김명진 지음

11강

캔버스 Part-2
- 드로잉 확장

학습 목표

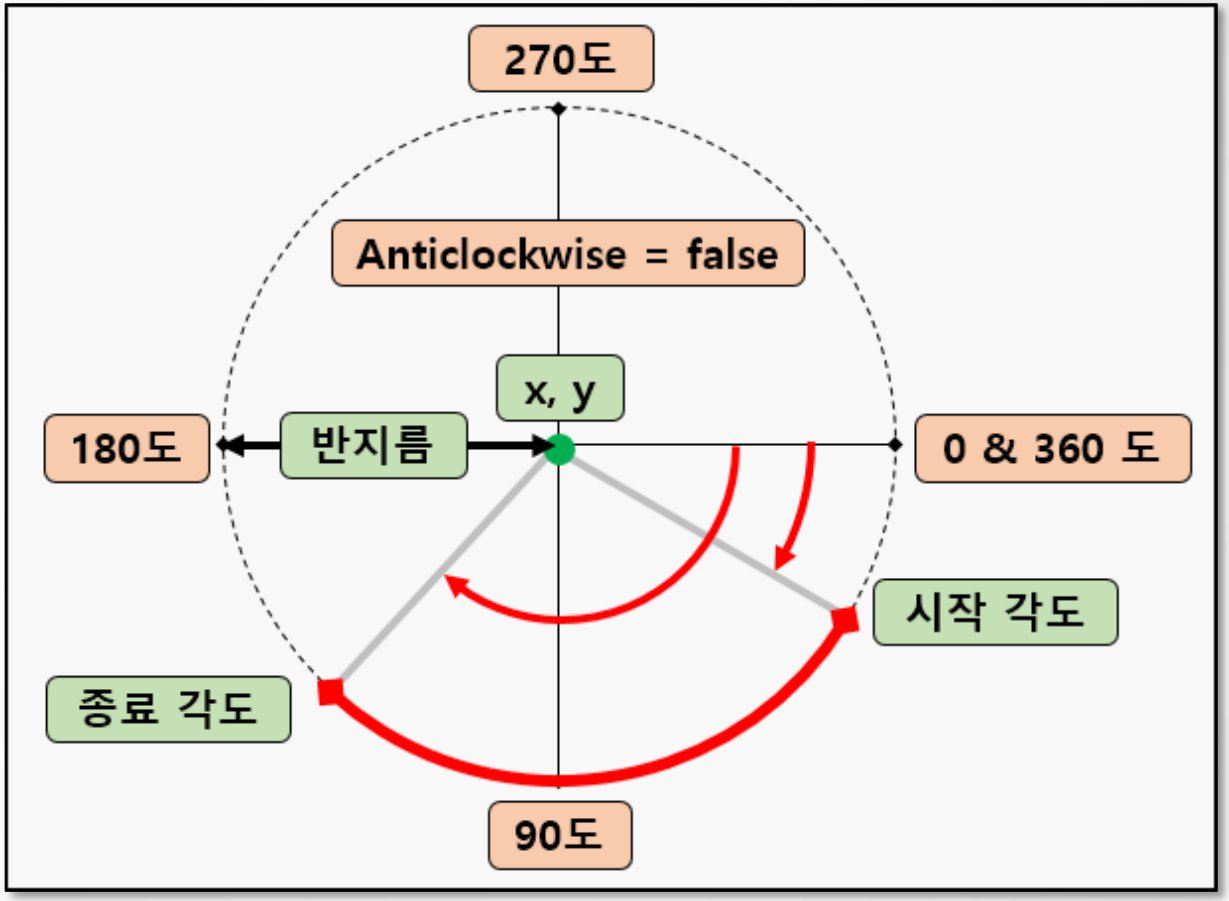
앞장에서 캔버스에서 드로잉 작업에 필요한 기본적인 내용들을 살펴보았다. 이번 장에서는 기본 드로잉 기능에 원 및 원호를 그리는 방법, 베지에 곡선을 그리는 방법을 학습한다. 그리고 다양한 색상으로 도형을 채울 수 있는 그라데이션 스타일, 와인딩에 따른 도형의 다양한 채우기 스타일, 패턴에 의한 스타일, 그리고 그림자 스타일에 대하여 학습하도록 한다.

Section

1. 원 그리기
2. 베지에 곡선
3. 스타일 지정하기

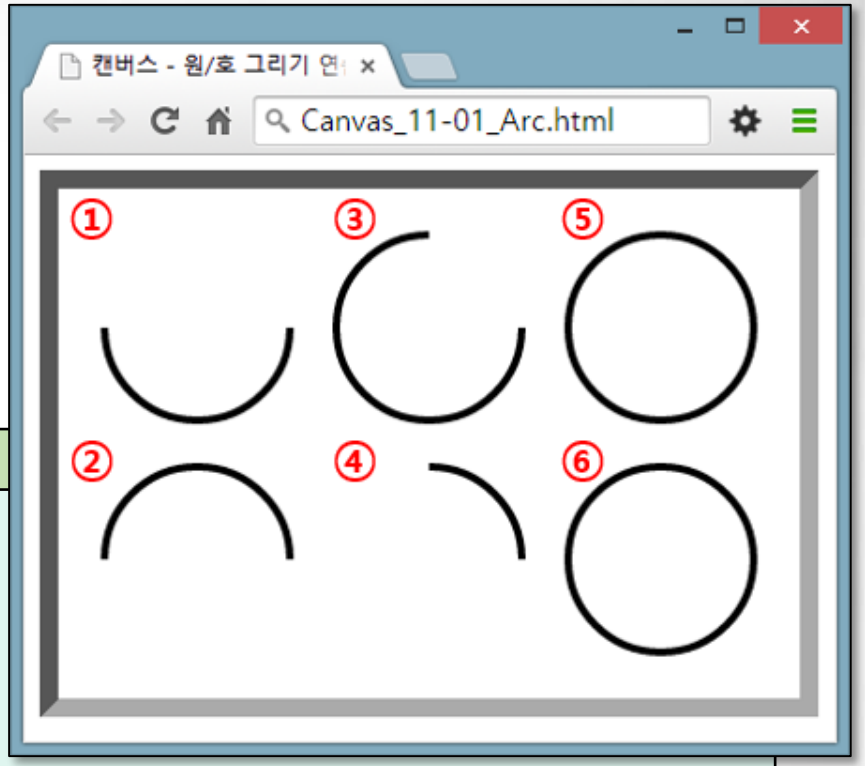
원/원호 그리기 – arc() 메서드

context.arc(x, y, radius, startAngle, endAngle [, anticlockwise])	호를 그린다.



원/원호 그리기 – arc() 메서드

context.arc(x, y, radius, startAngle, endAngle [, anticlockwise])	호를 그린다.



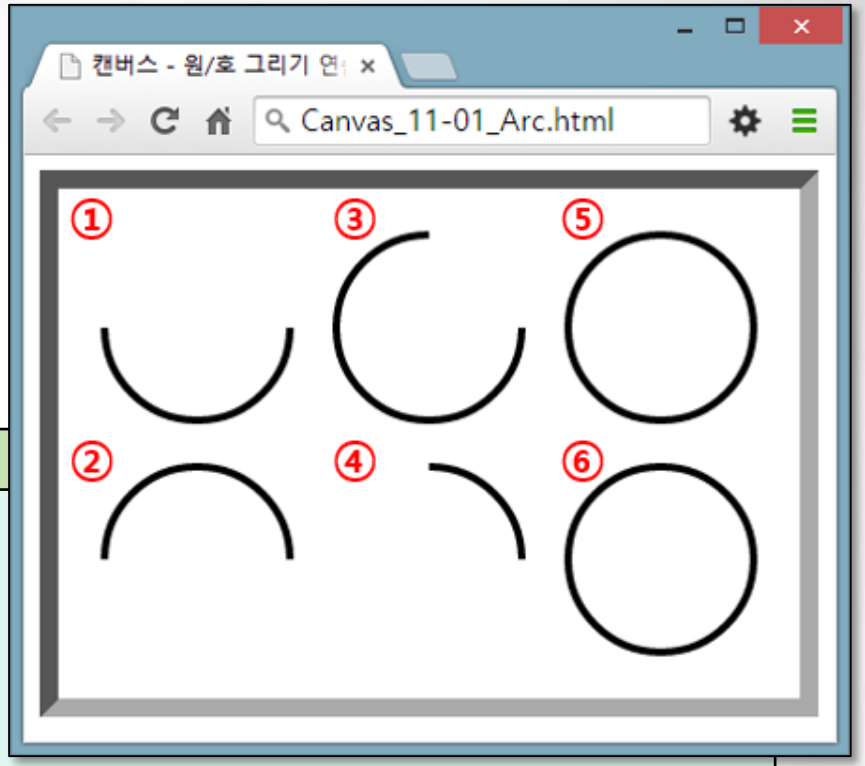
예제 Canvas_11-01_Arc.html

Script

```
for (var i = 0; i < 2; i++) {
   for (var j = 0; j < 3; j++) {
      context.beginPath();
      var x = 75 + j * 125; // 좌표 X
      var y = 75 + i * 125; // 좌표 Y
      var radius = 50;      // 반지름
      var startAngle = 0;   // 시작 각도
      var endAngle = Math.PI + (Math.PI * j) / 2; // 종료 각도
      var anticlockwise = i % 2 == 0 ? false : true; // 그리는 방향

      context.lineWidth = 4;
      context.arc(x, y, radius, startAngle, endAngle, anticlockwise); // 호를 그린다
      context.stroke();
   }
}
```

원/원호 그리기 – arc() 메서드

context.arc(x, y, radius, startAngle, endAngle [, anticlockwise])	호를 그린다.



예제 살펴보기

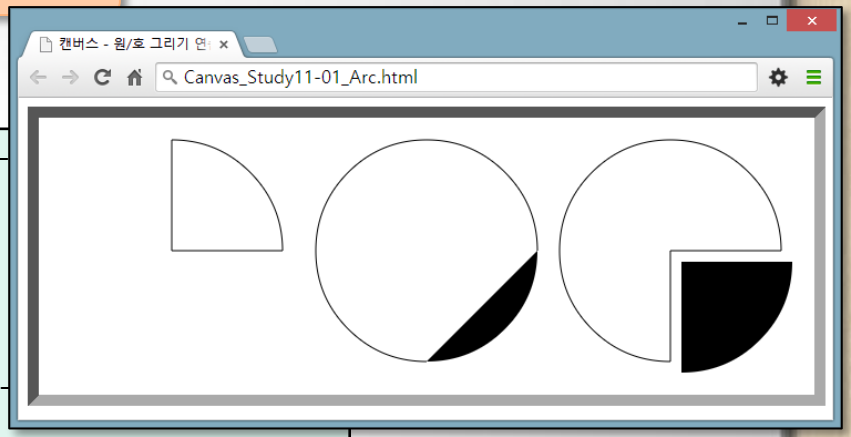
예제: Canvas_11-01_Arc.html

Script

```
for (var i = 0; i < 2; i++) {
   for (var j = 0; j < 3; j++) {
      context.beginPath();
      var x = 75 + j * 125; // 좌표 X
      var y = 75 + i * 125; // 좌표 Y
      var radius = 50;      // 반지름
      var startAngle = 0;   // 시작 각도
      var endAngle = Math.PI + (Math.PI * j) / 2; // 종료 각도
      var anticlockwise = i % 2 == 0 ? false : true; // 그리는 방향

      context.lineWidth = 4;
      context.arc(x, y, radius, startAngle, endAngle, anticlockwise); // 호를 그린다
      context.stroke();
   }
}
```

원/원호 그리기 – arc() 메서드

실습 – Canvas_Study11-01_Arc.html



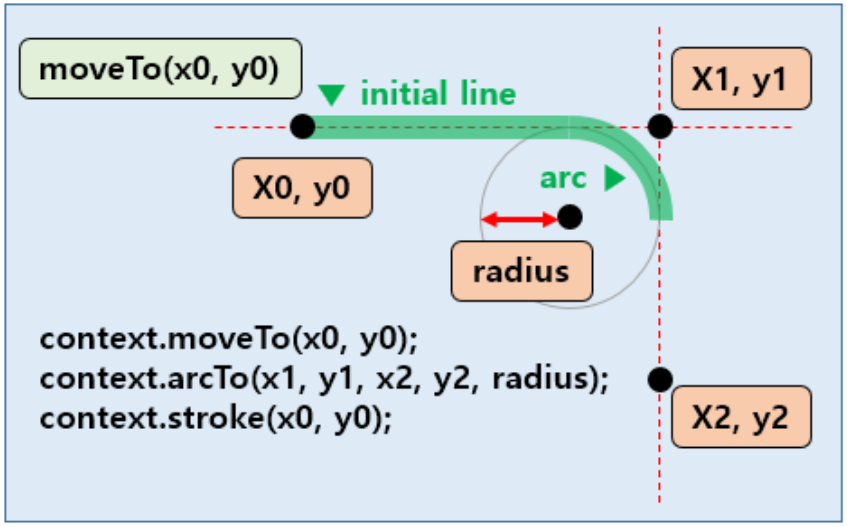
Script

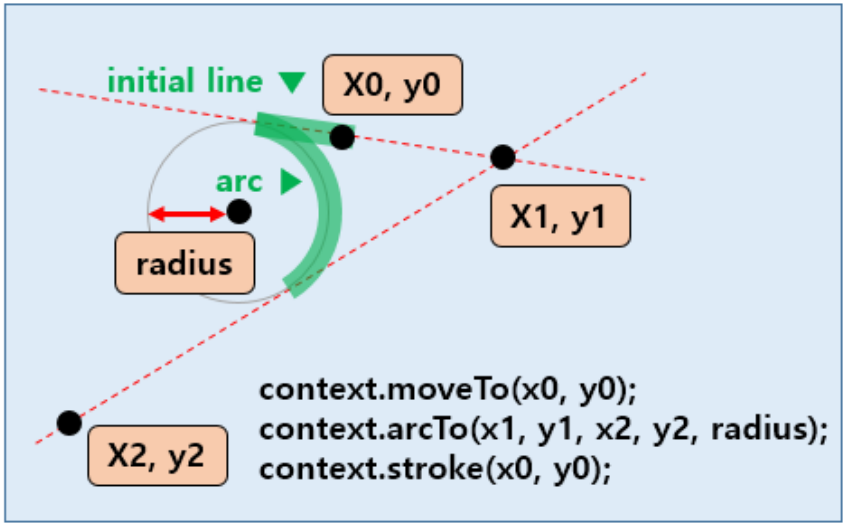
```
1  // 첫 번째 부채꼴을 그린다 ----------
2  context.beginPath();
3  context.moveTo(120,120);
4  context.arc(120,120,100,0,270*Math.PI/180, true);
5  context.closePath();
6  context.stroke();
7
8  // 두 번째 원의 첫 번째 호를 그린다 ----------
9  context.beginPath();
10 context.arc(350,120,100,90*Math.PI/180,360*Math.PI/180,false);
11 context.stroke();
12 // 두 번째 원의 두 번째 채워진 호를 그린다 ----------
13 context.beginPath();
14 context.arc(350,120,100,90*Math.PI/180,360*Math.PI/180,true);
15 context.fill();
16
17 // 세 번째 원의 첫 번째 부채꼴을 그린다 ----------
18 context.beginPath();
19 context.moveTo(570,120);
20 context.arc(570,120,100,90*Math.PI/180,360*Math.PI/180,false);
21 context.closePath();
22 context.stroke();
23
24 // 세 번째 원의 두 번째 채워진 부채꼴을 그린다 ----------
25 context.beginPath();
26 context.moveTo(580,130);
27 context.arc(580,130,100,90*Math.PI/180,360*Math.PI/180,true);
28 context.closePath();
29 context.fill();
```

직선과 연결된 원호 그리기 – arcTo() 메서드

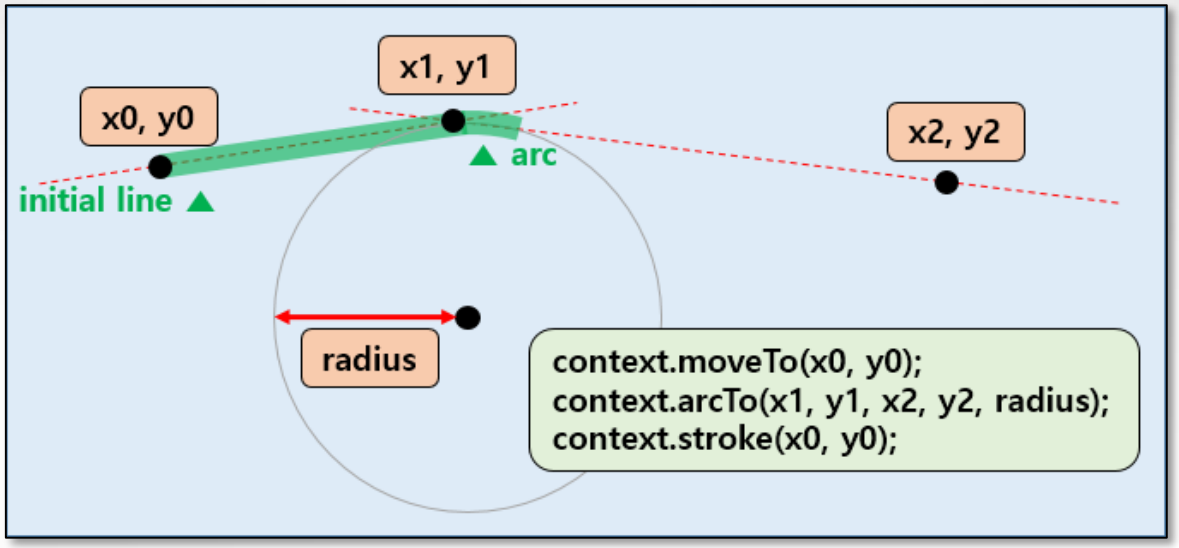
context.arcTo(x1, y1, x2, y2, radius)	직선과 연결된 호를 그린다.

- ✓ 현재 경로에 하나의 지점이 추가되고 그 지점은 직선에 의해 직전 지점에 연결된다. 그리고 현재 경로에 두 번째 지점이 추가되고 그 지점은 속성이 인수에 지정된 호에 의해 직전 지점에 연결된다.
- ✓ 주어진 반지름의 크기가 음수인 경우, indexSizeError 예외 오류가 발생한다.

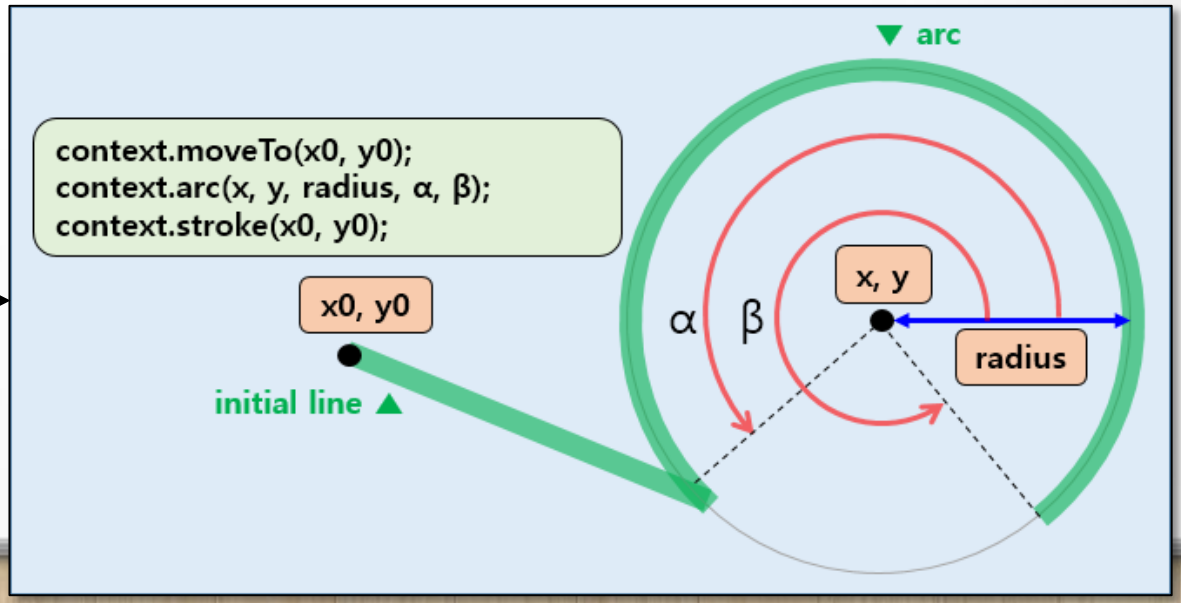




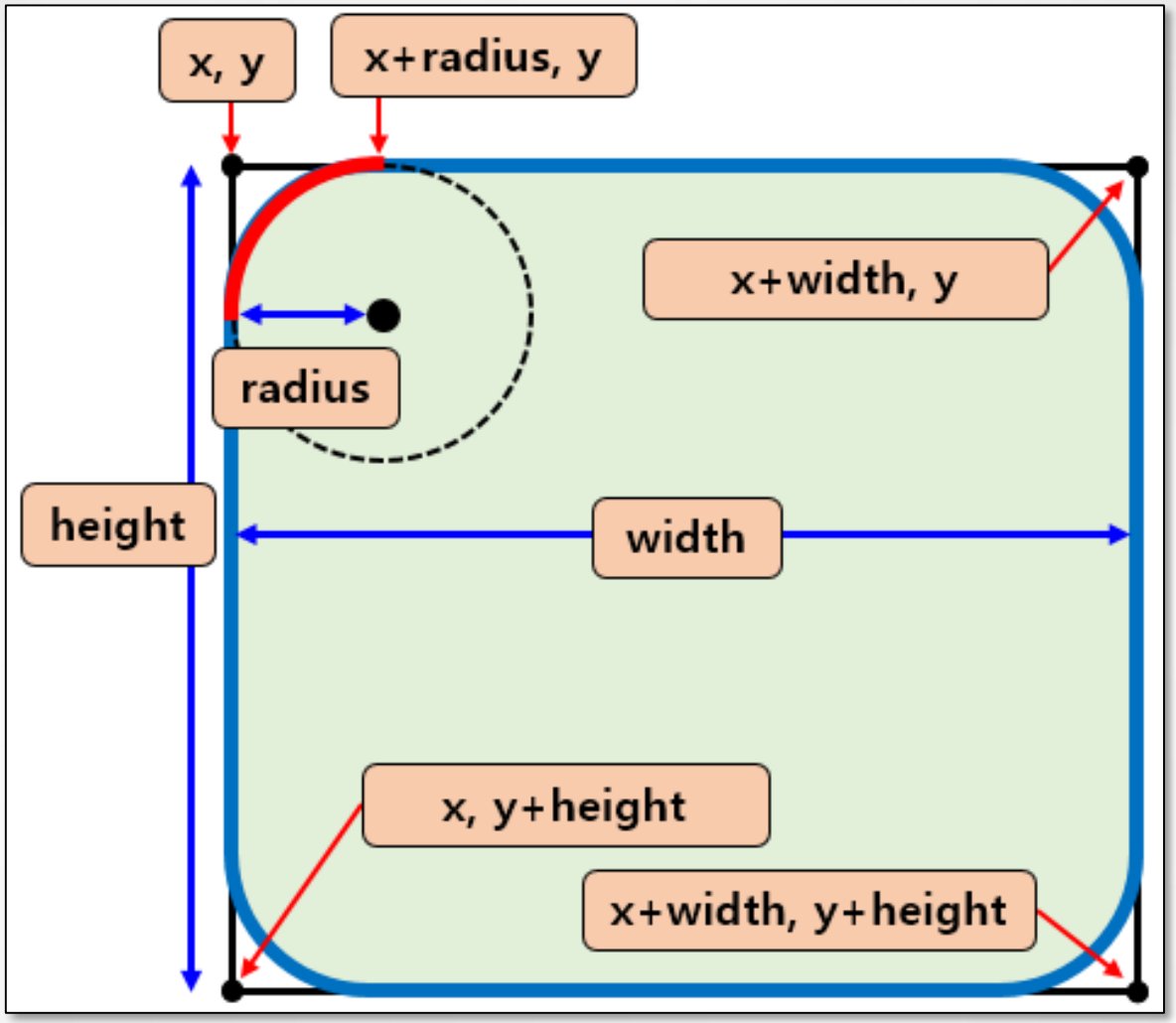
직선과 연결된 원호 그리기 – arcTo() 메서드



- arcTo() vs. arc()



직선과 연결된 원호 그리기 – arcTo() 메서드

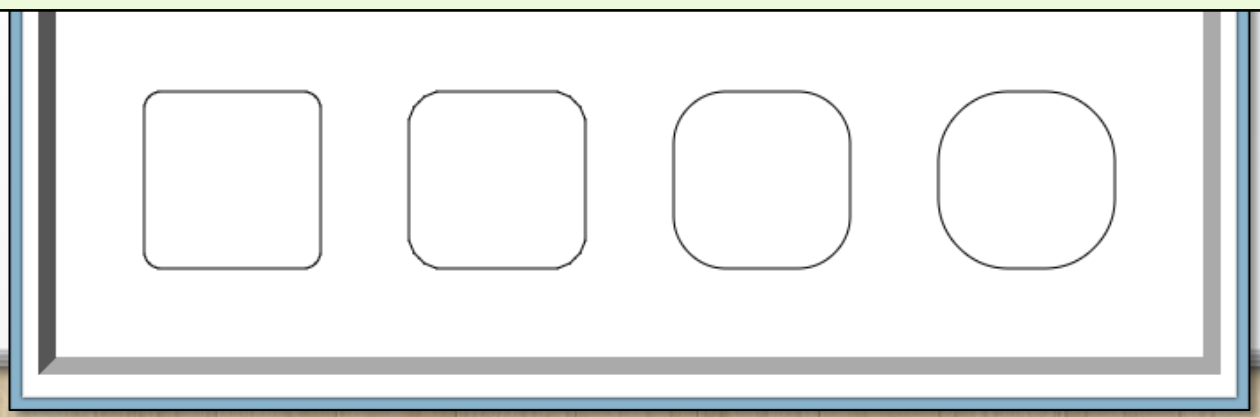


직선과 연결된 원호 그리기 – arcTo() 메서드

실습 – Canvas_Study11-02_RoundRect.html

Script

```
1  function roundRect(context, x, y, width, height, radius) {
2      if(width < 1) return;
3      context.beginPath();
4              context.moveTo(x + radius, y); //오른쪽 상단 모서리를 그리기 위한 시작점으로 이동
5              context.arcTo((x+width), y, (x+width), (y+height), radius);  //오른쪽 상단 모서리
6              context.arcTo((x+width), (y+height), x, (y+height), radius); //오른쪽 하단 모서리
7              context.arcTo(x, (y+height), x, y, radius); //왼쪽 하단 모서리
8              context.arcTo(x, y, (x+radius), y, radius); //왼쪽 상단 모서리
9      context.stroke();
10 }
11 function draw() {
12             var canvas = document.getElementById("myCanvas");
13             var context = canvas.getContext("2d");
14
15             roundRect(context, 50, 50, 100, 100, 10);
16             roundRect(context, 200, 50, 100, 100, 20);
17             roundRect(context, 350, 50, 100, 100, 30);
18             roundRect(context, 500, 50, 100, 100, 40);
19 }
```

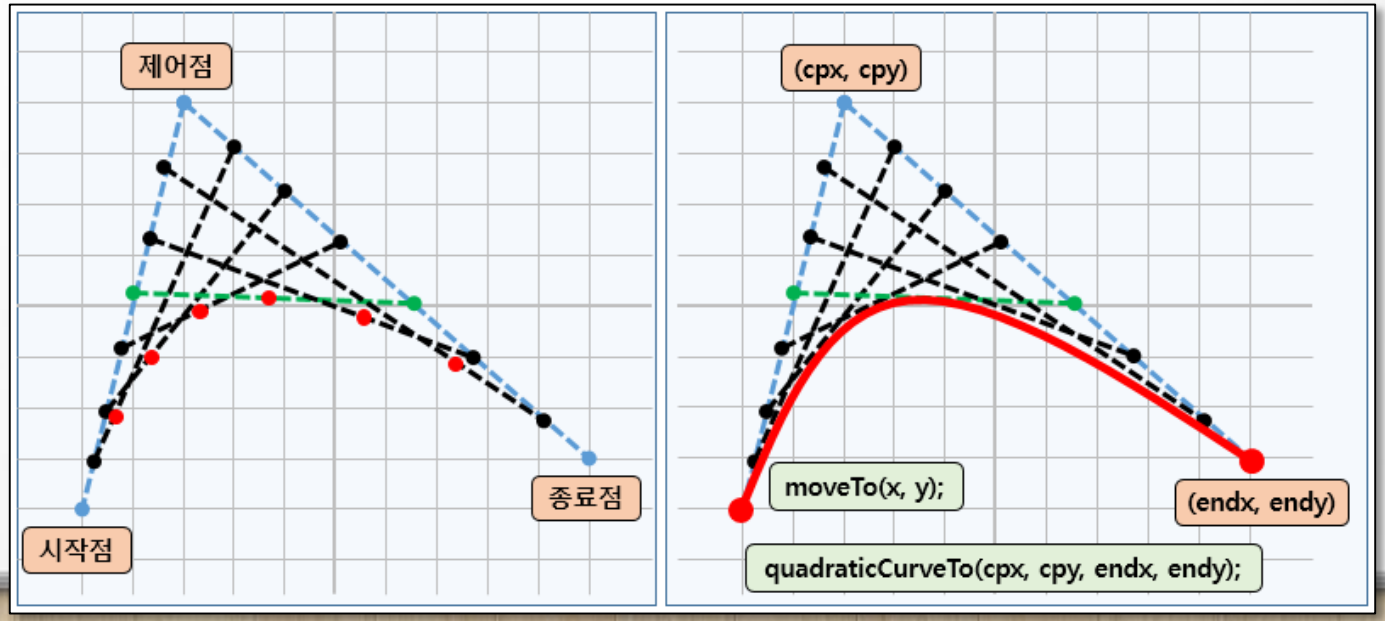


2차 베지에 곡선

- **베지에 곡선은 n 개의 점으로부터 얻어지는 (n-1)차 곡선을 의미**
  - 베지에 곡선은 2차 곡선과 3차(다항) 곡선으로 이루어져 있다.
  - 2차 베지에 곡선은 2개의 기준점(시작점과 종료점)과 한 개의 제어점을 필요로 한다.

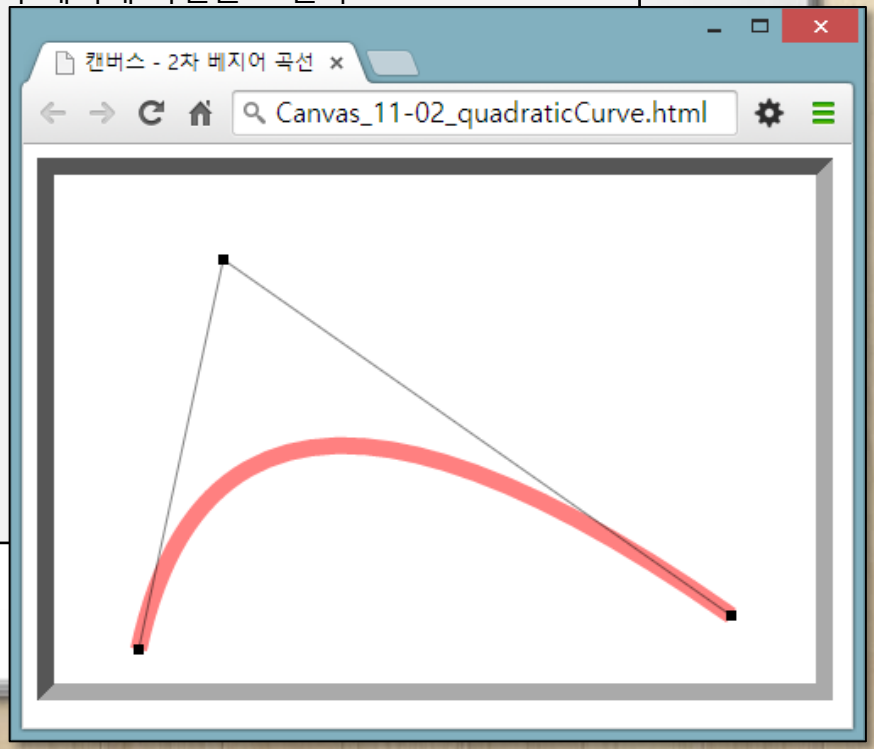
context.quadraticCurveTo(cpx, cpy, endx, endy)	2차 베지어 곡선을 그린다.

✓ 주어진 제어 포인트와 2차(quadratic) 베지어 곡선에 연결되는 현재의 서브 패스로 지정된 포인트를 추가한다.



2차 베지에 곡선

예제		Canvas_11-02_quadraticCurve.html
Script	1	var X = 50, Y = 280; //시작점의 위치 지정
	2	var cpX = 100, cpY = 50; //제어점의 위치 지정
	3	var endX = 400, endY = 260; //종료점의 위치 지정
	4	
	5	// 2차 베지어 곡선을 그린다.
	6	context.beginPath();
	7	context.lineWidth = 10; //곡선 두께 지정
	8	context.strokeStyle = "rgba(255, 0, 0, 0.5)"; //선 색 지정
	9	context.**moveTo**(X, Y); //시작점으로 이동
	10	context.**quadraticCurveTo(cpX, cpY, endX, endY);** //2차 베지에 곡선을 그린다
	11	context.stroke();
	12	
	13	// 가상의 보조선을 그린다.
	14	context.lineWidth = 1; //보조선 두께 지정
	15	context.strokeStyle = "rgba(0, 0, 0, 0.5)";
	16	context.beginPath();
	17	context.**moveTo**(X, Y); //시작점으로 이동
	18	context.**lineTo**(cpX, cpY); //제어점까지 연결선
	19	context.**lineTo**(endX, endY); //종료점까지 연결선
	20	context.stroke();
	21	
	22	//시작점, 보조점, 종료점을 표시한다
	23	context.**fillRect**(X-3, Y-3, 6, 6); //시작점 표시
	24	context.**fillRect**(cpX-3, cpY-3, 6, 6); //제어점 표시
	25	context.**fillRect**(endX-3, endY-3, 6, 6); //제어점 표시

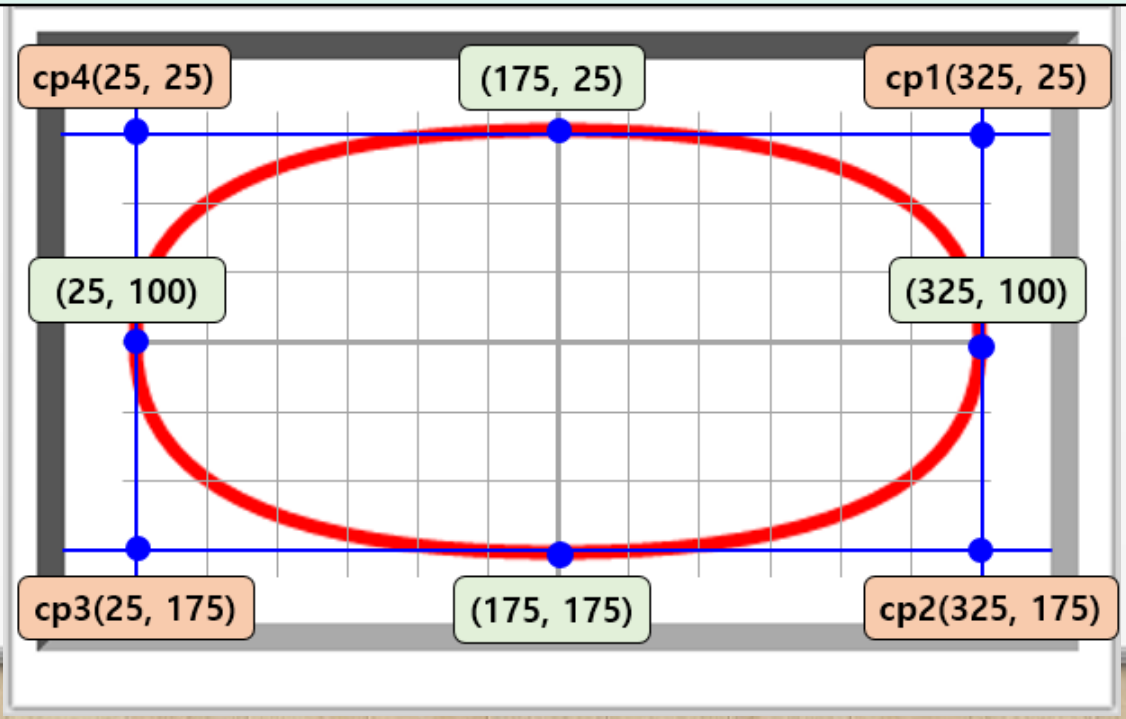


2차 베지에 곡선

실습 – Canvas_Study11-03_quadraticCurve.html

Script

```
1  context.lineWidth = 5.0;
2  context.strokeStyle = "red"
3  context.beginPath();
4                  context.moveTo(175, 25); //시작점을 중앙 상단으로 옮긴다
5                  context.quadraticCurveTo(325, 25, 325, 100);   //제어점 cp1과 종료점(325,100)
6                  context.quadraticCurveTo(325, 175, 175, 175);  //제어점 cp2와 종료점(175,175)
7                  context.quadraticCurveTo(25, 175, 25, 100);    //제어점 cp3과 종료점(25,100)
8                  context.quadraticCurveTo(25, 25, 175, 25);     //제어점 cp4과 종료점(25,25)
9  context.stroke();
```



베지에 곡선

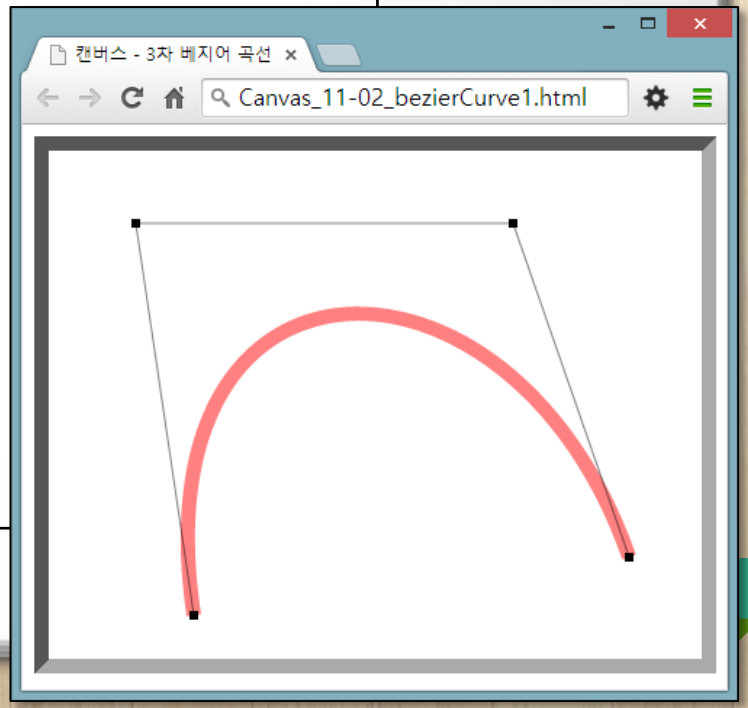
3차 베지에 곡선

- **베지에 곡선은 n 개의 점으로부터 얻어지는 (n-1)차 곡선을 의미**
  - 3차 베지에 곡선은 2개의 기준점(시작점, 종료점)과 2개의 제어점으로 정의된다.
  - 4개의 점에서 중간 점 3개를 구하고, 중간 점 3개에서의 중간 점을 2개를 다시 구한다.
  - 마지막으로 중간점 2개에서 마지막 중간 점을 구하는 식

context.bezierCurveTo(cp1x, cp1y, cp2x, cp2y, endx, endy)	3차 베지에 곡선을 그린다.
✓ 직선에 의한 이전 포인트와 연결된 현재의 서브 패스로 지정된 포인트를 추가한다.	

3차 베지에 곡선

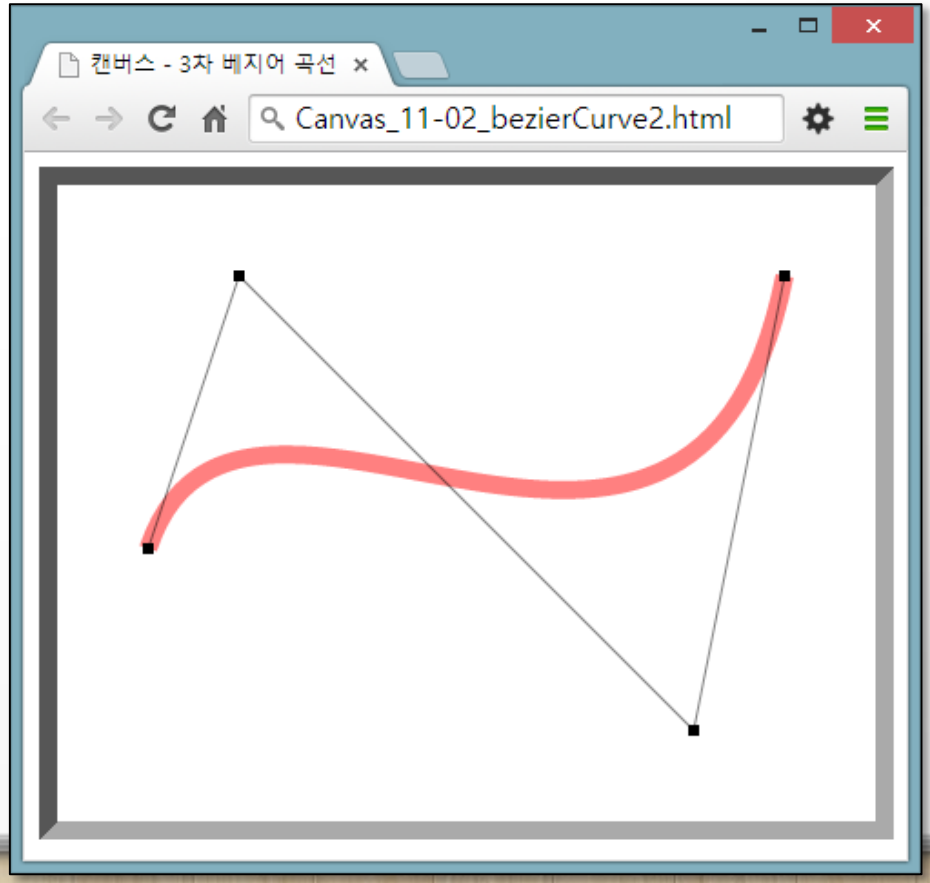
예제		Canvas_11-02_bezierCurve1.html
Script	1	var X = 100, Y = 320; //시작점의 위치 지정
	2	var cp1X = 60, cp1Y = 50; //제어점1의 위치 지정
	3	var cp2X = 320, cp2Y = 50; //제어점1의 위치 지정
	4	var endX = 400, endY = 280; //종료점의 위치 지정
	5	
	6	// 3차 베지어 곡선을 그린다.
	7	context.beginPath();
	8	context.lineWidth = 10; //곡선 두께 지정
	9	context.strokeStyle = "rgba(255, 0, 0, 0.5)"; //선 색 지정
	10	context.**moveTo**(X, Y); //시작점으로 이동
	11	context.**bezierCurveTo**(**cp1X, cp1Y, cp2X, cp2Y, endX, endY**);
	12	context.stroke();
	13	
	14	// 가상의 보조선과 각 지점을 그린다.
	15	context.lineWidth = 1; //보조선 두께 지정
	16	context.strokeStyle = "rgba(0, 0, 0, 0.5)";
	17	context.beginPath();
	18	context.moveTo(X, Y);
	19	context.lineTo(cp1X, cp1Y);
	20	context.lineTo(cp2X, cp2Y);
	21	context.lineTo(endX, endY);
	22	context.stroke();
	23	context.fillRect(X-3, Y-3, 6, 6); //시작점의 표시
	24	context.fillRect(cp1X-3, cp1Y-3, 6, 6); //제어점1의 표시
	25	context.fillRect(cp2X-3, cp2Y-3, 6, 6); //제어점2의 표시
	26	context.fillRect(endX-3, endY-3, 6, 6); //종료점의 표시



3차 베지에 곡선

예제		Canvas_11-02_bezierCurve2.html

Script	1
	2
	3
	4
		var X = 100, Y = 320; //시작점의 위치 지정
		var cp1X = 60, cp1Y = 50; //제어점1의 위치 지정
		var cp2X = 320, cp2Y = 50; //제어점2의 위치 지정
		var endX = 400, endY = 280; //종료점의 위치 지정



채우기 스타일(Fill Style) 지정

context.fillStyle [= value]	도형을 채우기 위해 사용되는 현재의 채우기 스타일을 반환한다.

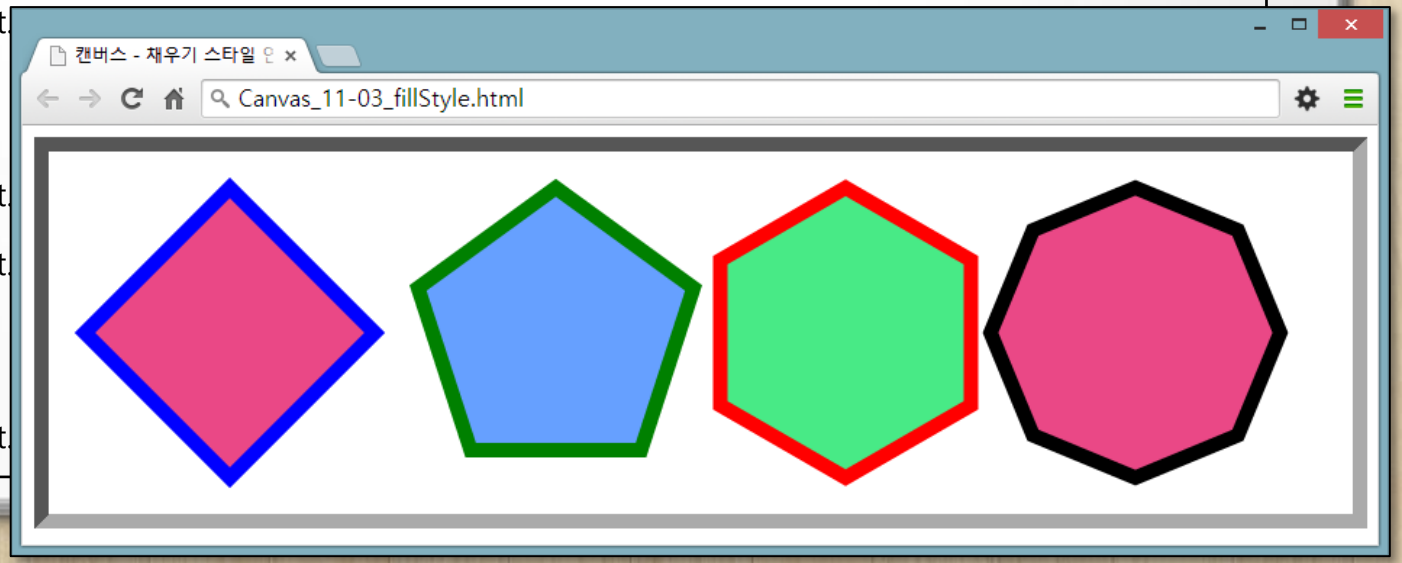
- ✓ fillStyle 속성은 도형 내부에 사용하는 색상이나 스타일을 나타내는 것으로 지정한 값(value)으로 변경할 수 있다.
- ✓ 스타일은 CSS 색상을 포함하는 문자열이나 CanvasGradient 또는 CanvasPattern 객체가 될 수 있으며, 잘못된 값은 무시된다.

context.globalAlpha [= value]	렌더링 처리에 적용하는 현재의 투명도 값을 반환한다.

- ✓ 전역(global) 알파(Alpha) 값을 설정하는 속성으로, 0(완전 투명) ~ 1.0(완전 불투명) 사이의 값을 지정할 수 있으며 기본 값은 1.0이다.
- ✓ rgba() 또는 hsla()를 이용하여 투명도를 지정할 수 있지만, 이 속성을 사용하면, 이후에 계속해서 그리게 되는 모든 도형과 이미지에 이 속성에 지정된 값이 적용된다.

채우기 스타일(Fill Style) 지정

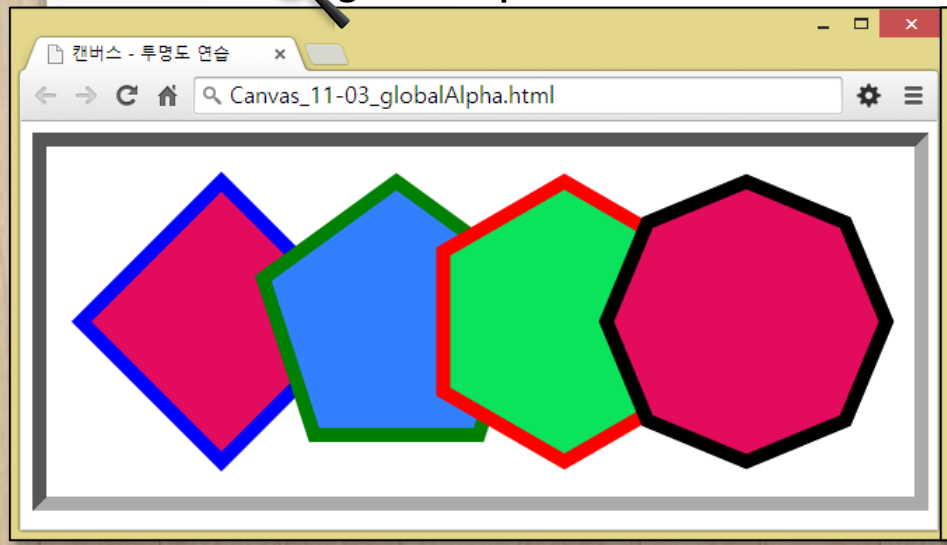
예제		Canvas_11-03_fillStyle.html
Script	1	context.lineWidth = 10;
	2	context.beginPath();
	3	context.**strokeStyle** = "blue"
	4	**polygon**(context,125, 125, 100, 4, -Math.PI/2); //사각형을 그린다
	5	context.**fillStyle="rgba(227,11,93,0.75)";**
	6	context.**fill**();
	7	context.stroke();
	8	
	9	context.beginPath();
	10	context.**strokeStyle** = "green"
	11	**polygon**(context,350, 125, 100, 5, -Math.PI/2); //오각형을 그린다
	12	context.**fillStyle="rgba(51,128,255,0.75)";**
	13	context.**fill**();
	14	context.stroke();
	15	
	16	context.
	17	
	18	
	19	
	20	
	21	context.
	22	
	23	context.
	24	
	25	
	26	
	27	
	28	context.



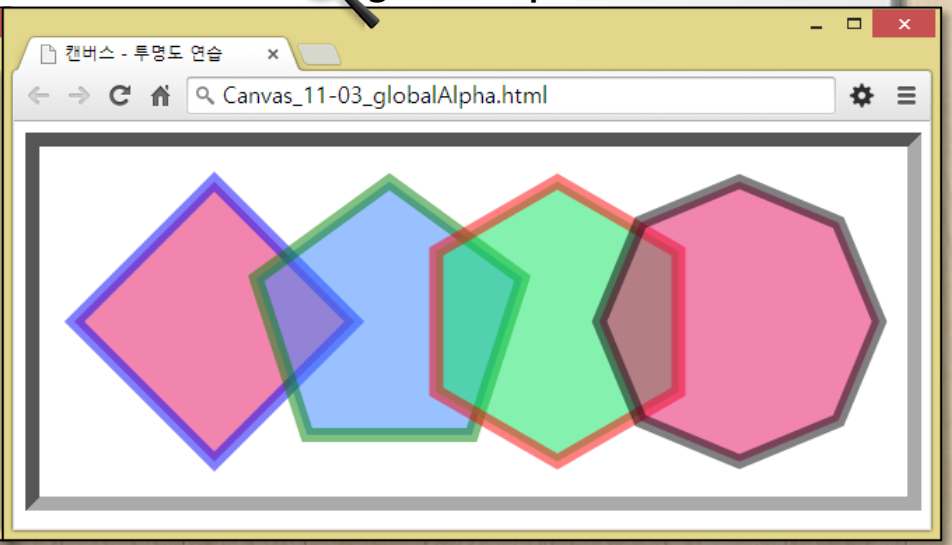
채우기 스타일(Fill Style) 지정

예제		Canvas_11-03_fillStyle.html
Script	1	context.lineWidth = 10;
	2	//context.**globalAlpha = 1;** //투명도를 완전 불투명하게 지정
	3	context.**globalAlpha = 0.5;** //투명도를 50% 투명하게 지정
	4	
	5	context.beginPath();
	6	context.**strokeStyle** = "blue"
	7	**polygon**(context,125, 125, 100, 4, -Math.PI/2); //사각형을 그린다
	8	context.**fillStyle="rgb(227,11,93)";**
	9	context.**fill**();
	10	context.stroke();
	11	
	12	... 이하 생략(앞의 예제 참고)

globalAlpha = 1.0

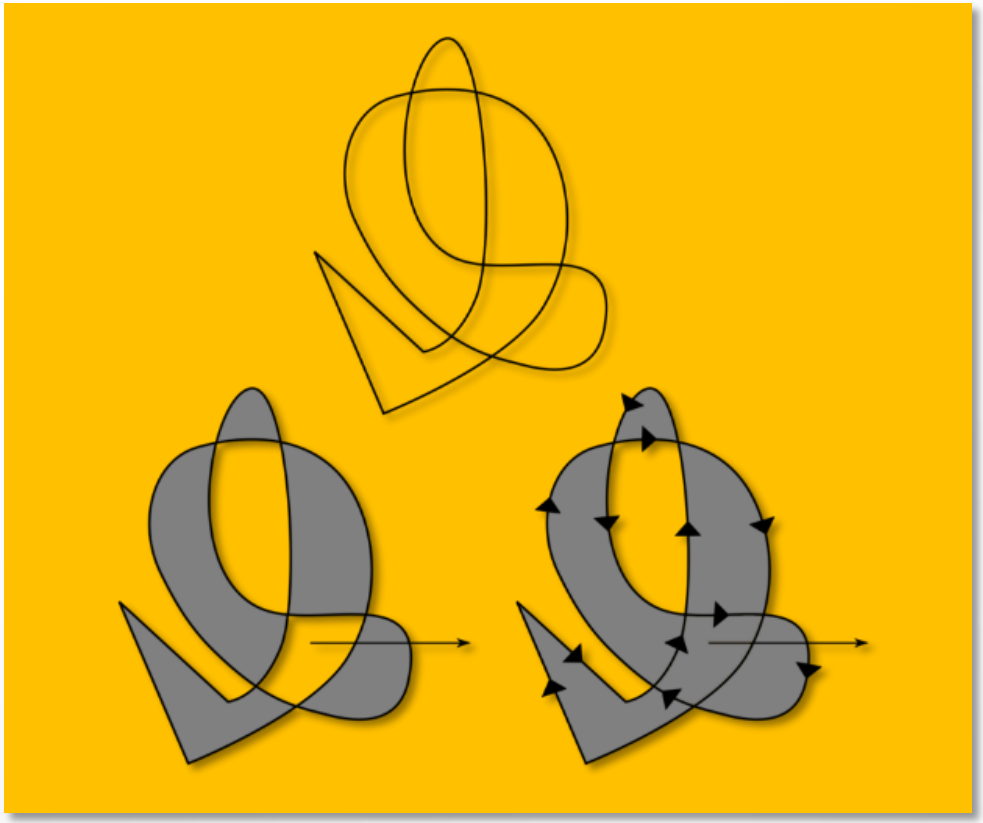


globalAlpha = 0.5



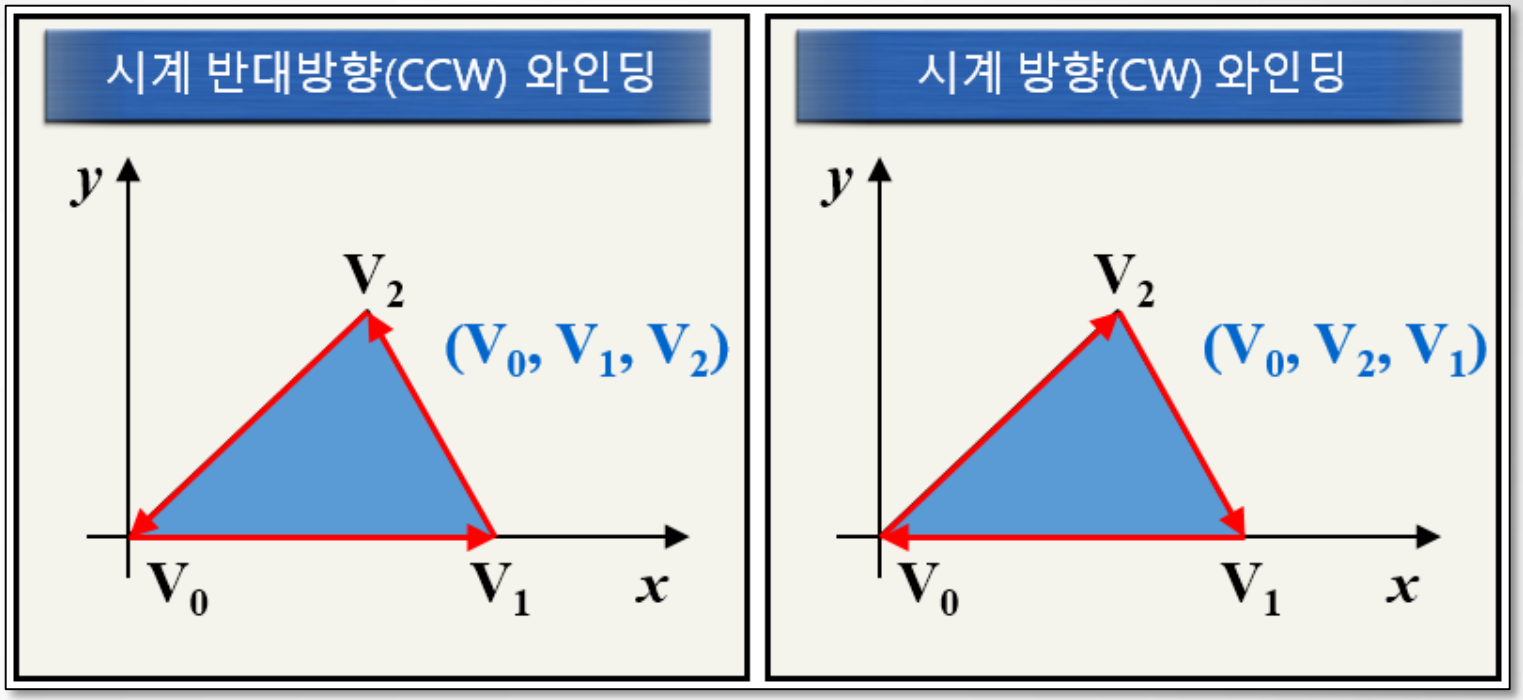
와인딩 규칙(Winding Rule)

- 현재 패스가 자기 자신을 통과하거나 현재 패스와 교차하는 여러 개의 서브 패스가 있을 경우에, fill() 메서드는 내부를 어떻게 채울까?



와인딩 규칙(Winding Rule)

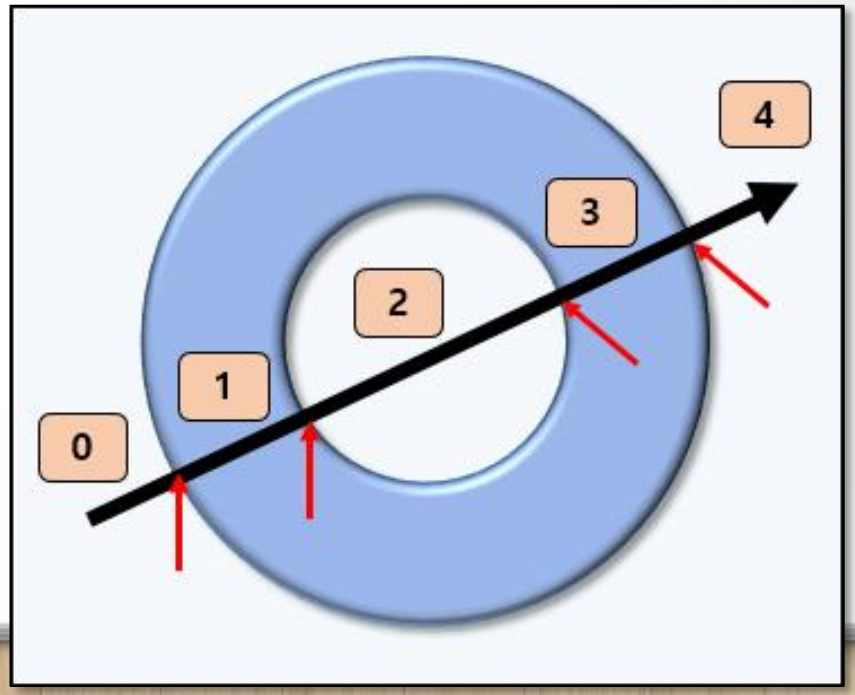
- 컴퓨터그래픽에서의 와인딩 방식



와인딩 규칙(Winding Rule)

짝-홀 와인딩(even-odd winding)

- 어떤 한 지점이 패스 내부에 있는지를 확인하려면, 다음 그림과 같이 다시 한번 그 지점을 통해서 선을 그려보면 알 수 있다.
- 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.



스타일 지정하기

와인딩 규칙(Winding Rule)

넌제로 와인딩(non-zero winding)

- 패스의 각 부분에 대한 드로잉을 하는 방향(시계 방향 및 시계 반대 방향)에 의존하는 방법
- 모든 시계 방향 회전의 경우에는 1씩 감소시키고, 모든 시계 반대 방향 회전의 경우에는 1씩 증가시킨다.
- 마지막으로 계산된 카운터가 0이 아니면, 해당 영역은 패스 안에 존재한다고 판단하여 브라우저에서는 fill() 메서드를 호출해서 해당 영역의 내부를 채운다.
- 그러나, 마지막 카운터가 0이라면 해당 영역은 패스 안에 존재하지 않는 것으로 판단하여 브라우저에서는 해당 영역 내부를 채우지 않는다.

와인딩 규칙(Winding Rule)

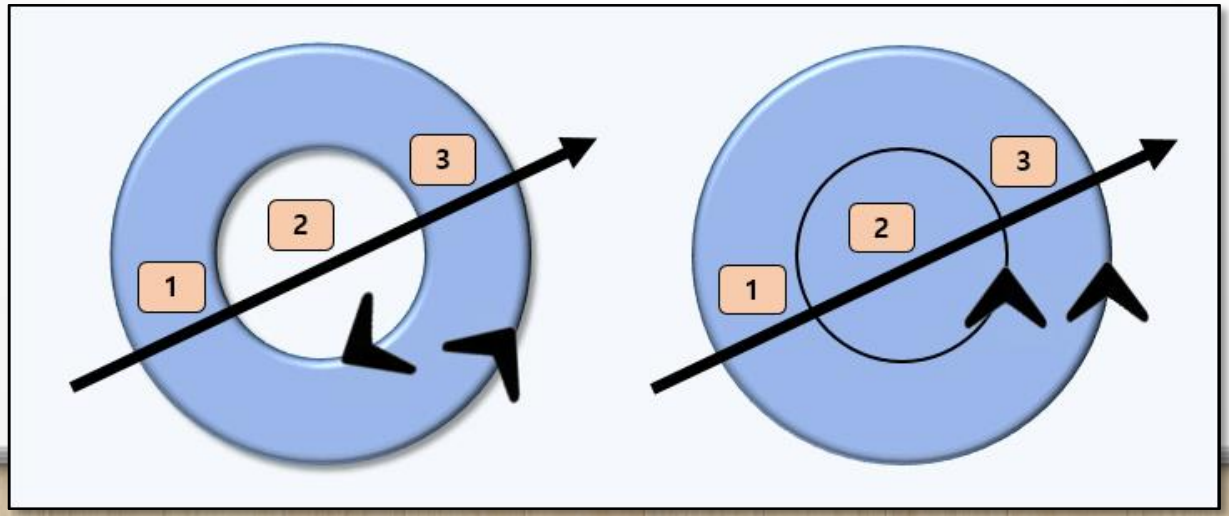
넌제로 와인딩(non-zero winding)

❖ 왼쪽 그림

- 지점 1. Total = 1, 따라서 내부라고 판단하여 내부를 채운다.
- 지점 2. Total = 1 - 1(시계방향) = 0, 따라서 외부라고 판단하여 내부를 채우지 않는다.
- 지점 3. Total = 1 - 1(시계방향) - 1(시계방향) = -1, 따라서 내부라고 판단하여 내부를 채운다.

❖ 오른쪽 그림

- 지점 1. Total = 1, 따라서 내부라고 판단하여 내부를 채운다.
- 지점 2. Total = 1 + 1(시계 반대 방향) = 2, 따라서 내부라고 판단하여 내부를 채운다.
- 지점 3. Total = 1 + 1(시계 반대 방향) + 1(시계 반대 방향) = 3,
  따라서 내부라고 판단하여 내부를 채운다.



와인딩 규칙(Winding Rule)

수정된 다각형 그리기

//function polygon(컨텍스트, 좌표(x, y), 반지름, 면의 수, 시작 각도, 그리는 방향) **function polygon(context, x, y, radius, sides, startAngle, anticlockwise)** {	수정

```
        if (sides < 3) return; //3각형 이하는 그리지 않도록 한다.
        var degree = (Math.PI * 2)/sides; //각 면에 대한 각도를 계산한다.
```

degree = anticlockwise ? -degree : degree; //각도의 방향을 반대로 계산하도록 한다.	**추가**

```
        context.save(); //드로잉 상태를 저정한다.
                context.translate(x,y);      //드로잉 좌표 공간을 다각형 중심좌표로 이동한다.
                context.rotate(startAngle); //시작 각도를 중심으로 그리도록 하기 위하여 회전한다.
                context.moveTo(radius,0);    //다각형의 시작 위치로 이동한다.
                for (var i = 1; i < sides; i++) { //면의 수 만큼 루프를 반복한다
                        //다음 꼭지점까지 선을 그린다.
                        context.lineTo(radius*Math.cos(degree*i),radius*Math.sin(degree*i));
                }
                context.closePath(); //패스를 닫는다.
        context.restore(); //기존 드로잉 상태를 복구한다.
}
```

와인딩 규칙(Winding Rule)

수정된 다각형 그리기

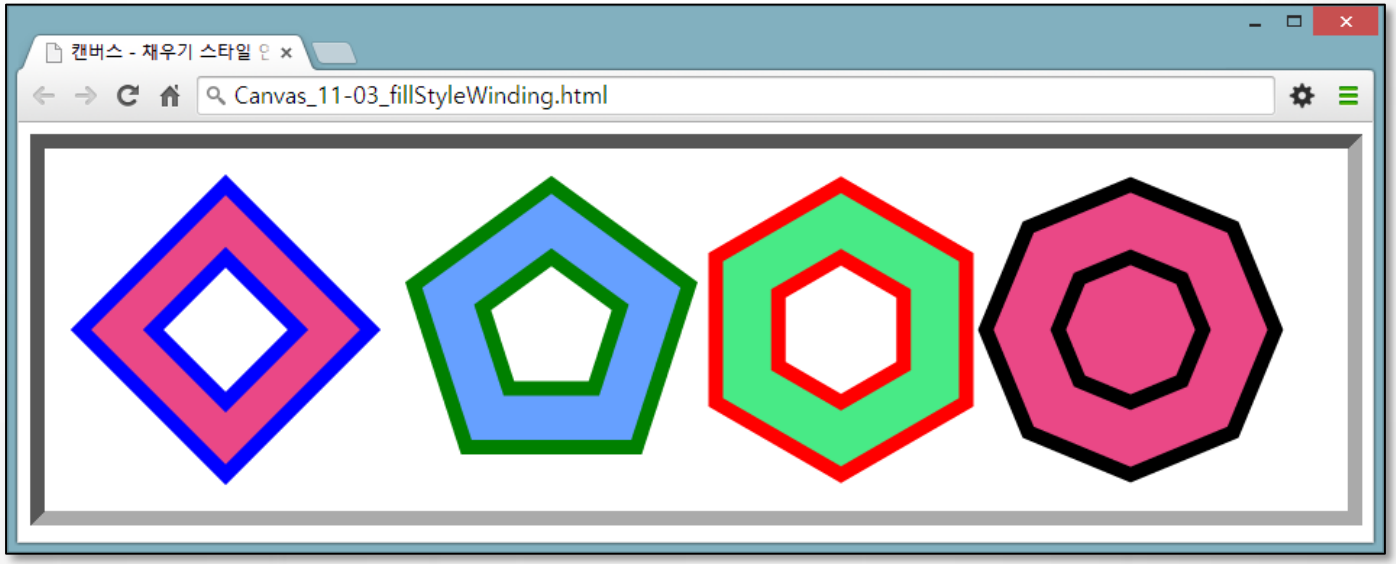
예제		Canvas_11-03_fillStyleWinding.html

Script	사각형	polygon(context,125, 125, **100**, 4, -Math.PI/2, **false**); //사각형 외부를 그린다
		polygon(context,125, 125, **50**, 4, -Math.PI/2, **true**); //사각형 내부를 그린다

	오각형	polygon(context,350, 125, **100**, 5, -Math.PI/2, **false**); //오각형 외부를 그린다
		polygon(context,350, 125, **50**, 5, -Math.PI/2, **true**); //오각형 내부를 그린다

	육각형	polygon(context,550, 125, **100**, 6, -Math.PI/2, **false**); //육각형 외부를 그린다
		polygon(context,550, 125, **50**, 6, -Math.PI/2, **true**); //육각형 내부를 그린다

	팔각형	polygon(context,750, 125, **100**, 8, -Math.PI/2, **false**); //팔각형 외부를 그린다
		polygon(context,750, 125, **50**, 8, -Math.PI/2, **false**); //팔각형 내부를 그린다



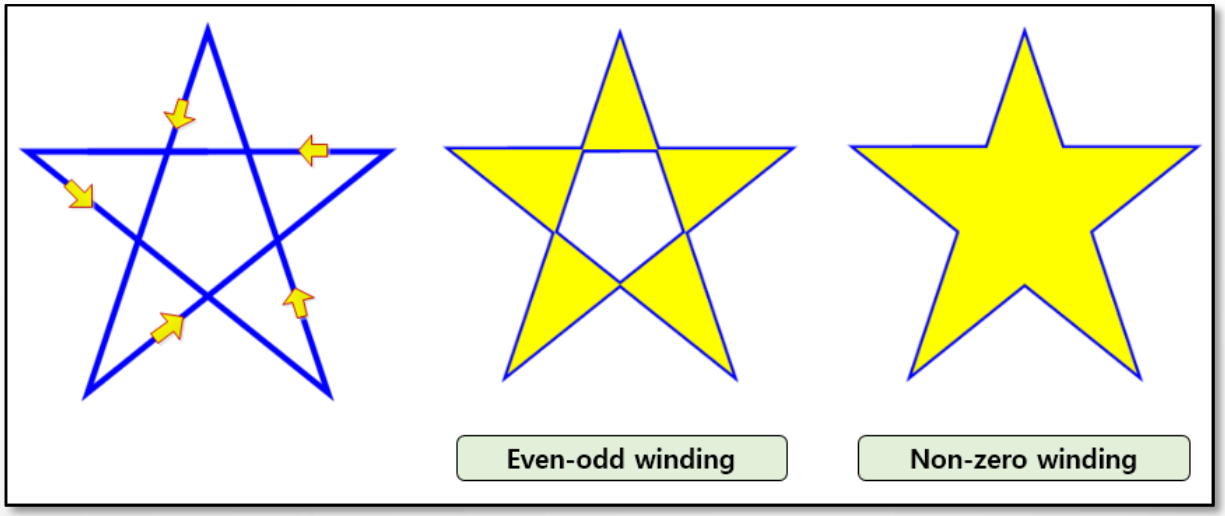
와인딩 규칙(Winding Rule)

HTML Living Standard 명세서

```
enum CanvasFillRule { "nonzero", "evenodd" };

void fill(optional CanvasFillRule fillRule = "nonzero");
void fill(Path2D path, optional CanvasFillRule fillRule = "nonzero");
```

짝-홀 와인딩	*context*.fill("evenodd");
넌제로 와인딩	*context*.fill("nonzero");

예제 Canvas_11-03_fillRule.html



그라데이션(Gradient) 스타일 지정

선형 그라데이션(Linear Gradient)

❖ 시작 좌표(x0, y0)와 종료 좌표(x1, y1)를 지정함으로써, 시작 좌표와 종료 좌표간의 위치에 따라서 색상 변화가 있는 그라데이션 효과

gradient = context.createLinearGradient(x0, y0, x1, y1)	선형 그라데이션 객체를 생성한다.

✓ 인수로 표시되는 좌표로 지정된 라인을 따라서 그리는 선형 그라데이션을 나타내는 CanvasGradient 객체를 반환한다.
✓ 만일, x0 = x1 그리고 y0 = y1 이면, 선형 그라데이션은 아무것도 그리지 않는다.

방사형 그라데이션(Radial Gradient)

❖ 원의 중심 좌표(x0, y0), 원의 반지름(r0), 또 다른 원의 중심 좌표(x1, y1), 또 다른 원의 반지름(r1)을 지정함으로써, 두 개의 가상의 원이 생성되고 그 두 개의 가상의 원 사이의 위치에 따라서 색상 변화가 있는 그라데이션으로 효과

gradient = context.createRadialGradient(x0, y0, r0, x1, y1, r1)	방사형 그라데이션 객체를 생성한다.

✓ 인수로 표시되는 원에 의해 주어진 원뿔을 따라서 그리는 방사형 그라데이션을 나타내는 CanvasGradient 객체를 반환한다. 지정된 반지름(r0, r1)중에 하나가 음수일 경우에는 IndexSizeError 예외 오류가 발생한다.

그라데이션(Gradient) 스타일 지정

변환점 색(경계 색) 지정

❖ 두 좌표간 또는 두 가상의 원 사이의 변환점 색상을 지정

context.addColorStop(*offset, color*)	주어진 오프셋 위치에 변환점 색상(color-stop)을 추가한다.

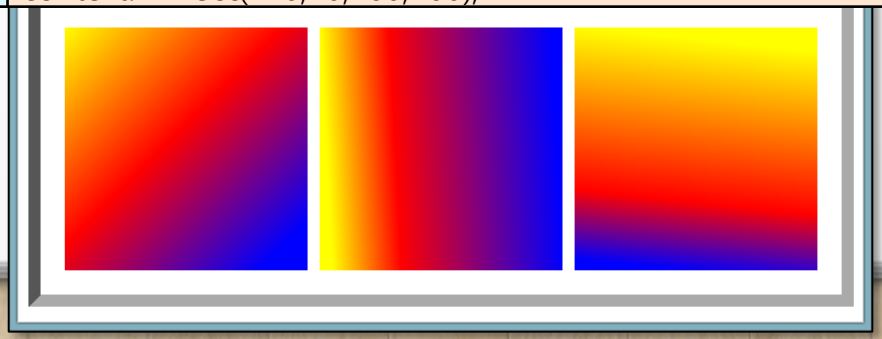
✓ 주어진 오프셋(offset) 위치에 그라데이션의 변환점 색상을 추가한다. 0.0은 한쪽 끝의 오프셋 위치이고 1.0은 다른 한쪽 끝의 오프셋 위치에 해당된다. 중간의 변환점 색상을 지정하기 위해서는 소수점으로 지정하면 된다(변환점 색상은 색상과 색상 사이의 경계색을 의미)

그라데이션 적용 방법

```
var 객체 = context.createLinearGradient(x0, y0, x1, y1);
    //또는 context.creatRadialGradient(x0, y0, r0, x1, y1, r1);
객체.addColorStop(시작 지점 오프셋, 시작 색상);
객체.addColorStop(끝 지점 오프셋, 색상);
context.fillStyle = 객체;
```

그라데이션(Gradient) 스타일 지정

예제		Canvas_11-03_GradientLinear.html
Script	1	context.beginPath(); //첫 번째 사각형을 그린다.
	2	var ***rectstyle*** = context.**createLinearGradient(20,20,200,200);** //선형 그라데이션 객체 생성
	3	***rectstyle*****.addColorStop(0,"yellow");** //시작 노란색의 옵셋을 지정
	4	***rectstyle*****.addColorStop(0.5,"red");** //중간 빨간색의 옵셋을 지정
	5	***rectstyle*****.addColorStop(1,"blue");** //종료 파란색의 옵셋을 지정
	6	context.**fillStyle** = ***rectstyle***; //그라데이션 객체를 채우기 스타일로 지정
	7	context.**fillRect**(20,20,200,200);
	8	
	9	context.beginPath(); //두 번째 사각형을 그린다.
	10	rectstyle = context.**createLinearGradient(230,20,420,10);**
	11	**rectstyle.addColorStop(0,"yellow");** //시작 노란색의 옵셋을 지정
	12	**rectstyle.addColorStop(0.3,"red");** //30% 위치의 빨간색상을 지정
	13	**rectstyle.addColorStop(1,"blue");** //종료 파란색의 옵셋을 지정
	14	context.**fillStyle** = ***rectstyle***; //그라데이션 객체를 채우기 스타일로 지정
	15	context.**fillRect**(230,20,200,200);
	16	
	17	context.beginPath(); //세 번째 사각형을 그린다.
	18	***rectstyle*** = context.**createLinearGradient(450,20,430,210);**
	19	***rectstyle*****.addColorStop(0,"yellow");** //시작 노란색의 옵셋을 지정
	20	***rectstyle*****.addColorStop(0.7,"red");** //70% 위치의 빨간색상을 지정
	21	***rectstyle*****.addColorStop(1,"blue");** //종료 파란색의 옵셋을 지정
	22	context.**fillStyle** = ***rectstyle***; //그라데이션 객체를 채우기 스타일로 지정
	23	context.**fillRect**(440,20,200,200);

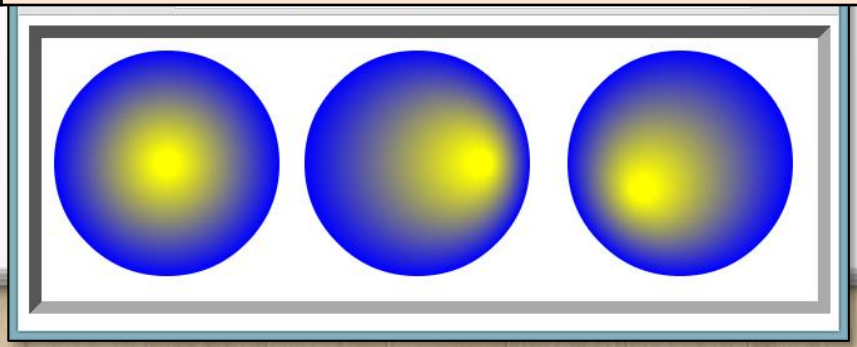


그라데이션(Gradient) 스타일 지정

예제		Canvas_11-03_GradientRadial.html

Script

```
1  context.beginPath(); //첫 번째 원을 그린다.
2                       //방사형 그라데이션 객체를 만든다.
3                       var gradient = context.createRadialGradient( 100, 100, 10, 100, 100, 90 );
4                       gradient.addColorStop(0, "yellow");
5                       gradient.addColorStop(1, "blue");
6                       context.fillStyle = gradient;
7                       context.arc(100, 100, 90, 0, 360*Math.PI/180, true);
8  context.fill();
9  context.beginPath(); //두 번째 원을 그린다.
10                      gradient = context.createRadialGradient( 350, 100, 10, 300, 100, 90 );
11                      gradient.addColorStop(0, "yellow");
12                      gradient.addColorStop(1, "blue");
13                      context.fillStyle = gradient;
14                      context.arc(300, 100, 90, 0, 360*Math.PI/180, true);
15 context.fill();
16 context.beginPath();  //세 번째 원을 그린다.
17                      gradient = context.createRadialGradient( 480, 120, 10, 510, 100, 90 );
18                      gradient.addColorStop(0, "yellow");
19                      gradient.addColorStop(1, "blue");
20                      context.fillStyle = gradient;
21                      context.arc(510, 100, 90, 0, 360*Math.PI/180, true);
22 context.fill();
```



그라데이션(Gradient) 스타일 지정

실습 – Canvas_Study11-04_LinearGradient.html

HTML

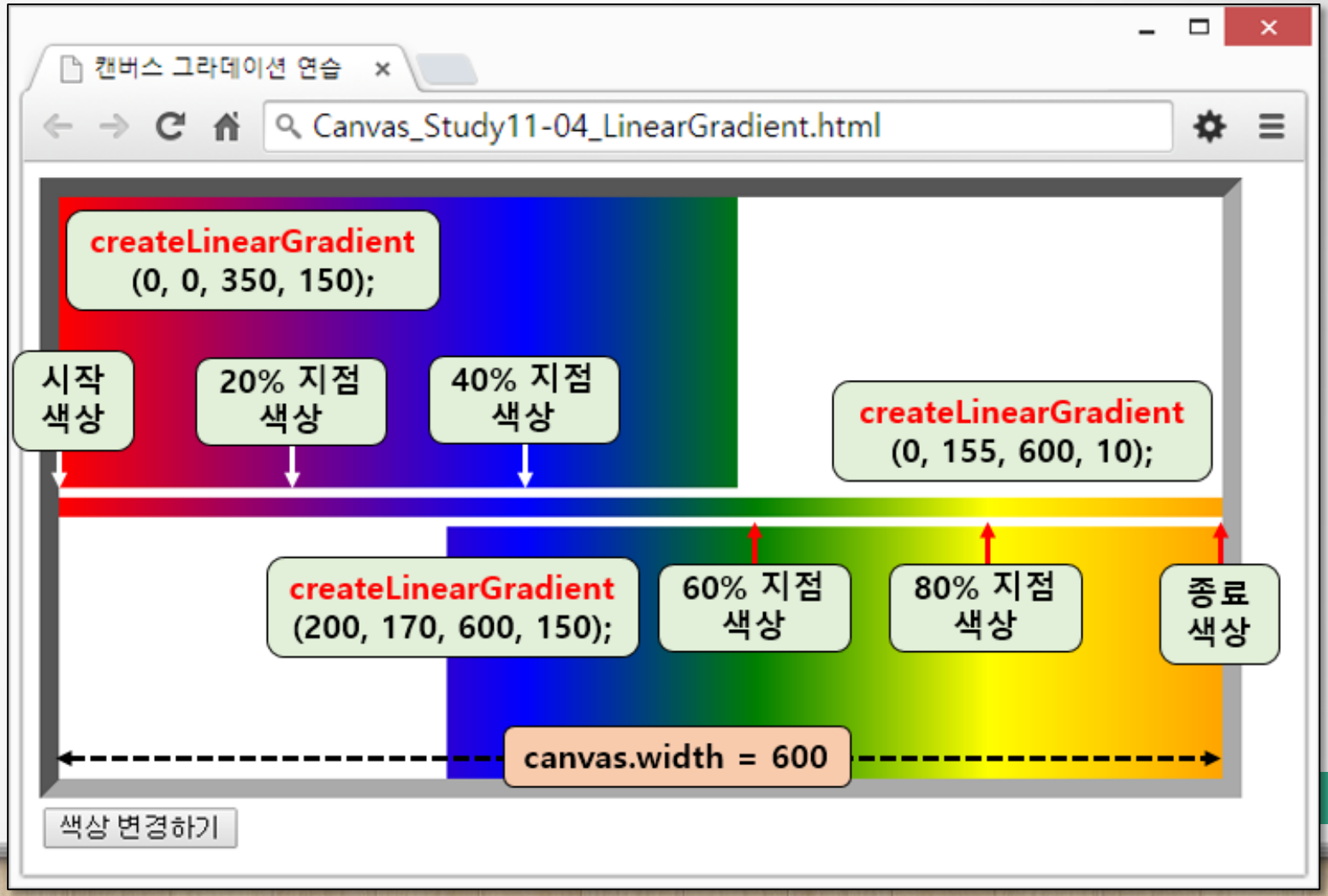
```
<button id="mybutton">색상 변경하기</button>
```

Script

```
var gradArray = ["red","purple","blue","green","yellow","orange"];
var gradient; //그라데이션을 위한 변수

document.getElementById("mybutton").addEventListener("click", changeColor, false);

doGradient();  //처음 지정된 색상으로 그라데이션 효과를 적용한다.

//배열 gradArray[]에 있는 색상으로 그라데이션 효과를 적용한다.
function doGradient() {
    gradient = context.createLinearGradient(0, 0, canvas.width, 0); //그라데이션 객체 생성
    gradient.addColorStop("0", gradArray[0]);    //시작 색상
    gradient.addColorStop(".20", gradArray[1]); //20% 지점의 색상
    gradient.addColorStop(".40", gradArray[2]); //40% 지점의 색상
    gradient.addColorStop(".60", gradArray[3]); //60% 지점의 색상
    gradient.addColorStop(".80", gradArray[4]); //80% 지점의 색상
    gradient.addColorStop("1.0", gradArray[5]); //마지막 색상

    context.fillStyle = gradient; //채우기 속성에 그라데이션 객체 스타일을 적용한다.
    context.fillRect(0, 0, 350, 150);      //그라데이션 스타일로 채워진 첫 번째 사각형을 그린다.
    context.fillRect(0, 155, 600, 10);      //그라데이션 영역을 비교하기 위한 사각형을 그린다.
    context.fillRect(200, 170, 600, 150); //그라데이션 스타일로 채워진 두 번째 사각형을 그린다.
}

//배열의 첫번째 색상을 제거해서 마지막 색상으로 이동시켜서 배열의 색상을 변경하고 그라데이션 효과를 적용한다.
function changeColor() {
    var temp = gradArray.shift();
    gradArray.push(temp);
    doGradient();    //변경된 색상으로 그라데이션 효과를 다시 적용시킨다.
}
```

그라데이션(Gradient) 스타일 지정

실습 – Canvas_Study11-04_LinearGradient.html



패턴(Pattern) 스타일 지정

- **패턴은 불투명한 CanvasPattern 인터페이스를 구현하는 객체로 표현**

*pattern = context.*createPattern(*image, repetition*)	CanvasPattern 객체를 반환한다.

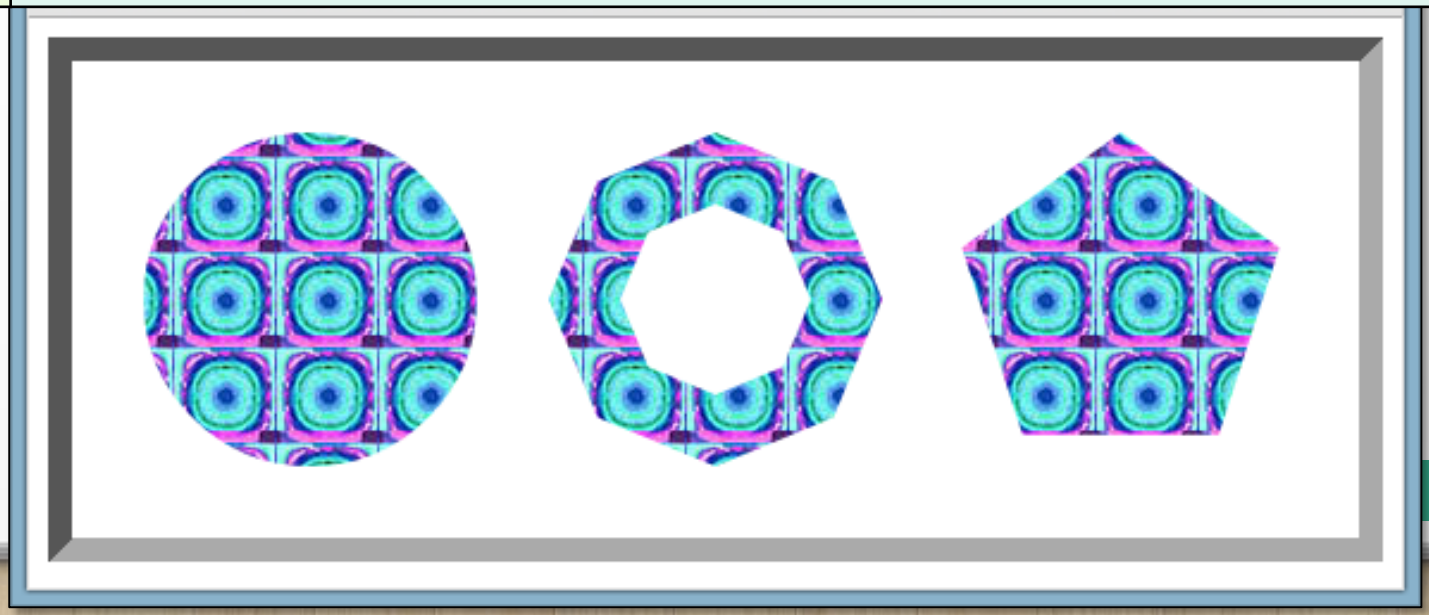
- ✓ 첫 번째 인수 image는 패턴으로 사용할 이미지를 지정한다. 이미지는 HTMLImageElement, HTMLCanvasElement, 그리고 HTMLVideoElement 중에서 지정해야 한다.
- ✓ 두 번째 인수 repetition는 주어진 image의 반복되는 방향을 나타낸다.
- ✓ 반복(repetition)으로 지정될 수 있는 값은 다음과 같으며, 그 이외의 값을 지정하면 SyntaxError 예외 오류가 발생한다. 이 값이 생략되면 기본적으로 repeat 값이 된다.
  - repeat(양방향), repeat-x(수평 방향만), repeat-y(수직 방향만), no-repeat(반복 안함)

- **패턴 채우기 적용 과정**

1. 이미지 파일의 Image 객체를 생성한다.
2. onload 이벤트 처리기를 사용하여 이미지 로드가 완료되면 createPattern() 메서드를 사용하여 패턴 객체를 생성한다.
3. fillStyle 속성에 패턴 객체를 지정한다.
4. 원하는 도형등을 그리면 내부를 패턴으로 채울 수 있게 된다.

패턴(Pattern) 스타일 지정

예제		Canvas_11-03_Pattern.html
Script	1	var *img* = new Image(); //이미지 객체를 만든다.
	2	*img*.src = "images/pattern.png"; //이미지 객체에 이미지를 지정한다.
	3	
	4	*img*.onload = function () { // 이미지가 로드될 때까지 기다렸다가 처리를 계속한다.
	5	context.beginPath ();
	6	var pattern = context. createPattern (*img*, ""); //이미지로 패턴 객체를 생성한다.
	7	context.fillStyle = pattern; //패턴 객체를 채우기 스타일로 지정한다.
	8	context.arc(100, 100, 70, 0, 2 * Math.PI, false); //원을 그린다.
	9	polygon(context, 270, 100, 70, 8, -Math.PI/2, false); //팔각형 외부를 그린다
	10	polygon(context, 270, 100, 40, 8, -Math.PI/2, true); //팔각형 내부를 그린다
	11	polygon(context, 440, 100, 70, 5, -Math.PI/2, false); //오각형을 그린다
	12	context.fill (); //원 내부를 채운다.
	13	}



패턴(Pattern) 스타일 지정

실습 – Canvas_Study11-05_Pattern.html

HTML

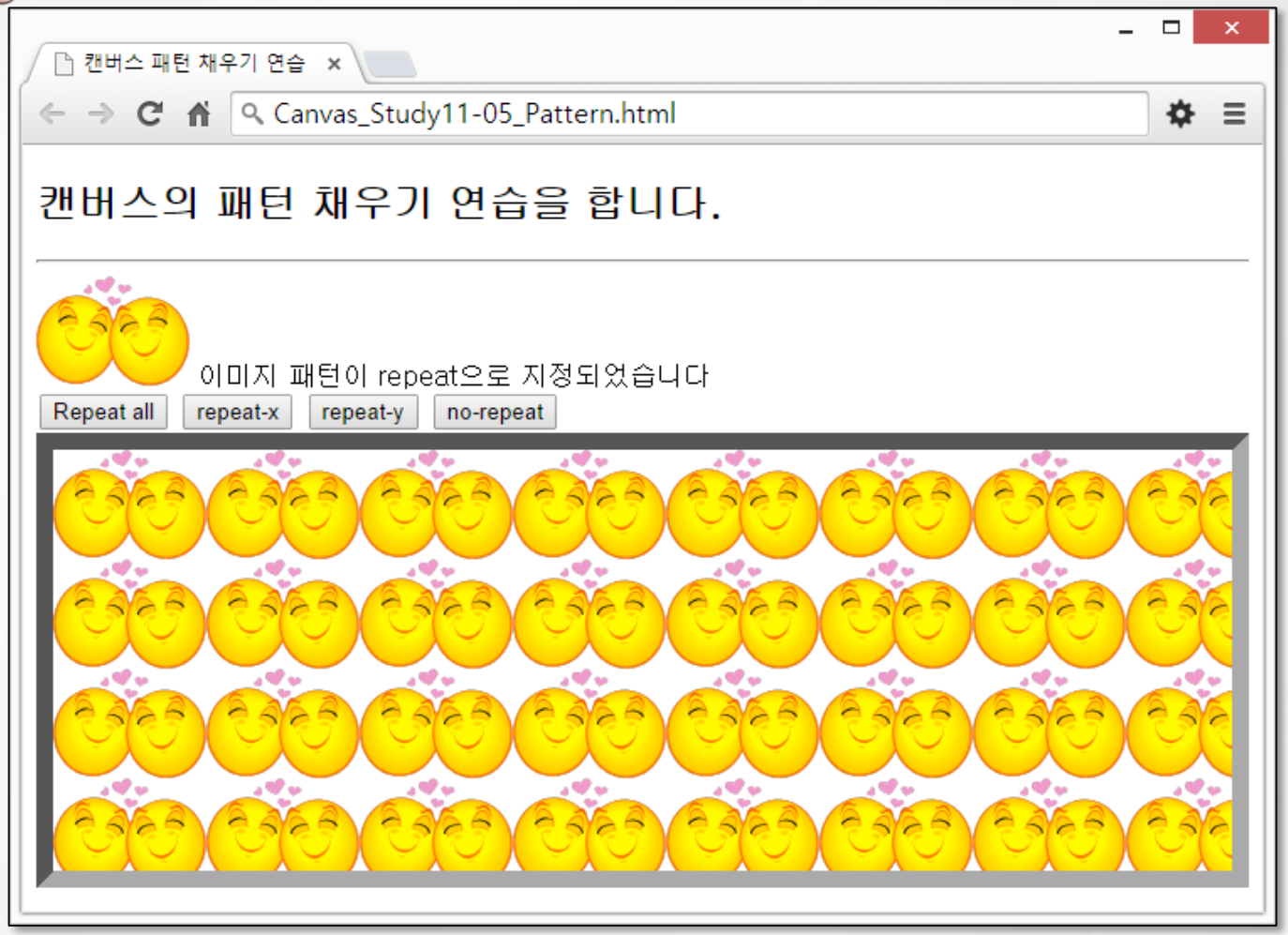
```
<div>
        <img id="myImage" src="images/love.png" />
        <p id="msg">이미지 패턴이 repeat 으로 지정되었습니다.</p>
</div>
<button onclick="draw('repeat')">Repeat all</button>
<button onclick="draw('repeat-x')">repeat-x</button>
<button onclick="draw('repeat-y')">repeat-y</button>
<button onclick="draw('no-repeat')">no-repeat</button><br />
```

Script

```
var image, pattern, message; //이미지와 패턴, 메시지를 위한 객체를 생성한다..
function draw(direction) {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");

        context.clearRect(0, 0, canvas.width, canvas.height);  //캔버스 영역을 지운다.
        pattern = context.createPattern(image, direction);    //패턴 객체를 생성한다.
        context.fillStyle = pattern; //채우기 스타일을 패턴으로 지정한다.
        context.fillRect(0, 0, canvas.width, canvas.height);  //캔버스 영역을 지운다.
        message.innerHTML = "이미지 패턴이 " + direction + "으로 지정되었습니다";
}

message = document.getElementById("msg");   //P 요소의 아이디를 가져온다.
image = document.getElementById("myImage"); //이미지의 아이디를 가져온다.

//이미지 리스너를 통해 이미지가 로드되면, draw() 메서드를 호출한다.
image.addEventListener("load", function () {
        draw("repeat"); // 초기 repeat 값으로 패턴을 채우도록 draw() 메서드를 호출한다.
}, false);
```

패턴(Pattern) 스타일 지정

실습 – Canvas_Study11-05_Pattern.html



그림자 스타일 지정

- **어떤 도형이나 텍스트, 또는 이미지 등에 입체감을 주기 위한 방법**
  - ➢ **그림자를 지정하면, 도형이나 텍스트 등이 캔버스 위에 마치 떠 있는 듯한 느낌이 들도록 하는 시각적 효과를 가진다**

context.shadowColor [= value]	현재 그림자의 색상을 반환한다.
✓ CSS로 지정할 수 있는 값을 사용하여 그림자의 색상을 변경할 수 있다. CSS 색상으로 구문 분석할 수 없는 값은 무시된다. ✓ 기본 값은 완전히 투명한 검정(rgba(0, 0, 0, 0))으로 초기화된다.	

context.shadowOffsetX [= value]	현재 그림자의 수평(X 좌표) 오프셋을 반환한다.
✓ 도형이나 텍스트로부터 그림자까지의 수평 오프셋을 픽셀 단위로 지정한다. ✓ 기본 값은 0으로 초기화되며, 변환 행렬의 영향을 받지 않는다.	

context. shadowOffsetY [= value]	현재 그림자의 수직(Y 좌표) 오프셋을 반환한다.
✓ 도형이나 텍스트로부터 그림자까지의 수직 오프셋을 픽셀 단위로 지정한다. ✓ 기본 값은 0으로 초기화되며, 변환 행렬의 영향을 받지 않는다.	

context.shadowBlur [= value]	그림자의 흐림 정도(크기 또는 범위)를 지정한다.
✓ 값을 지정하여 그림자의 흐림 정도를 지정할 수 있다. 값이 작을수록 그림자는 선명해진다. ✓ 컨텍스트가 생성될 때, shadowBlur 속성은 0으로 초기화된다.	

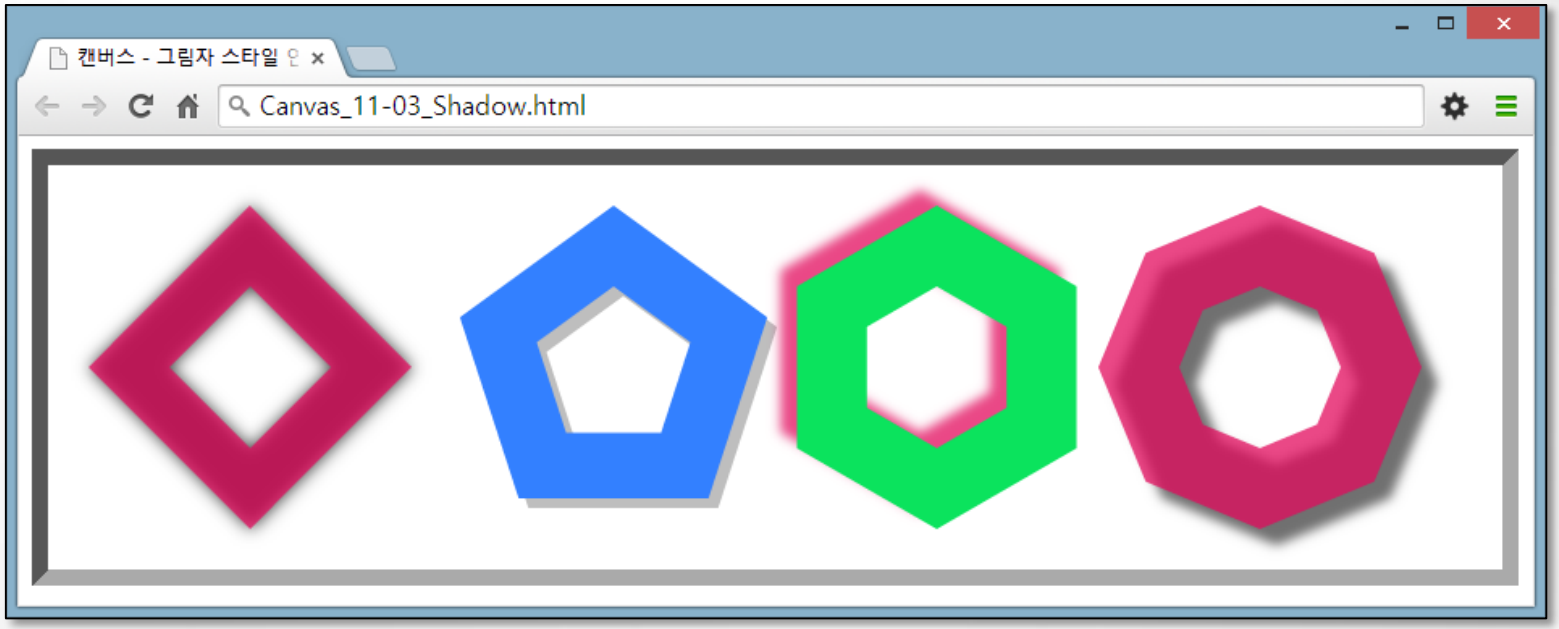
그림자 스타일 지정

예제		Canvas_11-03_Shadow.html
Script	1	context.beginPath();
	2	context.shadowBlur = 15; //그림자의 흐림 정도를 크게 지정하여 많이 흐리게 한다.
	3	context.shadowColor = "rgb(0, 0, 0)"; //그림자의 색상을 검은색으로 지정한다.
	4	polygon(context, 125, 125, 100, 4, -Math.PI/2, false); //사각형 외부를 그린다
	5	polygon(context, 125, 125, 50, 4, -Math.PI/2, true); //사각형 내부를 그린다
	6	context.fillStyle="rgba(227,11,93,0.75)";
	7	context.fill();
	8	
	9	context.beginPath();
	10	context.shadowBlur = 0; //그림자의 흐림 정도를 주지 않는다.
	11	context.shadowOffsetX = 6; //그림자를 도형의 6픽셀만큼 오른쪽으로 위치시킨다.
	12	context.shadowOffsetY = 6; //그림자를 도형의 6픽셀만큼 아래쪽으로 위치시킨다.
	13	context.shadowColor = "rgba(125, 125, 125, 0.5)"; //그림자를 50% 투명하게 지정한다.
	14	polygon(context, 350, 125, 100, 5, -Math.PI/2, false); //오각형 외부를 그린다
	15	polygon(context, 350, 125, 50, 5, -Math.PI/2, true); //오각형 내부를 그린다
	16	context.fillStyle="rgba(51,128,255,1.0)";
	17	context.fill();
	18	
	19	context.beginPath();
	20	context.shadowBlur = 10;
	21	context.shadowOffsetX = -10; //그림자를 도형의 10픽셀만큼 왼쪽으로 위치시킨다.
	22	context.shadowOffsetY = -10; //그림자를 도형의 10픽셀만큼 위쪽으로 위치시킨다.
	23	context.shadowColor = "rgba(227,11,93,0.75)"; //그림자를 25% 투명하게 지정한다.
	24	polygon(context, 550, 125, 100, 6, -Math.PI/2, false); //육각형 외부를 그린다
	25	polygon(context, 550, 125, 50, 6, -Math.PI/2, true); //육각형 내부를 그린다
	26	context.fillStyle="rgba(11,227,93,1.0)";
	27	context.fill();

그림자 스타일 지정

예제		Canvas_11-03_Shadow.html

Script	29
	30
	31
	32
	33
	34
	35
	36
	37
		context.beginPath();
		context.shadowBlur = 10;
		context.shadowOffsetX = 10; //그림자를 도형의 10픽셀만큼 오른쪽으로 위치시킨다.
		context.shadowOffsetY = 10; //그림자를 도형의 10픽셀만큼 아래쪽으로 위치시킨다.
		context.shadowColor = 'rgba(0,0,0,0.75)'; //그림자의 색상을 25% 투명한 검은색으로 지정한다.
		polygon(context, 750, 125, 100, 8, -Math.PI/2, false); //팔각형 외부를 그린다
		polygon(context, 750, 125, 50, 8, -Math.PI/2, true); //팔각형 내부를 그린다
		context.fillStyle="rgba(227,11,93,0.75)";
		context.fill();



다음 학습

1. 도형 합성 및 변환
2. 텍스트 그리기
3. 이미지 처리하기
4. 애니메이션(Animation)
5. 히트 영역(Hit Regions)