생능출판

HTML 표준화 문서를 기반으로 하는 지침서

속이 깊은 HTML5 & CSS3

김명진 지음

10강

캔버스 Part-1

- 캔버스 구성 및 기본 드로잉

학습 목표

HTML5에서 가장 흥미롭게 주목 받고 있는 기능 중에 하나가 바로 캔버스일 것이다. 유화를 그릴 때 사용되는 천이란 뜻의 캔버스는 그림을 그리기 위하여 놓는 그림판 또는 도화지라고 생각하면 된다. 이번 장에서는 브라우저에 캔버스를 생성하고 기본적인 도형 및 선을 그리는 방법에 대해서 살펴보도록 하겠다.

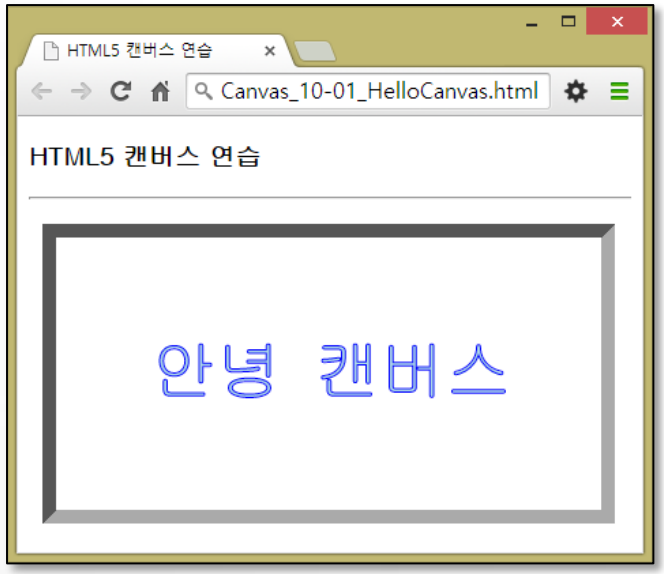
Section

1. 캔버스(Canvas) 기본 내용
2. 사각형 그리기
3. 선 그리기

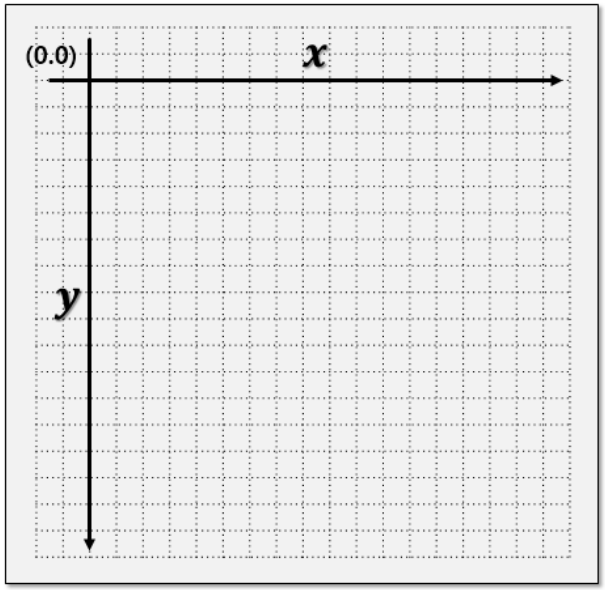
캔버스(Canvas) 기본 내용

표준화 문서	HTML Canvas 2D Context - http://www.w3.org/html/wg/drafts/2dcontext/html5_canvas_CR/
표준화 단계	W3C Last Call Working Draft(Editors Draft), (2014-06-11)
HTML Living Standard	http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html Last Updated (2014-06-12)

캔버스 요소의 좌표 시스템과 크기 지정하기



캔버스 예제



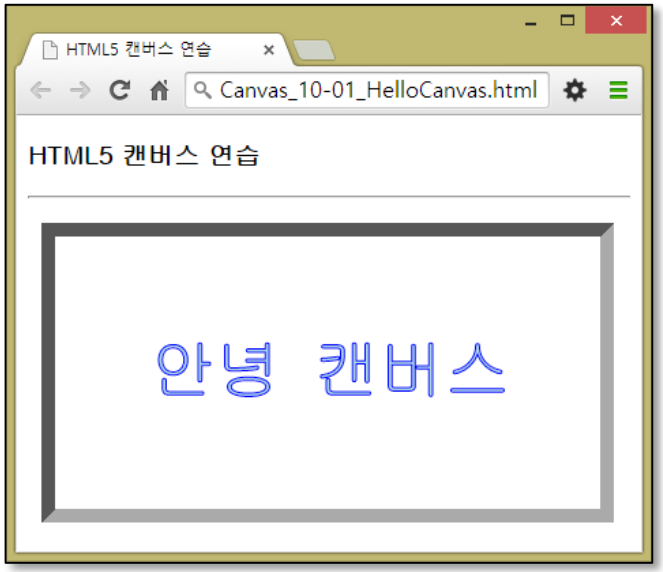
캔버스 좌표 시스템

캔버스(Canvas) 기본 내용

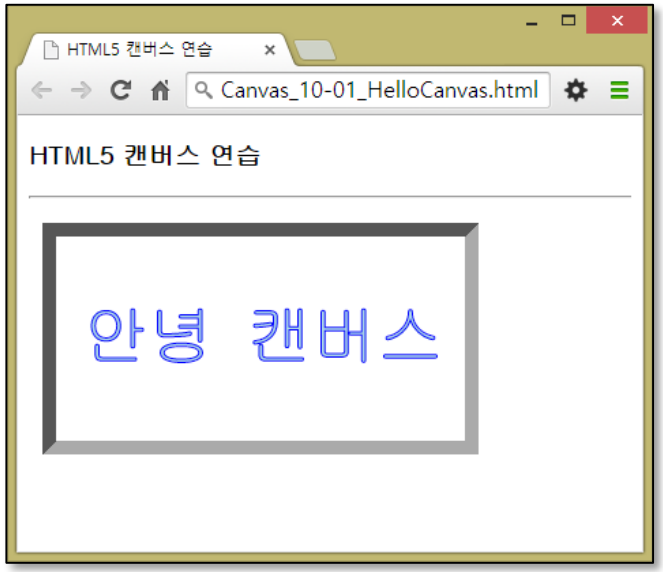
캔버스 요소의 좌표 시스템과 크기 지정하기

예제 Canvas_10-01_HelloCanvas.html

```
1  <canvas id="myCanvas" width="400" height="200">
2                  이 브라우저는 캔버스를 지원하지 않습니다.
3  </canvas>
4  <script>
5      var canvas = document.getElementById("myCanvas");
6      var context = canvas.getContext("2d");
7      context.font = "36pt 굴림체";
8      context.fillStyle = "lightblue";
9      context.strokeStyle = "blue";
10     context.fillText("안녕 캔버스", canvas.width/2 - 130, canvas.height/2 + 15);
11     context.strokeText("안녕 캔버스", canvas.width/2 - 130, canvas.height/2 + 15);
12 </script>
```



canvas 요소의 속성을 이용한 크기 변경



CSS를 이용한 캔버스 크기 변경

캔버스 요소의 좌표 시스템과 크기 지정하기

캔버스 속성

속성 값	설명
width	드로잉 표면의 가로(폭) 크기를 나타내며, 기본적으로 브라우저에서는 캔버스 요소의 크기를 드로잉 표면과 같은 크기로 생성한다. 기본 값은 300이며, 음수가 아닌 정수여야 한다.
height	드로잉 표면의 세로(높이) 크기를 나타내며, 기본적으로 브라우저에서는 캔버스 요소의 크기를 드로잉 표면과 같은 크기로 생성한다. 기본 값은 150이며, 음수가 아닌 정수여야 한다.

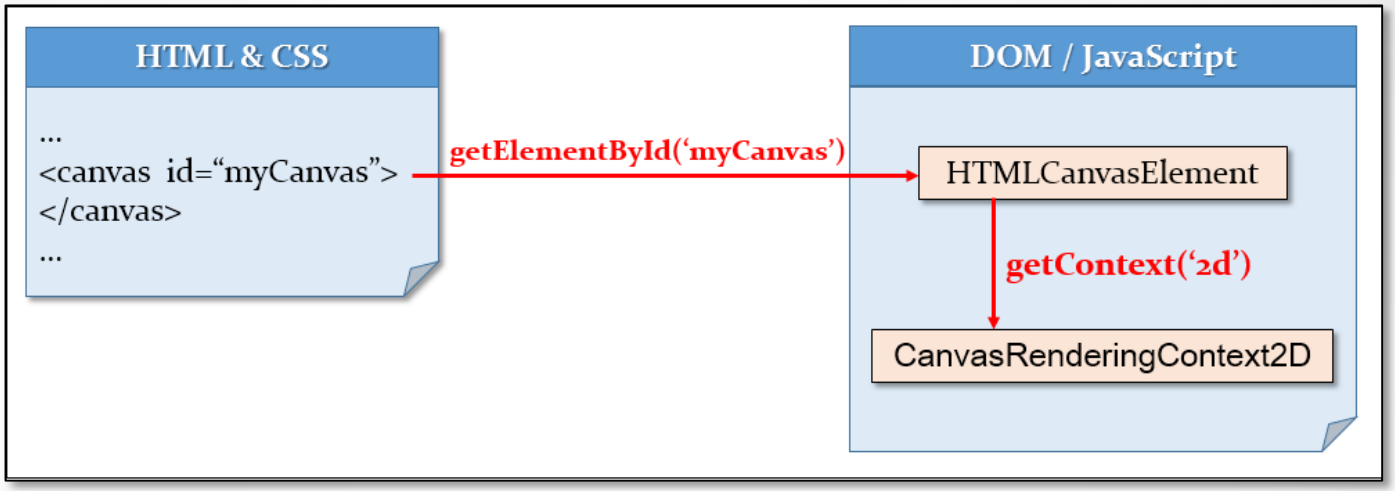
캔버스 API

메서드	설명
getContext( )	캔버스의 컨텍스트를 반환한다. 각 캔버스에는 하나의 컨텍스트가 있으며 각 컨텍스트는 하나의 캔버스와 관련되어 있다. ContextId 2D에 대한 새로운 객체를 반환하는 경우에, 브라우저는 새로운 CanvasRenderingContext2D 개체를 반환하고 추가적인 인수는 무시된다.
toDataURL (type, quality)	img 요소의 src 속성에 할당할 수 있는 데이터 URL을 반환한다. ✓ type - 이미지의 종류를 나타내는 것으로, "image/jpeg" 또는 "image/png"와 같이 지정할 수 있다. 지정되지 않으면, 기본적으로 "image/png"로 설정된다. ✓ quality - 이미지의 품질을 나타내는 것으로, 0.0 ~ 1.0(손실이 없는 가장 좋은 품질) 사이의 실수 값을 반드시 지정해야 한다.
toBlob (callback,type, args….)	캔버스의 이미지를 포함하고 있는 파일을 나타내는 Blob를 생성한다. ✓ callback - 브라우저에서 Blob에 대한 참조를 호출하는 콜백 함수를 나타낸다. ✓ type - 이미지의 종류를 나타내는 것으로, "image/jpeg" 또는 "image/png"와 같이 지정할 수 있다. 지정되지 않으면, 기본적으로 "image/png"로 설정된다. ✓ args … - 이미지 특성을 제어하기 위해 quality 등의 다른 인수를 추가할 수 있다.

캔버스(Canvas) 기본 내용

캔버스 컨텍스트

- 컨텍스트는 모든 그래픽 능력을 제공하며, 캔버스는 컨텍스트를 위한 컨테이너로서의 역할만 하고, 실제 캔버스에 그리는 등의 기능들은 모두 컨텍스트를 통해서 처리



방법1	var **캔버스객체** = **document.getElementById("캔버스 아이디");** var 컨텍스트 = **캔버스객체.getContext("2d");**
방법2	var 컨텍스트 = **document.getElementById("캔버스 아이디").getContext("2d");**

예제1	var ***canvas*** = document.**getElementById**("myCanvas"); var context = ***canvas***.**getContext**("2d");
예제2	var context = **document.getElementById("myCanvas").getContext("2d");**

캔버스 상태 저장 및 복원

- 캔버스 컨텍스트는 드로잉 상태(drawing state)의 스택을 관리하기 위하여 다음과 같이 컨텍스트의 드로잉 상태를 저장하고 복원할 수 있는 save() 메서드와 restoe() 메서드를 제공

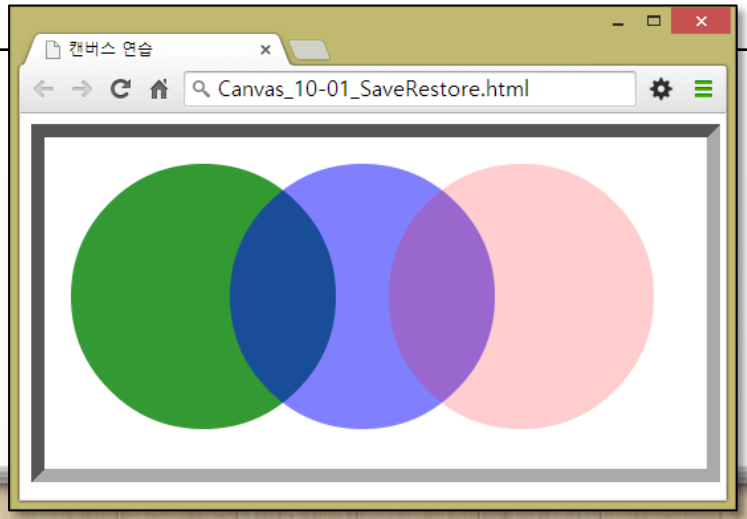
메서드	설명
***context*.save()**	현재 드로잉 상태의 복사본을 드로링 상태의 저장소인 스택의 마지막에 저장한다.
***context*.restore()**	스택에 저장되어 있는 마지막 상태를 꺼내옴으로써, 현재 상태를 다시 복원한다.

```
1  function draw_area() { stroke_style, fill_style} {
2      context.save(); /컨텍스트를 스택에 저장한다.
3          context.fillStyle = fill_style;
4          context.strokeStyle = stroke_style;
5          // 다른 그리기 작업 진행 ....
6      context.restore(); //스택으로부터 저장한 컨텍스트를 복원한다.
7  }
```

캔버스(Canvas) 기본 내용

캔버스 상태 저장 및 복원

예제		Canvas_10-01_SaveRestore.html
Script	1	var canvas = document.getElementById("myCanvas"); //캔버스 객체 생성
	2	var **context** = canvas.**getContext**("2d"); //컨텍스트 생성
	3	
	4	// 드로잉 상태(색상 및 투명도)를 3 개 저장한다
	5	var colors = new Array ("red", "blue", "green"); //색상 3개
	6	var alphas = new Array (0.2, 0.5, 0.8); //투명도 3개
	7	for (var i = 0; i < 3; i++) {
	8	**context**.fillStyle = colors [i];
	9	**context**.globalAlpha = alphas [i];
	10	**context**.**save**();
	11	}
	12	// 저장된 드로잉 상태를 복원하여 3개의 원을 그린다
	13	for (var i = 0; i < 3; i++) {
	14	**context**.**restore**(); // 드로잉 상태를 꺼내온다. 녹색, 파란색, 빨간색 순으로 꺼내온다
	15	**context**.beginPath ();
	16	**context**.arc ((i+1) * 120, 120, 100, 0, Math.PI * 2, false);
	17	**context**.fill ();
	18	}



드로잉 기본 작업

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

① 캔버스 정의

사용 예제

```
<canvas id="myCanvas" width="400" height="200">
        이 브라우저는 캔버스를 지원하지 않습니다.
    <img src="sample.jpg" width="400" height="200">
</canvas>
```

② 그리기를 컨텍스트 생성

사용 예제

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

③ 자바스크립트를 이용한 그림 그리기

캔버스(Canvas) 기본 내용

드로잉 기본 작업

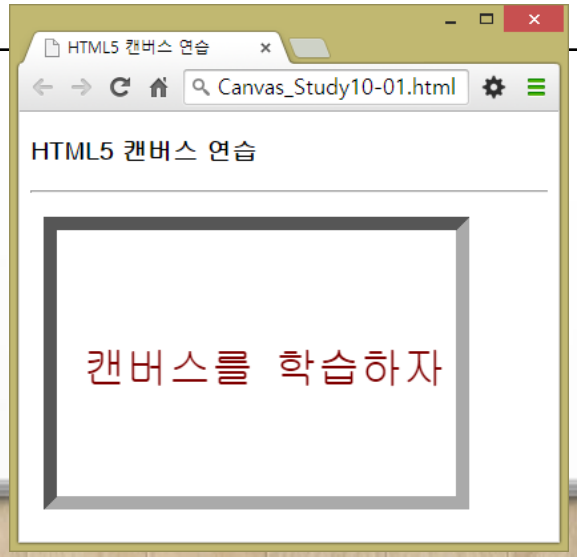
실습 – Canvas_Study10-01.html

CSS

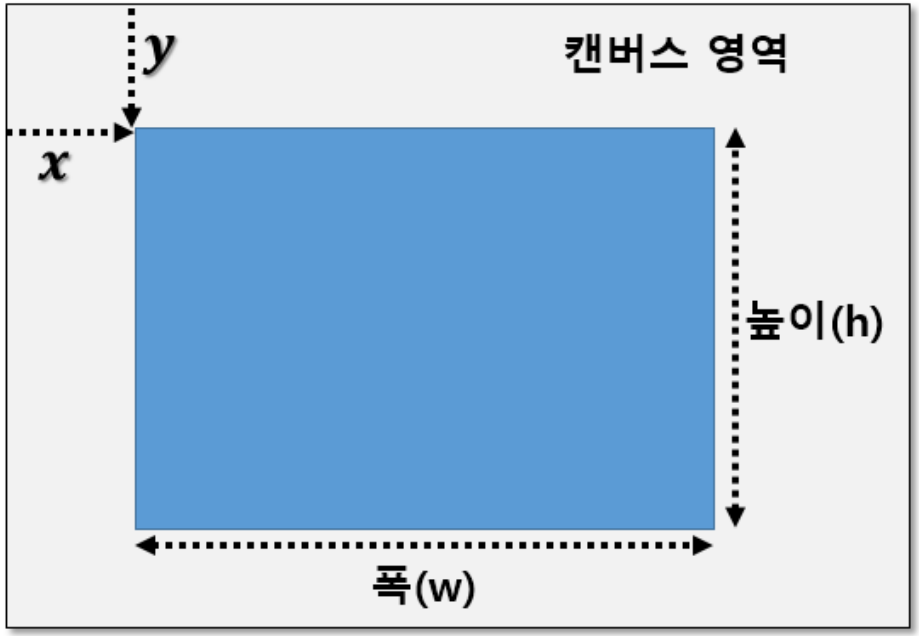
```
1  canvas { margin: 10px; padding 10px; border: 10px inset #aaa }
```

HTML

```
1  <canvas id="myCanvas" width="300" height="200">
2              이 브라우저는 캔버스를 지원하지 않습니다.
3  </canvas>
4  <script>
5     var canvas = document.getElementById("myCanvas");
6     var context = canvas.getContext("2d");
7     context.font = "24pt 굴림체";
8     context.fillStyle = "maroon";
9     context.fillText("캔버스를 학습하자", canvas.width/2 - 130, canvas.height/2 + 15);
10 </script>
```



사각형 그리기 메서드

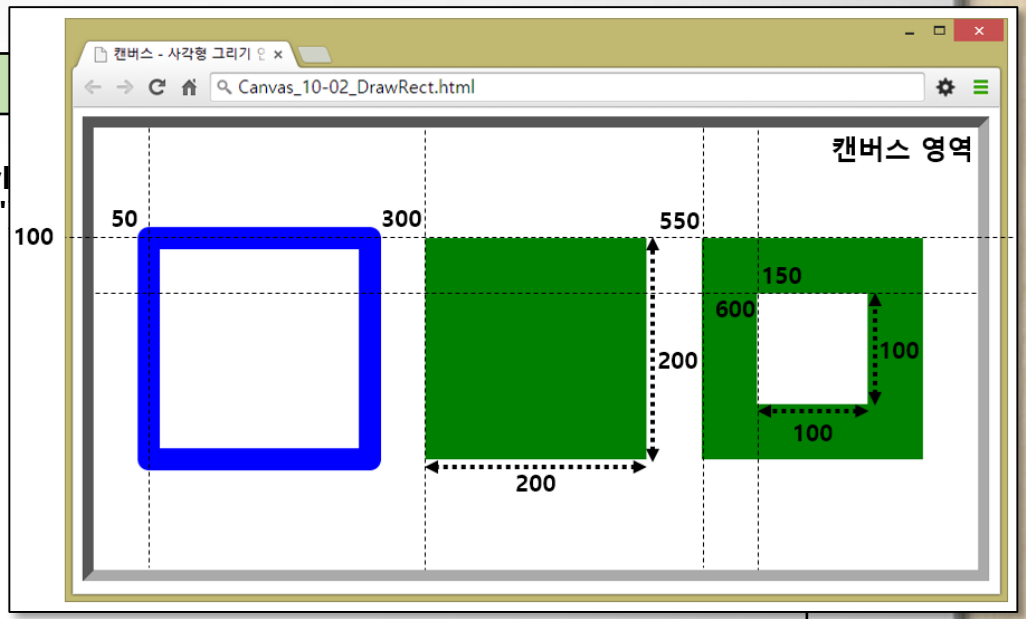


context.fillRect(x, y, w, h)	색이 채워진 사각형 영역을 그린다.
context.strokeRect(x, y, w, h)	테두리만 있는 사각형 영역을 그린다.
context.clearRect(x, y, w, h)	지정한 사각형 영역을 지운다.

사각형 그리기 메서드

예제 살펴보기

예 제		Canvas_10-02_DrawRect.html
Script	1	function rect() {
	2	var *canvas* = document.**getElementBy**
	3	var *context* = **canvas.getContext("2d"**
	4	
	5	***context*.lineJoin** = "round";
	6	***context*.lineWidth** = 20;
	7	***context*.strokeStyle** = "blue";
	8	***context*.fillStyle** = "green";
	9	
	10	***context*.strokeRect**(50, 100, 200, 200);
	11	***context*.fillRect**(300, 100, 200, 200);
	12	
	13	***context*.fillRect**(550, 100, 200, 200);
	14	***context*.clearRect**(600, 150, 100, 100);
	15	}
HTML	1	<body onload="rect();">
	2	<**canvas** id="**myCanvas**" width="800" height="400" style="border: 10px inset #aaa">
	3	캔버스에 사각형 그리기 연습
	4	</canvas>
	5	</body>



패스와 서브 패스

- 패스(Path) – 각 도형들을 이루는 선들의 집합
- 서브 패스 – 각각의 선
  - ✓ CanvasDrawingStyles 인터페이스를 구현하는 객체는 단 한 개의 기본 패스(path)를 가지고 있으며, 이를 현재 패스라고 부른다.
  - ✓ 현재 패스는 여러 개의 서브 패스로 구성될 수 있고 각 패스는 0개 이상의 서브 패스의 목록을 가지고 있다.

context.beginPath( )	현재 패스를 초기화한다.
context.closePath()	현재의 패스를 닫는다.
context.lineTo(x, y)	직선을 연결한다.
context.moveTo(x, y)	주어진 점으로 시작하는 새로운 서브 패스를 만든다.
context.stroke( )	현재 패스 또는 주어진 패스의 서브 패스를 현재의 선 스타일로 그린다.
context.fill( )	현재 패스 또는 주어진 패스의 서브 패스를 현재의 채우기 스타일로 채운다.
context.rect(x, y, w, h)	사각형을 그린다.

패스와 서브 패스

예제 Canvas_10-03_Rect1.html

context.**beginPath**();	현재 패스를 초기화 하기 때문에 모든 서브 패스를 지운다.
context.**rect**(10, 10, 100, 100);	4개의 점으로 구성된 서브 패스를 추가한다.
context.**stroke**();	서브 패스의 윤곽선을 그린다.
context.**beginPath**();	현재 패스를 가진 모든 서브 패스(앞의 4개의 서브 패스)를 지운다.
context.**rect**(110, 110, 100, 100);	4개의 점으로 구성된 서브 패스를 추가한다.
context.**stroke**();	서브 패스의 윤곽선을 그린다.

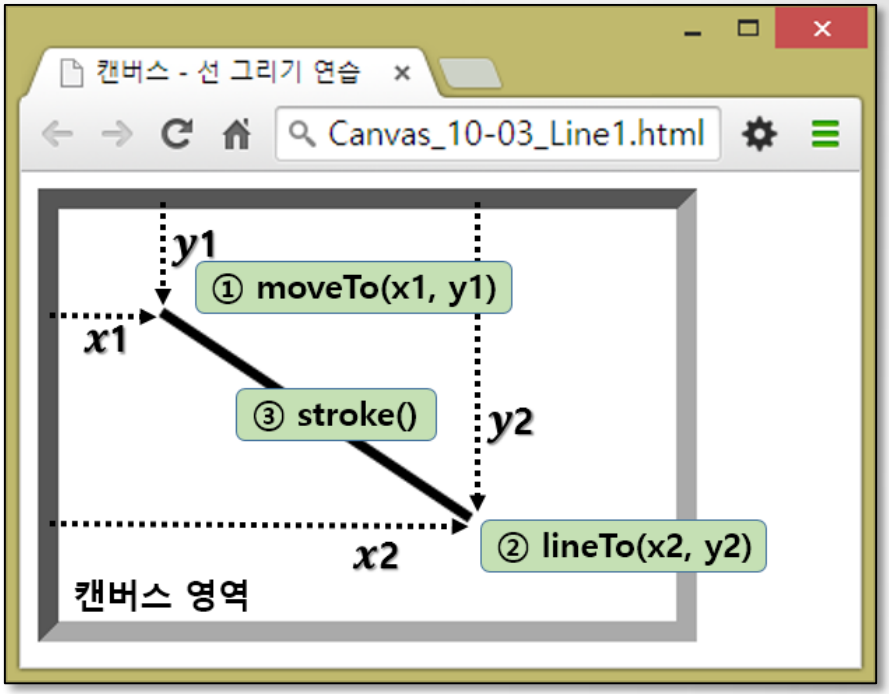
예제 Canvas_10-03_Rect2.html

context.**beginPath**();	현재 패스를 초기화 하기 때문에 모든 서브 패스를 지운다.
context.**rect**(10, 10, 100, 100);	4개의 점으로 구성된 서브 패스를 추가한다.
context.**stroke**();	서브 패스의 윤곽선을 그린다.
context.**rect**(110, 110, 100, 100);	4개의 점으로 구성된 서브 패스를 추가한다.
context.**stroke**();	서브 패스의 윤곽선을 그린다.

선 그리기

예 제	Canvas_10-03_Line1.html

context.beginPath();
① *context*.moveTo(x1, y1);
② *context*.lineTo(x2, y2);
③ *context*.stroke();

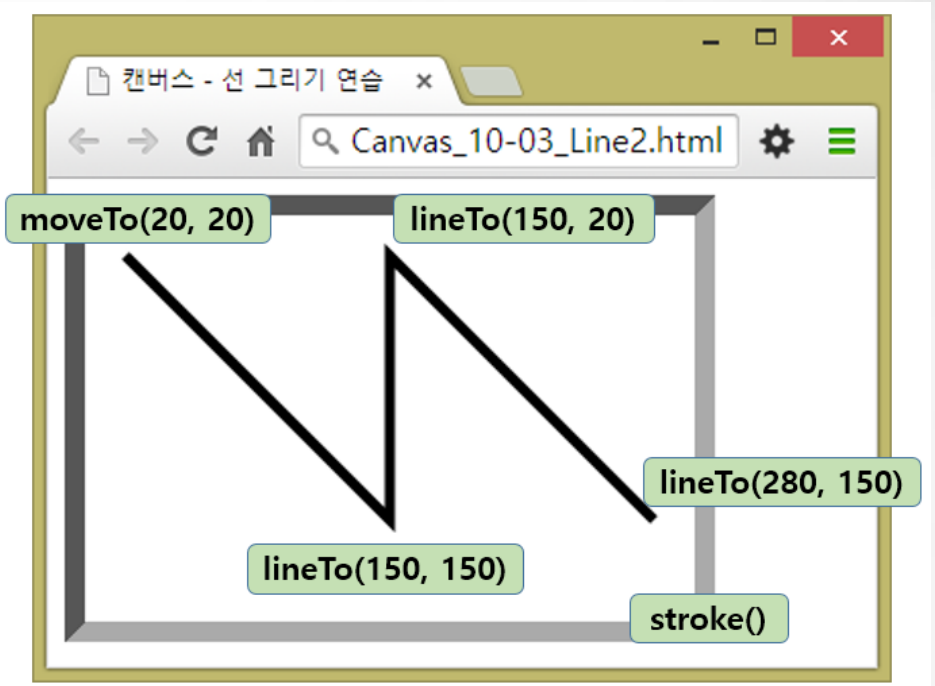


- stroke() 메서드를 호출하기 전에는 패스를 그리지 않는다.

선 그리기

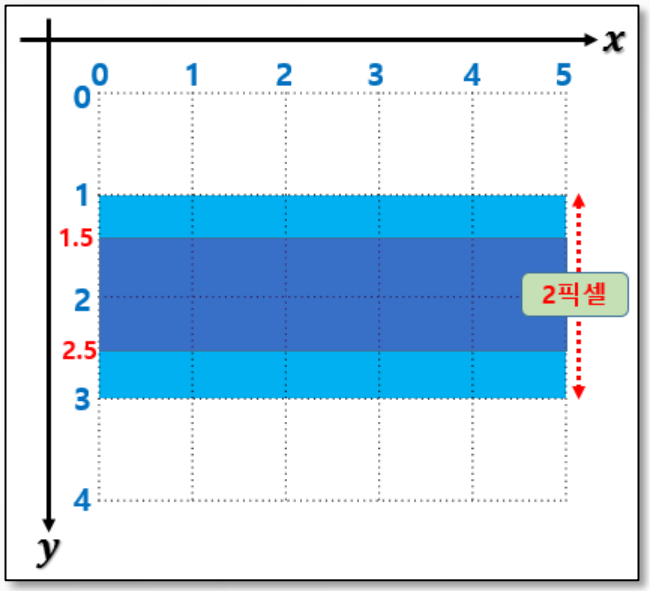
예 제	Canvas_10-03_Line2.html

```
context.beginPath();
context.moveTo(20, 20);
context.lineTo(150, 150);
context.lineTo(150, 20);
context.lineTo(280, 150);
context.stroke();
```

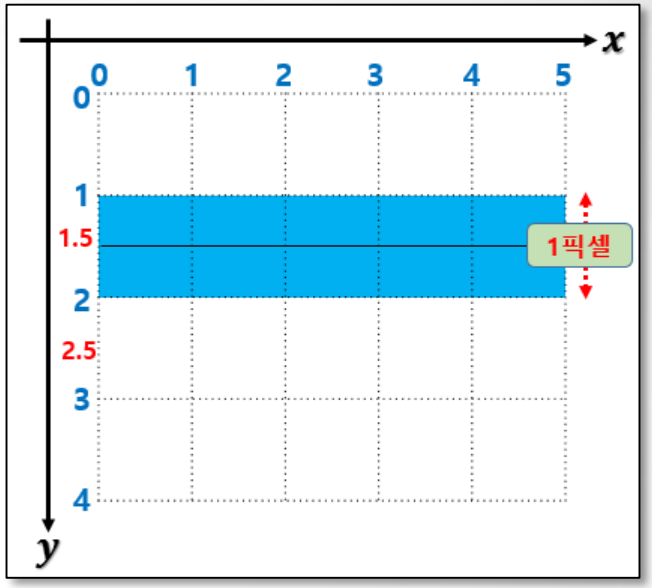


- 생성된 패스는 stroke() 메서드나 fill() 메서드를 호출해야 캔버스에 표시된다는 점을 잊지 말자.

선의 경계와 픽셀 경계



픽셀 경계에 선 그리기



픽셀 사이에 선 그리기

선의 경계와 픽셀 경계

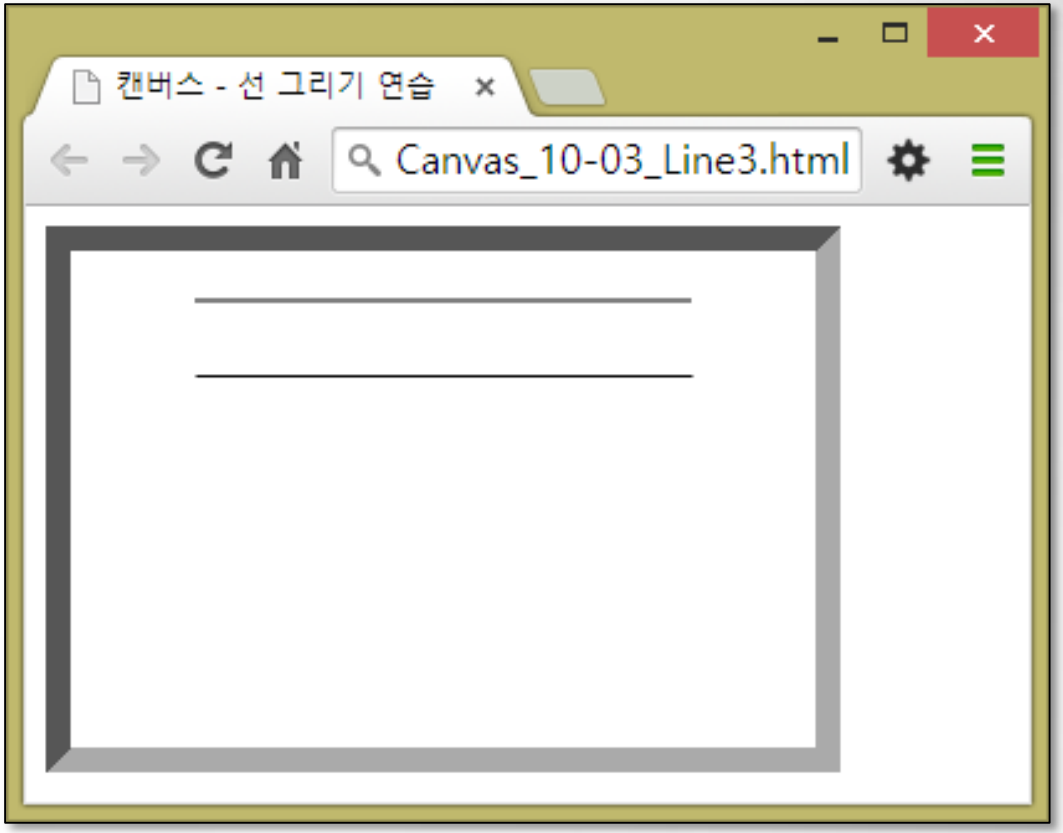
예 제	Canvas_10-03_Line3.html

```
context.lineWidth = 1;


context.beginPath();
context.moveTo(50, 20);
context.lineTo(250, 20);
context.stroke();

context.beginPath();
context.moveTo(50.5, 50.5);
context.lineTo(250.5, 50.5);
context.stroke();


...
```



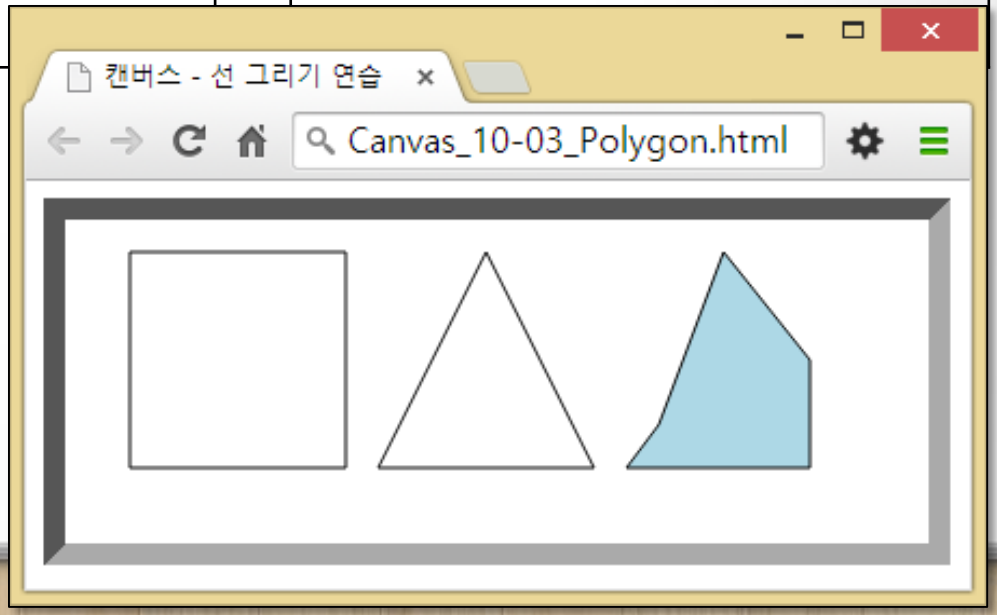
다각형 그리기

- 다각형을 그리기 위해서는 앞에서 학습한 선 그리기에서 closePath() 메서드를 사용

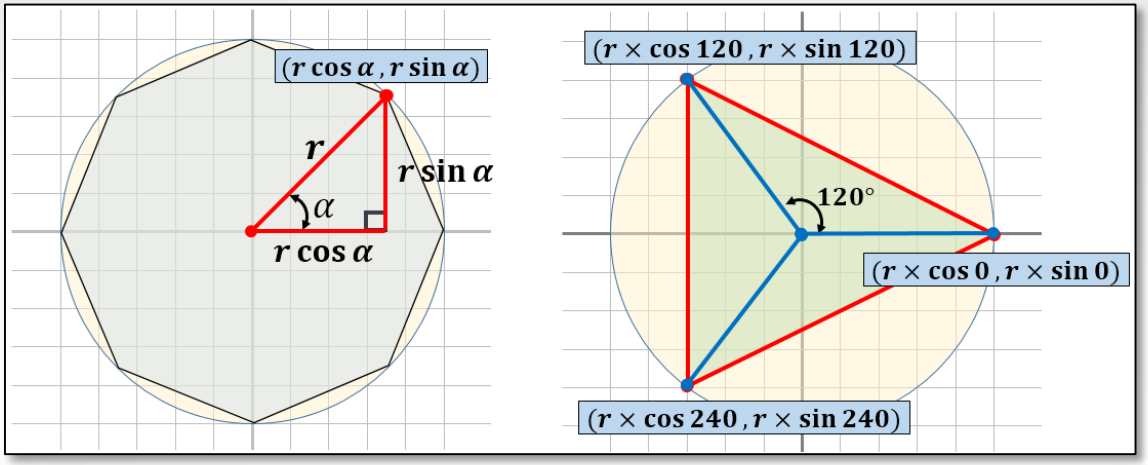
예 제	Canvas_10-3_Polygon.html

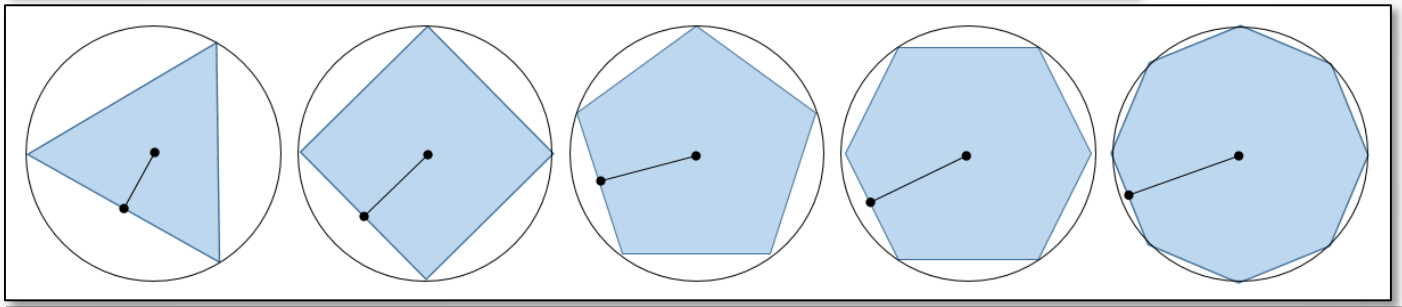
```
1  context.beginPath(); // 사각형 그리기
2      context.moveTo(30,15);
3      context.lineTo(130,15);
4      context.lineTo(130,115);
5      context.lineTo(30,115);
6  context.closePath();
7  context.stroke();
8  context.beginPath(); // 삼각형 그리기
9      context.moveTo(145,115);
10     context.lineTo(195,15);
11     context.lineTo(245,115);
12 context.closePath();
13 context.stroke();
```

```
14 context.beginPath(); // 다각형 그리기
15 context.fillStyle = 'lightblue';
16     context.moveTo(260,115);
17     context.lineTo(275,95);
18     context.lineTo(305,15);
19     context.lineTo(345,65);
20     context.lineTo(345,115);
21 context.closePath();
22 context.fill();
   context.stroke();
```



다각형 그리기





알고리즘

- 가장 먼저 원의 반지름(R)과 면의 수(N)를 계산한다(3각형은 3면, 4각형은 4면, …)
- 원의 중심으로부터 정다각형의 각 측면에 의한 각도(360/N)를 계산한다.
- 첫 번째 꼭지점의 위치를 (R, 0)으로 지정한다.
- 면의 수(N)만큼 루프를 통해서 꼭지점이 위치하는 각도를 계산한다.
- 각 꼭지점들끼리의 드로잉(직선 및 채우기 등)을 수행한다.

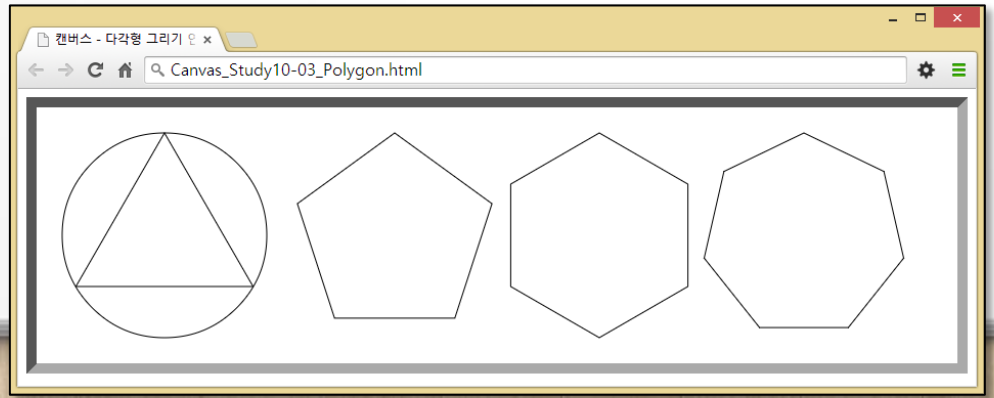
다각형 그리기

실습 – Canvas_Study10-03_Polygon.html

Script

```
1  function polygon(context, x, y, radius, sides, startAngle) {
2      if (sides < 3) return; //3각형 이하는 그리지 않도록 한다.
3      var degree = (Math.PI * 2)/sides; //각 면에 대한 각도를 계산한다.
4      context.save();             //드로잉 상태를 저정한다.
5      context.translate(x,y);     //드로잉 좌표 공간을 다각형 중심좌표로 이동한다.
6      context.rotate(startAngle); //시작 각도를 중심으로 그리도록 하기 위하여 회전한다.
7      context.moveTo(radius, 0); //다각형의 시작 위치로 이동한다.
8      for (var i = 1;  i < sides;  i++) { //면의 수 만큼 루프를 반복한다
9          //다음 꼭지점까지 선을 그린다.
10         context.lineTo( radius * Math.cos(degree * i), radius * Math.sin(degree * i) );
11     }
12     context.closePath(); //패스를 닫는다.
13     context.restore();   //기존 드로잉 상태를 복구한다.
14 }
```

- polygon(context, 125, 125, 100, 3, -Math.PI/2); //삼각형을 그린다
- polygon(context, 350, 125, 100, 5, -Math.PI/2); //오각형을 그린다
- polygon(context, 550, 125, 100, 6, -Math.PI/2); //육각형을 그린다
- polygon(context, 750, 125, 100, 7, -Math.PI/2); //칠각형을 그린다



점선 그리기

context.lineDashOffset [= value]	점선 모양의 패턴과 동일한 단위에서의 위상 오프셋을 반환한다.
context.setLineDash (*segments*)	현재의 점선 패턴을 설정한다.

- ✓ 전달되는 인수 segments 값은
  점선의 패턴으로 선이 그려지는 부분과 그려지지 않는 부분이 반복되는 배열이 된다.
- ✓ 점선의 패턴을 지우기 위해서는 segments 값을 [0,0]으로 지정하면 된다.

segments = *context*.getLineDash()	현재의 점선 패턴을 반환한다.

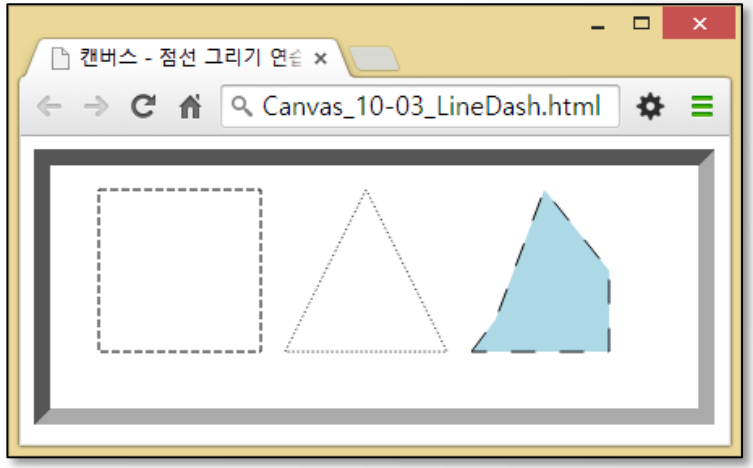
- ✓ 반환 된 배열의 목록들은 항상 짝수 값을 가지고 있어야 한다.

예 제 Canvas_10-3_LineDash.html

```
//사각형 그리기
context.setLineDash([5, 2]);
context.rect(30, 15, 100, 100);

//삼각형 그리기
context.setLineDash([1, 2]);
context.moveTo(145, 115);
context.lineTo(195, 15);
context.lineTo(245, 115);

//다각형 그리기
context.setLineDash([15]);
context.moveTo(260, 115);
context.lineTo(275, 95);
context.lineTo(305, 15);
context.lineTo(345, 65);
context.lineTo(345, 115);
```



선 스타일 지정하기

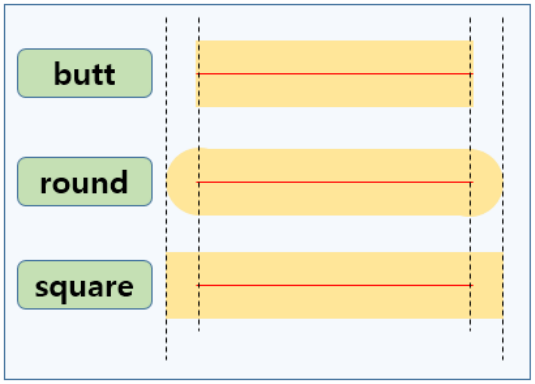
선의 색상 및 두께 지정

context.strokeStyle [= value]	도형을 그리기 위해 사용되는 현재의 선 스타일을 반환한다.
context.lineWidth [= value]	현재 선의 두께를 반환한다.

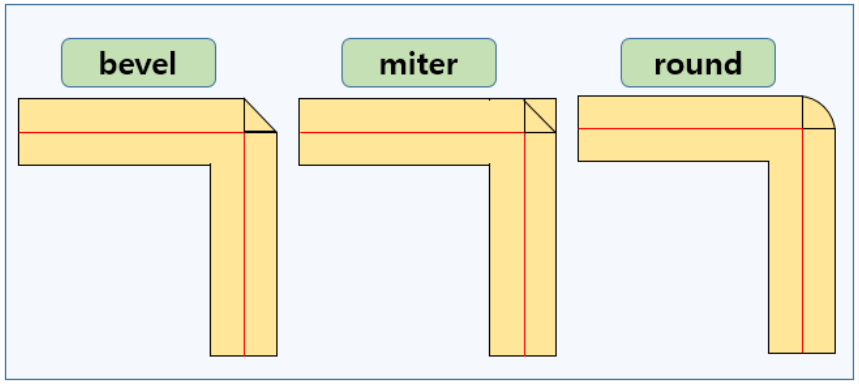
선의 끝 부분 및 연결 부분 스타일 지정

context.lineCap [= value]	현재 선의 끝 부분 스타일을 반환한다.
context.lineJoin [= value]	현재 선의 연결 부분 스타일을 반환한다.

- lineCap 속성

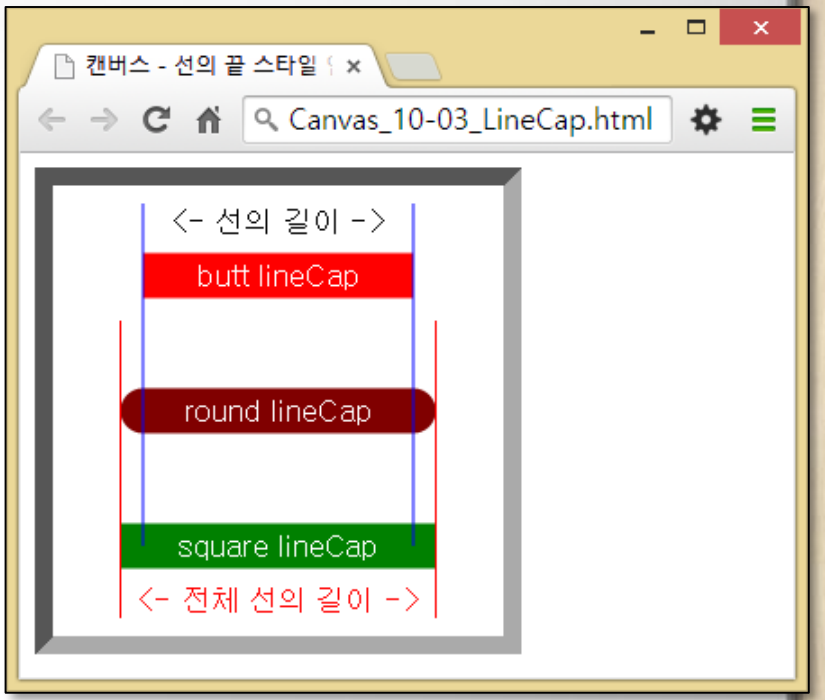


- lineJoin 속성



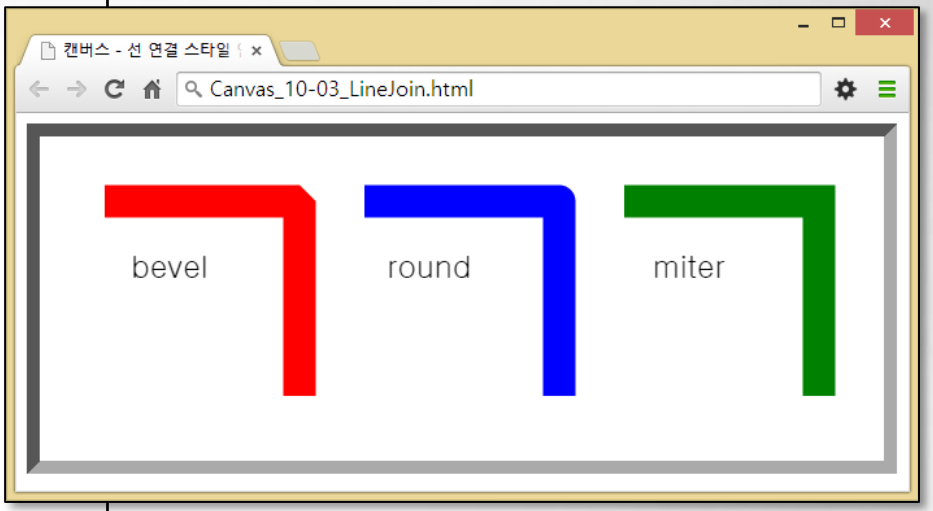
선 스타일 지정하기

예 제		Canvas_10-03_LineCap.html
Script	1	var lineStart = 50, lineEnd = 200, yStart = 50;
	2	context.**lineWidth** = "25";
	3	
	4	context.beginPath();
	5	context.**lineCap** = **"butt"**;
	6	context.**strokeStyle** = "Red";
	7	context.moveTo(lineStart, yStart);
	8	context.lineTo(lineEnd, yStart);
	9	context.stroke();
	10	
	11	context.beginPath();
	12	context.**lineCap** = **"round"**;
	13	context.**strokeStyle** = "maroon";
	14	context.moveTo(lineStart, yStart + 75);
	15	context.lineTo(lineEnd, yStart + 75);
	16	context.stroke();
	17	
	18	context.beginPath();
	19	context.**lineCap** = **"square"**;
	20	context.**strokeStyle** = "green";
	21	context.moveTo(lineStart, yStart + 150);
	22	context.lineTo(lineEnd, yStart + 150);
	23	context.stroke();



선 스타일 지정하기

예 제		Canvas_10-03_LineJoin.html
Script	1	var lineStart = 50, lineEnd = 200, yStart = 50;
	2	context.**lineWidth** = "25";
	3	
	4	context.beginPath();
	5	context.**strokeStyle** = "Red";
	6	context.**lineJoin** = "bevel";
	7	context.moveTo(lineStart, yStart);
	8	context.lineTo(lineEnd, yStart);
	9	context.lineTo(lineEnd, yStart + 150);
	10	context.stroke();
	11	
	12	context.beginPath();
	13	context.**strokeStyle** = "blue";
	14	context.**lineJoin** = "round";
	15	context.moveTo(lineStart + 200, yStart);
	16	context.lineTo(lineEnd + 200, yStart);
	17	context.lineTo(lineEnd + 200, yStart + 150);
	18	context.stroke();
	19	
	20	context.beginPath();
	21	context.**strokeStyle** = "green";
	22	context.**lineJoin** = "miter";
	23	context.moveTo(lineStart + 400, yStart);
	24	context.lineTo(lineEnd + 400, yStart);
	25	context.lineTo(lineEnd + 400, yStart + 150);
	26	context.stroke();

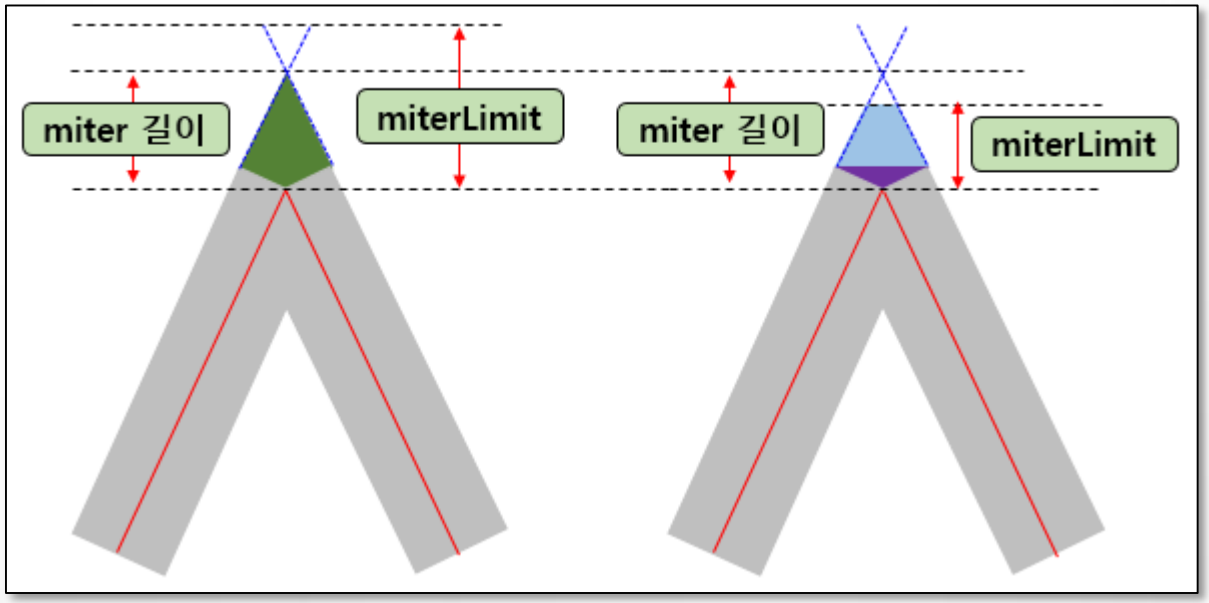


선 스타일 지정하기

선의 연결 부분 스타일 miter의 한계 비율 지정

context.miterLimit [= value]	현재 선의 miter 연결 부분의 한계 비율을 반환한다.

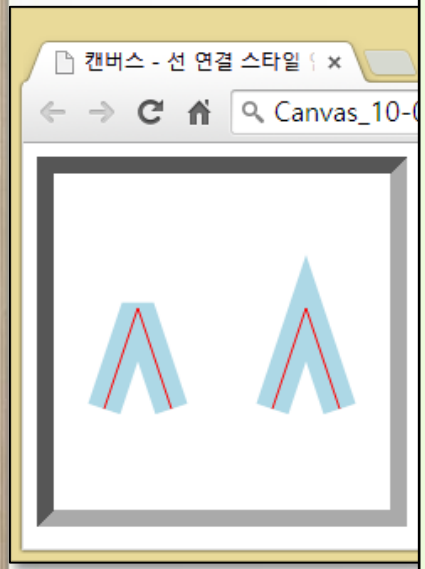
- ✓ lineJoin 속성 값이 miter인 경우, 선 두께의 1/2로 나눈 miter 높이의 비율로 지정할 수 있다. 비율은 (miter 길이 / 선 두께의 ½크기)와 같이 계산된다.
- ✓ CanvasDrawingStyles 객체가 생성될 때, miterLimit 속성 값은 10.0으로 초기화 된다. 따라서 miterLimit 속성 값이 선 두께의 반(1/2) 길이보다 10배 더 길다는 의미이다.



선 스타일 지정하기

선의 연결 부분 스타일 miter의 한계 비율 지정

context.miterLimit [= value]	현재 선의 miter 연결 부분의 한계 비율을 반환한다.



예 제 Canvas_10-03_miterLimit.html

Script

```
1  // 첫 번째 연결선을 그린다.
2  context.lineWidth = 20;
3  context.miterLimit = 3.0;
4  context.beginPath();
5  context.strokeStyle = "lightblue";
6  context.moveTo(30, 140);
7  context.lineTo(50, 80);
8  context.lineTo(70, 140);
9  context.stroke();
10
11 // 보조선을 선 중간에 그린다
12 context.beginPath();
13 context.lineWidth = 1;
14 context.strokeStyle = "red";
15 context.moveTo(30, 140);
16 context.lineTo(50, 80);
17 context.lineTo(70, 140);
18 context.stroke();
```

```
18 //두 번째 연결선을 그린다.
19 context.lineWidth = 20;
20 context.miterLimit = 10.0;
21 context.beginPath();
22 context.strokeStyle = "lightblue";
23 context.moveTo(130, 140);
24 context.lineTo(150, 80);
25 context.lineTo(170, 140);
26 context.stroke();
27
28 // 보조선을 선 중간에 그린다
29 context.beginPath();
30 context.lineWidth = 1;
31 context.strokeStyle = "red";
32 context.moveTo(130, 140);
33 context.lineTo(150, 80);
34 context.lineTo(170, 140);
   context.stroke();
```

다음 학습

1. 원 그리기
2. 베지에 곡선
3. 스타일 지정하기