

HTML 표준화 문서를 기반으로 하는 지침서



속이 깊은

HTML5 & CSS3

김명진 지음

12강

캔버스 Part-3

- 드로잉 응용

학습 목표

캔버스는 좌표 공간을 이동하거나 회전 및 확대/축소할 수 있는 기능들과 텍스트를 다양하게 그리거나 배치하기 위한 기능들을 지원한다. 그리고 이미지를 그리는 다양한 방법 이외에도 캔버스의 이미지 데이터에 접근하고 처리할 수 있는 다양한 기능들을 제공한다. 또한, 캔버스의 드로잉 영역을 제한하도록 하는 클리핑 기능과 애니메이션을 지원해주는 기능들을 제공한다. 마지막으로 캔버스의 비트맵 영역에 대한 히트 영역을 지원하여 다양한 이벤트 처리를 할 수 있도록 하고 있다. 이번 장에서는 이러한 고급 기능들에 대하여 학습하도록 한다.

Section

1 도형 합성 및 변환

2 텍스트 그리기

3 이미지 처리하기

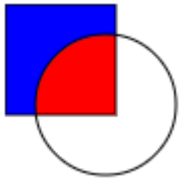
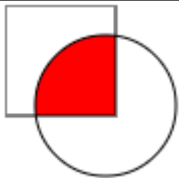
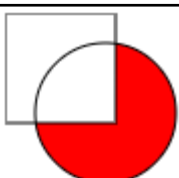
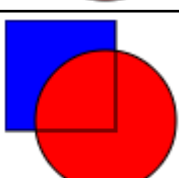
4 애니메이션(Animation)

5 히트 영역(Hit Regions)



도형 합성하기

- **globalCompositeOperation** – 다양한 기본 합성 동작을 지정
 - 도형을 그린 순서와 상관없이 겹쳐지는 부분에 대한 처리
 - 드로잉 원본 이미지(source image) A는 앞으로 드로잉을 하고자 하는 도형이나 이미지를 나타내고, 드로잉 대상 이미지(destination image) B는 현재의 비트맵 상태를 나타내는 것

합성 모드	합성 예	설명
source-atop		A atop B, 처음 드로잉 된 대상 이미지 B 위에 드로잉 원본 이미지 A가 표시되도록 하는 합성 모드
source-in		A in B, 처음 드로잉 된 대상 이미지 B 위에 드로잉 원본 이미지 A의 겹쳐진 부분만 표시되도록 하는 합성 모드
source-out		A out B, 처음 드로잉 된 대상 이미지 B와 겹쳐지지 않은 드로잉 원본 이미지 A 부분만 표시되도록 하는 합성 모드
source-over		A over B, 처음 드로잉 된 대상 이미지 B 위에 드로잉 원본 이미지 A가 표시되도록 하는 합성 모드. 이 모드는 합성 모드의 기본값이 된다.



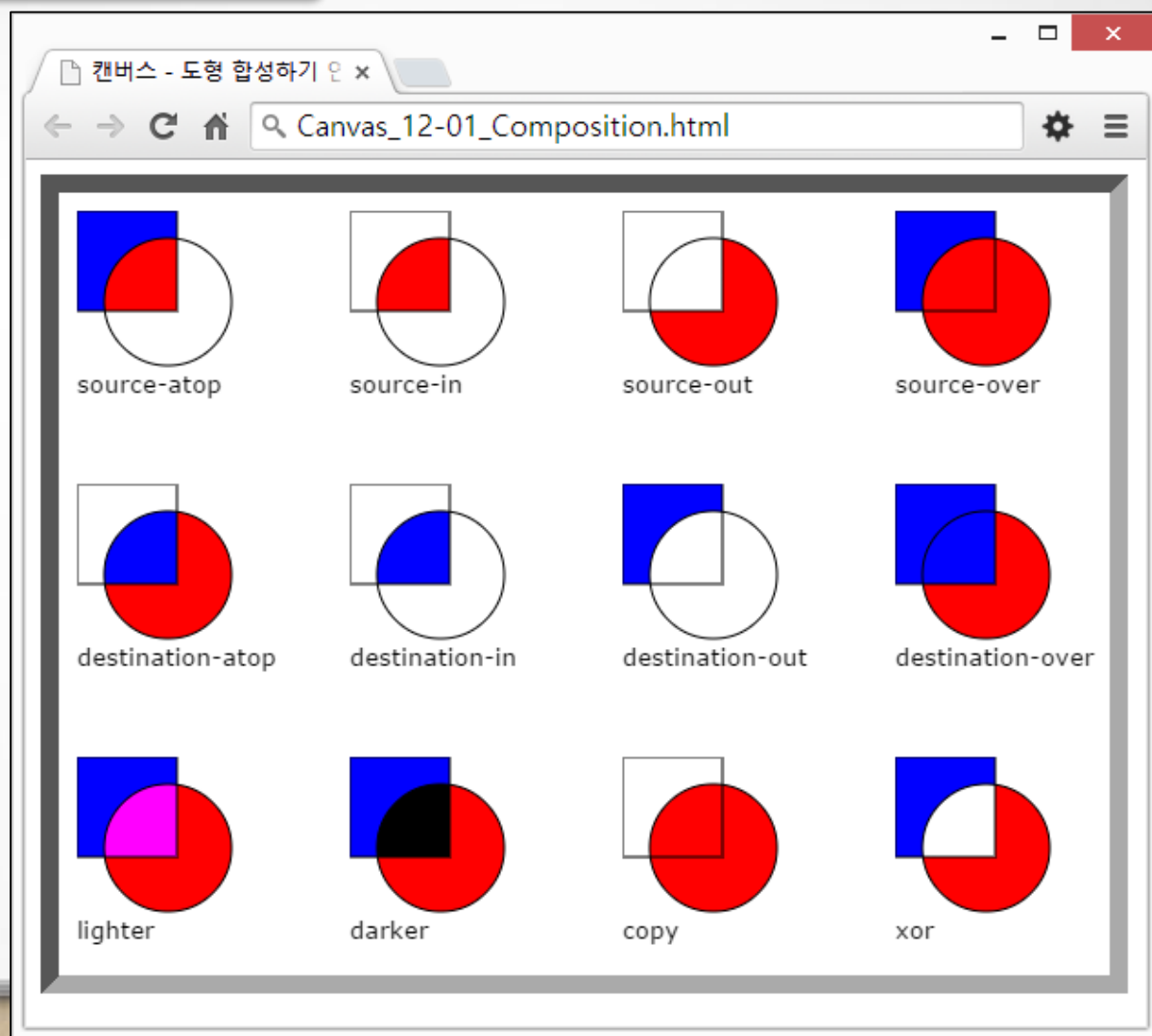
합성 모드	합성 예	설명
destination-atop		B atop A, 드로잉 원본 이미지 A 위에 처음 드로잉 된 대상 이미지가 B가 표시 되도록 하는 합성 모드이다. 그러나 드로잉 원본 이미지 A 영역만큼만 표시되고 그 이외의 부분은 투명하게 된다.
destination-in		B in A, 드로잉 원본 이미지 A 위에 처음 드로잉 된 대상 이미지 B의 겹쳐진 부분만 표시되도록 하는 합성 모드이다. 그 이외의 부분은 투명하게 된다.
destination-out		B out A, 드로잉 원본 이미지 A와 겹쳐지지 않은 처음 드로잉 된 대상 이미지 B 부분만 표시되도록 하는 합성 모드이다. 그 이외의 부분은 투명하게 된다.
destination-over		B over A, 드로잉 원본 이미지 A 위에 처음 드로잉 된 대상 이미지 B가 표시되도록 하는 합성 모드이다. 처음 그려진 도형이 나중에 그려진 도형 위에 표시됨
lighter		A plus B, 처음 드로잉 된 대상 이미지 B와 드로잉 원본 이미지 A의 겹쳐지는 부분의 색상이 두 이미지의 합으로 계산된 값으로 표시되도록 하는 합성 모드이다.
copy		A (B는 무시됨) 처음 드로잉 된 대상 이미지 B 대신에 드로잉 원본 이미지 A만 표시되도록 하는 합성 모드이다.
xor		A xor B(배타적 논리합), 처음 드로잉 된 대상 이미지 B와 드로잉 원본 이미지 A가 겹쳐지는 부분은 투명하게 표시되도록 하는 합성 모드이다.
darker		A minus B, 처음 드로잉 된 대상 이미지 B와 드로잉 원본 이미지 A가 겹쳐지는 부분의 색상이 두 이미지의 차이 값을 구하여 표시되도록 하는 합성 모드이다.
vendorName - operationName		이 합성 모드는 독자적으로 합성 모드를 확장할 때 사용된다.



도형 합성하기



예제 살펴보기





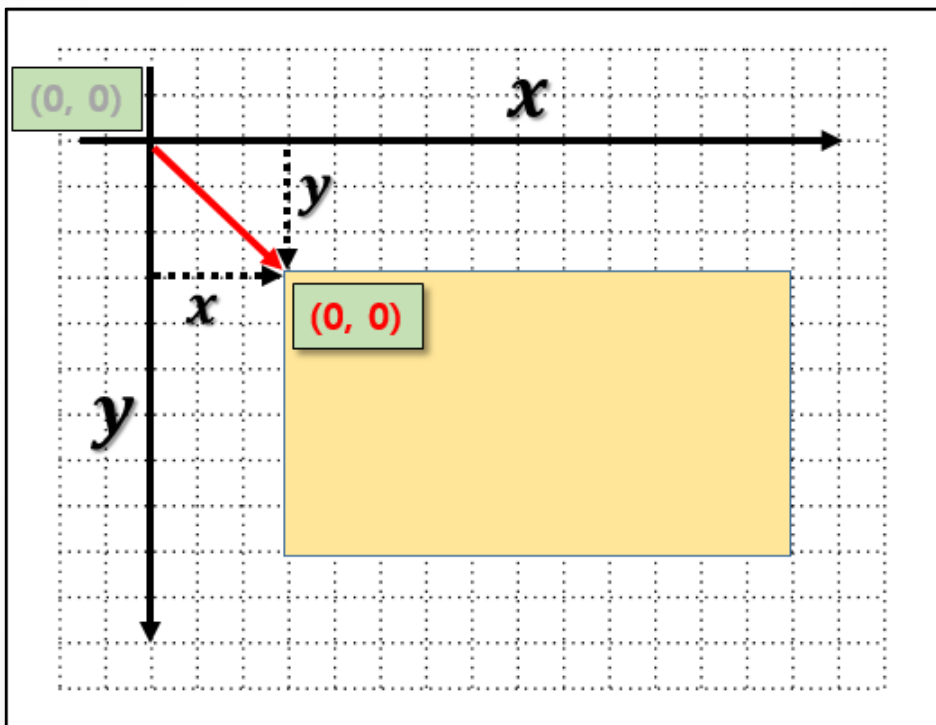
도형 변환하기



좌표 공간의 이동 - `translate()` 메서드

`context.translate(x, y)` 주어진 좌표(x, y)를 사용해서 이동 변환을 적용하여 변환 행렬을 변경한다.

- ✓ 변환 인수 x, y 는 공간 좌표 단위를 나타내는 것으로, x 는 수평 방향의 변환 거리를 나타내고, y 는 수직 방향의 변환 거리를 나타낸다. 아래 그림과 같이 좌표(x, y) 공간을 이동시킨다.



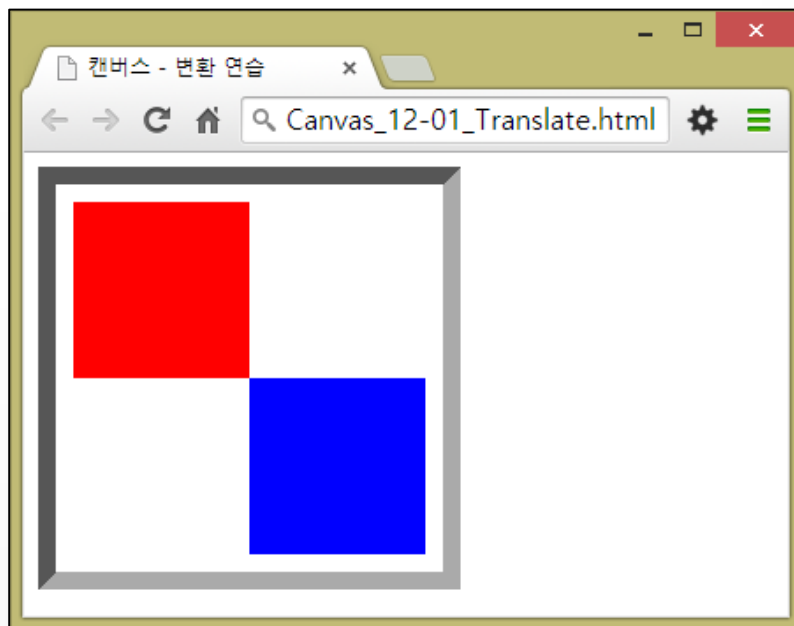


도형 변환하기



좌표 공간의 이동 - translate() 메서드

예제	Canvas_12-01_Translate.html
Script	<pre>1 context.fillStyle = "red"; 2 context.fillRect(10, 10, 100, 100); //빨간 색상으로 사각형을 그린다. 3 context.translate(100, 100); //좌표 공간을 오른쪽으로 100픽셀, 아래쪽으로 100픽셀 이동 4 context.fillStyle = "blue"; 5 context.fillRect (10, 10, 100, 100); //동일한 좌표공간에서 파란 색상으로 사각형을 그린다.</pre>





도형 변환하기

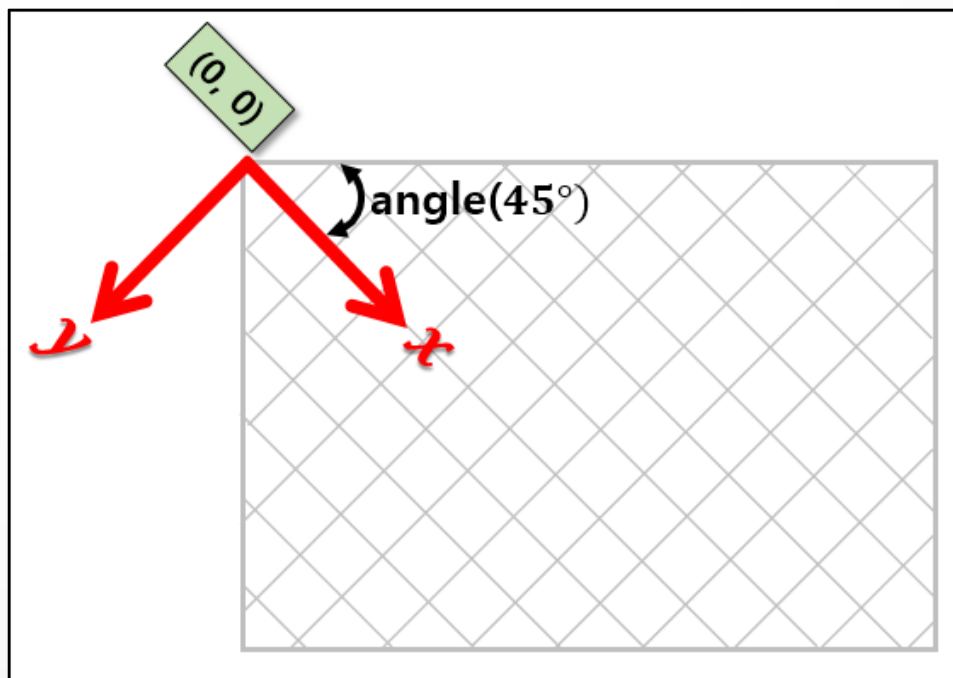


좌표 공간의 회전 - rotate() 메서드

context.rotate(angle)

주어진 각도를 사용해서 회전 변환을 적용하여 변환 행렬을 변경한다.

- ✓ 회전 인수 각도(angle)는 라디안(radians)으로 다음 그림과 같이 시계 방향의 회전 각도를 나타낸다.



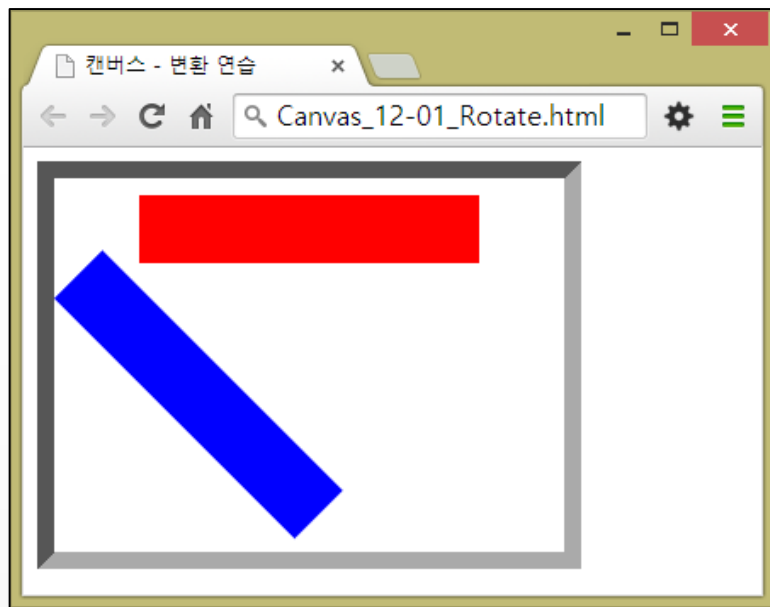


도형 변환하기



좌표 공간의 회전 - rotate() 메서드

예제	Canvas_12-01_Rotate.html
Script	<pre>1 context.fillStyle = "red"; 2 context.fillRect(50, 10, 200, 40); //빨간 색상으로 사각형을 그린다. 3 4 context.rotate(45 * Math.PI/180); //좌표 공간을 시계방향으로 45도 회전한다. 5 6 context.fillStyle = "blue"; 7 context.fillRect(50, 10, 200, 40); //동일한 좌표공간에서 파란 색상으로 사각형을 그린다.</pre>





도형 변환하기

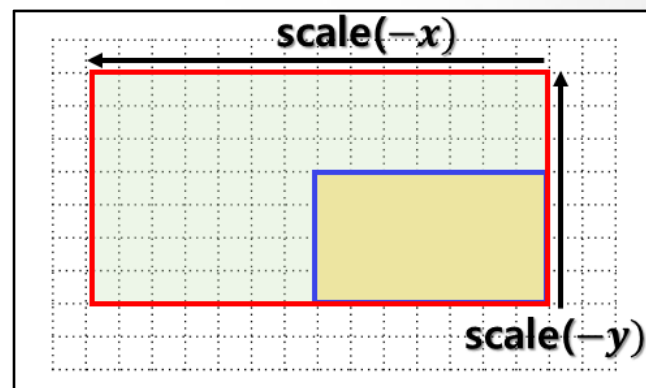
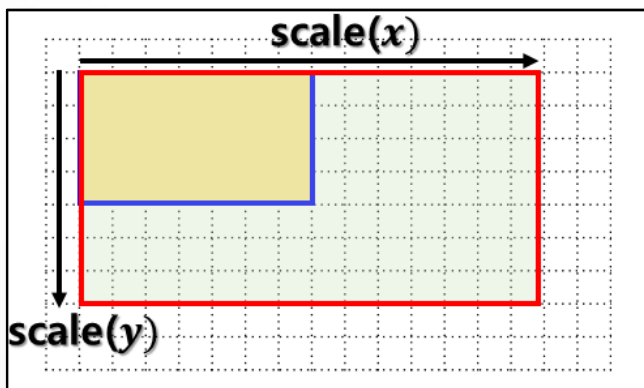


좌표 공간의 확대/축소 - `scale()` 메서드

`context.scale(x, y)`

주어진 변환 요인(x, y)을 사용해서 크기 변환을 적용하여 변환 행렬을 변경한다.

- ✓ 인수 x는 수평 방향의 크기 변환 요인(factor)를 나타내고, 인수 y는 수직 방향의 크기 변환 인수를 나타낸다.
- ✓ 크기 변환 요인은 얼마만큼의 크기로 변경할지를 결정하는 크기 배수에 해당된다.



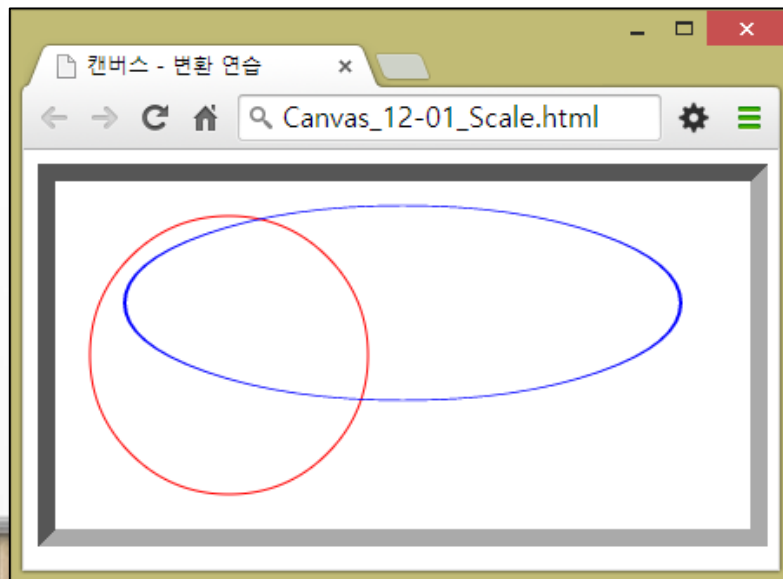


도형 변환하기



좌표 공간의 확대/축소 – scale() 메서드

예제	Canvas_12-01_Scale.html
Script	<pre>1 context.beginPath(); 2 context.strokeStyle = "red"; 3 context.arc(100, 100, 80, 0, 2 * Math.PI, false); //첫 번째 원을 그린다. 4 context.stroke(); 5 6 context.scale(2, 0.7); //좌표 공간을 수평으로 2배, 수직으로 0.7배로 축소한다. 7 8 context.beginPath(); 9 context.strokeStyle = "blue"; 10 context.arc(100, 100, 80, 0, 2 * Math.PI, false); //두 번째 원을 그린다. 11 context.stroke();</pre>





도형 변환하기



실습하기 - CSS_Study12_Rotate.html

Script	<pre> 1 //별을 그리는 메서드(컨텍스트, 별의 중심 좌표(x, y), 별의 반지름, 별을 그리는 방향) 2 function drawStar(context, center_x, center_y, radius, anticlockwise) { 3 var vertAngle = (2 * Math.PI) / 10; 4 var pox_x = new Array(11); //x 좌표를 위한 배열 5 var pos_y = new Array(11); //y 좌표를 위한 배열 6 7 context.save(); 8 for (var i = 11; i != 0; i--) { //10개의 꼭지점 좌표를 계산한다. 9 var r = radius * (i % 2 + 1) / 2; //현재 꼭지점의 반지름을 계산한다. 10 var curAngle = vertAngle * i; //현재 꼭지점의 각도를 계산한다. 11 pox_x[i] = (r * Math.sin(curAngle)) + center_x; //별 꼭지점의 X좌표 저장 12 pos_y[i] = (r * Math.cos(curAngle)) + center_y; //별 꼭지점의 Y좌표 저장 13 } 14 //시계 반대방향이면 좌표 순서를 뒤에서 그리도록 한다. 15 if(anticlockwise) { pox_x.reverse(); pos_y.reverse(); } 16 context.moveTo(pox_x[1], pos_y[1]); //시작점으로 이동한다. 17 //반목루프를 이용하여 별을 그리도록 한다. 18 for (var i = 10; i != 0; i--) { context.lineTo(pox_x[i], pos_y[i]); } 19 context.closePath(); 20 context.restore(); 21 }</pre>
--------	---



도형 변환하기



실습하기 - CSS_Study12_Rotate.html

Script

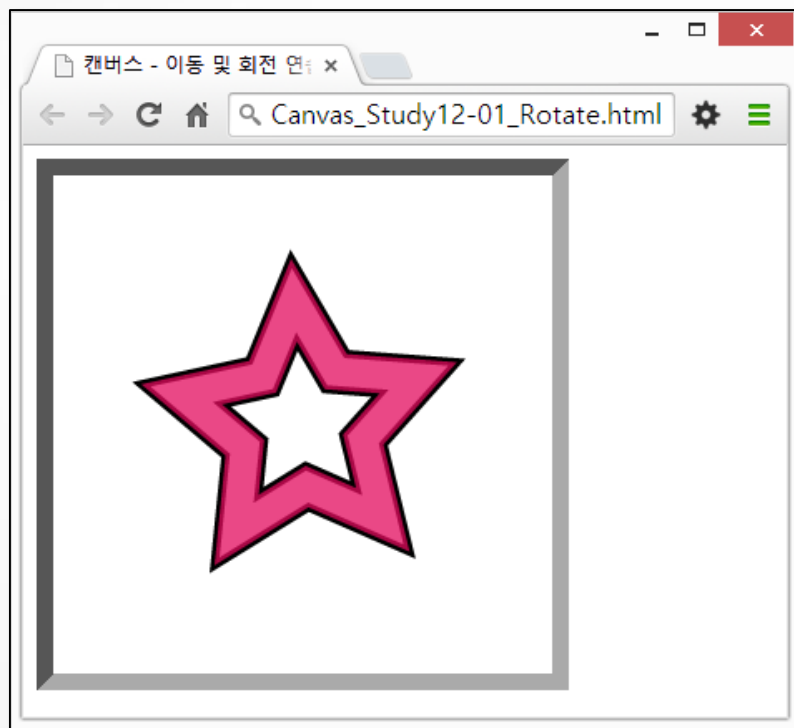
```
23 function Rotate() {
24     var cW = canvas.width/2; //캔버스 중앙으로 위치시키기 위한 수평 좌표
25     var cH = canvas.height/2; //캔버스 중앙으로 위치시키기 위한 수직 좌표
26
27     animate();
28
29     function animate() {
30         context.clearRect(0, 0, canvas.width, canvas.height); //캔버스를 지운다.
31         context.translate(cW, cH); //캔버스의 좌표 공간을 이동시킨다.
32         context.rotate((1 * Math.PI) / 180); //좌표 공간을 1도 회전시킨다.
33
34         context.lineWidth = 5;
35         context.fillStyle = "rgba(227,11,93,0.75)";
36
37         context.beginPath();
38         drawStar(context, 0, 0, 100, false); //바깥쪽 별을 그린다.
39         drawStar(context, 0, 0, 50, true); //안쪽 별을 그린다.
40         context.stroke();
41         context.fill();
42
43         context.translate(-cW, -cH); //별을 그린 후에 이동시킨 좌표 공간을 다시 이동
44         window.requestAnimationFrame(animate);
45     }
46 }
```



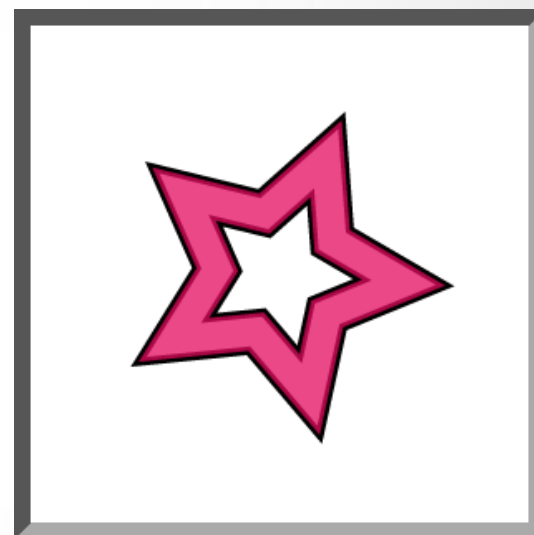
도형 변환하기



실습하기 - CSS_Studyl2_Rotate.html



회전





도형 변환하기



사용자 정의 변환 – transform(), setTransform()

context.transform(a, b, c, d, e, f)

지정된 인수 행렬로 변환 행렬을 변경한다.

- ✓ 다음에 나타낸 행렬과 현재의 변환 행렬을 곱한 결과로 현재 변환 행렬을 대체한다. 인수 a, b, c, d, e 및 f는 때때로 m11, m12, m21, m22, dx 그리고 dy 또는 m11, m21, m12, m22, dx 그리고 dy로 불린다.

a / m11	c / m21	e / dx
b / m12	d / m22	f / dy
0	0	1

context.setTransform(a, b, c, d, e, f)

지정된 인수 행렬로 변환 행렬을 변경한다.

- ✓ 이 메서드는 인수로 주어진 행렬로 현재의 변환 행렬을 대체하고 동일한 인수를 이용하여 transform(a, b, c, d, e, f) 메서드를 호출한다.

두 메서드의 차이

- transform() 메서드가 지정된 인수의 행렬로 현재 변환 행렬에 적용시키기 때문에 연속해서 이 메서드를 호출하면 변환 행렬 값이 누적된다.
- 반면에, setTransform() 메서드는 현재 변환 행렬을 원래 값인 단위 행렬로 재설정한다. 다음 transform() 메서드를 호출한다.
- ✓ 따라서 이 메서드를 호출할 때마다 매번 새로운 변환 행렬이 적용된다.

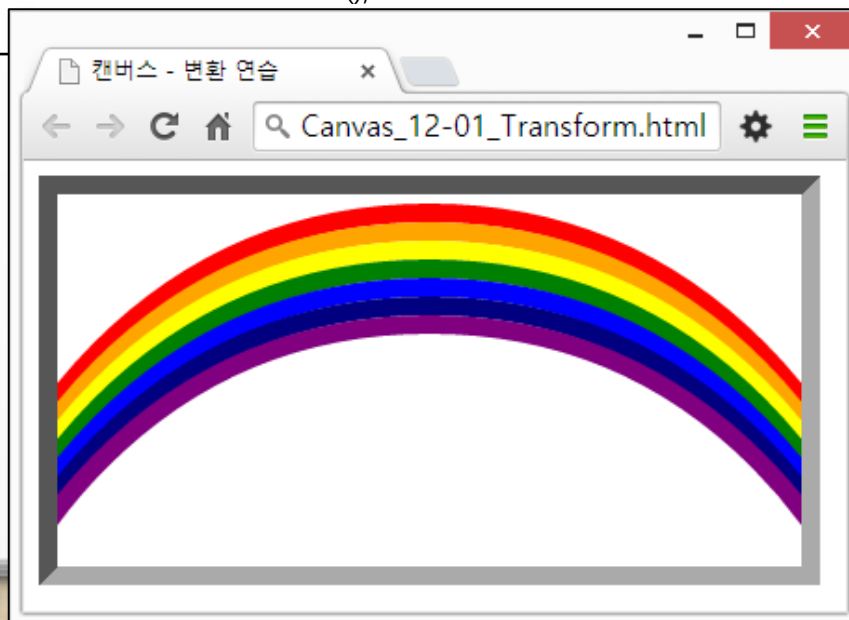


도형 변환하기



사용자 정의 변환 – `transform()`, `setTransform()`

예제		Canvas_12-01_Transform.html
Script	1	var colors = new Array("red", "orange", "yellow", "green", "blue", "navy", "purple");
	2	
	3	context.lineWidth = 10;
	4	for (var i = 0; i < colors.length; i++) { //호를 반복해서 그리도록 한다.
	5	// 아래쪽으로 10픽셀만큼 이동하는 변환 행렬을 정의한다
	6	context. transform (1, 0, 0, 1, 0, 10);
	7	context.strokeStyle = colors[i] ; //색상을 적용시킨다.
	8	context.beginPath();
	9	context.arc(200, 250, 250, 0, Math.PI, true); // 호를 그린다.
	10	context.stroke();
	11	}



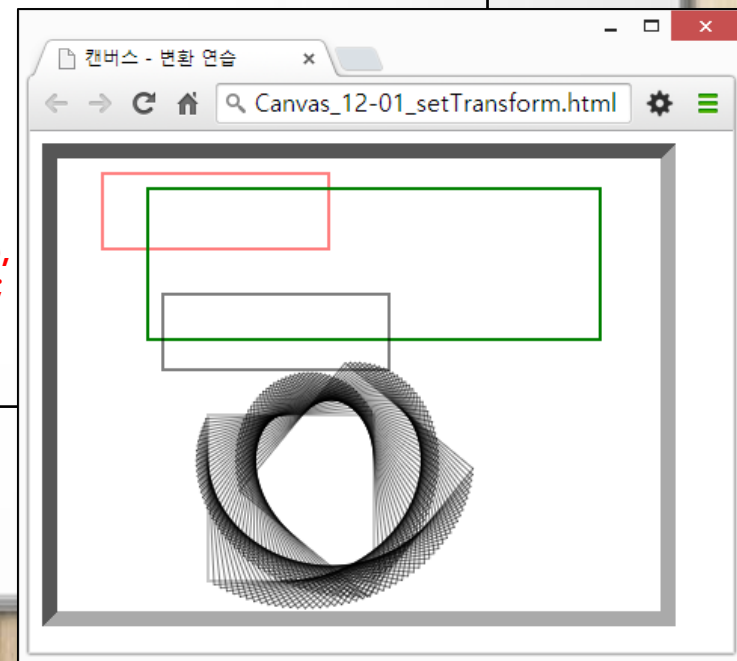


도형 변환하기



사용자 정의 변환 – transform(), setTransform()

예제	Canvas_12-01_setTransform.html
Script	<pre> 1 context.strokeStyle = "red"; 2 context.strokeRect(30, 10, 150, 50); //빨간 색상으로 사각형을 그린다. 3 4 //수평 및 수직으로 2배 확대하는 변환 행렬을 정의한다. 5 context.setTransform(2.0, 0, 0, 2.0, 0, 0); 6 context.strokeStyle = "green"; 7 context.strokeRect(30, 10, 150, 50); //녹색으로 사각형을 그린다. 8 9 //오른쪽으로 40픽셀, 아래쪽으로 80픽셀만큼 이동하는 변환 행렬을 정의한다. 10 context.setTransform(1, 0, 0, 1, 40, 80); 11 context.strokeStyle = "black"; 12 context.strokeRect(30, 10, 150, 50); 13 14 context.lineWidth = 0.5; 15 context.strokeStyle = "black"; 16 for(var i = 0; i < 50; i++) { //사각형을 회전시킨다. 17 context.setTransform(Math.cos(angle), -Math.sin(angle), 18 Math.sin(angle), Math.cos(angle), i + 150, 220); 19 context.strokeRect(-50, -50, 110, 110); 20 angle += 0.05; 21 }</pre>





■ 텍스트 테두리 및 채우기

`context.fillText(text, x, y, [, maxWidth])`

지정된 위치에 텍스트를 채운다.

- ✓ `maxWidth`가 지정된 경우, 텍스트는 `maxWidth` 폭 크기에 맞게 조정된 텍스트를 채운다..

`context.strokeText(text, x, y, [, maxWidth])`

지정된 위치에 텍스트의 테두리를 그린다.

- ✓ `maxWidth`가 지정된 경우, 텍스트는 `maxWidth` 폭 크기에 맞게 조정된 텍스트의 테두리를 그린다.

`metrics = context.measureText(text)`

지정된 텍스트의 폭을 반환한다.

- ✓ 현재 글꼴에서 주어진 텍스트의 폭을 가진 `TextMetrics` 객체를 반환한다.

`metrics.width`

텍스트의 폭을 반환한다.

- ✓ `measureText()` 메서드를 통해 반환된 텍스트의 유효한 폭(CSS 픽셀에서의 인라인 박스의 폭)을 반환한다.

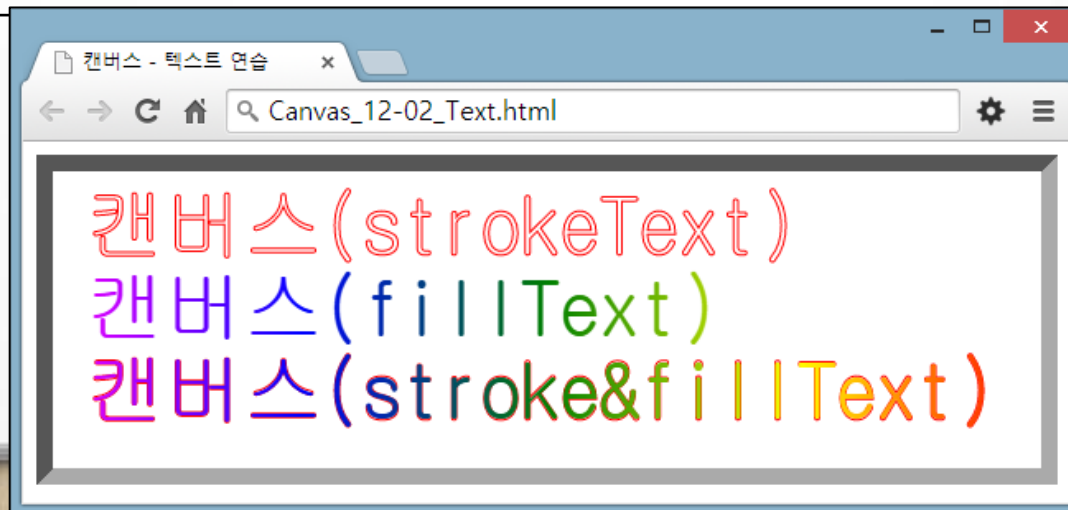
주의사항

현재의 명세서에 따르면, `fillText()` 와 `strokeText()` 메서드를 호출하여 그려지는 텍스트는 글꼴의 크기(em square 크기)와 `measureText()` 메서드에 의해 반환된 텍스트 폭에 의한 박스 크기 밖으로 벗어날 수 있다는 점에 유의하자. 현재 버전의 명세서는 텍스트의 경계 박스 크기를 구하는 방법을 제공하지 않는다. 따라서, 텍스트를 그리거나 제거함으로써, 클리핑 영역을 덮는 캔버스 전체 영역을 대체할 때 주의할 필요가 있다.



■ 텍스트 테두리 및 채우기

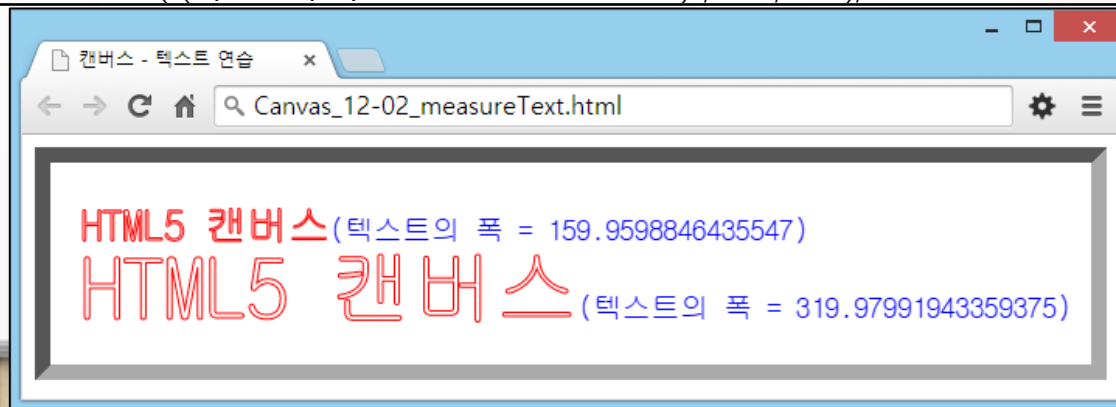
예제	Canvas_12-01_Text.html
Script	<pre>1 gradient = context.createLinearGradient(0, 0, canvas.width, 0); 2 gradient.addColorStop("0", "magenta"); 3 gradient.addColorStop(".25", "blue"); 4 gradient.addColorStop(".50", "green"); 5 gradient.addColorStop(".75", "yellow"); 6 gradient.addColorStop("1.0", "red"); 7 8 context.font = "36pt 굴림체"; 9 context.fillStyle = gradient; 10 context.strokeStyle = "red"; 11 12 context.strokeText("캔버스(strokeText)", 20, 50); //테두리만 있는 텍스트를 그린다. 13 context.fillText("캔버스(fillText)", 20, 100); //내부가 채워진 텍스트를 그린다. 14 15 context.strokeText("캔버스(stroke&fillText)", 20, 150); //텍스트 테두리를 그리고 16 context.fillText("캔버스(stroke&fillText)", 20, 150); //텍스트 내부를 채운다.</pre>





■ 텍스트 테두리 및 채우기

예제	Canvas_12-01_Text.html
Script	<pre> 1 var text = "HTML5 캔버스"; 2 var metrics, textWidth1, textWidth2; 3 4 context.strokeStyle = "red"; 5 context.fillStyle = "blue"; 6 context.font = "20pt 굴림체"; //글꼴의 크기를 20포인트로 지정한다. 7 context.strokeText(text, 20, 50); //첫 번째 텍스트를 출력한다. 8 9 metrics = context.measureText(text); //텍스트의 TextMetrics 객체를 반환한다. 10 textWidth1 = metrics.width; //TextMetrics 객체의 텍스트 폭을 반환한다. 11 12 context.font = "40pt 굴림체"; //글꼴의 크기를 40포인트로 지정한다. 13 context.strokeText(text, 20, 100); //두 번째 텍스트를 출력한다. 14 15 metrics = context.measureText(text); //텍스트의 TextMetrics 객체를 반환한다. 16 textWidth2 = metrics.width; //TextMetrics 객체의 텍스트 폭을 반환한다. 17 18 //첫 번째 및 두 번째 텍스트의 폭을 출력한다. 19 context.font = "14pt 굴림체"; 20 context.fillText("(텍스트의 폭 = " + textWidth1 + ")", 180, 50); 21 context.fillText("(텍스트의 폭 = " + textWidth2 + ")", 340, 100); </pre>





글꼴 설정 및 텍스트 배치하기

context.font [= value]

현재 글꼴의 설정 값을 반환한다.

- ✓ 글꼴을 변경하기 위하여 CSS의 font 속성과 동일한 방법으로 설정하고 CSS 글꼴 값으로 해석할 수 없는 값은 무시된다.
- ✓ 상대적인 키워드와 길이는 캔버스 요소의 글꼴을 기준으로 계산된다.



텍스트의 수평 위치 지정

context.textAlign [= value]

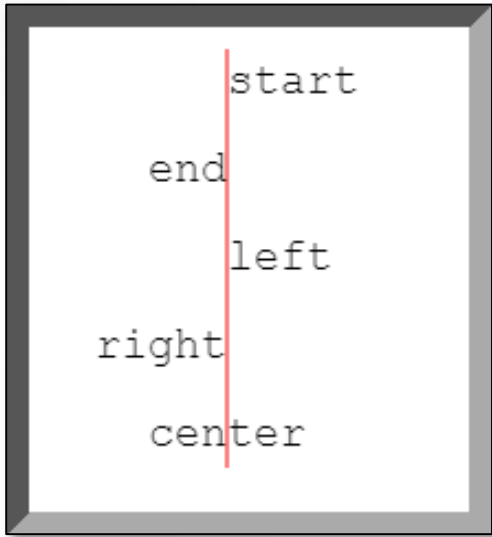
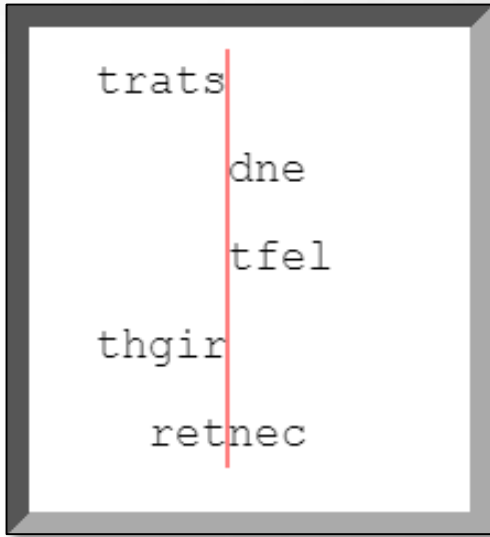
현재 텍스트의 수평에 대한 맞춤(정렬) 설정 값을 반환한다.

- ✓ 텍스트의 맞춤을 변경하기 위하여 설정될 수 있고, 기본 값은 start이다.
- ✓ 지정할 수 있는 값은 start, end, left, right, 그리고 center 이며, 그 이외의 값은 무시된다.

키워드	설명
start	<p>캔버스 요소의 방향성에 따라서 앵커 포인트는 다음과 같이 결정된다.</p> <ul style="list-style-type: none"> ✓ ltr(left-to-right) 왼쪽에서 오른쪽 방향 - 텍스트의 왼쪽 가장자리가 기준이 된다. ✓ rtl(right-to-left) 오른쪽에서 왼쪽 방향 - 텍스트의 오른쪽 가장자리가 기준이 된다.
end	<p>캔버스 요소의 방향성에 따라서 앵커 포인트는 다음과 같이 결정된다.</p> <ul style="list-style-type: none"> ✓ ltr(left-to-right) 왼쪽에서 오른쪽 방향 - 텍스트의 오른쪽 가장자리가 기준이 된다. ✓ rtl(right-to-left) 오른쪽에서 왼쪽 방향 - 텍스트의 왼쪽 가장자리가 기준이 된다.
left	앵커 포인트는 텍스트의 왼쪽 가장자리가 기준이 된다.
right	앵커 포인트는 텍스트의 오른쪽 가장자리가 기준이 된다.
center	앵커 포인트는 텍스트의 중심에 있다.



글꼴 설정 및 텍스트 배치하기

Canvas_12-02_textAlign1.html	Canvas_12-02_textAlign2.html
	
canvas 요소의 방향 = 왼쪽에서 오른쪽(ltr)	canvas 요소의 방향 = 오른쪽에서 왼쪽(rtl)



글꼴 설정 및 텍스트 배치하기



텍스트의 수직 위치 지정

context.textBaseline [= value] 현재 텍스트의 수직에 대한 기준(baseline) 정렬 설정 값을 반환한다.

- ✓ 수직 위치의 기준 정렬을 변경하기 위하여 설정될 수 있고, 기본 값은 alphabetic이다.
- ✓ 지정할 수 있는 값은 top, hanging, middle, alphabetic, ideographic, 그리고 bottom이며, 그 이외의 값은 무시된다.

키워드	설명
top	em 박스의 세로 상단을 기준으로 한다.
hanging	hanging 기준선을 기준으로 한다(주로 인도의 다양한 언어에서 사용된다).
middle	em 박스의 세로 가운데를 기준으로 한다.
alphabetic	영어 알파벳 글자의 기준선을 기준으로 한다.
ideographic	표의 문자의 기준선을 기준으로 한다(주로 일본어와 중국어 등과 같은 언어에서 사용된다).
bottom	em 박스의 세로 바닥을 기준으로 한다.

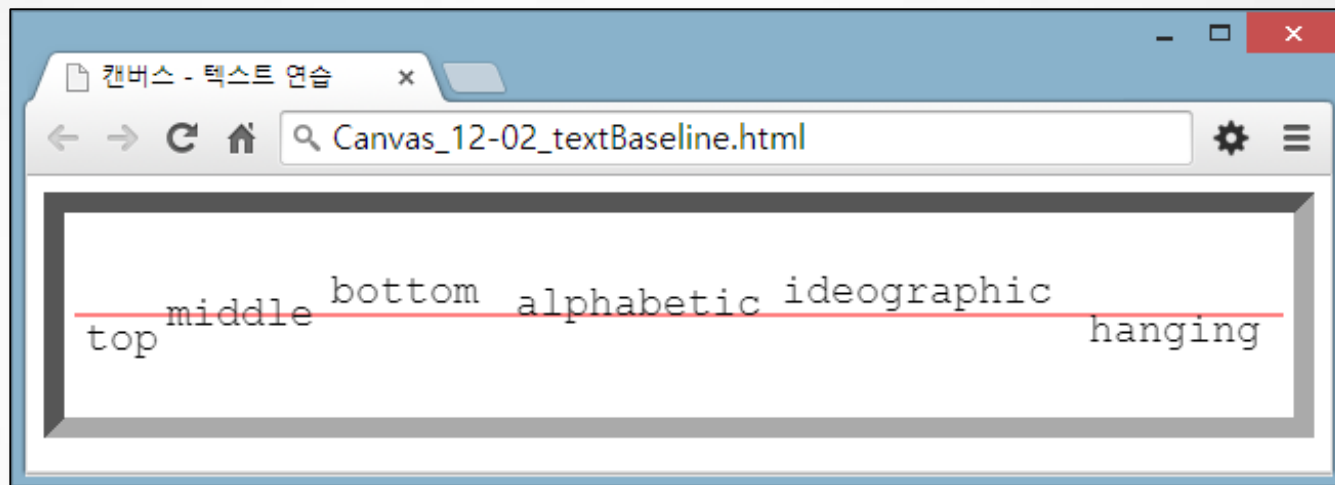




글꼴 설정 및 텍스트 배치하기



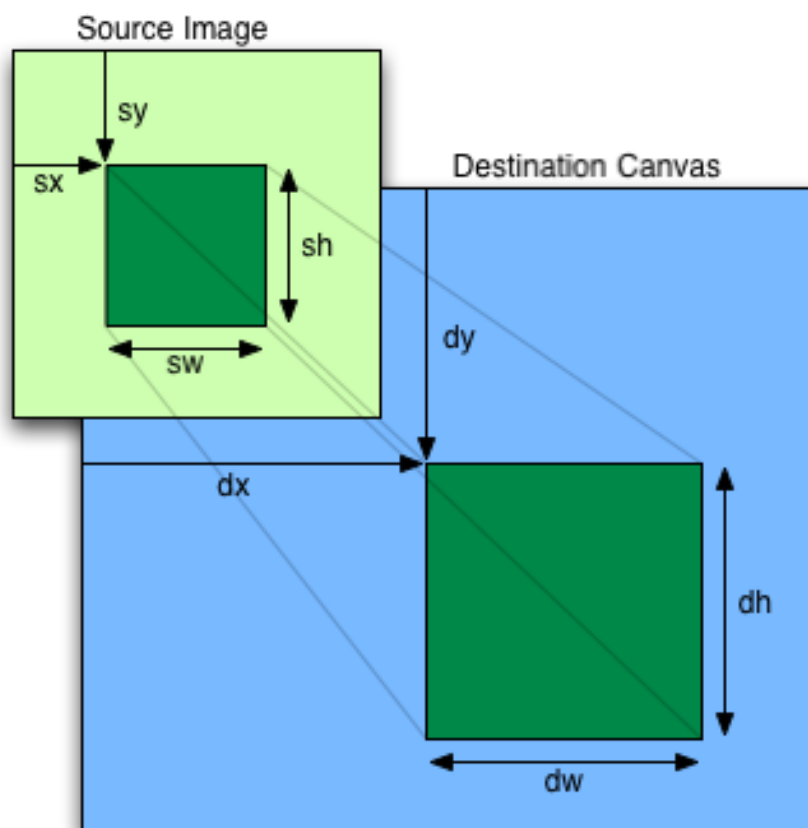
텍스트의 수직 위치 지정





이미지 그리기

<code>context.drawImage(image, dx, dy)</code>	지정한 이미지를 원래의 크기로 그린다.
<code>context.drawImage(image, dx, dy, dw, dh)</code>	지정한 이미지를 지정한 크기로 그린다.
<code>context.drawImage(image, sx, sy, sw, sh, dx, dy, dw, dh)</code>	이미지의 일부를 지정한 크기로 그린다.





이미지 그리기



원본 이미지를 목적 캔버스에 그리기

context.drawImage(image, dx, dy) | 지정한 이미지를 원래의 크기로 그린다.

- ① var **image** = new Image();
- ② **image.src** = 이미지 주소;
- ③ context.**drawImage**(**image**, 목적 캔버스 위치(dx, dy));

예제

Canvas_12-03_drawImage1.html

Script

```

1 function drawImage() {
2     var canvas = document.getElementById('myCanvas');
3     var context = canvas.getContext('2d');
4
5     var image = new Image();
6     image.src = 'images/WorldCup2014.png'; //570 x 220
7
8     image.onload = function(e) { context.drawImage( image, 0, 0 ); }
9 }
```

HTML

```

1 <canvas id="myCanvas" width="700" height="300" style="border: 10px inset #aaa">
```





■ 이미지 그리기



원본 이미지를 지정한 크기로 목적 캔버스에 그리기

context.drawImage(image, dx, dy, dw, dh) | 지정한 이미지를 지정한 크기로 그린다.

③ context.**drawImage**(*image*, 목적 캔버스 위치(dx, dy), 드로잉 이미지 폭과 높이(dw, dh));

예제	Canvas_12-03_drawImage2.html
Script	<pre> 1 function drawImage() { 2 var canvas = document.getElementById('myCanvas'); 3 var context = canvas.getContext('2d'); 4 5 var <i>image</i> = new Image(); 6 <i>image</i>.src = 'images/WorldCup2014.png'; //570 x 220 7 8 <i>image</i>.onload = function(e) { 9 context.drawImage(<i>image</i>, 0, 0, <i>canvas.width</i>, <i>canvas.height</i>); } 10 }</pre>
HTML	<pre> 1 <canvas id="myCanvas" width="700" height="300" style="border: 10px inset #aaa"></pre>





이미지 그리기



원본 이미지의 일부를 목적 캔버스에 지정한 크기로 그리기

context.drawImage(image, sx, sy, sw, sh, dx, dy, dw, dh) | 이미지의 일부를 지정한 크기로 그린다.

③

context.drawImage(*image*, 원본 이미지 위치(*sx, sy*), 원본 이미지 크기(*sw, sh*),
목적 캔버스 위치(*dx, dy*), 드로잉 이미지 폭과 높이(*dw, dh*));

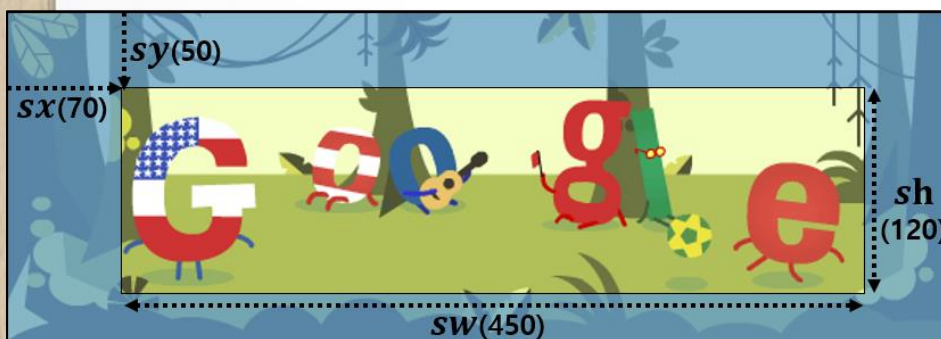
예제

Canvas_12-03_drawImage3.html

Script

8
9

```
image.onload = function(e) {  
    context.drawImage( image, 70, 50, 450, 120, 0, 0, canvas.width, canvas.height); }
```





이미지 그리기



실습하기 - CSS_Study12_O3_drawImage.html

Script	<pre> 1 var image = new Image(); 2 image.src = 'images/WorldCup2014.png'; //570 x 220 3 4 image.onload = function(e) { 5 context.drawImage(image, 0, 0, canvas.width, canvas.height); //원본 이미지 드로잉 6 } 7 8 canvas.addEventListener("click", function () { 9 var canvas2 = document.getElementById("yourCanvas"); 10 var context2 = canvas2.getContext("2d"); 11 12 context2.drawImage(image, 50, 50, 300, 120); //이미지를 축소하여 드로잉 한다. 13 context2.drawImage(image, 0, 0, 100, 100, 0, 0, 50, 50); //왼쪽 상단 이미지 14 context2.drawImage(image, 470, 0, 100, 100, 350, 0, 50, 50); //오른쪽 상단 이미지 15 context2.drawImage(image, 0, 120, 100, 100, 0, 170, 50, 50); //왼쪽 하단 이미지 16 context2.drawImage(image, 470, 120, 100, 100, 350, 170, 50, 50); //오른쪽 하단 이미지 17 }, false); </pre>
HTML	<pre> 1 <canvas id="myCanvas" width="400" height="200" style="border: 10px inset #aaa">캔버스 연습</canvas> 2 <canvas id="yourCanvas" width="400" height="200" style="border: 10px outset #aaa">캔버스 연습</canvas> 3 <p>왼쪽 캔버스를 마우스로 클릭하면 오른쪽 캔버스에 이미지들이 그려집니다.</p> </pre>



이미지 그리기



실습하기 – CSS_Studyl2_O3_drawImage.html





■ 이미지 조작하기

<code>imagedata = context.createImageData(sw, sh)</code>	ImageData 객체를 반환한다.
<code>imagedata = context.createImageData(imagedata)</code>	
<ul style="list-style-type: none">✓ 2개의 인수 sw와 sh를 지정하면 지정한 크기의 사각형(CSS 픽셀 단위)을 가지는 ImageData 객체를 반환한다.✓ 지정된 imagedata크기와 동일한 이미지 데이터(ImageData) 객체를 반환한다.✓ 반환된 객체의 모든 픽셀은 투명한 검은색으로 초기화된다.	
<code>imagedata = context.getImageData(sx, sy, sw, sh)</code>	지정된 영역에 대한 ImageData 객체를 반환한다.
<ul style="list-style-type: none">✓ 캔버스 영역에 지정된 사각형 영역(sw, sy, sw, sh)에 대한 이미지를 포함한 ImageData 객체를 반환한다.✓ 사각형 영역의 각의 4개의 모서리는 캔버스의 좌표 공간 단위에서 (sx, sy), (sx+sw, sy), (sx+sw, sy+sh), (sx, sy+sh)가 된다.	
<code>imagedata = context.putImageData(imagedata, dx, dy [, dirtyX, dirtyY, dirtyWidth, dirtyHeight])</code>	
<ul style="list-style-type: none">✓ 캔버스에 지정된 ImageData 객체(imagedata)의 데이터를 그린다.✓ 만일, 옵션 dirty로 주어지는 사각형(dirtyX, dirtyY, dirtyWidth, dirtyHeight) 영역이 지정되면, 지정된 사각형 영역의 픽셀만 그려지게 된다.	
<code>imagedata.width</code>	ImageData 객체 데이터의 실제 폭(width)과 높이(height)를 반환한다.
<code>imagedata.height</code>	읽기 전용의 부호가 없는 long형의 실수 값이며, 단위는 장치의 픽셀이다.
<code>imagedata.data</code>	각 디바이스의 픽셀의 색상 구성 요소인 RGBA 순서의 데이터를 포함하는 1차원 배열(CanvasPixelArray 객체)을 반환한다. 따라서 각 픽셀은 빨강, 녹색, 파랑 및 투명 요소의 순서로 주어진다. 각각의 값은 8비트 정수로 이루어져 있기 때문에 0~255의 범위 이내의 값을 갖는다.



이미지 조작하기



이미지 데이터(ImageData) 처리

```
imageCopy = context.createImageData(canvas.width, canvas.height)
```

imageCopy.data

data[0]	data[1]	data[2]	data[3]	...	data[n-4]	data[n-3]	data[n-2]	data[n-1]
빨간색	녹색	파란색	투명도	...	빨간색	녹색	파란색	투명도

```
imageData = context.getImageData(0, 0, canvas.width, canvas.height)
```

➤ 이미지 배열 데이터 복사

```
for(var i=0; i < imageData.data.length; i++) { //캔버스에서 가져온 이미지 데이터 크기만큼 반복한다.
    imageCopy.data[i] = imageData.data[i]; //imageData의 데이터를 imageCopy에 복사한다
}
```

➤ 이미지 투명도 높이기

```
for(var i=3; i < imageData.data.length - 4; i+=4) { //이미지 데이터 크기만큼 반복.
    imageData.data[i] = imageData.data[i] / 2; //imageData.data에 저장된 픽셀 데이터의 투명도를
    높인다.
}
```

➤ 이미지 네거티브 필터

```
for(var i=0; i < imageData.data.length - 4; i+=4) { //이미지 데이터 크기만큼 반복.
    imageData.data[i] = 255 - imageData.data[i]; //빨간 색상을 반전한다.
    imageData.data[i+1] = 255 - imageData.data[i+1]; //녹색 색상을 반전한다.
    imageData.data[i+2] = 255 - imageData.data[i+2]; //파란 색상을 반전한다.
}
```

➤ 이미지 흑백 필터

```
for(var i=0; i < imageData.data.length - 4; i+=4) { //이미지 데이터 크기만큼 반복.
    average = ( imageData.data[i] + imageData.data[i+1] + imageData.data[i+2] ) / 3; //픽셀 값 평균
    imageData.data[i] = imageData.data[i+1] = imageData.data[i+2] = average; //평균 값 저장
}
```



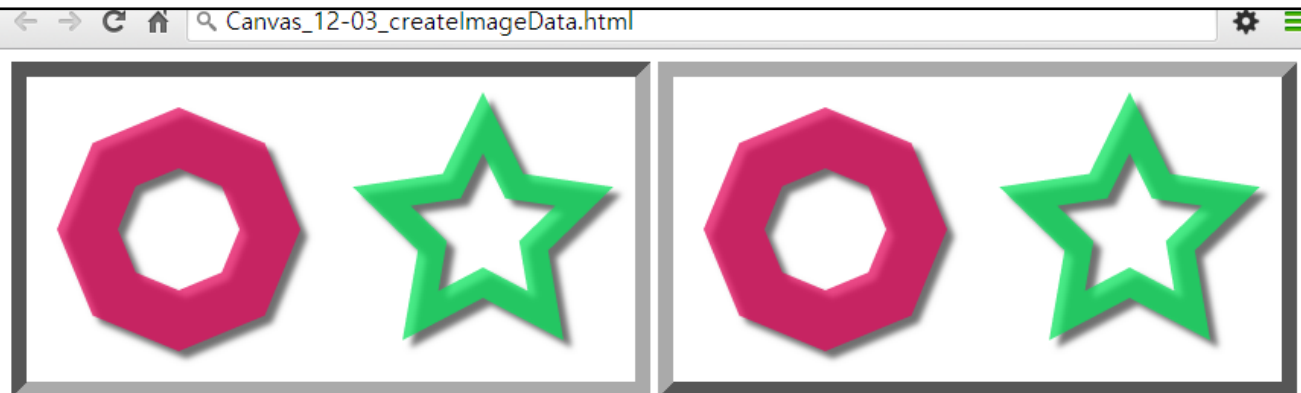

■ 이미지 조작하기



이미지 데이터(ImageData) 처리

```
context.putImageData(imageCopy, 0, 0)
```

예제	Canvas_12-03_createImageData.html
Script	<pre> 25 ... 26 //캔버스에 그려진 이미지 데이터를 가져온다. 27 var imageData = context.getImageData(0, 0, canvas.width, canvas.height); 28 29 //캔버스에 이미지 데이터 크기만큼의 빈 이미지 데이터 객체를 생성한다. 30 var imageCopy = context.createImageData(imageData); 31 //가져온 이미지(imageBuffer)를 생성된 빈 이미지 데이터(imageCopy)에 복사한다. 32 for(var i=0; i < imageData.data.length; i++) { 33 imageCopy.data[i] = imageData.data[i]; 34 } 35 context2.putImageData(imageCopy, 0, 0); //다른 캔버스에 복사한 이미지를 그린다. 36 ... </pre>



왼쪽 캔버스를 마우스로 클릭하면 오른쪽 캔버스에 이미지를 복사합니다.

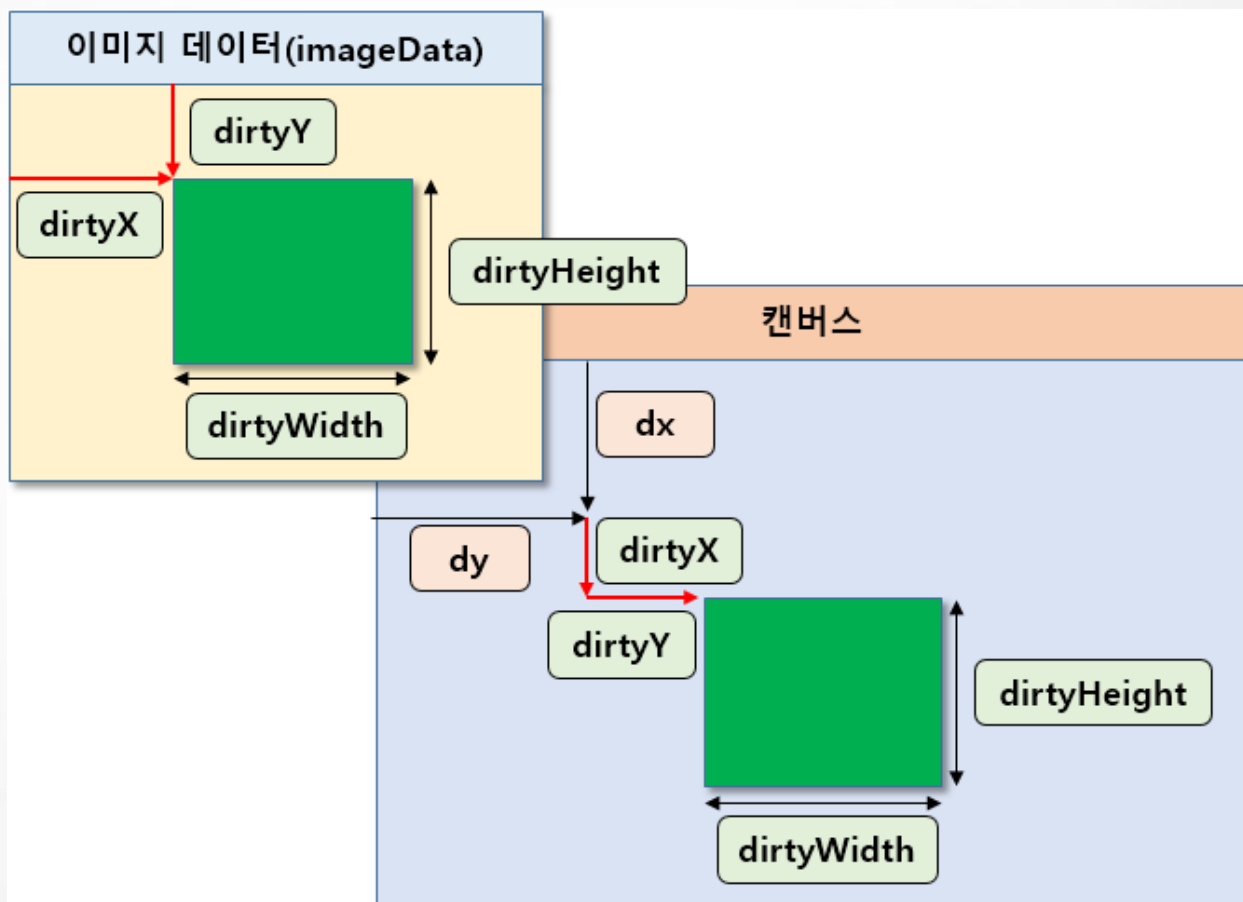


이미지 조작하기



이미지 데이터(ImageData) 처리

```
context.putImageData(imagedata, dx, dy [, dirtyX, dirtyY, dirtyWidth, dirtyHeight])
```





■ 이미지 조작하기



이미지 데이터(ImageData) 처리

context.putImageData(imagedata, dx, dy [, dirtyX, dirtyY, dirtyWidth, dirtyHeight])

예제	Canvas_12-03_putImageData.html
Script	<pre> 1 ... 앞의 예제 참고 2 canvas.addEventListener("click", function () { //캔버스의 이벤트리스너에서 마우스 클릭시 3 var canvas2 = document.getElementById("yourCanvas"); 4 var context2 = canvas2.getContext("2d"); 5 6 //캔버스에 그려진 이미지 데이터를 가져온다. 7 var imageData = context.getImageData(0, 0, canvas.width, canvas.height); 8 9 //이미지 데이터를 원하는 영역만 다른 캔버스에 출력한다. 10 context2.putImageData(imageData, 0, 0, 100, 0, 200, 200); }, false); </pre>





■ 이미지 조작하기



이미지 데이터 저장 - toDataURL()

url = canvas.toDataURL(type, quality) 캔버스에 그려진 내용을 URL 문자열로 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.

data:image/png;base64,iVBORw0KGGoAAAANSUhEUgAAAZAAAADICAYAAADGFbfIAAAgAEIEQVR.....

예제		Canvas_12-03_toDataURL.html
Script	<pre> 1 ... 앞의 예제 참고 2 canvas.addEventListener("click", function (){ //캔버스의 이벤트리스너에서 마우스 클릭시 3 var myImage = document.getElementById("myImage"); 4 var textArea = document.getElementById("textArea"); 5 6 //이미지의 src와 텍스트 필드의 value에 URL 문자열을 할당한다. 7 textArea.value = myImage.src = canvas.toDataURL(); 8 }, false); </pre>	
HTML	<pre> 1 2 <textarea id="textArea" cols="110" rows="10"> </textarea> </pre>	



이미지 조작하기



이미지 데이터 저장 - toDataURL()

캔버스를 마우스로 클릭하면 오른쪽에 이미지로 복사하고 이미지 문자열을 텍스트 상자에 복사합니다.



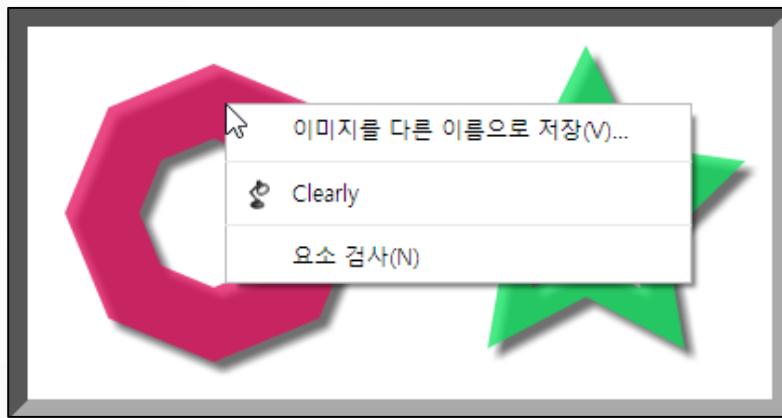
```
data:image/png;base64,iVBORw0KGgoAAAANSUHEUgAAAAZAAADICAYAAADGFbfIAAAgAEIBQVPAxu19C7gcZZImVY9zzkIDwjYAAoaEuzxkwTs3H
XUSZnQGDZiQRAWdcXTAnXVQxFVnfXZhPxZHXBR2dZhcELIIGBEZKAEMxoAFMAIGQCBnJzUC4BEL00X2r2u/9q74+f/+nuruqu6q7qs/fz90cw0I1
ddX3//W///td3s809EtEbQftAWQbQFugBQuYLXxGFORbQftAWQbQFvAQACIJ4G2gLaAtOC2QEsWQADSktN0h7QFtAWQbQFNIDoOaAtOC2gLaAtCUI
FNIQQZDb9IWOBBQftAWQbQFugBQuYLXxGFORbQftAWQbQFvAQACIJ4G2gLaAtOC2QEsWQADSktN0h7QFtAWQbQFNIDoOaAtOC2gLaAtCUI
Cwe9EcX0ih6T9bl4gLGhcXqQFkHAyyvkYtgV6wwMD0xTcKADnHP1/ohfvphXvQANILo6jvQWugxyQg2IedEgBimNYXIIWsr3PLdo+blIa3pwEkVs0hL
OZbQFvAyyKCFZimcGEZtr2OWMgX6beUcqxF/4+3BpEOTSMNIBoYtP4abQFtdYsUMM+3FNV+968dvjt9zwLPuL+CaBRpnfRBZHwvKx/KpAFNIAEMpc+
WFtAW6DTFqhhH4KBGKY9/Hp26LQ7/ocCICPO/OPOLmkWOpIR0gDSGTvrb9EW0BZowQJe7M0w7JRdsjKVe7f/z8I/PL7VPS3Vx5v0fp3ew5qFtGDsFJ6
iAaQFo+mPaAtOC3TGAp7so2yIjZKYtV8aemH4gl98370SsI8D9H6F3gc1C+nM+GgA6Yvd9bdoC2gLBLRAXfZrTLIAEKNQyRv2nBr+ZbNO+jUh+i9n9
4v0ftVzUICGrvFwzWAtGg4/TfTAW2BaC3Qv3Px9aZpnI9f9FsQ+wD6KVg7gQSCSg/zh5ecLf71qBRCD19WL9P6jCySahUQ7POLsGkA6VQG/X7FzYnFc0
05dimNt07rrhEM/w00hX9oC49ICA9svm2Zk8rfv3DxiH2AfrQKPGpWzS/SzbGckf/Wbn1mbDuymY/e5A1KfmoVQVQZoA0mAkett9xfaBy6aI7dx8GoR5
```



이미지 조작하기



이미지 데이터 저장 – `toDataURL()`



캔버스에서 오른쪽 버튼 클릭 시



이미지에서 오른쪽 버튼 클릭 시



클리핑



클리핑 영역 지정하기

- 클리핑 영역은 캔버스 영역에서 드로잉을 제한하도록 해주는 패스로 정의된 특수한 영역을 말한다.
 - 컨텍스트가 초기화될 때, 클리핑 영역은 기본적으로 상단 왼쪽(0,0)에서 좌표 공간의 폭과 높이를 가지는 구조로 설정되기 때문에 클리핑 영역의 크기와 캔버스의 크기와 동일하게 된다.
- 클리핑 영역은 현재의 클리핑 영역과 현재 패스에서 그려진 영역이 겹치는 부분을 계산하여 새로운 클리핑 영역으로 생성한다.
 - 만일 열려 있는 서브 패스가 있다면, 클리핑 영역을 계산할 때 암시적으로 닫혀있는 것으로 처리한다. 그러나 서브 패스에는 영향을 주지 않는다.
- 클리핑 영역 지정하기
 1. 원하는 클리핑 모양의 패스를 지정한다.
 2. `context.clip()`

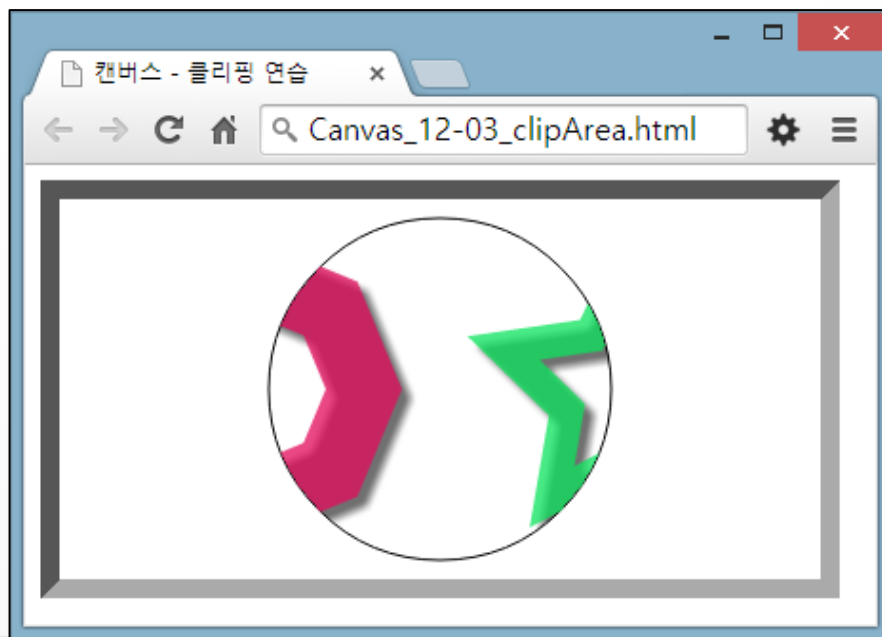


클리핑



클리핑 영역 지정하기

예제		Canvas_12-03_clipArea.html
Script	1 2 3 4 5	<pre>... 앞의 예제 참고 ... // 둥근 클리핑 패스를 생성한다. context.beginPath (); context.arc(200, 100, 90, 0, Math.PI * 2, false); //원 모양의 클리핑 영역 패스를 만든다. context.stroke(); context.clip(); //클리핑 영역을 설정한다. ... 앞의 예제 참고 ...</pre>





클리핑



클리핑 영역 지우기

- 클리핑 영역을 설정하고 `clearRect()` 메서드를 호출하면 지워지는 영역이 클리핑 영역으로 제한된다.

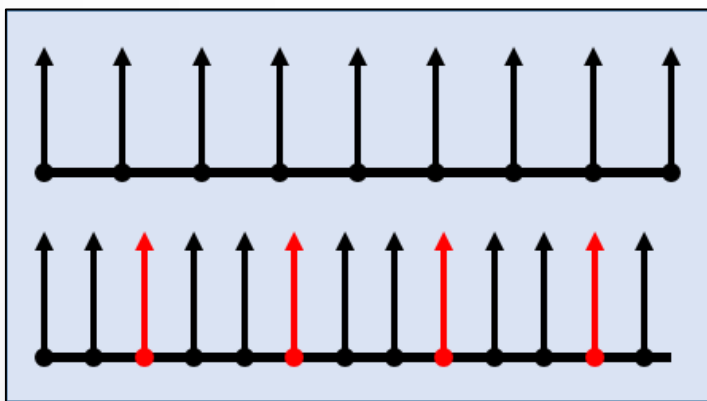




표준화 문서	Timing control for script-based animations - http://www.w3.org/TR/animation-timing/
표준화 단계	W3C Candidate Recommendation, (2013-10-31)

■ 기존 방법의 문제

- SVG에서의 `<animate>` 요소와 스크립트에서 구현되는 스크립트 기반의 `setTimeout()` 메서드와 `setInterval()` 메서드이다.
 - `setTimeout()` 메서드가 특정 시간에 메서드를 한 번만 호출하는 반면에 `setInterval()` 메서드는 특정 시간마다 메서드를 반복적으로 호출한다.
 - 이러한 메서드들의 문제는 애니메이션을 업데이트하기 위한 최적의 주기(frequency)가 무엇인지 모르기 때문에 개발자는 가장 간단한 방법으로 10ms와 같은 최소한의 시간으로 고정시킨다.



타이밍 불일치 발생



기존 방법의 문제

setTimeout based animations - 1ms / 10ms / 15ms

Expected callbacks: 8183
Actual callbacks: 5474
Callback Efficiency: 59.7%
CPU Efficiency: Medium
Power Consumption: Medium
Background Interference: High



setTimeout 기반의 애니메이션

requestAnimationFrame based animations

Actual callbacks: 8184
Expected callbacks: 8184
Callback Efficiency: 100%
CPU Efficiency: High
Power Consumption: Low
Background Interference: Low



requestAnimationFrame 기반의 애니메이션



■ 스크립트 기반 애니메이션용 타이밍 컨트롤

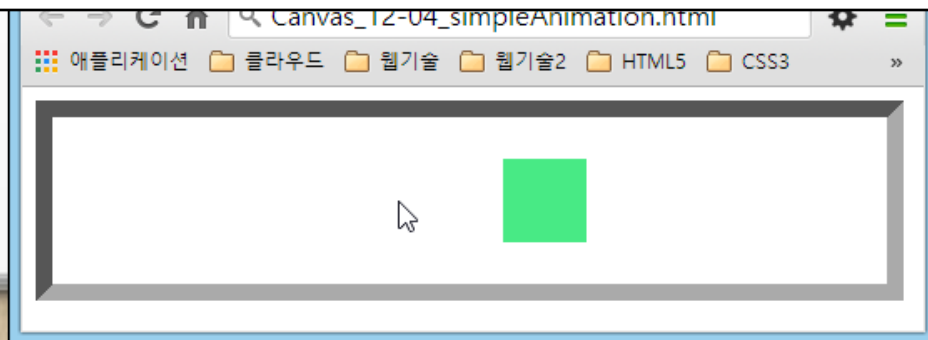
```
partial interface Window {  
    long requestAnimationFrame(FrameRequestCallback callback);  
    void cancelAnimationFrame(long handle);  
};  
  
callback FrameRequestCallback = void (DOMHighResTimeStamp time);
```

메서드	설명
requestAnimationFrame	다음 애니메이션 프레임을 그릴 준비가 되면 브라우저에게 특정 콜백을 호출할 수 있도록 요청한다. long 타입의 핸들을 반환한다. cancelAnimationFrame 메서드에서 사용된다. 다음 애니메이션을 그릴 시간을 알 필요가 없다는 장점이 있다.
cancelAnimationFrame	애니메이션 프레임을 업데이트하도록 이전에 예약되어 있는 콜백 요청을 취소하는데 사용된다.



■ 스크립트 기반 애니메이션용 타이밍 컨트롤

예제	Canvas_12-04_simpleAnimation.html
Script	<pre> 6 ... 7 function draw() { 8 context.clearRect(0, 0, canvas.width, canvas.height); 9 context.fillRect(x, y, rectWidth, rectWidth); 10 } 11 12 function animate() { 13 requestId = window.requestAnimationFrame(animate); 14 x += speed; 15 if(x <= 0 x >= (canvas.width - rectWidth)){ speed = -speed; } //방향을 바꾸도록 한다. 16 17 draw(); 18 } 19 20 canvas.addEventListener("click", function () { //캔버스의 이벤트리스너에서 마우스 클릭시 21 if(requestId > 0) { //애니메이션이 동작중일 때 22 window.cancelAnimationFrame(requestId); //애니메이션을 정지한다.. 23 requestId = 0; //핸들 번호를 초기화 시킨다. 24 } else { //애니메이션이 정지 상태이면 다시 시작한다. 25 requestId = window.requestAnimationFrame(animate); 26 } 27 }, false); </pre>

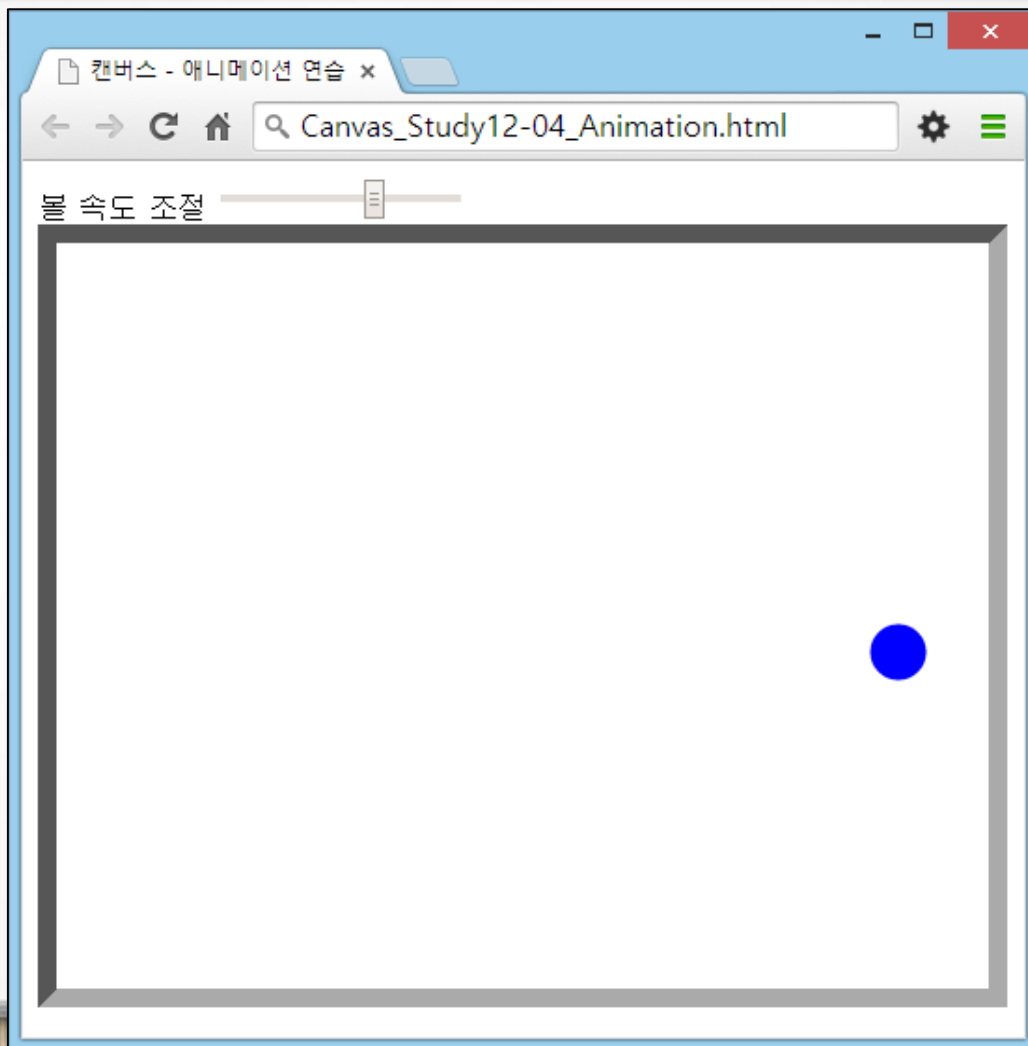




■ 스크립트 기반 애니메이션용 타이밍 컨트롤



실습하기 - CSS_Study12-O4_Animation.html



■ 히트 영역

- HTML5 캔버스에서는 히트 영역을 위한 기능 지원이 확대되었다.
이 기능을 사용하면 사용자의 이벤트에 응답하는 임의의 영역을 지정할 수 있게 된다. 즉, 그래픽서 핫-스팟(hot-spots)을 정의할 수 있다.
 - 히트 영역은 최근에 추가된 기능이기에 아직은 주요 브라우저에서 지원되지 않는다.
- 각각의 히트 영역은 다음 정보로 구성되어 있다.
 - 캔버스의 비트맵에 있는 패스
 - 히트 영역의 패스를 둘러싸고 있는 캔버스 요소의 비트맵에 있는 경계의 둘레
 - 선택적으로, 다른 영역으로부터 영역을 구별하기 위한 ID를 나타내는 비어 있지 않은 문자열
 - 선택적인 컨트롤(특정 조건에서의 요소 노드에 대한 참조, 캔버스 요소의 하위 요소가 아니면 무시된다) .

히트 영역

context.addHitRegion(options)

현재 기본 패스를 기반으로 캔버스 비트맵에 대한 히트 영역을 추가한다.

- ✓ 이 메서드의 options 인수는 다음과 같은 멤버를 가진 객체이다.
- ✓ **id(기본적으로 비어 있는 문자열)** – 히트 영역에서 사용할 수 있는 아이디를 나타낸다. 캔버스의 이벤트 영역(event.region)에서의 MouseEvent 이벤트와 이 메서드에 대한 이후에 호출되는 영역을 식별하는 방법으로 사용된다. 이 옵션을 사용하면 마우스 이벤트에서 마우스(ID 포함)가 히트 영역에 있을 때, 브라우저가 이 영역을 검사하게 함으로써, 히트 감지를 쉽게 만들 수 있다.
- ✓ **control(기본 값은 Null)** – 이벤트에 대한 캔버스의 하위 요소로써, 히트 영역과의 상호작용을 위한 대리자로서의 사용할 수 있는 접근성 도구를 나타낸다.

context.removeHitRegion(id)

캔버스 비트맵에서 히트 영역을 제거한다.

- ✓ 인수 id는 addHitRegion() 메서드를 사용하여 추가된 히트 영역의 ID를 나타낸다.

context.clearHitRegions()

캔버스 비트맵에서의 모든 히트 영역을 제거한다.

- ✓ 이 메서드로 사용하여 히트 영역들에 있는 패스들은 효과적으로 지워진다.

MouseEvent 인터페이스

```

partial interface MouseEvent { readonly attribute DOMString? region; };
partial dictionary MouseEventInit { DOMString ? region; };

```

event.region

만일, 마우스가 히트 영역에 있으면 히트 영역의 ID를 반환한다. 그 외에는 null값을 반환한다.

히트 영역

addHitRegion() 메서드의 옵션 인수

속성	기본 값	설명
path	null	히트 영역을 위해 응답이 가능한 패스/도형을 결정하는 캔버스 Path 객체를 나타낸다. 이 값이 지정되지 않으면 현재 패스가 사용된다.
fillRule	nonzero	패스의 내부를 채우는(fill) 규칙을 나타낸다. 자세한 내용은 11장 3절에 설명되어 있는 와인딩 규칙을 참고하도록 한다.
id	빈 문자열	이벤트 및 addHitRegion 메서드로 정의된 히트 영역을 식별하는데 사용된다.
parentID	null	부모 히트 영역의 ID가 될 수 있는 ParentID를 나타낸다. 부모 히트 영역으로 사용될 수 있는 잠재적인 하나는 포인터에 대한 다른 스타일을 지정할 수 있도록 하는 커서의 대체 요소(fallback)를 가질 수 있다는 것이다.
cursor	inherit	히트 영역에 대한 커서의 스타일을 나타낸다. 예를 들어, 마우스 오버시 커서의 모양을 변경할 수 있다.
control	null	클릭이나 마우스 이벤트에 대한 캔버스의 하위 요소로써, 히트 영역과의 상호작용을 위한 대리자로서의 사용할 수 있는 접근성 도구를 나타낸다. <button>과 같은 컨트롤이 될 수 있다. 따라서, "proxy" 같은 요소를 사용한 다음에 이 요소에 대한 이벤트 리스너 코드를 추가하면 된다. 기본 값은 null이다.
label	null	주어진 control이 없을 경우에, 히트 영역의 설명으로 사용되는 "텍스트 레이블"과 같은 접근성 도구를 나타낸다.
role	null	주어진 control이 없을 경우에, 히트 영역을 어떻게 표현하는 지를 결정하기 위해 사용되는 접근성 툴을 위한 ARIA 역할을 나타낸다.

히트 영역

예제		Canvas_12-05_simpleHitRegion.html
Script	1	<script>
	2	context.beginPath();
	3	context.rect(10,10,100,100); //첫 번째 히트 영역
	4	context.fill();
	5	context.addHitRegion({'id': 'First Button', 'cursor': 'pointer'});
	6	
	7	context.beginPath();
	8	context.rect(120,10,100,100); //두 번째 히트 영역
	9	context.fill();
	10	context.addHitRegion({'id': 'Second Button', 'cursor': 'pointer'});
	11	
	12	canvas.onclick = function (event)
	13	{
	14	if (event. region) { alert('You clicked ' + event. region); }
	15	}
	16	</script>



다음 학습



- 1 미디어 요소
- 2 오디오
- 3 비디오