HTML5 웹 프로그래밍 입문(교수용)

10장. 캔버스

목차

10.1 캔버스 이해하기

10.1.1 캔버스의 특징

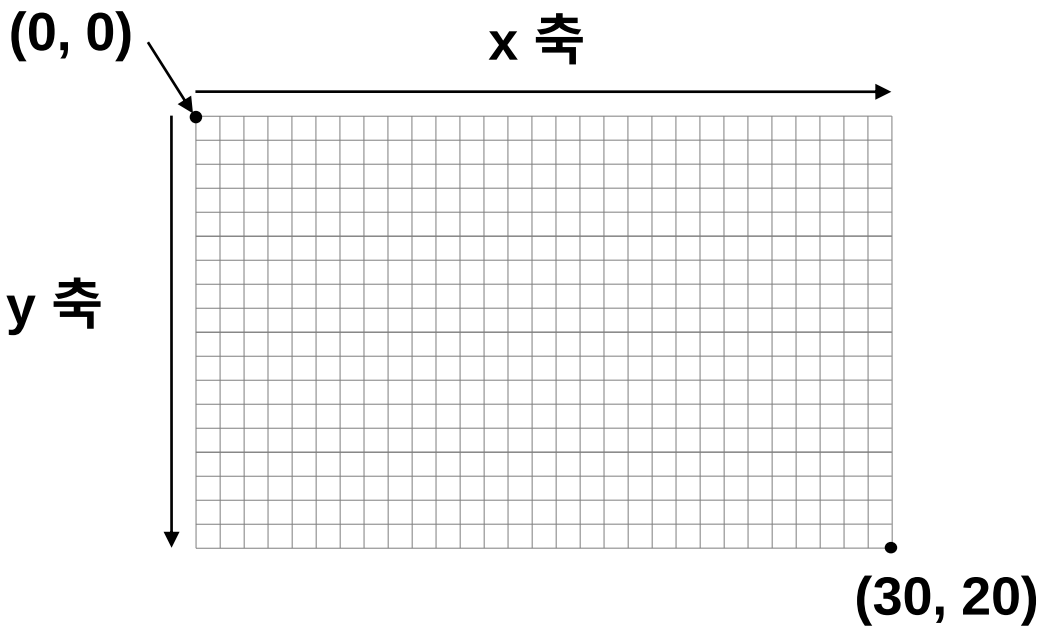
10.1.2 캔버스와 컨텍스트 객체

HTML5 캔버스

- 자바스크립트를 이용해서 웹 문서상에 그림 그리는 기능
  - HTML5 이전
    - 직접 이미지 파일을 <img> 태그를 이용해서 문서상에 포함
    - 자바 애플릿 이용
    - 플래시 이용
  - HTML5 캔버스
    - 자바스크립트만을 이용해서 그림을 그릴 수 있다
    - 별도의 플러그인이나 프로그램 설치 없이 가능
    - 이미지나 그림을 합성, 변환 조작도 가능

캔버스 좌표계

- 사각 평면의 2차원 좌표계 사용
  - 이차원 (2D) 이미지 표현
  - x, y 2개의 축으로 구성
  - 왼쪽 상단 모서리가 원점 (0, 0)



비트맵 그래픽

- 픽셀(pixel)
  - 좌표계 상의 각각의 정사각형 네모 칸
  - 이미지를 구성하는 점이며 색상을 가진다.
  - 각 픽셀의 색상 값을 바꾸어 다양한 이미지를 표현
- 비트맥 그래픽
  - Bitmap graphics
  - 픽셀만으로 이미지를 표현하고 저장하는 형태
- 캔버스의 도형이나 그림, 글씨 등 2차원 비트맵으로 저장
  - 이미 그려진 도형이나 그림을 확대하는 등은 작업은 불가능

캔버스로 그림 그리기 준비

- 캔버스 요소
  - <canvas> 태그를 이용해서 캔버스 요소 추가
  - width와 height 속성을 이용해 캔버스 좌표계의 크기 지정
  - DOM을 통한 접근을 위해 id 지정

```
<canvas id="myCanvas" width="30" height="20"></canvas>
```

- 컨텍스트(context) 객체
  - 캔버스에 내용을 채우기 위한 객체
  - 캔버스 요소 객체의 getContext() 메소드를 이용
    - <canvas> 요소 객체에 접근한 후 getContext("2d") 메소드 실행

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

10.2 캔버스 기본 API 사용하기

10.2.1 기본 도형 그리기
10.2.2 기본 도형 꾸미기
10.2.3 이미지와 글자 그리기

캔버스 기본 도형 그리기

- 캔버스 기본 API
  - 컨텍스트 객체의 메소드를 호출함로써 그림이 그려짐
- 캔버스 컨텍스트의 선 그리기 메소드
  - 선긋기, 경로, 곡선 등
  - 현재 시작 지점에서 다음 지점까지 선을 연결하는 방식
    - 현재위치 이동
      - moveTo(x,y): 현재 시작 지점을 이동시키는 메소드
      - 선을 그린 마지막 지점으로 현재 위치 이동

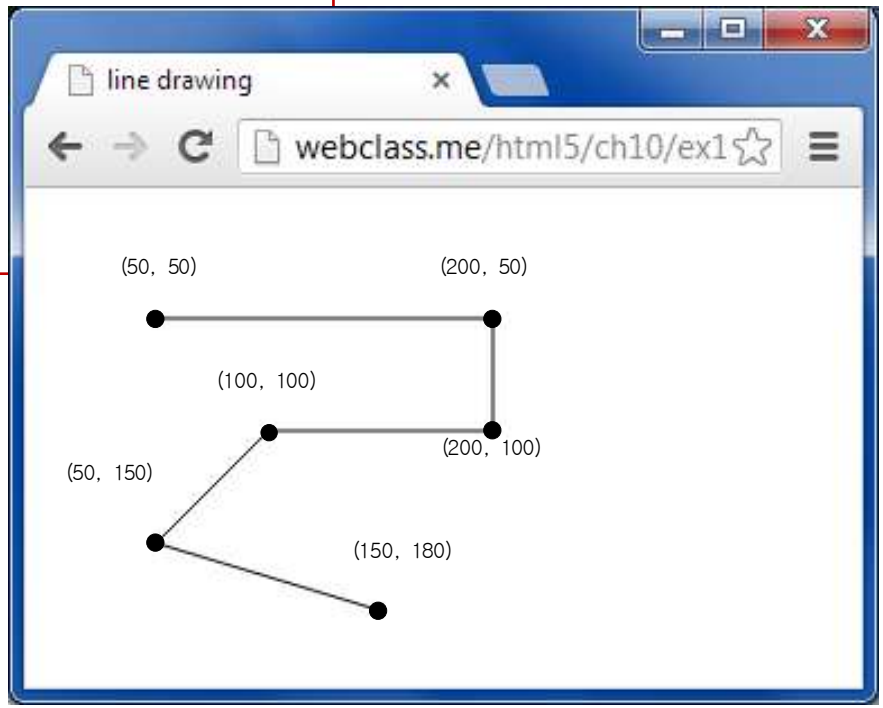
캔버스 기본 도형 그리기

- 기본 도형 그리기 메소드

캔버스 컨텍스트 메소드	기능 및 설명
context.moveTo(x, y)	선의 시작 지점을 (x, y) 좌표로 이동시킨다.
context.lineTo(x, y)	현재의 시작점에서 (x, y) 지점까지 선을 그린다.
context.rect(x, y, width, height);	왼쪽 위 모서리를 (x, y) 지점으로 하고 가로와 세로 변의 크기가 각각 width, height인 사각형을 그린다. 현재의 시작점을 (x, y)로 이동시킨다 .
context.stroke();	현재 지정된 색상과 선 끝 모양으로 선을 그린다. stroke() 메소드를 실행하지 않으면 선이 그려지지 않는다. 기본 색상은 검정색이다.

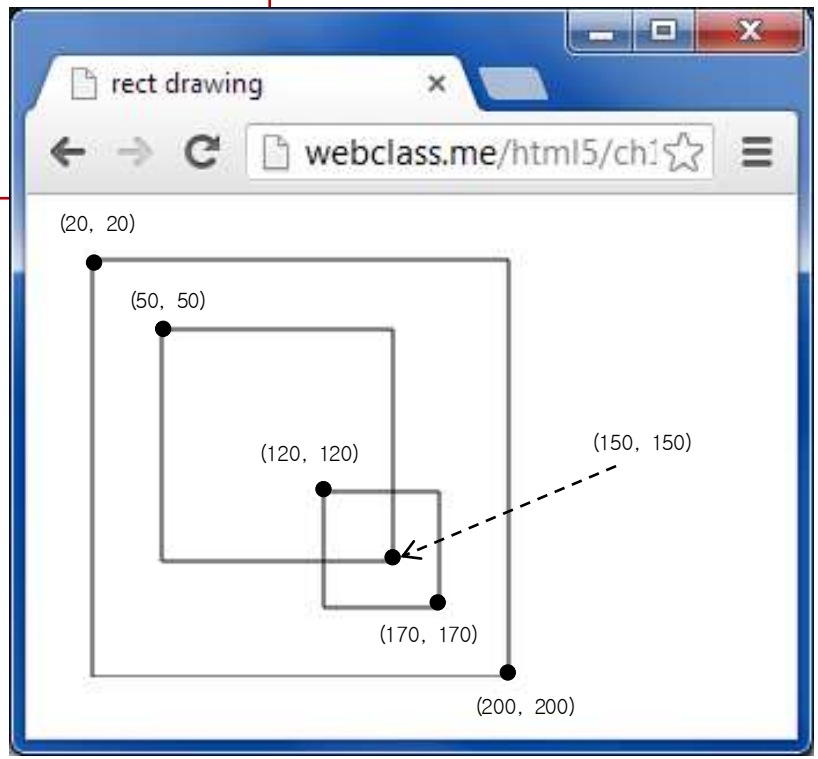
직선 그리기 예제

```
1  var canvas = document.getElementById("myCanvas");
2  var context = canvas.getContext("2d");
3
4  context.moveTo(50, 50);
5  context.lineTo(200, 50);
6  context.lineTo(200, 100);
7  context.lineTo(100, 100);
8  context.lineTo(50, 150);
9  context.lineTo(150, 180);
10 context.stroke();
```



사각형 그리기 예제

```
1  var canvas = document.getElementById("myCanvas");
2  var context = canvas.getContext("2d");
3
4  context.rect(50, 50, 100, 100);
5  context.rect(20, 20, 180, 180);
6   context.rect(120, 120, 50, 50);
7  context.stroke();
```



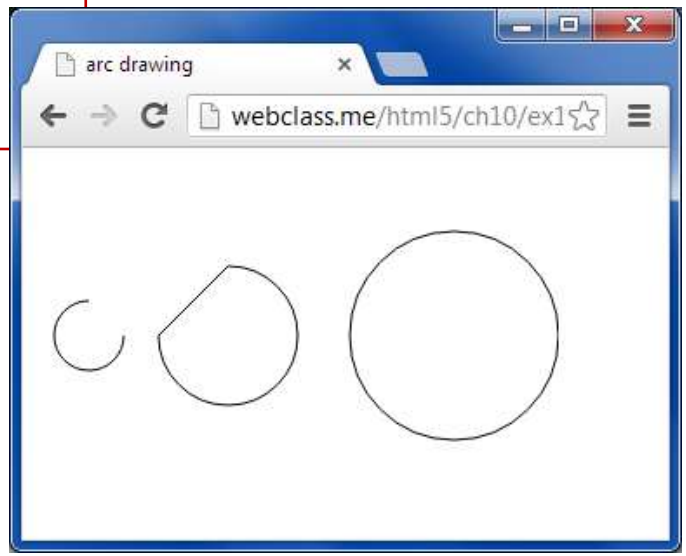
원호와 곡선 그리기

- stroke() 메소드를 호출하지 않으면 실제로 캔버스에 선이 그려지지 않음에 유의

캔버스 컨텍스트 메소드	기능 및 설명
context.arc(x, y, r, startAngle, endAngle, antiClockwise)	(x, y)를 원점으로 하고 반지름 r인 원호를 그린다. 시작 각도와 끝 각도를 지정하여 원호를 그린다. antiClockwise 값을 false로 설정하면 시계방향으로 원호를 그린다. 기본값은 시계방향이다.
context.quadraticCurveTo(cx, cy, x, y);	하나의 제어점을 가지는 곡선을 그린다. 시작점은 현재 위치이며 끝 점은 (x, y)이다. (cx, cy)이 제어점이 된다.
context.bezierCurveTo(cx1, cy1, cx2, cy2, x, y);	두개의 제어점을 가지는 곡선을 그린다. 시작점은 현재 위치이며 끝 점은 (x, y)이다. 두개의 제어점은 (cx1, cy1)과 (cx2, xy2)로 지정한다.

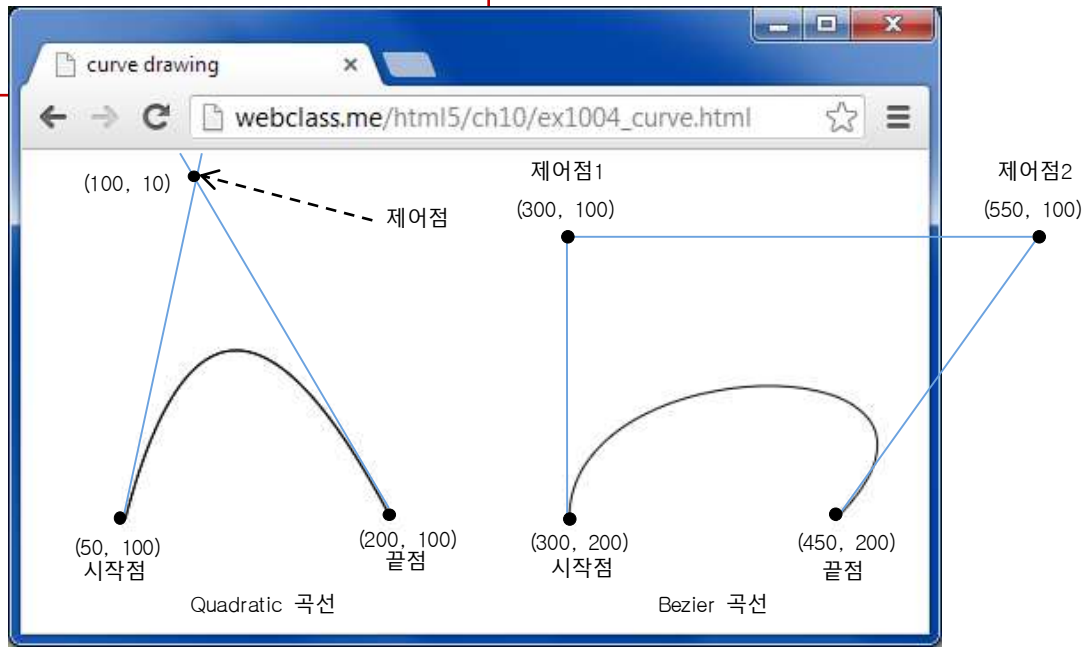
원호 그리기 예제

```
1  var canvas = document.getElementById("myCanvas");
2  var context = canvas.getContext("2d");
3
4  context.beginPath();
5  context.arc(30, 100, 20, 0, 1.5*Math.PI); // Math.PI 상수를 이용해 각도지정
6  context.stroke();
7
8  context.beginPath();
9  context.arc(110, 100, 40, 1*Math.PI, 1.5*Math.PI, true);// 반시계방향 원호
10 context.closePath(); // 경로 시작점까지 직선으로 연결하며 경로를 종료한다.
11 context.stroke();
12
13 context.beginPath();
14 context.arc(240, 100, 60, 0, 2*Math.PI); // 360도 원호를 그려 원 그리기
15 context.stroke();
```



곡선 그리기 예제

```
1  var canvas = document.getElementById("myCanvas");
2  var context = canvas.getContext("2d");
3
4  context.moveTo(50, 200);
5  context.quadraticCurveTo(100, 10, 200, 200);
6
7  context.stroke();
8
9  context.moveTo(300, 200);
10 context.bezierCurveTo(300, 100, 600, 100, 450, 200);
11
12 context.stroke();
```



경로 그리기

- 연속된 선 그리기를 통한 경로 그리기
  - beginPath()
    - 경로의 시작 설정
  - closePath()
    - 경로 지정을 종료
    - 처음 경로 시작 지점으로 선을 연결하여 경로를 완성

캔버스 컨텍스트 메소드	기능 및 설명
context.beginPath();	경로 지정을 시작하는 메소드이다.
context.closePath();	경로 지정의 종료를 의미하는 메소드이며 현재까지 그려진 경로의 마지막 위치에서 경로의 시작점까지 직선으로 연결한다. 그리고, 현재 위치는 경로의 시작점으로 이동 시킨다.

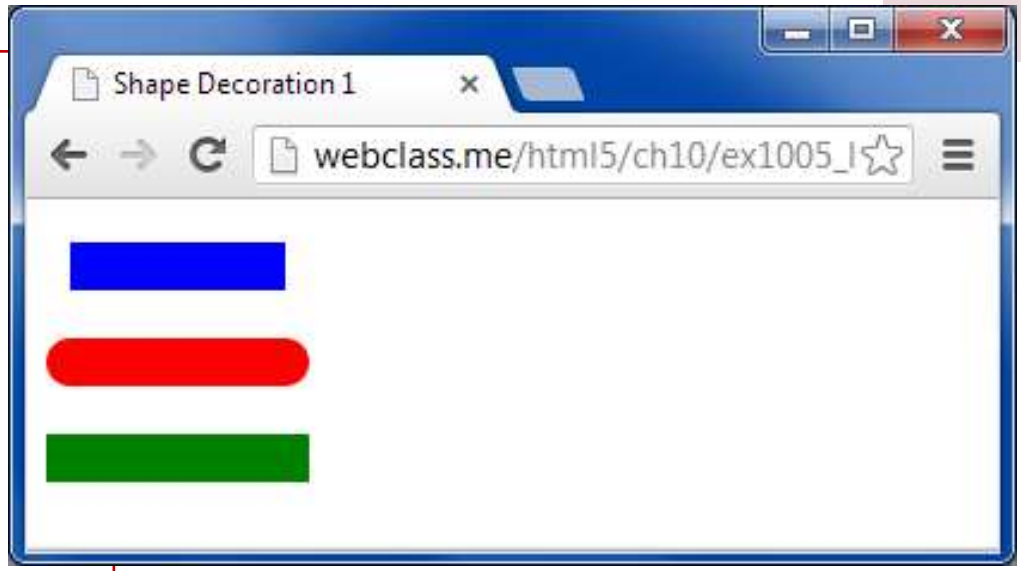
기본 도형 꾸미기

- 선 꾸미기와 색칠하기

캔버스 컨텍스트 속성 및 메소드	기능 및 설명
context.lineWidth	선의 두께를 픽셀 개수로 설정한다.
context.strokeStyle	선의 색상을 지정한다. 색상을 지정하는 방법은 일반적인 웹 문서에서와 동일하다. 예) "blue" 혹은 "#0000ff" 등
context.lineCap	선의 양쪽 끝 모양을 지정한다. 지정할 수 있는 형태는 "butt", "round", "square"이며 기본 값은 "butt"이다.
context.lineJoin	선이 꺽이는 모서리 지점에서의 모양을 지정한다. "miter", "round", "bevel" 세가지 중의 한가지 값으로 지정할 수 있ㄷ. 기본값은 "miter" 스타일이다.
context.fillStyle	경로, 원, 사각형 등의 도형의 내부를 색칠할 색상 값을 지정한다. 스타일값으로 그라데이션이나 패턴을 지정할 수도 있다. 색상은 웹 문서와 동일하게 지정할 수 있다. 예) "red" 혹은 "#ff0000" 등
context.fill();	현재 지정된 fillStyle 색상으로 도형을 채운다. 색칠할 도형은 fill() 메소드를 실행하기 이전에 그려지 모든 도형들이다.

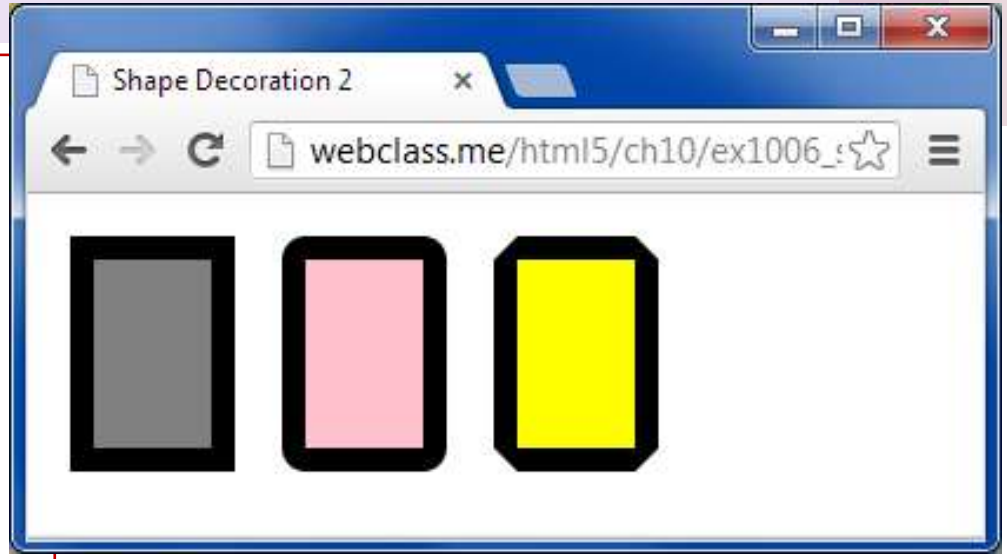
선 꾸미기 예제

```
1   context.beginPath();
2    context.moveTo(10, 20);
3    context.lineTo(100,20);
4    context.lineWidth = 20;
5    context.strokeStyle = "blue";
6    context.lineCap = "butt";
7    context.stroke();
8
9    context.beginPath();
10   context.moveTo(10, 60);
11   context.lineTo(100, 60);
12   context.strokeStyle = "red";
13   context.lineCap = "round";
14   context.stroke();
15
16   context.beginPath();
17   context.moveTo(10, 100);
18   context.lineTo(100, 100);
19   context.strokeStyle = "green";
20   context.lineCap = "square";
21   context.stroke();
```



도형 꾸미기 예제

```
1   context.beginPath();
2    context.rect(20, 20, 50, 80);
3    context.lineWidth = 20;
4    context.strokeStyle = "black";
5    context.lineJoin = "miter";
6    context.fillStyle = "grey";
7    context.stroke();
8    context.fill();
9
10   context.beginPath();
11   context.rect(110, 20, 50, 80);
12   context.strokeStyle = "black";
13   context.lineJoin = "round";
14   context.fillStyle = "pink";
15   context.stroke();
16   context.fill();
17
18   context.beginPath();
19   context.rect(200, 20, 50, 80);
20   context.strokeStyle = "black";
21   context.lineJoin = "bevel";
22   context.fillStyle = "yellow";
23   context.stroke();
24   context.fill();
```



이미지 그리기

- 기존에는 이미지를 그리기 위해서는 <img> 태그를 이용
- 캔버스에 이미지 그리기
  - 사이즈 조정, 크롭(crop) 등의 기능도 가능
  - Image 객체를 이용
    - Image() 생성자를 이용해서 생성

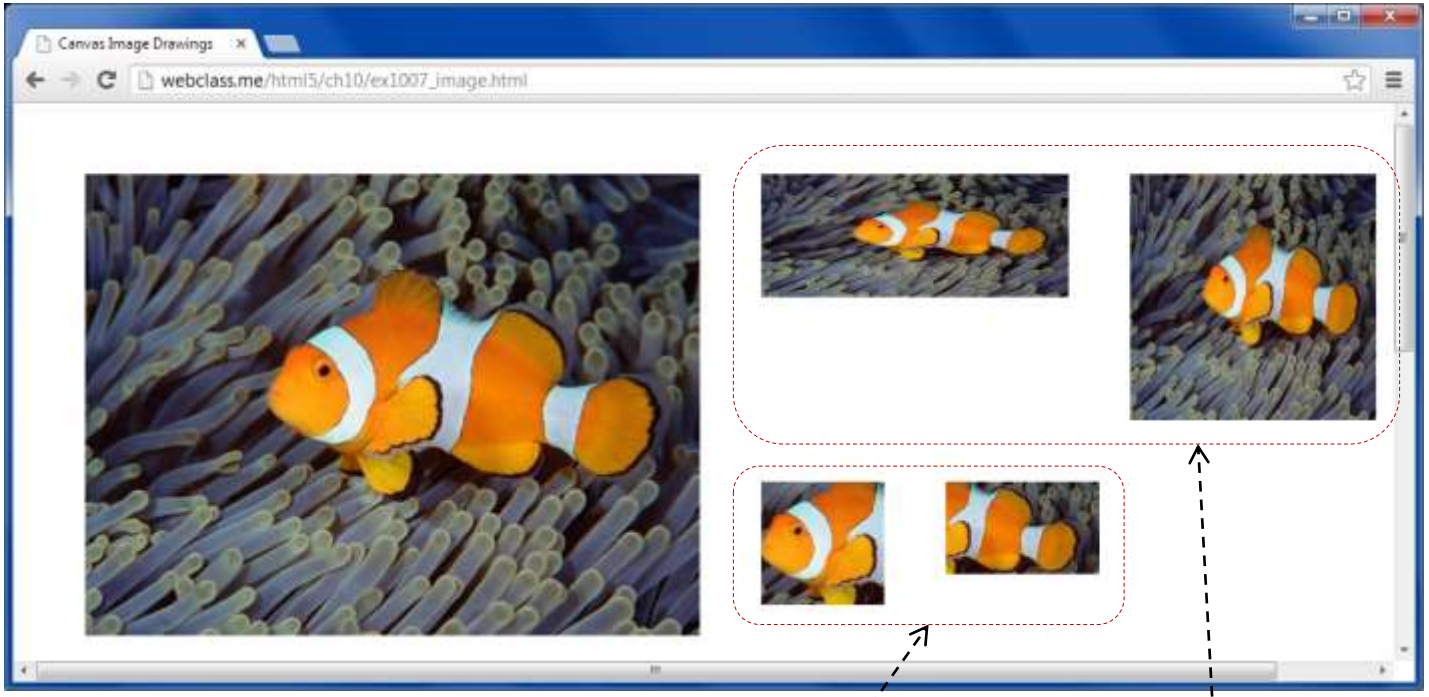
```
var imgObj = new Image();
```

  - drawImage() 메소드
    - 캔버스 컨텍스트에서 이미지를 그리는 메소드

이미지 그리기 예제

```
1   var canvas = document.getElementById("myCanvas");
2   var context = canvas.getContext("2d");
3
4   var imgObj = new Image();
5   imgObj.src = "clownfish.jpg";
6
7   imgObj.onload = function() {
8     // (50, 50) 지점에 원래 크기 그대로 이미지 그리기
9     context.drawImage(imgObj, 50, 50);
10
11    // 사이즈 조정하기: (50, 450) 지점에 250 x 100 크기로 이미지 그리기
12    context.drawImage(imgObj, 50, 450, 250, 100);
13
14    // 사이즈 조정하기: (350, 450) 지점에 200 x 200 크기로 이미지 그리기
15    context.drawImage(imgObj, 350, 450, 200, 200);
16
17    // 이미지 크롭후 사이즈 조정:
18    // 1) 원본 이미지 (150, 100) 지점에서 150 x 50 크기의 이미지를 크롭
19    // 2) Canvas의 (50, 700) 지점에 100 x 75 크리로 그리기
20    context.drawImage(imgObj, 150, 100, 150, 150, 50, 700, 100, 100);
21
22    // 이미지 크롭후 사이즈 조정:
23    // 1) 원본 이미지 (250, 100) 지점에서 250 x 150 크기의 이미지를 크롭
24    // 2) Canvas의 (200, 700) 지점에 125 x 75 크리로 그리기
25    context.drawImage(imgObj, 250, 100, 250, 150, 200, 700, 125, 75);
26  }
```

이미지 그리기 예제 실행 결과



캔버스에 글자 그리기

- 비트맵 방식으로 캔버스에 텍스트 그리기
  - 삽입된 글자를 수정하거나 크기를 조정하는 것은 불가능
  - 텍스트를 그려 넣을기 전에 폰트, 크기, 정렬방법 등을 결정

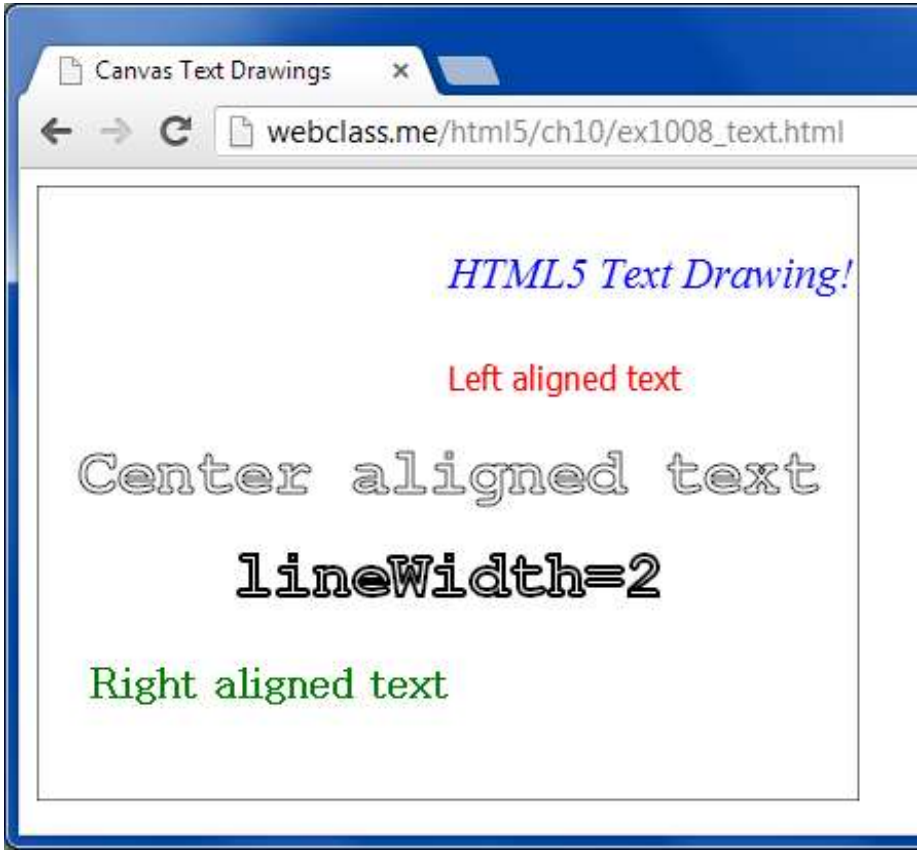
캔버스 컨텍스트 속성 및 메소드	기능 및 설명
context.font	그려 넣을 텍스트의 글자체를 지정한다. 이탤릭체 여부, 글자 크기, 폰트 등을 한번에 지정하게 된다.
context.textAlign	텍스트의 정렬방식을 지정한다. "left", "right", "center", "start", "end"의 값을 가질 수 있다. 기본 값은 "start"이다.
context.fillStyle	글자의 색상을 지정한다. 색상을 지정하는 방법은 일반적인 웹 문서에서와 동일하다. 예) "blue" 혹은 "#0000ff" 등
context.fillText()	현재 지정된 fillStyle 색상으로 캔버스의 지정된 위치에 글자를 를 그려 넣는다. 글자의 왼쪽 위 모서리 지점이 그려넣는 기준점이 된다.

캔버스에 글자 장식하기

- 색상 및 외곽선 두께 등을 지정

캔버스 컨텍스트 속성 및 메소드	기능 및 설명
context.strokeStyle = 색상값;	글자의 외곽선을 그릴 색상을 지정한다. 색상을 지정하는 방법은 일반적인 웹 문서에서와 동일하다. 예) "blue" 혹은 "#0000ff" 등
Context.lineWidth = 선두께;	글자의 외곽선을 그릴 선의 두께를 지정한다.
context.strokeText(text, x, y);	현재 지정된 strokeStyle 색상으로 캔버스의 (x, y) 위치에 text의 외곽선을 그려 넣는다. 글자의 외곽선만 그려지게 되므로 내부가 비어있는 형태가 된다. 텍스트의 왼쪽 위 모서리 지점이 텍스트를 그려넣는 기준점이 된다.

글자 그려넣기 예제




```
1   context.rect(0, 0, 400, 300);
2   context.stroke();
3
4   var text1 = "HTML5 Text Drawing!";
5   var text2 = "Left aligned text";
6   var text3 = "Center aligned text";
7   var text4 = "Right aligned text";
8
9   context.font = "italic 16pt Times New Roman";
10  context.fillStyle = "blue";
11  context.fillText(text1, 200, 50);
12
13  context.font = "12pt Tahoma";
14  context.fillStyle = "red";
15  context.textAlign = "left";
16  context.fillText(text2, 200, 100);
17
18  context.font = "bold 24pt Courier New";
19  context.strokeStyle = "black";
20  context.textAlign = "center";
21  context.lineWidth = 1;
22  context.strokeText(text3, 200, 150);
23  context.lineWidth = 2;
24  context.strokeText("lineWidth=2", 200, 200);
25
26  context.font = "bold 16pt Batang";
27  context.fillStyle = "green";
28  context.textAlign = "right";
29  context.fillText(text4, 200, 250);
```

10.3 캔버스 고급 기능 사용하기

10.3.1 그리기 효과

10.3.2 변환 효과

10.3.3 기타 고급 기능

그리기 효과

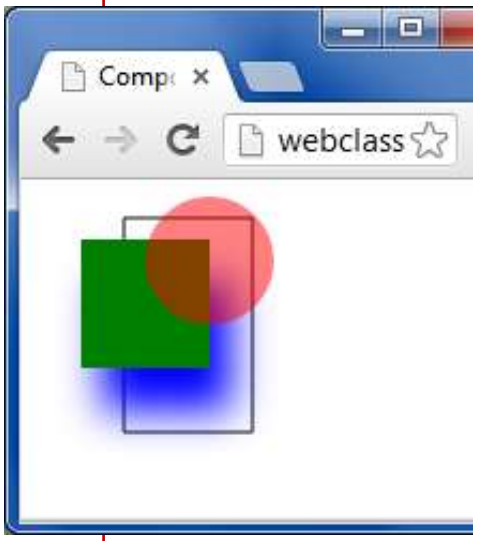
- 합성 (composition) 효과
  - 그림자 효과를 줄 수 있는 **shadow**
  - 투명도 조절을 위한 **globalAlpha**
  - 지정한 도형 모양으로 잘라내는 **clip**(클립)
  - 도형간의 연산을 위한 **operation**

캔버스 컨텍스트 속성 및 메소드	기능 및 설명
context.shadowColor **context.shadowBlur** **context.shadowOffsetX** **context.shadowOffsetY**	그림자 효과를 줄 때 사용하는 속성들이다. 그림자의 색상, 흐림정도, 그림자의 크기를 지정할 수 있다. shadowOffsetX, shadowOffsetY 값을 조정함으로써 그림자의 크기를 조절할 수 있다.
context.globalAlpha	투명도를 조절하기 위해서는globalAlpha 속성값을 조절하면 된다. 0과 1 사이의 실수값을 가져야 하며 0이 완전 투명한 상태, 1이 완전히 불투명한 상태를 뜻한다.
context.clip()	clip() 메소드가 실행되기 바로 이전에 정의된 경로로도형 자르기를 수행한다. 경로는 path() 등을 이용해 지정하게 된다.

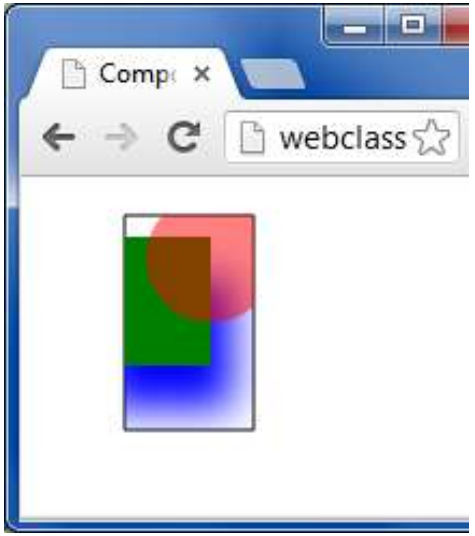
그리기 효과 예제

```
1  <script type="text/javascript">
2    var canvas = document.getElementById("myCanvas");
3    var context = canvas.getContext("2d");
4
5    context.beginPath();
6    context.rect(40, 10, 60, 100);
7    context.closePath();
8    context.stroke();
9    context.clip();
10
11   context.beginPath();
12   context.rect(20, 20, 60, 60);
13   context.fillStyle = "green";
14   context.shadowColor = "blue";
15   context.shadowBlur = 30;
16   context.shadowOffsetX = 10;
17   context.shadowOffsetY = 20;
18   context.fill();
19
20   context.beginPath();
21   context.arc(80, 30, 30, 0, 2*Math.PI);
22   context.fillStyle = "red";
23   context.globalAlpha = 0.5;
24   context.shadowColor = "transparent";
25   context.fill();
26 </script>
```

- 클립 효과 사용시 유의 사항
  - 잘라내고자 하는 그림을 그리기 이전에 clip() 메서드를 실행해 함
  - clip() 메서드 실행 이전에 그려진 그림은 자르기 효과가 적용 안됨



(a) clip() 메소드를 실행하지 않은 경우



(b) clip() 메소드를 실행한 경우

도형간 연산

- 연속해서 그려 넣는 도형간의 연산을 통해 합성을 수행
  - globalCompositeOperation 속성값을 이용
    - 본 속성값을 지정한 시점의 전과 후에 대해 연산을 수행
    - 속성 지정 이전이 source, 속성 지정 이후가 destination
    - 기본 연산: source-over 연산

캔버스 컨텍스트 속성	가능한 속성 값
context.globalCompositeOperation	'source-atop' 'source-in' 'source-out' 'source-over' 'destination-atop' 'destination-in' 'destination-out' 'destination-over' 'lighter' 'darker' 'xor' 'copy';

변환 효과

- 변환(transformation) 효과
  - 주로 그림을 그려넣을때 위치 이동, 회전, 대칭 등의 기능을 수행

캔버스 컨텍스트 속성 및 메소드	기능	기능 및 설명
context.translate(x, y);	이동 변환	기준좌표를 (x, y) 만큼 이동시켜 도형이나 그림의 위치를 이동시킨다.
context.scale(x, y);	크기 변환	도형의 크기를 조절한다. 가로 세로 방향의 배율을 (x, y)값으로 조절가능하며 (1, 1)이 기준 값이며 1보다 크면 도형의 크기가 커지며 1보다 작은 값으로 설정하면 작아지게 된다.
context.rotate(회전각도);	회전 변환	도형과 그림을 회전시켜 그려 넣는다. 회전각도는 라디안(radian) 값으로 지정한다. 360°는 2π 즉 2*Math.PI 로 지정할 수 있다. 회전하는 중심점은 context의 왼쪽 위 모서리이다.

상하/좌우 대칭 변환

- scale() 메소드의 인자 값을 조정하여 구현

```
// 좌우 대칭
context.scale(-1,1);
// 상하 대칭
context.scale(1,-1);
```

사용자 정의 변환

- transform() 메소드

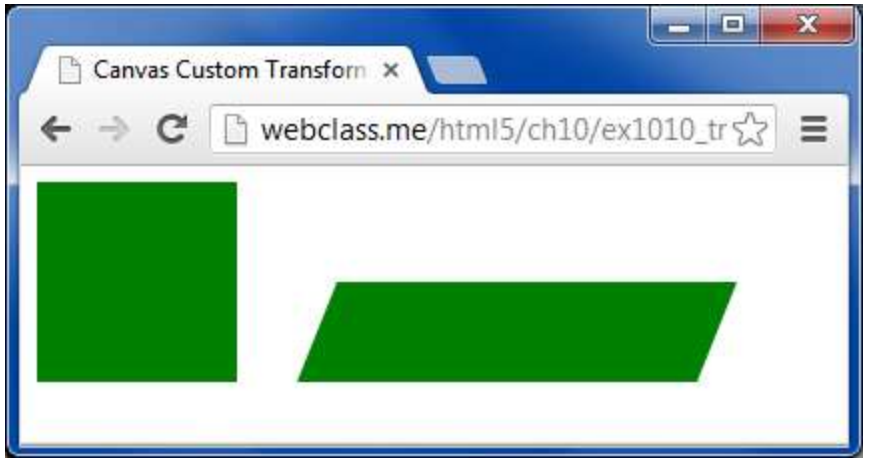
```
context.transform(a, b, c, d, e, f);
```

- 임의의 사용자 정의 변환 행렬을 지정
  - $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$
    - x, y는 변환되기 이전 좌표
    - x', y'는 사용자 정의 변환에 의해 변환된 이후의 좌표 값

사용자 정의 변환 예제

- 변환식
  - 크기변환
    - $x' = 2x + 0.5y, y' = 0.5y$
  - 이동 변환
    - (150, 50) 만큼 이동
- 유의 사항
  - 그림을 그려 넣기 전에 transform() 메소드를 실행해야 함

```
1  // original drawing
2    context.rect(0, 0, 100, 100);
3    context.fillStyle = "green";
4    context.fill();
5
6    // drawing after custom transformation
7    context.transform(2, 0, -0.2, 0.5, 150, 50);
8    context.rect(0, 0, 100, 100);
9    context.fill();
```



변환의 초기화

- 현재까지 지정된 변환이나 사용자 정의 변환 행렬을 초기화
  - setTransform() 메소드 이용
  - 아무런 변환을 지정하지 않은 기본 상태로 초기화

```
context.setTransform(1, 0, 0, 1, 0, 0);
```

픽셀 데이터 접근하기

- 캔버스에 그려진 그림의 픽셀 데이터에 접근
  - getImageData() 메소드
    - 리턴 객체의 data 속성에 각 픽셀 별 데이터가 1차원 배열로 저장되어 있음
      - data 속성 배열의 길이는 캔버스의 가로*세로*4
    - 픽셀의 구성: red, green, blue, alpha 값의 4개의 요소

```
var imgData= context.getImageData(0, 0, canvas.width, canvas.height);
var data = imgData.data;
```

  - putImageData() 메소드
    - 픽셀 데이터의 값을 수정 후 다시 컨텍스트에 반영

```
context.putImageData(imgData, 0, 0);
```

데이터 URL로 저장하기

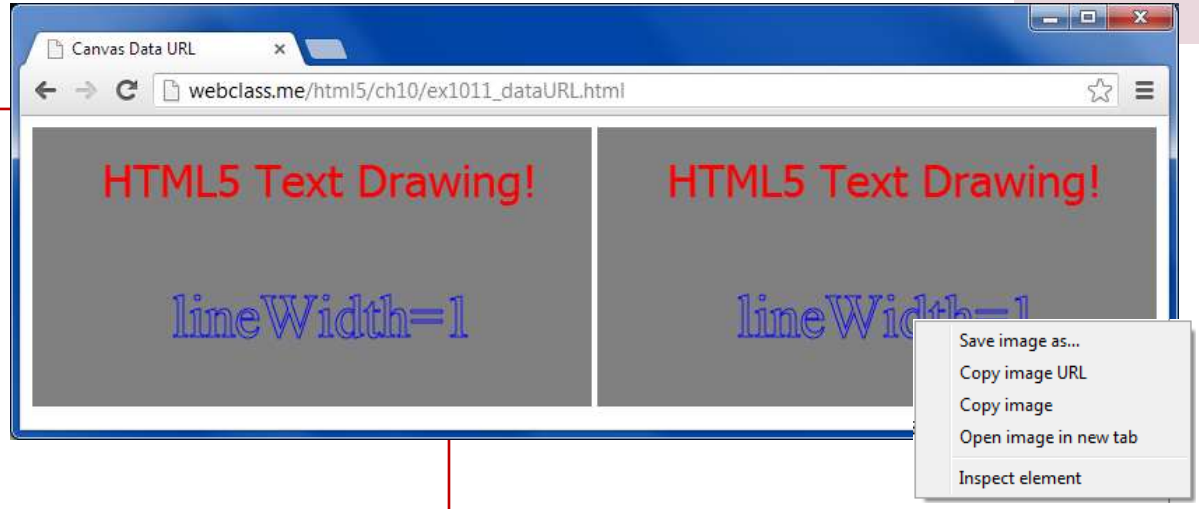
- 그림을 PNG(Portable Network Graphics) 등의 형식으로 저장
  - 캔버스의 toDataURL() 메소드 이용
    - 그림을 toDataURL() 메소드를 이용해서 PNG 형태의 데이터로 변환
    - 이를 캔버스 요소의 src 속성으로 지정하면 파일로 저장이 가능

```
var dataURL = canvas.toDataURL();
canvasDom.src = dataURL;
```

  - 유의 사항
    - **toDataURL()** 메소드는 캔버스 컨텍스트의 메소드가 아닌 캔버스 객체의 메소드

데이터 URL 저장 예제

```
1   context.rect(0, 0, 400, 200);
2   context.fillStyle = "grey";
3   context.fill();
4
5   var text1 = "HTML5 Text Drawing!";
6
7   context.font = "24pt Tahoma";
8   context.fillStyle = "red";
9   context.fillText(text1, 50, 50);
10
11  context.lineWidth = 1;
12  context.font = "32pt San Serif";
13  context.strokeStyle = "blue";
14  context.strokeText("lineWidth=1", 100, 150);
15
16  // Canvas 이미지를 data URL로 저장한다. 기본 형식은 PNG 포맷이다.
17  var dataURL = canvas.toDataURL();
18
19  // dataURL을 "canvasImage" 엘리먼트의 src 속성으로 지정하여
20  // 마우스 오른쪽 버튼을 이용하여 PNG file로 저장될 수 있도록 한다.
21  document.getElementById("canvasImage").src = dataURL;
```



캔버스 이미지 (왼쪽)과 데이터 URL 방식으로 저장한 PNG 이미지 (오른쪽)

캔버스 비트맵 초기화

- 캔버스 비트맵을 초기화할 수 있는 가장 간편한 방법은 clearRect() 메소드를 이용
  - (x, y) 위치를 기준으로 width, height의 폭과 높이의 비트맵을 초기화 한다

```
context.clearRect(x, y, width, height);
```