



Canvas

Canvas



자바스크립트를 이용해 웹 문서상 그래픽 시각화

- HTML5 이전
 - > 직접 이미지 파일을 태그를 이용해 문서상 포함
 - > 자바 애플릿 이용
 - > 플래시 이용
- HTML5 Canvas
 - > 자바스크립트 만 이용해 그래픽 시각화 및 처리 가능
 - > 별도 플러그인이나 프로그램 설치 없이 가능
 - > 이미지나 그림 합성 or 변환 조작도 가능

Canvas



What?

- 웹 상에서 그림을 그리기 위한 API
- Apples 사의 Mac OS Dashboard에 내장된 '위젯'이라고 부르는 간단한 애플리케이션을 위해 개발됨
- 이 후 Safari / Firefox를 지원하여 HTML5 API로 채택
- JavaScript로 이미지를 그림
- Path라는 경로를 설정하고 그것을 따라가는 형태로 선을 그리거나 색을 칠함
- 이미지도 붙여 넣을 수 있음

Canvas



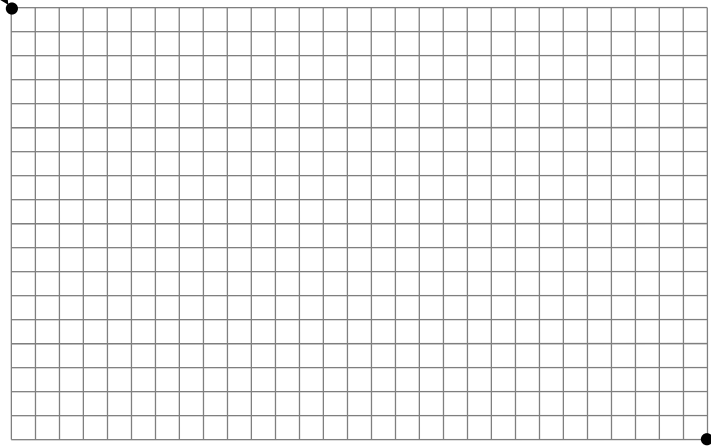
사각 평면의 2차원 좌표계 사용

- 2D 이미지 표현
- X, Y 축으로 구성
- 왼쪽 상단 모서리 원점

(0, 0)

x 축

y 축



(30, 20)



픽셀

- 좌표계 상 각각의 정사각형 네모칸
- 이미지를 구성하는 점 색상 가짐
(색상 값을 바꿔 다양한 이미지 표현)



비트맵 그래픽

- 픽셀만으로 이미지를 표현하고 저장하는 형태

Canvas



Hello 예제

```
<!DOCTYPE html>
<html>
<head> <title></title> </head>
<body>
<canvas id="canvas" width="300" height="300"></canvas>
<script type="text/javascript">
    var canvas = document.getElementById("canvas");
    var myContext = canvas.getContext("2d");
</script>
</body>
</html>
```

Canvas



캔버스 요소

- <canvas> 태그 이용 요소 추가
- width / height 속성 이용 캔버스 좌표계 크기 지정
- DOM을 통한 접근을 위해 id 지정

```
<canvas id="cvs" width="300" height="300"></canvas>
```



컨텍스트 객체

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

```
var wpcanvas = document.getElementById("cvs");  
var wpcontext = wpcanvas.getContext("2d");
```

Canvas



도형 그리기 관련 메소드

- **lineTo(x,y)**
- **beginPath()**
- **closePath()**
- **moveTo(x,y)**
- **rect(x,y,w,h)**
- **fillRect(x,y,w,h)**
- **strokeRect(x,y,w,h)**
- **clearRect(x,y,w,h)**
- **fill()**
- **stroke()**
- **arcTo(x1,y1,x2,y2,r)**
- **arc(x,y,r,sAngle,eAngle,aClockwise)**
- **quadraticCurveTo(cpx,cpy,x,y)**
- **bezierCurveTo(cpx1,cpy1,cpx2,cpy2,x,y)**
- **isPointInPath(x,y)**
- **clip()**

Canvas 기본 도형 그리기



캔버스 기본 API

- <https://developer.mozilla.org/ko/docs/Web/HTML/Canvas>
- 컨텍스트 객체 메소드를 호출함으로써 그림이 그려짐



선 그리기 메소드

- 선 굵기, 경로, 곡선 등
- 현재 시작 지점에서 다음 지점까지 선 연결하는 방식

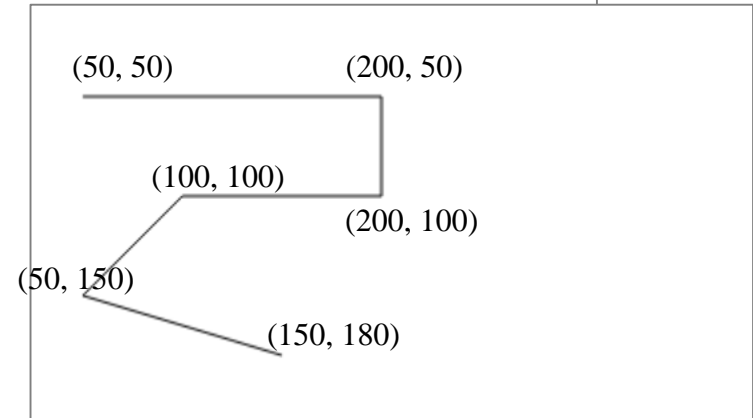
캔버스 컨텍스트 메소드	기능 및 설명
<code>myContext.moveTo(x, y)</code>	선의 시작 지점을 (x, y) 좌표로 이동시킨다.
<code>myContext.lineTo(x, y)</code>	현재의 시작점에서 (x, y) 지점까지 선을 그린다.
<code>myContext.rect(x, y, width, height);</code>	왼쪽 위 모서리를 (x, y) 지점으로 하고 가로와 세로 변의 크기가 각각 width, height인 사각형을 그린다. 현재의 시작점을 (x, y)로 이동시킨다.
<code>myContext.stroke();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

Canvas 기본 도형 그리기



직선 그리기 예제

```
<!DOCTYPE html>
<html>
<head> <title></title> </head>
<body>
<canvas id="myCanvas" width="300" height="200"></canvas>
<script type="text/javascript">
var canvas = document.getElementById("myCanvas");
var myContext = canvas.getContext("2d");
  myContext.moveTo(50, 50);
  myContext.lineTo(200, 50);
  myContext.lineTo(200, 100);
  myContext.lineTo(100, 100);
  myContext.lineTo(50, 150);
  myContext.lineTo(150, 180);
  myContext.stroke();
</script>
</body>
</html>
```



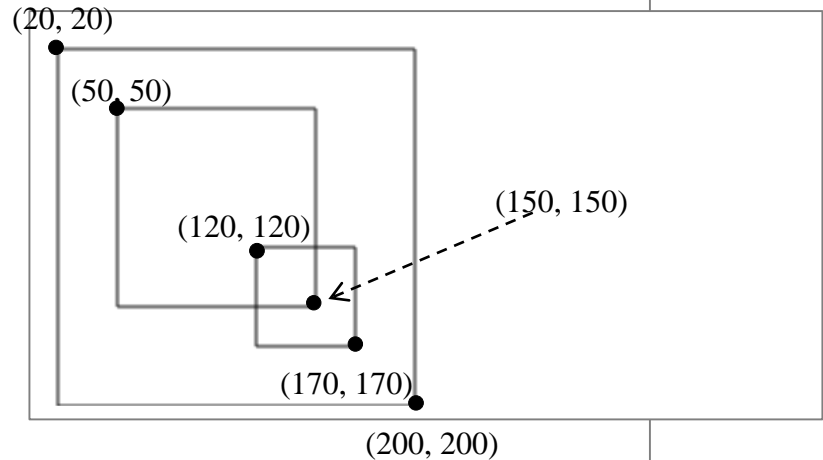
Canvas 기본 도형 그리기



사각형 그리기 예제

```
<!DOCTYPE html>
<html>
<head> <title></title> </head>
<body>
<canvas id="myCanvas" width="300" height="200"></canvas>
<script type="text/javascript">
var canvas = document.getElementById("myCanvas");
var myContext = canvas.getContext("2d");

myContext.rect(50, 50, 100, 100);
myContext.rect(20, 20, 180, 180);
myContext.rect(120, 120, 50, 50);
myContext.stroke();
</script>
</body>
</html>
```





연속된 선 그리기를 통한 경로 그리기

- `beginPath()`
> 경로 시작 설정
- `closePath()`
> 경로 지정 종료
> 처음 경로 시작 지점으로 선을 연결하여 경로를 완성

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



연속된 선 그리기를 통한 경로 그리기

```
<canvas id="myCanvas" width="300" height="200"></canvas>
```

```
<script type="text/javascript">
```

```
var canvas = document.getElementById("myCanvas");
```

```
var myContext = canvas.getContext("2d");
```

```
myContext.beginPath();
```

```
myContext.moveTo(50, 50);
```

```
myContext.lineTo(200, 50);
```

```
myContext.lineTo(200, 100);
```

```
myContext.closePath();
```

```
myContext.fill();
```

```
myContext.beginPath();
```

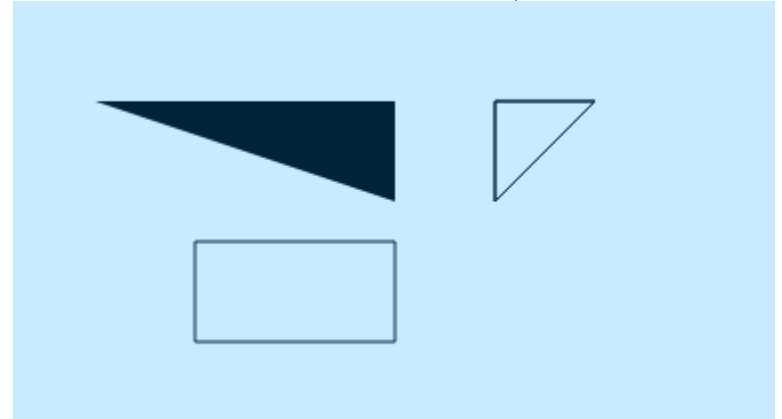
```
myContext.moveTo(250, 50);
```

```
myContext.lineTo(250, 100);
```

```
myContext.lineTo(300, 50);
```

```
myContext.closePath();
```

```
myContext.stroke();
```



```
myContext.rect(100, 120, 100, 50);
```

```
myContext.stroke();
```

```
</script>
```

```
</body>
```

```
</html>
```



캔버스 비트맵 초기화

- `clearRect()` 메소드 이용
> (x, y) 위치를 기준으로 width, height의 폭과 높이의 비트맵을 초기화함

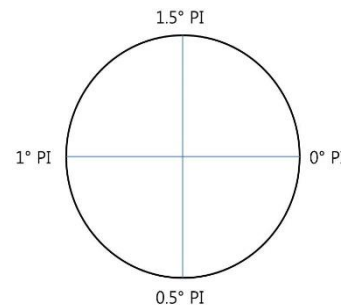
```
myContext.clearRect(x, y, width, height);
```



원호와 곡선 그리기

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

- **stroke() 메소드를 호출하지 않으면 실제로 캔버스에 선이 그려지지 않음에 유의**
- **Angle, radian = $(\text{Math.PI}/180) * \text{degree}$**

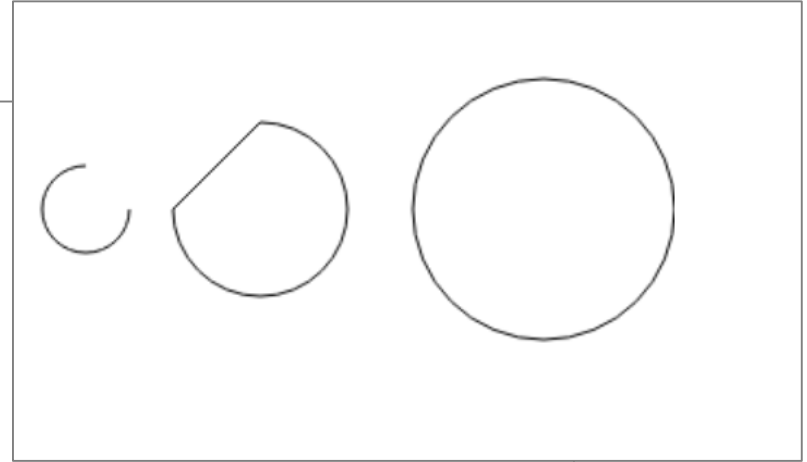


Canvas



원호 그리기 예제

```
...  
<script type="text/javascript">  
var canvas = document.getElementById("myCanvas");  
var canvas = document.getElementById("myCanvas");  
var myContext = canvas.getContext("2d");  
  
myContext.beginPath();  
myContext.arc(30, 100, 20, 0, 1.5*Math.PI); // Math.PI 상수를 이용해 각도지정  
myContext.stroke();  
  
myContext.beginPath();  
myContext.arc(110, 100, 40, 1*Math.PI, 1.5*Math.PI, true); // 반시계방향 원호  
myContext.closePath(); // 경로 시작점까지 직선으로 연결하며 경로를 종료한다.  
myContext.stroke();  
  
myContext.beginPath();  
myContext.arc(240, 100, 60, 0, 2*Math.PI); // 360도 원호를 그려 원 그리기  
myContext.stroke();  
</script>...
```



Canvas



곡선 그리기 예제

...

```
<script type="text/javascript">
```

```
var canvas = document.getElementById("myCanvas");
```

```
var myContext = canvas.getContext("2d");
```

```
myContext.moveTo(50, 200);
```

```
myContext.quadraticCurveTo(100, 10, 200, 200);
```

```
myContext.stroke();
```

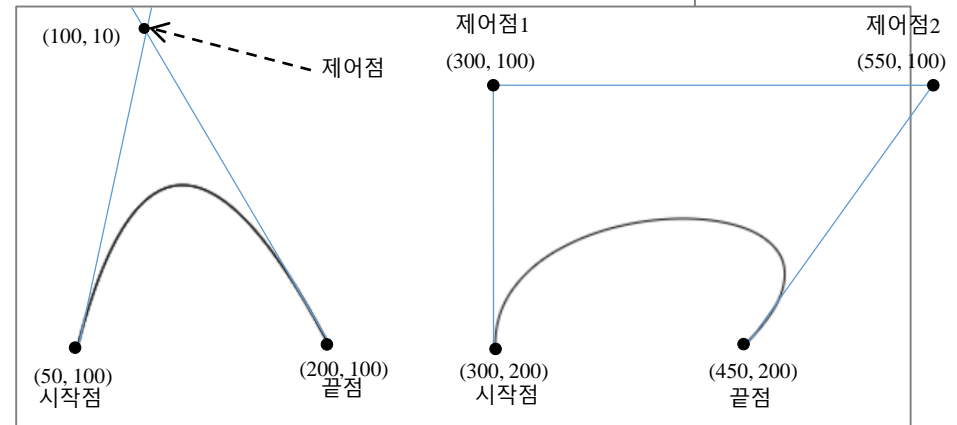
```
myContext.moveTo(300, 200);
```

```
myContext.bezierCurveTo(300, 100, 600, 100, 450, 200);
```

```
myContext.stroke();
```

```
</script>
```

...



Quadratic 곡선

Bezier 곡선

Canvas 기본 도형 꾸미기



선 꾸미기와 색칠하기

캔버스 컨텍스트 속성 및 메소드	기능 및 설명
<code>myContext.lineWidth</code>	선의 두께를 픽셀 개수로 설정한다.
<code>myContext.strokeStyle</code>	선의 색상을 지정한다. 색상을 지정하는 방법은 일반적인 웹 문서에서와 동일하다. 예) "blue" 혹은 "#0000ff" 등
<code>myContext.lineCap</code>	선의 양쪽 끝 모양을 지정한다. 지정할 수 있는 형태는 "butt", "round", "square"이며 기본 값은 "butt"이다.
<code>myContext.lineJoin</code>	선이 꺾이는 모서리 지점에서의 모양을 지정한다. "miter", "round", "bevel" 세 가지 중의 한가지 값으로 지정할 수 있다. 기본값은 "miter" 스타일이다.
<code>myContext.fillStyle</code>	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
<code>myContext.fill();</code>	현재 지정된 <code>fillStyle</code> 색상으로 도형을 채운다. 색칠할 도형은 <code>fill()</code> 메소드를 실행하기 이전에 그려지 모든 도형들이다.

Canvas 기본 도형 꾸미기



선 꾸미기 예제

...

```
<script type="text/javascript">
```

```
myContext.beginPath();
```

```
myContext.moveTo(10, 20);
```

```
myContext.lineTo(100,20);
```

```
myContext.lineWidth = 20;
```

```
myContext.strokeStyle = "blue";
```

```
myContext.lineCap = "butt";
```

```
myContext.stroke();
```

```
myContext.beginPath();
```

```
myContext.moveTo(10, 60);
```

```
myContext.lineTo(100, 60);
```

```
myContext.strokeStyle = "red";
```

```
myContext.lineCap = "round";
```

```
myContext.stroke();
```

```
myContext.beginPath();
```

```
myContext.moveTo(10, 100);
```

```
myContext.lineTo(100, 100);
```

```
myContext.strokeStyle = "green";
```

```
myContext.lineCap = "square";
```

```
myContext.stroke();
```

```
</script>...
```



Canvas 기본 도형 꾸미기



도형 꾸미기 예제

...

```
<script type="text/javascript">
```

```
myContext.beginPath();
```

```
myContext.rect(20, 20, 50, 80);
```

```
myContext.lineWidth = 20;
```

```
myContext.strokeStyle = "black";
```

```
myContext.lineJoin = "miter";
```

```
myContext.fillStyle = "grey";
```

```
myContext.stroke();
```

```
myContext.fill();
```

```
myContext.beginPath();
```

```
myContext.rect(110, 20, 50, 80);
```

```
myContext.strokeStyle = "black";
```

```
myContext.lineJoin = "round";
```

```
myContext.fillStyle = "pink";
```

```
myContext.stroke();
```

```
myContext.fill();
```

```
myContext.beginPath();  
myContext.rect(200, 20, 50, 80);  
myContext.strokeStyle = "black";  
myContext.lineJoin = "bevel";  
myContext.fillStyle = "yellow";  
myContext.stroke();  
myContext.fill();  
</script>...
```



Canvas 이미지 그리기



캔버스 이미지?

- HTML 태그 사용 “” 태그 이용하지만 캔버스를 사용하여 이미지를 그릴 수 있음



방법

- Image 객체 이용
 - > Image() 생성자 이용하여 생성

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

- drawImage() 메소드
 - > 캔버스 컨텍스트에서 이미지를 그리는 메소드
 - > 사이즈 조정, 크롭 등 기능 가능

Canvas 이미지 그리기



캔버스 이미지 예제

...

```
<script type="text/javascript">
```

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

```
imgObj.src = "clownfish.jpg";
```

```
imgObj.onload = function() {
```

```
    // (50, 50) 지점에 원래 크기 그대로 이미지 그리기  
    myContext.drawImage(imgObj, 50, 50);
```

```
    // 크기조정하기: (350, 50) 지점에 150 x 100 크기로 이미지 그리기  
    myContext.drawImage(imgObj, 350, 50, 150, 100);
```

```
    // 크기조정하기: (550, 50) 지점에 100 x 100 크기로 이미지 그리기  
    myContext.drawImage(imgObj, 550, 50, 100, 100);
```

```
    // 이미지 자르기 후 크기조정:
```

```
    // 1) 원본 이미지 (70, 50) 지점에서 70 x 70 크기의 이미지를 자른다.
```

```
    // 2) Canvas의 (350, 180) 지점에 70 x 70 크기로 그리기
```

```
    myContext.drawImage(imgObj, 70, 50, 70, 70, 350, 180, 70, 70);
```

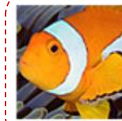
```
    // 이미지 자르기 후 크기조정:
```

```
    // 1) 원본 이미지 (120, 50) 지점에서 100 x 50 크기의 이미지를 자른다.
```

```
    // 2) Canvas의 (450, 180) 지점에 125 x 75 크기로 그리기
```

```
    myContext.drawImage(imgObj, 120, 50, 100, 50, 450, 180, 100, 50);
```

```
} ...
```



이미지 크롭

이미지 사이즈
조정

Canvas 글자 그리기



캔버스 글자?

- 비트맵 방식으로 글자를 그릴 수 있음
- 삽입된 글자를 수정하거나 크기 조정 불가능
- 그려 넣기 전 폰트, 크기, 정렬방법 등 결정



방법

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

Canvas 글자 그리기



색상 및 외곽선 두께 등 지정 가능

캔버스 컨텍스트 속성 및 메소드

기능 및 설명

`myContext.strokeStyle = 색상값;`

글자의 외곽선을 그릴 색상을 지정한다. 색상을 지정하는 방법은 일반적인 웹 문서에서와 동일하다. 예) "blue" 혹은 "#0000ff" 등

`myContext.lineWidth = 선두께;`

글자의 외곽선을 그릴 선의 두께를 지정한다.

`myContext.strokeText(text, x, y);`

현재 지정된 `strokeStyle` 색상으로 캔버스의 (x, y) 위치에 `text`의 외곽선을 그려 넣는다. 글자의 외곽선만 그려지게 되므로 내부가 비어있는 형태가 된다. 텍스트의 왼쪽 위 모서리 지점이 텍스트를 그려넣는 기준점이 된다.

Canvas 글자 그리기



캔버스 글자 예제

...

```
<script type="text/javascript">  
myContext.rect(0, 0, 400, 300);  
myContext.stroke();
```

```
var text1 = "HTML5 Text Drawing!";  
var text2 = "Left aligned text";  
var text3 = "Center aligned text";  
var text4 = "Right aligned text";
```

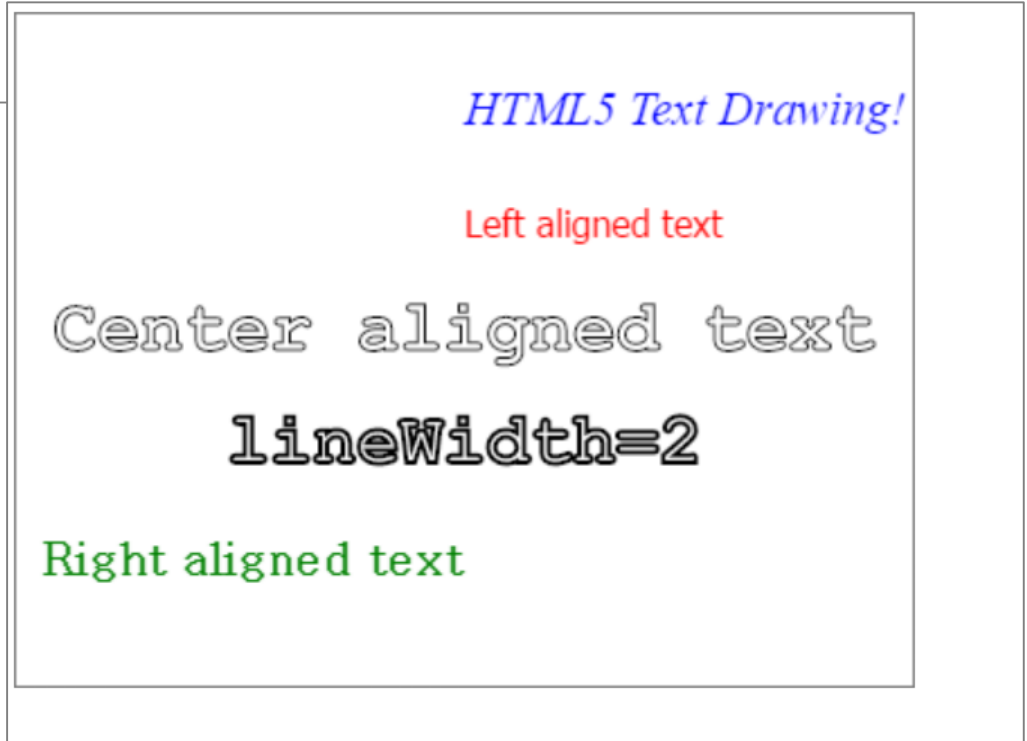
```
myContext.font = "italic 16pt Times New Roman";  
myContext.fillStyle = "blue";  
myContext.fillText(text1, 200, 50);
```

```
myContext.font = "12pt Tahoma";  
myContext.fillStyle = "red";  
myContext.textAlign = "left";  
myContext.fillText(text2, 200, 100);
```

```
myContext.font = "bold 24pt Courier New";  
myContext.strokeStyle = "black";  
myContext.textAlign = "center";  
myContext.lineWidth = 1;
```

```
myContext.strokeText(text3, 200, 150);  
myContext.lineWidth = 2;  
myContext.strokeText("lineWidth=2", 200, 200);
```

```
myContext.font = "bold 16pt Batang";  
myContext.fillStyle = "green";  
myContext.textAlign = "right";  
myContext.fillText(text4, 200, 250);
```



Canvas 고급 기능 (그리기 효과)



합성 효과

- 그림자 효과를 줄 수 있는 shadow
- 투명도 조절을 위한 globalAlpha
- 지정한 도형 모양으로 잘라내는 clip

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

Canvas 고급 기능 (그리기 효과)



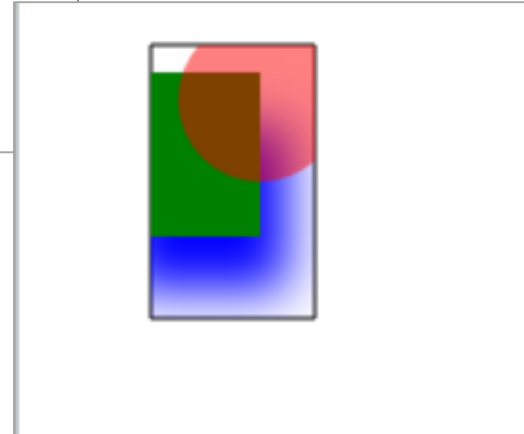
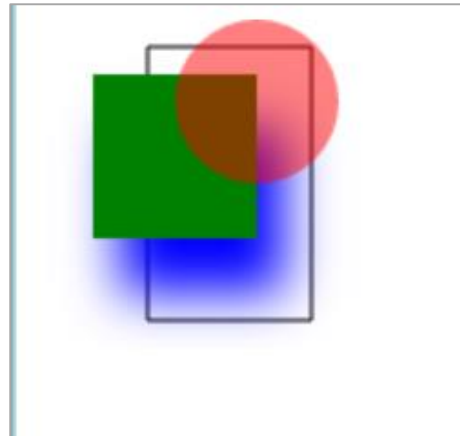
캔버스 그리기 효과 예제

...

```
<script type="text/javascript">  
var canvas = document.getElementById("myCanvas");  
var myContext = canvas.getContext("2d");
```

```
myContext.beginPath();  
myContext.rect(40, 10, 60, 100);  
myContext.closePath();  
myContext.stroke();  
myContext.clip();
```

```
myContext.beginPath();  
myContext.rect(20, 20, 60, 60);  
myContext.fillStyle = "green";  
myContext.shadowColor = "blue";  
myContext.shadowBlur = 30;  
myContext.shadowOffsetX = 10;  
myContext.shadowOffsetY = 20;  
myContext.fill();
```



```
myContext.beginPath();  
myContext.arc(80, 30, 30, 0, 2*Math.PI);  
myContext.fillStyle = "red";  
myContext.globalAlpha = 0.5;  
myContext.shadowColor = "transparent";  
myContext.fill();  
</script>
```

...

Canvas 고급 기능 (변환 효과)

기본 효과

- 그림을 그려넣을 때 위치 이동, 크기 변환, 회전 등 기능 수행

캔버스 컨텍스트 속성 및 메소드	기능	기능 및 설명
<code>myContext.translate(x, y);</code>	이동 변환	기준좌표를 (x, y) 만큼 이동시켜 도형이나 그림의 위치를 이동시킨다.
<code>myContext.scale(x, y);</code>	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.
<code>myContext.rotate(회전각도);</code>	회전 변환	도형과 그림을 회전시켜 그려 넣는다. 회전각도는 라디안(radian) 값으로 지정한다. 360° 는 2π 즉 $2 * \text{Math.PI}$ 로 지정할 수 있다. 회전하는 중심점은 context의 왼쪽 위 모서리이다.

Canvas 고급 기능 (변환 효과)



캔버스 기본 변환 예제

...

```
<script type="text/javascript">
```

```
var canvas = document.getElementById("myCanvas");
```

```
var myContext = canvas.getContext("2d");
```

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

```
imgObj.src = "clownfish.jpg";
```

```
imgObj.onload = function() {
```

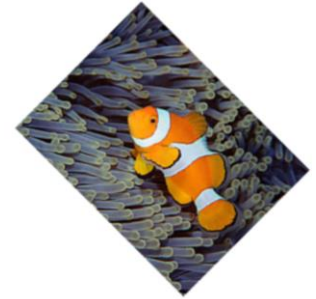
```
    // original drawing
```

```
    myContext.drawImage(imgObj, 50, 50);
```

```
    // 이동변환
```

```
    myContext.translate(300, 0); // (300, 50) 이동
```

```
    myContext.drawImage(imgObj, 50, 50);
```



```
// 크기변환
```

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

```
myContext.scale(1.2, 0.33); // 가로 1.2, 세로 1/3배 스케일
```

```
myContext.drawImage(imgObj, 40, 800);
```

```
// 회전변환
```

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

```
myContext.translate(750, 0);
```

```
myContext.rotate(Math.PI/4);
```

```
myContext.drawImage(imgObj, 0, 0);
```

```
}
```

```
...
```

Canvas 고급 기능 (변환 효과)



상하/좌우 대칭 변환

- Scale() 메소드의 인자 값 조정하여 구현

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

- 현재까지 지정된 변환 or 사용자 정의 변환 행렬 초기화
 - > setTransform() 메소드
 - > 아무런 변환하지 않은 기본 상태로 초기화

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

Canvas 고급 기능



데이터 URL로 저장

- Canvas 그림을 PNG 등 형식으로 저장
> toDataURL() 메소드 이용

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

- toDataURL() 메소드는 캔버스 컨텍스트 메소드가 아닌 캔버스 요소 객체 메소드

Canvas 고급 기능



데이터 URL로 저장

...

```
<script type="text/javascript">  
myContext.rect(0, 0, 400, 200);  
myContext.fillStyle = "grey";  
myContext.fill();
```

```
var text1 = "HTML5 Text Drawing!";
```

```
myContext.font = "24pt Tahoma";  
myContext.fillStyle = "red";  
myContext.fillText(text1, 50, 50);
```

```
myContext.lineWidth = 1;  
myContext.font = "32pt San Serif";  
myContext.strokeStyle = "blue";  
myContext.strokeText("lineWidth=1", 100, 150);
```

// Canvas 이미지를 data URL로 저장한다. 기본 형식은 PNG 포맷이다.

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

// dataURL을 "canvasImage" 엘리먼트의 src 속성으로 지정하여

// 마우스 오른쪽 버튼을 이용하여 PNG file로 저장될 수 있도록 한다.

```
document.getElementById("canvasImage").src = dataURL;
```

HTML5 Text Drawing!

lineWidth=1

HTML5 Text Drawing!

Open image in new tab
Save image as...
Copy image
Copy image address
Search Google for image

Inspect

Ctrl+Shift+I

Canvas 마우스 그림 그리기 예제



캔버스 상에 마우스 이벤트를 활용해 그림 그리기

- 마우스 버튼을 누른 후 이동 위치를 따라 연속으로 직선을 그림
- mousemove 이벤트가 발생할 때마다 lineto() 메소드 호출

```
<body onmousedown="start();" onmousemove="draw();" onmouseup="stop();">  
<canvas id="myCanvas" width="600" height="600" style="border-style: solid;"></canvas>
```

```
<script type="text/javascript">
```

(중간 생략)

```
var stopped = true;
```

```
function start() {
```

```
    e = window.event;
```

```
    myContext.moveTo(e.clientX-10, e.clientY-10);
```

```
    stopped = false;
```

```
}
```

(중간 생략)

```
function draw() {
```

```
    if (!stopped) {
```

```
        e = window.event;
```

```
        myContext.lineTo(e.clientX-10, e.clientY-10);
```

```
        myContext.stroke();
```

```
    }
```

```
}
```

```
</script>
```

```
</body>
```

