

***화면에 그림을 그려보자***

# <canvas> 태그

그래픽이 그려질 영역

속성	설명
id	한 페이지에 여러 개의 캔버스를 그렸을 때 캔버스 구별
width / height	캔버스 크기 지정

<!DOCTYPE HTML>

<html>

<head>

<script type="application/javascript">

function draw\_m( ) {

var m\_canvas = document.getElementById("m");

var m\_ctx = m\_canvas.getContext("2d");

m\_ctx.fillStyle = "rgb(200,0,0)";

m\_ctx.fillRect (10, 10, 100, 100);

}

</script>

</head>

<body onload="draw\_m( )">

<canvas id="m"

width="500" height="300"> </canvas>

</body>

도형을 그리는 함수 정의

ID가 m인 캔버스 객체를 불러와 m\_canvas라는 변수에 저장

드로잉 컨텍스트의 인스턴스 만듦

사각형을 채울 스타일 정의

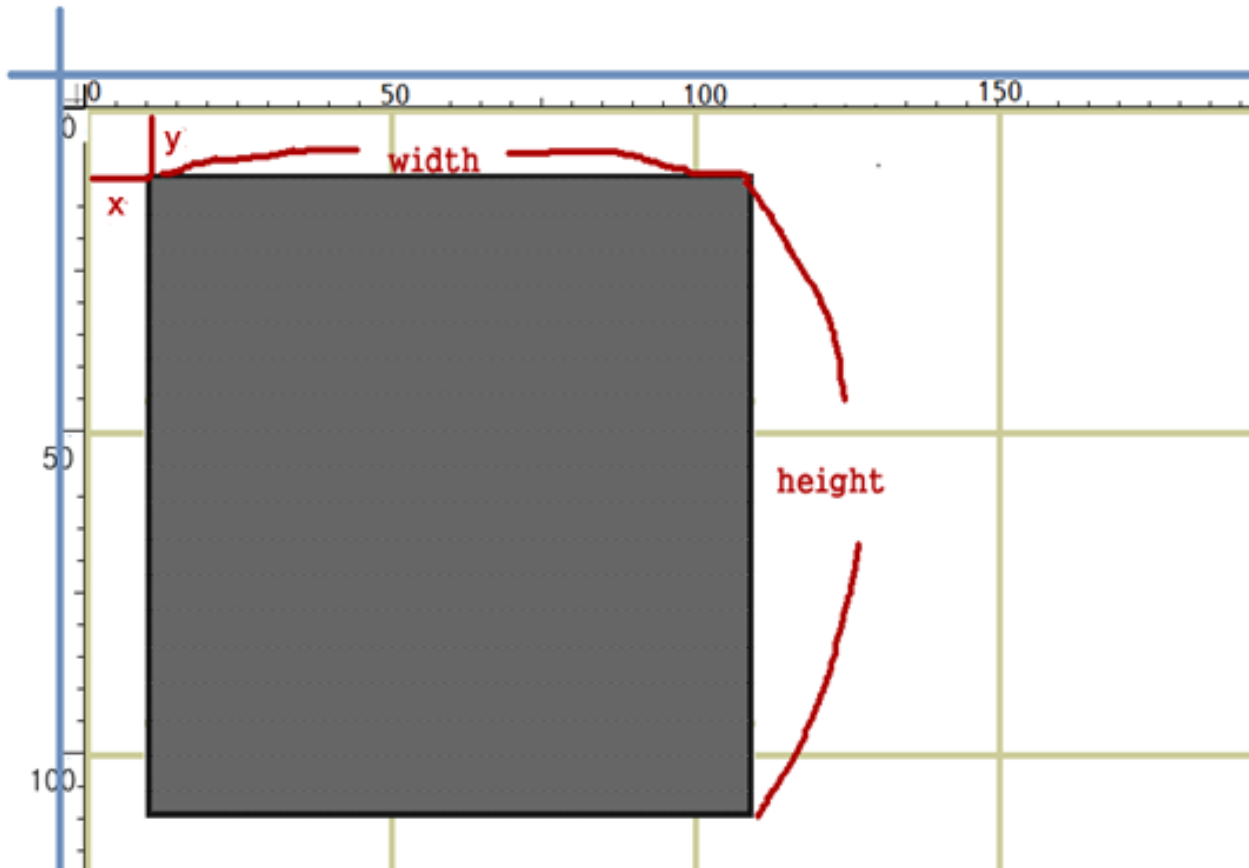
사각형 그림

onload 이벤트와 함께 함수 실행

# fillRect() 함수와 좌표

**fillRect (x, y, width, height);**

→ (x,y)라는 위치에서 시작하여 너비 width, 높이 height인 사각형

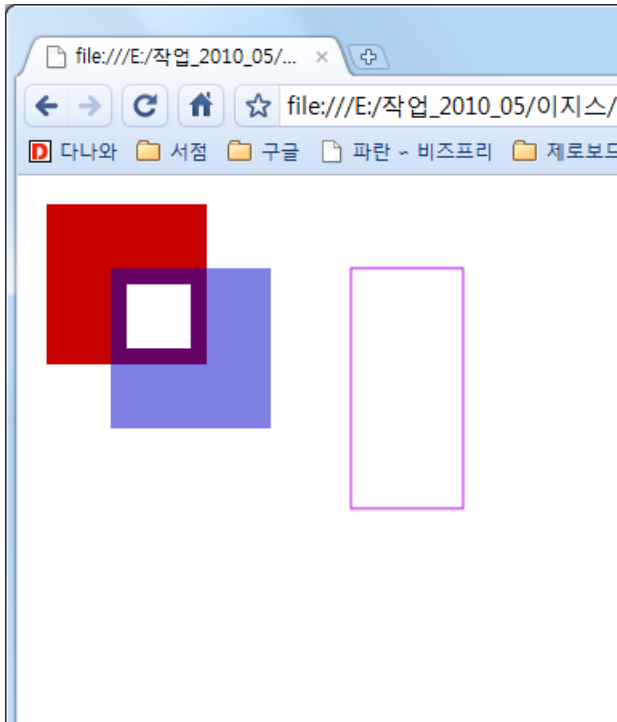


## <canvas> 요소를 지원하지 않는 브라우저를 고려해서

```
<script type="application/javascript">
function draw_m() {
    var canvas = document.getElementById("m");
    if(canvas.getContext) {
        var ctx = canvas.getContext("2d");
        ctx.fillRect (10, 10, 100, 100);
        ctx.fillStyle = "rgb(200,0,0)";
    }
    else {
        alert("사파리 브라우저나 파이어폭스 1.5 이상의 브라우저에서만 사
이트 내용을 제대로 볼 수 있습니다.")
    }
}
```

# 사각형 그리기

- **fillRect(x, y, width, height)** : 색이 칠해진 사각형 그림
- **strokeRect(x, y, width, height)** : 테두리만 있는 사각형 그림
- **clearRect(x, y, width, height)** : 특정 영역을 지우고 완전히 투명하게 만듦



```
function draw_m() {  
    var canvas = document.getElementById("m");  
    if(canvas.getContext) {  
        var ctx = canvas.getContext("2d");  
        ctx.fillStyle = "rgb(200,0,0)";  
        ctx.fillRect (10, 10, 100, 100);  
        ctx.fillStyle = "rgba(0,0,200, 0.5)";  
        ctx.fillRect (50, 50, 100, 100);  
        ctx.clearRect(60, 60, 40, 40);  
        ctx.strokeStyle="rgb(200,0,250)";  
        ctx.strokeRect(200, 50, 70, 150);  
    }  
}
```

# 경로 그리기

**beginPath()** : 경로 시작

**closePath()** : 경로 종료

**stroke()** : 경로 테두리 그리기

**fill()** : 도형 채우기

**moveto(x,y)** : (x,y) 위치로 시작점을 옮김

**lineto(x,y)** : x에서 y까지 직선을 그림

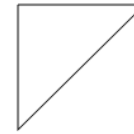
# 경로 그리기

<삼각형을 채울 때>



```
ctx.beginPath();  
ctx.moveTo(50,50);  
ctx.lineTo(150,50);  
ctx.lineTo(50,150);  
ctx.fill();
```

<삼각형의 테두리만 그릴 때>



```
ctx.beginPath();  
ctx.moveTo(50,50);  
ctx.lineTo(150,50);  
ctx.lineTo(50,150);  
ctx.lineTo(50,50);  
ctx.closePath();  
ctx.stroke();
```



# 원, 호 그리기

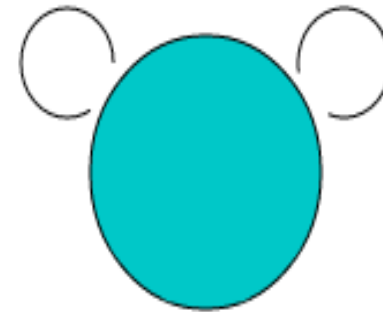
**`arc(x, y, r, startAngle, endAngle, anticlockwise)`**

- (x,y)에서 시작하여 반시계방향(anticlockwise)으로 반지름(r)만큼의 원을 그림.
- `startAngle / endAngle` : 호의 시작점과 끝점을 각도로 표시  
각도는 라디안으로 표시 `var radians = (Math.PI/180)*degrees`
- `Anticlockwise` : 반시계 방향으로 그릴지 여부. `true` 또는 `false`로 지정

```

function draw_m() {
  var canvas = document.getElementById("m");
  if(canvas.getContext){
    var ctx = canvas.getContext('2d');
    ctx.beginPath();
    ctx.arc(70, 70, 20, 0,(Math.PI/180)*60,true);
    ctx.stroke();
    ctx.beginPath();
    ctx.arc(130, 110, 50, 0, Math.PI*2,true);
    ctx.fillStyle="rgb(0,200,200)";
    ctx.fill();
    ctx.stroke();
    ctx.beginPath();
    ctx.arc(190, 70, 20, (Math.PI/180)*110,(Math.PI/180)*170,true);
    ctx.stroke();
    ctx.beginPath();
    ctx.arc(300, 150, 30, 0,(Math.PI/180)*200,false);
    ctx.fillStyle="rgb(0,200,0)";
    ctx.fill();
  } else {
    alert("사파리 브라우저나 파이어폭스 1.5 이상의 브라우저에서만 사이트 내용을 제대로 볼 수 있습니다.")
  }
}

```

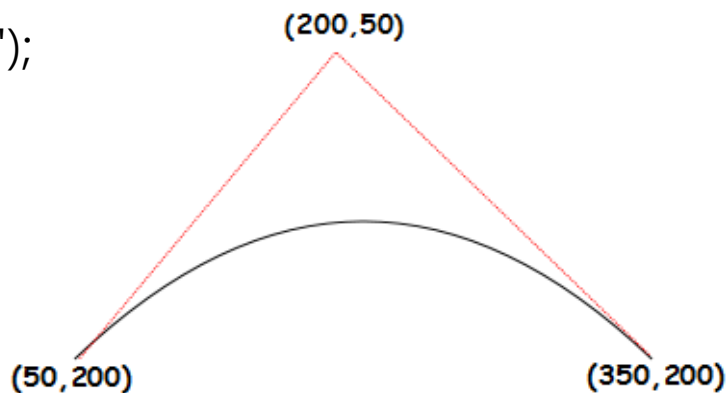


# 베지에 곡선 그리기

**quadraticCurveTo(cp1x, cp1y, x, y)**

한 개의 조절점(cp1x,cp1y)을 이용해 (x,y)까지의 곡선을 그림

```
function draw_m() {  
  var canvas = document.getElementById("m");  
  if(canvas.getContext){  
    var ctx = canvas.getContext('2d');  
    ctx.beginPath();  
    ctx.moveTo(50,200);  
    ctx.quadraticCurveTo(200,50,350,200);  
    ctx.stroke();  
  }  
}
```

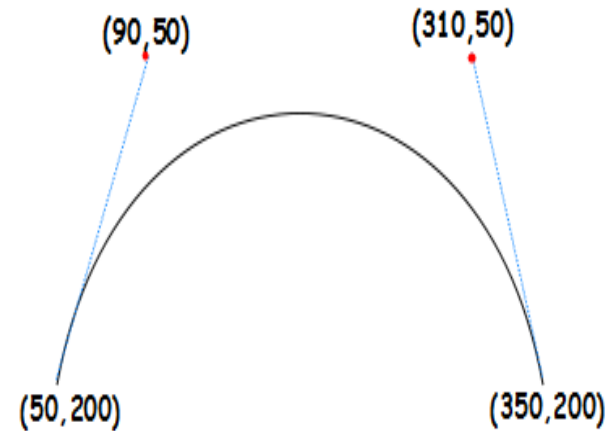


# 베지에 곡선 그리기

**bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y)**

두 개의 조절점(cp1x,cp1y)와 (cp2x,cp2y)를 이용해 (x,y)까지의 곡선을 그림

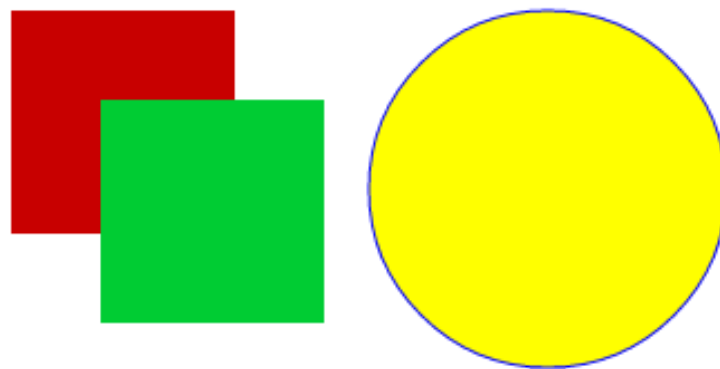
```
function draw_m() {  
  var canvas = document.getElementById("m");  
  if(canvas.getContext){  
    var ctx = canvas.getContext('2d');  
    ctx.beginPath();  
    ctx.moveTo(50,200);  
    ctx.bezierCurveTo(90,50,310,50,350,200)  
    ctx.stroke();  
  }  
}
```



# 스타일 적용하기 - 색상

<b>fillStyle= color,      strokeStyle = color</b>
---

```
var m_canvas = document.getElementById("m");  
if(m_canvas.getContext) {  
    var m_ctx = m_canvas.getContext("2d");  
    m_ctx.fillStyle = "rgb(200,0,0)";  
    m_ctx.fillRect (10, 10, 100, 100);  
    m_ctx.fillStyle = "#0c3";  
    m_ctx.fillRect (50, 50, 100, 100);  
    m_ctx.beginPath();  
    m_ctx.arc(250,90,80,0,Math.PI*2,true);  
    m_ctx.fillStyle="yellow";  
    m_ctx.strokeStyle="blue";  
    m_ctx.fill();  
    m_ctx.stroke();  
}
```

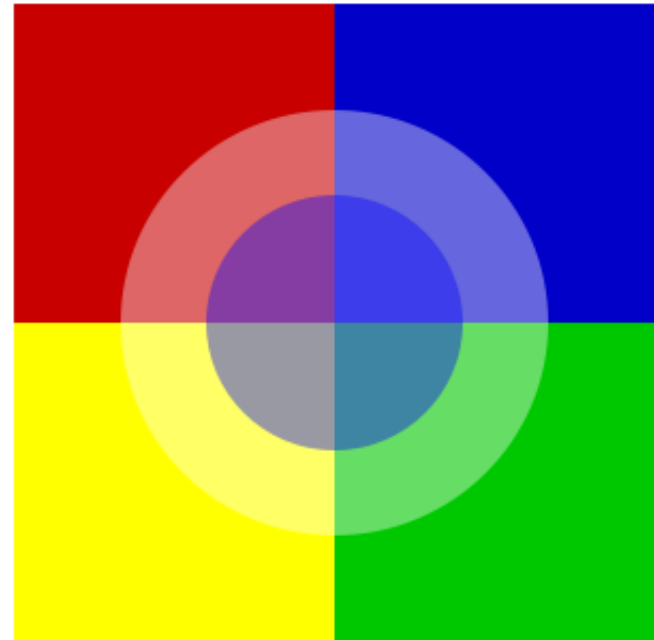


# 스타일 적용하기 - 투명도

**globalAlpha = value**

사용할 수 있는 값 : 0.0 ~ 1.0, 기본값 1.0

```
var m_canvas = document.getElementById("m");  
if(m_canvas.getContext) {  
    var m_ctx = m_canvas.getContext("2d");  
    m_ctx.fillStyle = "rgb(200,0,0)";  
    m_ctx.fillRect (10, 10, 150, 150);  
    m_ctx.fillStyle = "rgb(0,0,200)";  
    m_ctx.fillRect (160, 10, 150, 150);  
    m_ctx.fillStyle = "rgb(0,200,0)";  
    m_ctx.fillRect (160, 160, 150, 150);  
    m_ctx.fillStyle = "yellow";  
    m_ctx.fillRect (10, 160, 150, 150);  
    m_ctx.globalAlpha=0.4;  
    m_ctx.beginPath();  
    m_ctx.arc(160,160,100,0,Math.PI*2,true);  
    m_ctx.fillStyle="white";  
    m_ctx.fill();  
    m_ctx.beginPath();  
    m_ctx.arc(160,160,60,0,Math.PI*2,true);  
    m_ctx.fillStyle="blue";  
    m_ctx.fill(); } }
```



# 스타일 적용하기 - 투명도

**globalAlpha = value**

사용할 수 있는 값 : 0.0 ~ 1.0, 기본값 1.0

```
var m_canvas = document.getElementById("m");  
if(m_canvas.getContext) {  
    var m_ctx = m_canvas.getContext("2d");  
  
    m_ctx.fillStyle = "rgba(255,51,0,1.0)";  
    m_ctx.fillRect (10, 10, 150, 30);  
    m_ctx.fillStyle = "rgba(255,51,0,0.8)";  
    m_ctx.fillRect (10, 40, 150, 30);  
    m_ctx.fillStyle = "rgba(255,51,0,0.6)";  
    m_ctx.fillRect (10, 70, 150, 30);  
    m_ctx.fillStyle = "rgba(255,51,0,0.4)";  
    m_ctx.fillRect (10, 100, 150, 30);  
    m_ctx.fillStyle = "rgba(255,51,0,0.2)";  
    m_ctx.fillRect (10, 130, 150, 30);  
}
```



# 스타일 적용하기 - 그라디언트 1

## ① 그라디언트 객체를 만듦

**createLinearGradient(x1, y1, x2, y2)**

**createRadialGradient(x1, y1, r1, x2, y2, r2)**

## ② 객체에 원하는만큼 색을 할당

**addColorStop(position, color)**

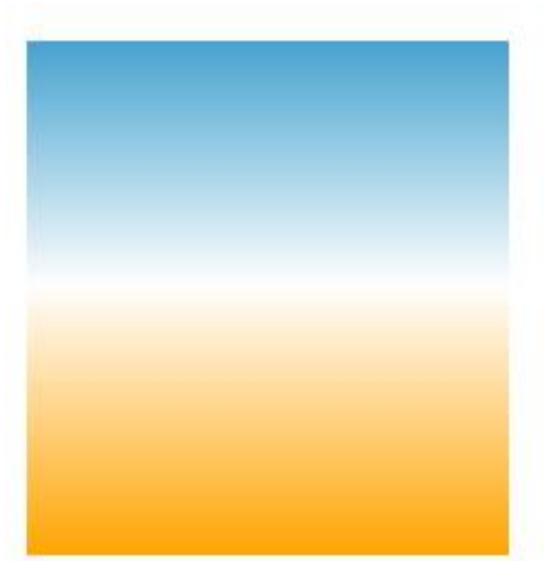
```
if(m_canvas.getContext) {  
    var ctx = m_canvas.getContext("2d");  
    var grad = ctx.createLinearGradient(0,0,0,200);  
    grad.addColorStop(0, '#39c');  
    grad.addColorStop(1, 'orange');  
    ctx.fillStyle=grad;  
    ctx.fillRect(10,10,200,190);  
}
```





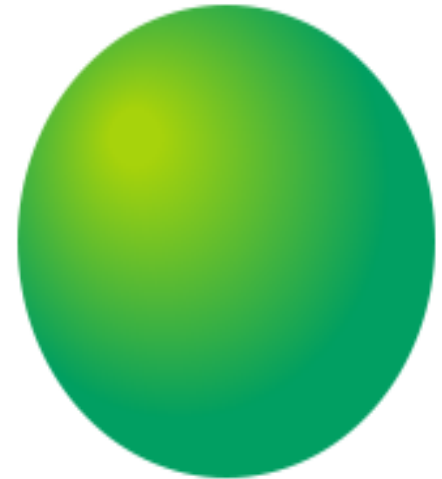
## 스타일 적용하기 - 그라디언트 2

```
if(m_canvas.getContext) {  
    var ctx = m_canvas.getContext("2d");  
    var grad = ctx.createLinearGradient(0,0,0,200);  
  
    grad.addColorStop(0, '#39c');  
    grad.addColorStop(0.5, '#fff');  
    grad.addColorStop(1, 'orange');  
  
    ctx.fillStyle=grad;  
    ctx.fillRect(10,10,200,190);  
}
```



# 스타일 적용하기 - 그라디언트 3

```
if(m_canvas.getContext) {  
    var ctx = m_canvas.getContext("2d");  
    var radgrad = ctx.createRadialGradient(45,45,10,62,60,80);  
  
    radgrad.addColorStop(0, '#A7D30C');  
    radgrad.addColorStop(1, 'rgb(1,159,98)');  
  
    ctx.fillStyle = radgrad;  
    ctx.arc(80,80,80,0,Math.PI*2,true);  
    ctx.fill();  
}
```

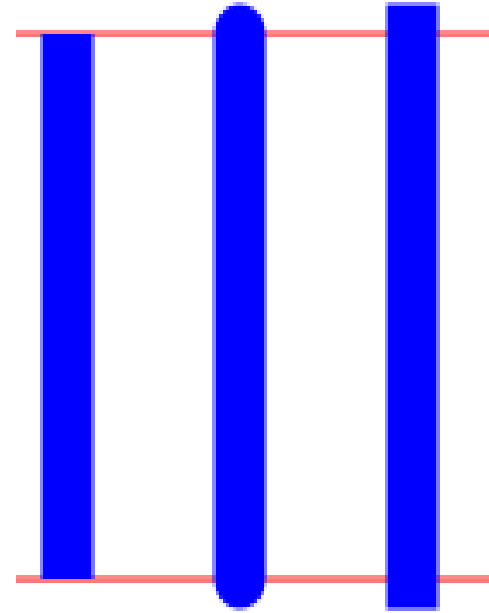


# 스타일 적용하기 - 선 스타일

- **lineWidth = value** : 선 두께 (사용할 수 있는 값은 양수, 기본값 1.0)
- **lineCap = type** : 선의 끝 모양 (사용할 수 있는 값 butt, round, square, 기본값 butt)
- **lineJoin = type** : 선과 선의 만남 (사용할 수 있는 round, bevel, miter. 기본값 miter)
- **miterLimit = value** : 연결 부분이 잘리는 크기

# 스타일 적용하기 - 선 스타일

```
var lineCap=['butt', 'round', 'square'];
var lineJoin=['round', 'bevel', 'miter'];
ctx.strokeStyle="#f00";
ctx.beginPath();
ctx.moveTo(10,10);
ctx.lineTo(150,10);
ctx.moveTo(10,150);
ctx.lineTo(150,150);
ctx.stroke();
ctx.strokeStyle="blue";
for (var i=0;i<lineCap.length;i++) {
  ctx.lineWidth=15;
  ctx.lineCap=lineCap[i];
  ctx.beginPath();
  ctx.moveTo(25+50*i,10);
  ctx.lineTo(25+50*i,150);
  ctx.stroke();
}
```



# 스타일 적용하기 - 선 스타일

```
var lineCap=['butt', 'round', 'square'];
var lineJoin=['round', 'bevel', 'miter'];
ctx.strokeStyle="#f00";
ctx.beginPath();
ctx.moveTo(10,10);
ctx.lineTo(150,10);
ctx.moveTo(10,150);
ctx.lineTo(150,150);
ctx.stroke();
ctx.strokeStyle="green";
for (var j=0;j<lineJoin.length;j++) {
    ctx.lineWidth=20;
    ctx.lineJoin = lineJoin[j];
    ctx.beginPath();
    ctx.moveTo(180,60*j+50);
    ctx.lineTo(220,60*j+15);
    ctx.lineTo(260,60*j+50);
    ctx.stroke();
}
```



# 스타일 적용하기 - 패턴

**createPattern(image, type)**

```
if(canvas.getContext){  
    var ctx = canvas.getContext('2d');  
  
    var img = new Image();  
    img.src = 'images/pattern.png';  
    img.onload = function(){  
        var ptn = ctx.createPattern(img,'repeat');  
        ctx.fillStyle = ptn;  
        ctx.fillRect(0,0,300,500);  
    }  
}
```



# 스타일 적용하기 - 그림자 효과

**shadowOffsetX = float** : 객체로부터 그림자가 얼마나 떨어져 있는가.

양수값은 오른쪽에 그림자.

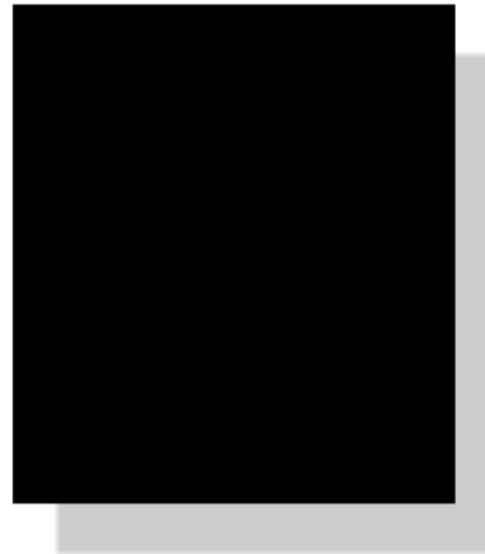
**shadowOffsetY = float** : 객체로부터 그림자가 얼마나 떨어져 있는가.

양수값은 아래에 그림자.

**shadowBlur = float** : 그림자가 얼마나 흐릿한가. 기본값은 0

**shadowColor = color** : 그림자 색상.

```
if(canvas.getContext){  
    var ctx = canvas.getContext('2d');  
    ctx.shadowOffsetX = 20;  
    ctx.shadowOffsetY = 20;  
    ctx.shadowBlur = 2;  
    ctx.shadowColor="#CCC";  
    ctx.fillRect(10,10,200,200);  
}
```



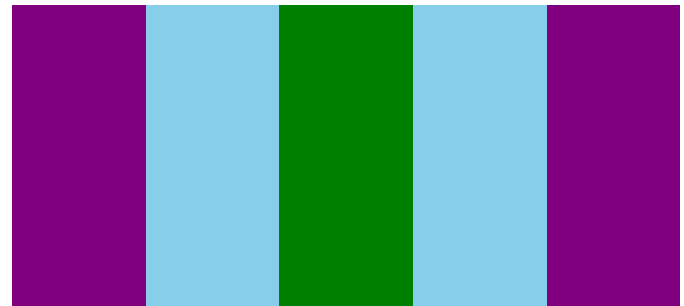
# 드로잉 상태 저장하기

**save(),    restore()**

```
if(canvas.getContext){  
    var ctx = canvas.getContext('2d');
```

```
    ctx.fillStyle="purple";  
    ctx.fillRect(10,10,50,100);  
    ctx.save();  
    ctx.fillStyle = "skyblue";  
    ctx.fillRect (60,10,50,100);  
    ctx.save();  
    ctx.fillStyle = "green";  
    ctx.fillRect (110,10,50,100);  
    ctx.restore();  
    ctx.fillRect (160,10,50,100);  
    ctx.restore();  
    ctx.fillRect (210,10,50,100);
```

```
}
```





# 이미지 표시하기

① 이미지 객체를 만듦

② **drawImage()** 메서드로 객체에 이미지를 그림

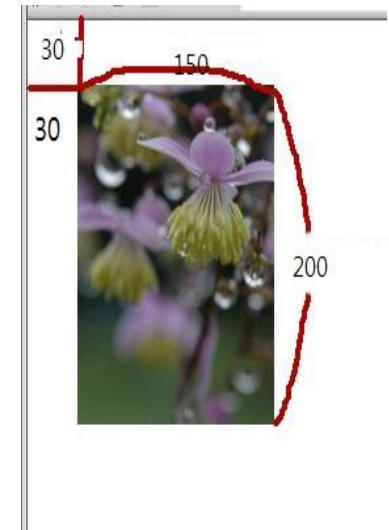
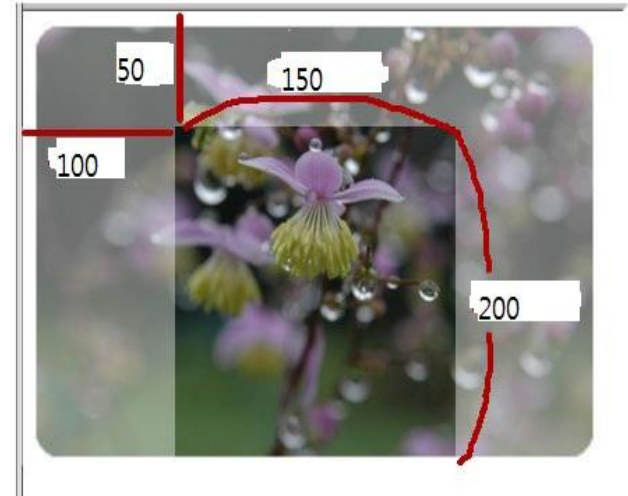
```
if(canvas.getContext){  
    var ctx = canvas.getContext('2d');  
    var img = new Image();  
    img.onload=function(){  
        ctx.drawImage(img,10,10)  
    }  
    img.src='images/flower.jpg'  
  
    var img2=new Image();  
    img2.onload=function(){  
        ctx.drawImage(img2,10,320,200,140);  
    }  
    img2.src='images/flower.jpg';  
}
```



# 이미지 슬라이스하기

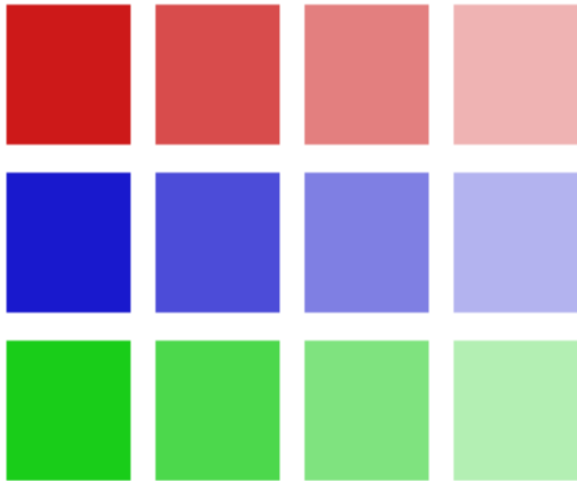
**drawImage(image, sx, sy, sWidth, sHeight, dx, dy, dWidth, dHeight)**

```
if(canvas.getContext){  
    var ctx = canvas.getContext('2d');  
    var img = new Image();  
    img.onload=function(){  
        ctx.drawImage(img,100,50,150,200,30,30,150,200);  
    }  
    img.src='images/flower.jpg';  
}
```



# 변형하기

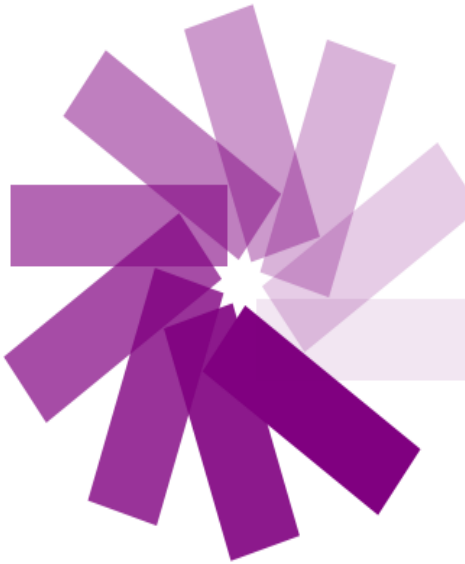
**translate(x,y)** : 위치 옮기기



```
if(m_canvas.getContext) {  
  var ctx = m_canvas.getContext('2d');  
  for(i=0;i<4;i++){  
    ctx.save();  
    ctx.fillStyle="rgb(200,0,0)";  
    ctx.globalAlpha=0.9-0.2*i;  
    ctx.translate(60*i,0);  
    ctx.fillRect(0,0,50,50);  
    ctx.restore(); }  
  for(i=0;i<4;i++){  
    ctx.save();  
    ctx.fillStyle="rgb(0,0,200)";  
    ctx.globalAlpha=0.9-0.2*i;  
    ctx.translate(60*i,60);  
    ctx.fillRect(0,0,50,50);  
    ctx.restore(); }  
  for(i=0;i<4;i++){  
    ctx.save();  
    ctx.fillStyle="rgb(0,200,0)";  
    ctx.globalAlpha=0.9-0.2*i;  
    ctx.translate(60*i,120);  
    ctx.fillRect(0,0,50,50);  
    ctx.restore(); }
```

# 변형하기

**rotate(각도)** : 회전시키기



```
if(m_canvas.getContext) {  
    var ctx = m_canvas.getContext('2d');  
    ctx.translate(200,200);  
    ctx.save();  
    ctx.fillStyle="purple";  
    for (i=0;i<10;i++){  
        ctx.globalAlpha = 1-(0.1*i)  
        ctx.rotate(Math.PI*2/10);  
        ctx.fillRect(10,10,150,50);  
    }  
}
```

# 변형하기

**scale(x, y) : 확대/축소**














```
if(m_canvas.getContext) {  
    var ctx = m_canvas.getContext('2d');  
    ctx.fillStyle="orange";  
    ctx.save();  
    ctx.fillRect(0,0,50,50);  
    ctx.translate(0,60);  
    ctx.scale(1.0, 0.5);  
    ctx.fillRect(0,0,50,50);  
    ctx.restore();  
    ctx.translate(0,100);  
    ctx.scale(2.0, 2.0);  
    ctx.fillRect(0,0,50,50);  
}
```

# 합성하기

## globalCompositeOperation 메서드

globalCompositeOperation의 결과 화면은 브라우저마다 조금씩 다릅니다.

					
Source-over	Source-in	Source-out	Source-atop	Destination-over	Destination-in
					이 결과 화면은 파이어폭스 3.6.6의 결과입니다.
Destination-out	Destination-atop	Lighter	Darker	copy	

# 클리핑

## clip()

```
if(canvas.getContext){  
    var ctx = canvas.getContext('2d');  
    var img = new Image();  
    img.onload=function(){  
        ctx.drawImage(img,10,10)  
    }  
    img.src='images/flower.jpg'  
    ctx.beginPath();  
    ctx.arc(200,110,100,0,Math.PI*2,true);  
    ctx.clip();  
}
```

