

canvas

1. canvas 기초
2. canvas 응용

성공회 대학교 소프트웨어공학과 외래교수
원철연(<http://fromyou.tistory.com>)





→ canvas 기초

canvas 요소는 기존의 HTML4.01 버전과 비교하여 새롭게 등장한 요소로 HTML5에서 차지하는 비중은 대단히 높음. 웹페이지에 간단한 직선, 사각형, 타원과 같은 기본적인 도형부터 차트, 애니메이션, 게임같은 조금은 복잡한 분야까지 직, 간접적으로 적용 가능.

현재 W3C에서 HTML canvas 2D Context 관련하여 표준화가 진행중
“<http://www.w3.org/TR/2dcontext/>”

3D(3차원) 작업은 “<https://www.khronos.org/registry/webgl/specs/latest/1.0/>”에 명시된대로 비영리 컨소시엄인 크로노스(khronos) 그룹에서 진행중

canvas 요소는 단지 너비(width)와 높이(height), 그리고 canvas를 식별하는데 사용되는 id 속성값은 갖는 단순한 요소.

이렇게 단순한 canvas에 간단한 도형부터 애니메이션, 게임 등이 나타나도록 하려면 중간 역할자가 필요한데 바로 Javascript(jQuery)를 이용해서 가능.
즉, canvas에 생기를 불어넣어주는 것이 바로 native JavaScript or jQuery 같은 library를 이용하여 가능

canvas 선언하기

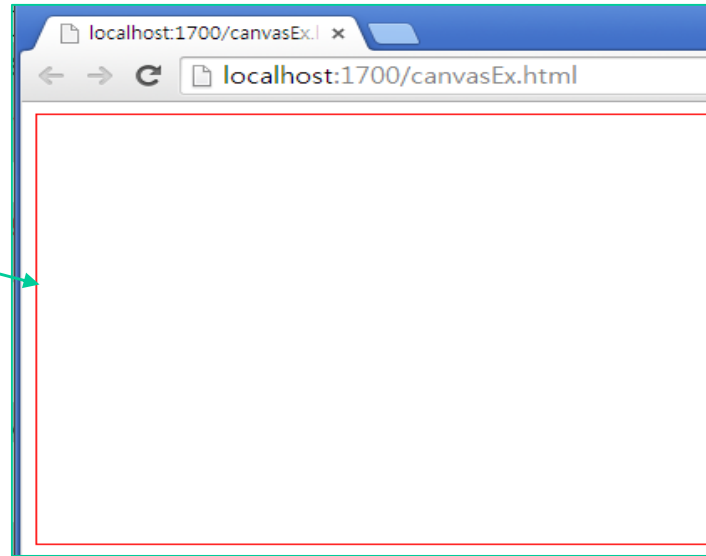
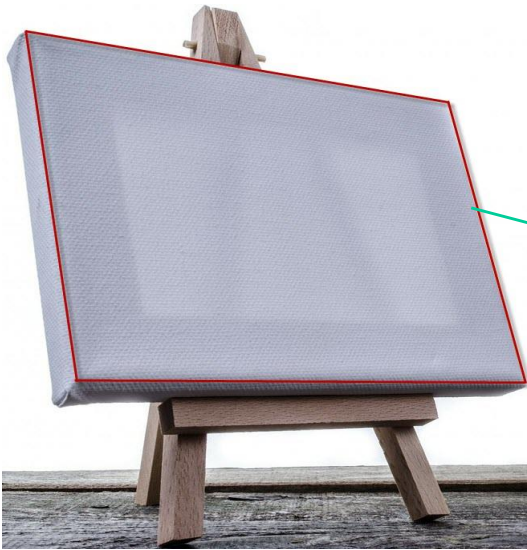
canvas는 id, width, height 속성을 이용하여 <body>...</body> 블록 내에 선언

```
<canvas id="id 속성값" width="canvas의 너비" height="canvas의 높이" ></canvas>
```

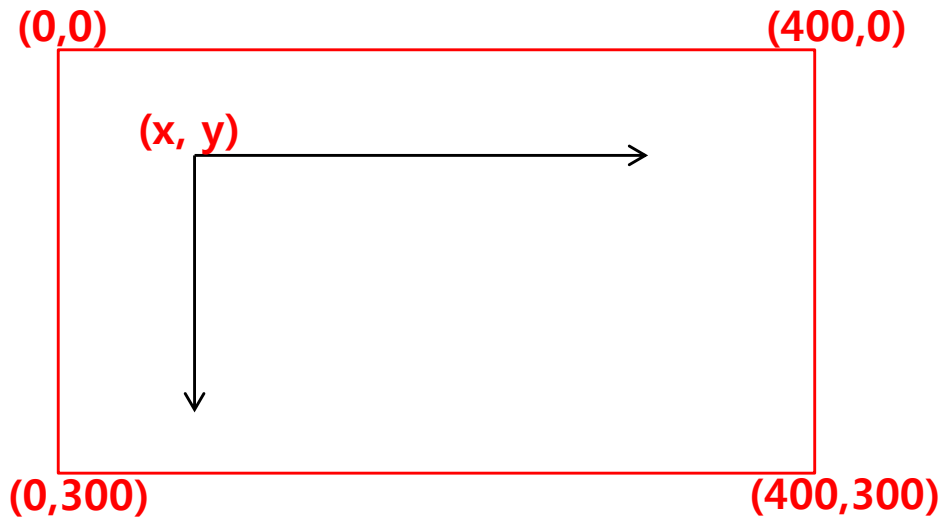
canvasEx.html

```
<link rel="Stylesheet" type="text/css" href="css/canvasEx.css" />
</head>
<body>
  <canvas id="yourcanvas" width="400px" height="300px"></canvas>
</body>

<!-- css/canvasEx.css -->
body{ }
canvas {
  border: 1px solid red;
}
```

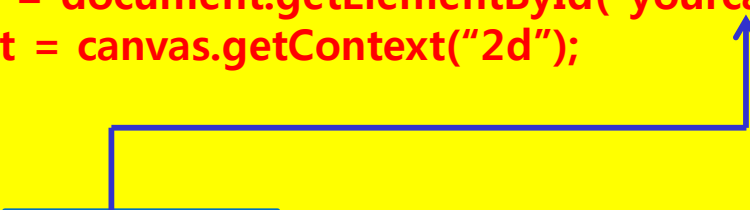
canvas 좌표해석



canvas 내에 그리기 준비 작업

canvas에 직선, 도형, 차트 등을 그리기 위해서는 기본적으로 조금 전의 좌표 해석이 필요하고 그 다음 실제 그리기 위해서는 다음과 같은 형식으로 그릴 준비가 되어 있어야 함.

```
var canvas = document.getElementById("yourcanvas");  
var context = canvas.getContext("2d");  
...  
</script>  
...  
<canvas id="yourcanvas" width="400px" height="300px"> </canvas>
```



어떤 색의 물감을 어디에 적용할까?

canvas에서 직선이나 도형에 색이나 스타일을 적용할 경우 다음 2 가지 속성 적용 가능

Property	Description
fillStyle	직선이나 도형의 내부 색이나 스타일 설정
strokeStyle	직선이나 도형의 라인 색이나 스타일 설정

사각형 그리기

canvas 내에 사각형을 그리기 위해서는 다음과 같은 3개의 메서드를 이용

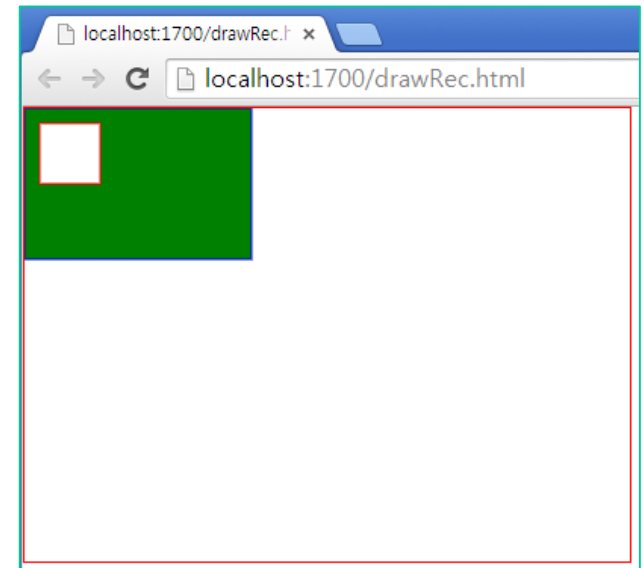
Method	Description
fillRect(x,y,w,h)	주어진 스타일로 사각형을 칠함
strokeRect(x,y,w,h)	주어진 스타일로 사각형 라인을 칠함
clearRect(x,y,w,h)	해당 사각형을 투명한 검은색으로 칠함. 실제로는 흰색으로 나타남

```
<style>
  canvas { border: 2px dotted red; }
</style>
<script type="text/javascript">
  window.onload = function () {
    var canvas = document.getElementById("yourcanvas");
    var context = canvas.getContext("2d");

    context.fillStyle = "Green";
    context.strokeStyle = "blue";

    context.fillRect(0, 0, 150, 100);
    context.strokeRect(0, 0, 150, 100);

    context.strokeStyle = "red";
    context.clearRect(10, 10, 40, 40);
    context.strokeRect(10, 10, 40, 40);
  }
</script>
</head>
<body>
  <canvas id="yourcanvas" width="400px" height="300px"> </canvas>
```



Path와 직선 그리기

Path는 하나 이상의 하부 경로들의 리스트. 하부 경로는 직선이나 곡선에 의해 연결되는 하나 또는 둘 이상의 점들의 리스트로 구성.

경로는 처음 시작점과 마침점이 같으면 닫힌 경로, 시작점과 마침점이 다르면 열린 경로로 구분하며 다음과 같은 메서드들을 이용하여 나타낼 수 있음

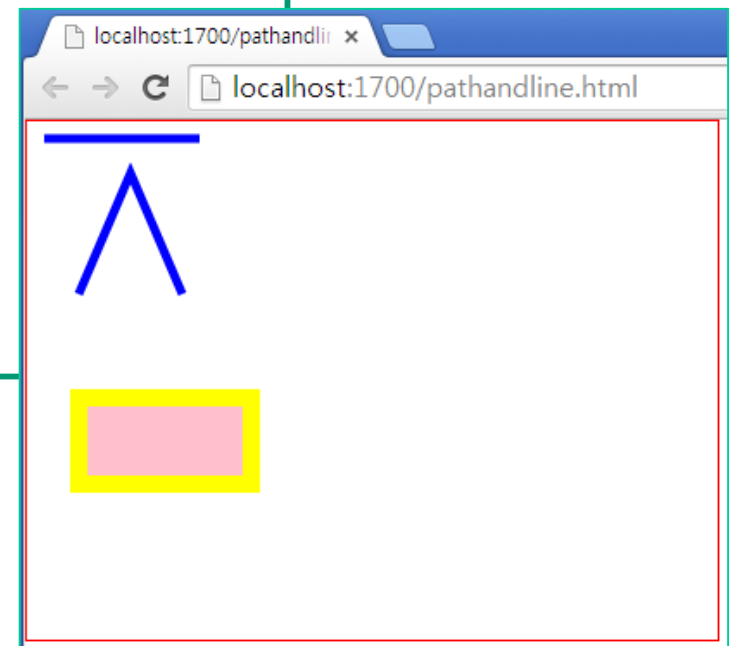
Method	Description
<code>beginPath()</code>	현재의 경로를 재설정하거나 경로를 시작할 때
<code>moveTo(x,y)</code>	(x,y) 좌표로 이동
<code>closePath()</code>	현재의 경로가 닫혔음을 마킹하고 새로운 경로 시작
<code>lineTo(x,y)</code>	(x,y)좌표까지 직선 그리기
<code>rect(x,y,w,h)</code>	(x,y)점을 시작으로 사각형의 닫힌 경로 구성
<code>fill()</code>	현재의 fill 스타일로 직선이나 도형의 내부 칠하기
<code>stroke()</code>	현재의 stroke 스타일로 직선이나 도형의 라인 칠하기
<code>isPointInPath(x,y)</code>	(x,y)점이 현재 경로 상에 존재하는 점이면 true, 그렇지 않으면 false

```
window.onload=function(){  
  ...  
  context.strokeStyle = "blue";  
  context.fillStyle = "red";  
  context.lineWidth = 5;
```

```
  //직선 그리기  
  context.beginPath();  
  context.moveTo(10, 10);  
  context.lineTo(100, 10);  
  context.stroke();
```

```
  //열린 경로  
  context.beginPath();  
  context.moveTo(30, 100);  
  context.lineTo(60, 30);  
  context.lineTo(90, 100);  
  context.stroke();
```

```
  //닫힌 경로  
  context.beginPath();  
  context.rect(30, 160, 100, 50);  
  context.fillStyle = "pink";  
  context.fill();  
  context.lineWidth = 10;  
  context.strokeStyle = "yellow";  
  context.stroke();  
}
```



호, 원 그리고 곡선 그리기

canvas 내에 호(arc), 원(circle) 그리고 곡선을 그리기 위해서는 다음과 같은 4가지 메서드를 이용하여 그리게 됨

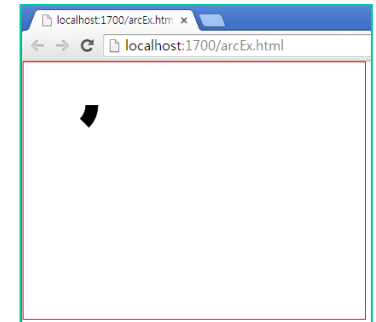
Method	Description
<code>arc(x,y,radius,startangle, endangle,[anticlockwise])</code>	x, y는 중심 좌표를 radius는 원의 반경을 startangle은 시작각, endangle은 끝각을 명시. 추가로 anticlockwise는 시계 반대방향을 나타냄
<code>arcTo(x1,y1,x2,y2, radius)</code>	보통 둥근 모서리를 만들 때 사용
<code>quadraticCurveTo(cpx,cpy, x,y)</code>	2차원 곡선을 그리는데 사용
<code>bezierCurveTo(cp1x,cp1y,cp2x, cp2y,x,y)</code>	3차원 곡선을 그리는데 사용

```
<script type="text/javascript">
  window.onload = function () {
    var canvas = document.getElementById("yourcanvas");
    var context = canvas.getContext("2d");

    context.strokeStyle = "black";
    context.lineWidth = 15;

    var centerX = 50;
    var centerY = 50;
    var radius = 30;
    var startAngle = 0;
    var endAngle = 0.25 * Math.PI;
    var counterClockwise = false;

    context.arc(centerX, centerY, radius, startAngle, endAngle, counterClockwise);
    context.stroke();
  }
</script>
</head>
<body>
  <canvas id="yourcanvas" width="400px" height="300px"> </canvas>
```



canvas 저장

canvas 내에 그린 직선이나 사각형 같은 이미지들을 이미지 형태로 저장

canvas.toDataURL(type)

Type은 기본형은 "image/png"이며 "image/jpeg" 등으로 올 수 있으며 반환값은 "data:image/png;base64,..." 형태로 반환

"pathandline.html" 에 다음과 같이 코드를 추가

```
<button name="btnSave" id="btnSave">Drawing 저장</button>
<script>
  function saveDrawing() {
    var canvas = document.getElementById('yourcanvas');
    window.open(canvas.toDataURL("image/png"));
  }

  document.getElementById('btnSave').addEventListener('click', saveDrawing, false);
</script>
```

→ Canvas 응용

그림자 효과(Shadow Effects)

포토샵이나 파워포인트에서 처럼 선, Text, 도형 등에 그림자 효과(Shadow Effects)를 줄 수 있음.

<http://www.w3.org/TR/2dcontext/#shadows>에 다음과 같은 context의 4 가지 속성으로 그림자 효과를 줄 수 있음

Method	Description
shadowColor	그림자의 색상을 설정. "blue"처럼 문자열 색상이나 RGB, RGBA로 설정
shadowOffsetX	그림자를 X 축 방향(너비 방향)으로 얼마나 나타낼 것인지 설정
shadowOffsetY	그림자를 Y 축 방향(높이 방향)으로 얼마나 나타낼 것인지 설정
shadowBlur	번짐(Blur)을 어느 정도 할 것인지 설정

shadowEx.html

```
<script type="text/javascript">  
  window.onload = function () {  
    var canvas = document.getElementById("yourcanvas");  
    var context = canvas.getContext("2d");
```



```
context.shadowColor = "rgba(0,0,255,1)";  
context.shadowOffsetX = 10;  
context.shadowOffsetY = 10;  
context.shadowBlur = 10;  
context.fillStyle = "#B3E3F5";  
context.fillRect(20, 20, 380, 100);
```

```
//텍스트 나타내기
```

```
var theString = "HTML5 canvas에 텍스트 쓰기";  
context.fillStyle = "black";  
context.shadowColor = "rgba(0,0,0,0.5)";  
context.shadowOffsetX = 5;  
context.shadowOffsetY = 5;  
context.shadowBlur = 5;  
context.font = "20pt Century Gothic";  
context.fillText(theString, 35, 75);  
context.stroke();
```

```
}
```

```
</script>
```

```
</head>
```

```
<body>
```

```
<canvas id="yourcanvas" width="400px" height="300px"> </canvas>
```

Gradient 효과(Effects)

사각형이나 도형 등에 색의 점진적인 변화(Gradient)를 주는 형태는 선형(linear)과 방사형(radial)이 있으며 다음과 같은 형식을 사용

```
var gradient=context.createLinearGradient(x0,y0,x1,y1);  
var gradient=context.createRadialGradient(x0,y0,r0,x1,y1,r1);
```

```
생성된 gradient객체.gradient.addColorStop(offset, color);
```

(x0,y0), (x1,y1)은 시작좌표와 끝좌표를 나타내고 r0, r1은 각각 반지름을 나타냄
위에서 addColorStop()은 색상을 설정하는데 사용. Offset은 0과 1사이의 값으로 설정.
예를 들어 0, 0.25, 0.5, 0.75, 1로 addColorStop() 설정하여 사용 가능

gradientEx.html

```
<script type="text/javascript">  
  window.onload = function () {  
    var canvas = document.getElementById("yourcanvas");  
    var context = canvas.getContext("2d");  
  }  
</script>  
...  
<canvas id="yourcanvas" width="400px" height="300px"> </canvas>
```

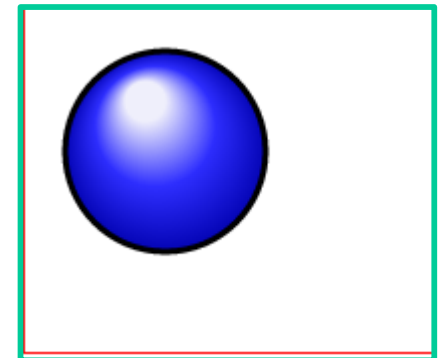
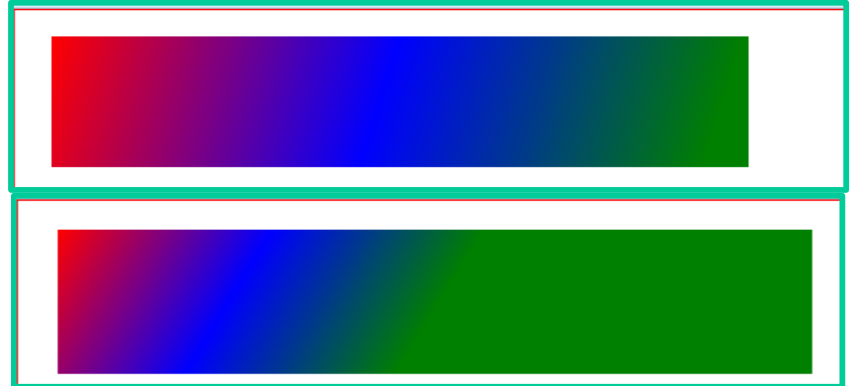
```
//LinearGradient 그리기  
var linearGrd = context.createLinearGradient(20, 20, 200, 100);  
//var linearGrd = context.createLinearGradient(20,20,380,100);  
linearGrd.addColorStop(0, "red");  
linearGrd.addColorStop(0.5, "blue");  
linearGrd.addColorStop(1, "green");
```

```
context.fillStyle = linearGrd;  
context.fillRect(20, 20, 380, 100);
```

```
//RadialGradient 그리기  
var radialGrad = context.createRadialGradient(60, 175, 10, 70, 200, 50);  
radialGrad.addColorStop(0.0, "#EFEFFB");  
radialGrad.addColorStop(0.5, "#2E2EFE");  
radialGrad.addColorStop(1.0, "#0404B4");  
context.lineWidth = 3;  
context.fillStyle = radialGrad;
```

```
context.beginPath();  
context.arc(70, 200, 50, 0, 2 * Math.PI);  
context.fill();  
context.stroke();
```

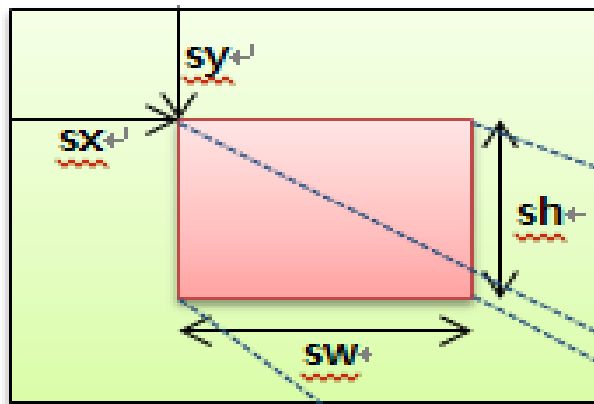
```
}
```



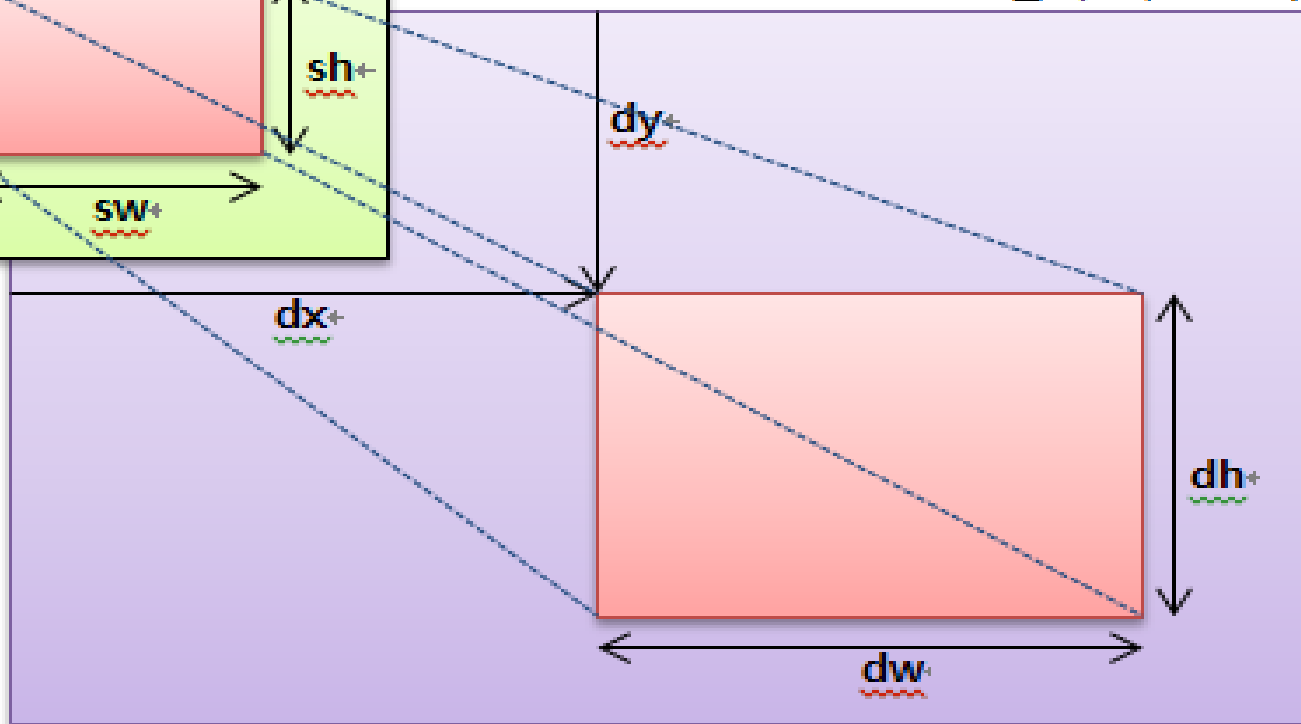
이미지 픽셀 처리

canvas 내에서 이미지 작업은 크게 drawImage 메서드를 이용하여 이미지를 그리는 작업, 이미지를 확대, 축소하는 등의 작업을 수행하는 변환, 그리고 픽셀 작업이 있음

소스 이미지(source Image)



캔버스(canvas)



drawImage 메서드를 이용한 이미지 나타내기

canvas에 이미지를 나타내기 위해서는 다음과 같은 3 가지 형태의 drawImage() 메서드 이용

```
context.drawImage(image,dx, dy);  
context.drawImage(image,dx, dy, dw, dh);  
context.drawImage(image,sx, sy, sw, sh, dx, dy, dw, dh);
```

입력파라미터	기능
sx	소스 이미지의 x좌표
sy	소스 이미지의 y좌표
sw	소스 이미지의 너비(width)
sh	소스 이미지의 높이(height)
dx	캔버스 내 이미지를 그릴 시작 x좌표
dy	캔버스 내 이미지를 그릴 시작 y좌표
dw	캔버스 내에 그려지는 이미지의 너비(width)
dh	캔버스 내에 그려지는 이미지의 높이(height)

```

<style>
  canvas{ border:2px solid red; }
</style>
<script type="text/javascript">
  window.onload = function () {
    var canvas = document.getElementById("yourcanvas")
    var context = canvas.getContext("2d");

    // canvas에 그릴 선이나 도형 작성
    var srcImg = document.getElementById("imgbus");
    context.drawImage(srcImg, 50,0);
    //context.drawImage(srcImg, 50,0,100,75); //크기 축소
    //context.drawImage(srcImg, 50,0,400,300); //크기 확대
    //context.drawImage(srcImg, 100,75,100,75,50,0,100,75); //그림의 일부분
    //context.drawImage(srcImg, 100, 75, 100, 75, 50, 0, 150, 125); //그림의 일부분 확대
  }
</script>
</head>
<body>
  <!---->
  
  <br />
  <canvas id="yourcanvas" width="400px" height="300px"> </canvas>

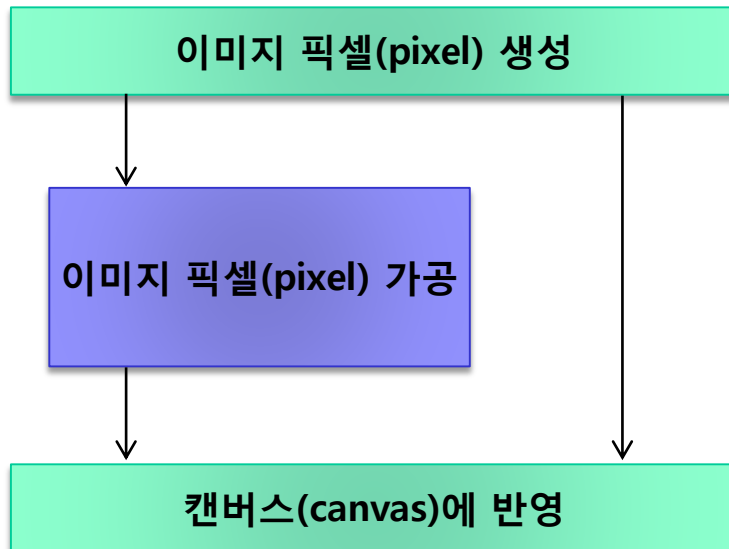
```



이미지 특정 영역만을 복사하기

canvas에 이미지의 특정 영역만을 복사하여 그리고자 할 경우
메모리상에 `createImageData()` 메서드를 이용하여 생성한 후 `putImageData()` 메서드를
이용하여 canvas에 그 크기만큼 복사 가능

```
imagedata = context.createImageData(sw, sh);  
imagedata = context.createImageData(imagedata);  
imagedata = context.getImageData(sx, sy, sw, sh);  
putImageData(imagedata, dx, dy [,dirtyX, dirtyY, dirtyWidth, dirtyHeight]);
```



```
<br />
<canvas id="yourcanvas" width="600" height="400"></canvas><br />
<button name="btnCopy" id="btnCopy">붙이기</button>
<script>
function copyImage() {
    var canvas = document.getElementById("yourcanvas")
    var context = canvas.getContext("2d");

    // canvas에 그릴 선이나 도형 작성
    var srcImg = document.getElementById("imgbus");
    context.drawImage(srcImg, 0, 0);

    //이미지 픽셀(pixel) 생성
    var imageData = context.getImageData(0, 0, 100, 50);

    //캔버스에 반영하기
    context.putImageData(imageData, 350, 100);
    document.getElementById('btnCopy').disabled = true;
}
document.getElementById('btnCopy').addEventListener('click', copyImage, false);
</script>
```

Pie Chart 만들기

차트(Chart)는 다양한 분야에서 사용되고 있고 특히 다수의 데이터를 간의 관계를 시각화 하는데 효과적이라서 실적이나 장래 성장 가능성 및 목표 매출 등에 사용됨

전세계 Web Browser 점유율(2015년 3월 기준 : <http://www.w3counter.com>)

Chrome	IE	Safari	Firefox	Opera	기타
43.9%	16.1%	15.7%	14.6%	3.2%	6.5%

```
<style>
  table, th, td{ border:1px solid black; border-collapse:collapse; padding: 5px; }
  table > caption{font-weight:bolder; margin-bottom:5px;}
  thead{ background:#0026ff; color:white;}
  tbody{ text-align:right; }
  table{ width : 500px; margin-bottom:10px; box-shadow:4px 4px 10px blue; }
  #chartcanvas {
    display:none;
    /*border: 1px solid red;*/
    padding: 0;
  }
</style>
<script>
  window.onload = function () {
    var thead = document.getElementsByTagName('thead')[0];
    thead.addEventListener('click', showChart, false);
    var canvas = document.getElementById('chartcanvas');
    canvas.addEventListener('click', hideCanvas, false);
  }
</script>
</head>
<body>
```

```
<table id="yourTable">
  <caption>전세계 웹브라우저 점유율 현황(2015년 3월 기준)</caption>
  <thead>
    <tr>
      <th>Chrome</th>
      <th>IE</th>
      <th>Safari</th>
      <th>Firefox</th>
      <th>Opera</th>
      <th>etc</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>43.9%</td>
      <td>16.1%</td>
      <td>15.7%</td>
      <td>14.6%</td>
      <td>3.2%</td>
      <td>6.5%</td>
    </tr>
  </tbody>
</table>
<canvas id="chartcanvas" width="500" height="400"> </canvas>
```

```
<script>
function showChart() {
    var canvas = document.getElementById('chartcanvas');
    var ctx = canvas.getContext("2d");
    var sw = canvas.width;
    var sh = canvas.height;
    var PADDING = 100;

    //Browser별 데이터 입력 chrome, IE, Firefox, Safari, Opera, etc 순
    var data = [43.9, 16.1, 15.7, 14.6, 3.2, 6.5];

    //Browser별 색상 lawngreen, blue, deeppink, aquamarine3, magenta, gold
    var colors = ["#7cfc00", "#0000ff", "#ff1493", "#66CDAA", "#ff00ff", "#FFD700"];

    var center_X = sw / 2; //원의 중심 x 좌표
    var center_Y = sh / 2; //원의 중심 y 좌표

    // 두 계산값 중 작은 값을 원의 반지름으로 설정
    var radius = Math.min(sw - (PADDING * 2), sh - (PADDING * 2)) / 2;
    var angle = 0;
    var total = 0;
    for (var i in data) { total += data[i]; } //데이터(data)의 총합 계산
```



```
for (var i = 0; i < data.length; i++) {  
    ctx.fillStyle = colors[i]; //생성되는 부분의 채울 색 설정  
    ctx.beginPath();  
    ctx.moveTo(center_X, center_Y); //원의 중심으로 이동  
    ctx.arc(center_X, center_Y, radius, angle, angle + (Math.PI * 2 * (data[i] / total)));  
    ctx.lineTo(center_X, center_Y);  
    ctx.fill();  
    angle += Math.PI * 2 * (data[i] / total);  
}  
  
//title 나타내기  
var strMessage = "Web Browser Market Share March 2015";  
ctx.textAlign = "center";  
ctx.fillStyle = "black";  
ctx.font = "14pt Century Gothic";  
ctx.fillText(strMessage, sw / 2, 30);
```

```
//브라우저 이름 및 점유율 나타내기
AddBrowserText(ctx, "Chrome", "(43.9%)", colors[0], 280, 320);
AddBrowserText(ctx, "Internet Explorer", "(16.1%)", colors[1], 20, 170);
AddBrowserText(ctx, "Firefox", "(15.7%)", colors[2], 140, 70);
AddBrowserText(ctx, "Safari", "(14.6%)", colors[3], 260, 70);
AddBrowserText(ctx, "Opera", "(3.2%)", colors[4], 360, 130);
AddBrowserText(ctx, "etc", "(6.5%)", colors[5], 370, 170);

canvas.style.display = "block";
}

function AddBrowserText(ctx, strBrowser, strRate, colors, text_xcoord, text_ycoord) {
    ctx.textAlign = "left";
    ctx.fillStyle = colors;
    ctx.font = "12pt Century Gothic";
    ctx.fillText(strBrowser, text_xcoord, text_ycoord);
    ctx.fillText(strRate, text_xcoord, text_ycoord + 20);
}

function hideCanvas() {
    this.style.display = "none";
}
</script>
```

Pie Chart 만들기 with json

사전 준비

Web.config 파일에 다음과 같은 코드 추가

```
<configuration>
```

```
  <system.webServer>
```

```
    <staticContent>
```

```
      <mimeMap fileExtension=".json" mimeType="application/json" />
```

```
    </staticContent>
```

```
  </system.webServer>
```

piechart.json

```
{  
  "heading": "Web Browser Market Share March 2015",  
  "browsers": [  
    { "browser": "chrome", "rate": 43.9 },  
    { "browser": "ie", "rate": 16.1 },  
    { "browser": "safari", "rate": 15.7 },  
    { "browser": "firefox", "rate": 14.6 },  
    { "browser": "opera", "rate": 3.2 },  
    { "browser": "etc", "rate": 6.5 }  
  ]  
}
```

XMLHttpRequest 객체의 open() 메서드로 전송방식, 경로, 비동기 처리 설정)

```
xar xhr = new XMLHttpRequest();  
xhr.open("GET", "files/piechart.json", true);
```

files/piechart.json을 "GET" 방식으로 비동기 처리 요청. 위에서 false 즉 동기 방식처리는 권장하지 않음 "POST"는 서버로 많은 양의 데이터를 보내거나 form data 같은 사용자 정보 할 때 등에 사용

XMLHttpRequest 객체의 readystate 속성(Property)

요청(request)에 대한 상태를 나타냄

Value	State	Description
0	UNSENT	open() 아직 실행되지 않음
1	OPEND	send() 아직 실행되지 않음
2	HEADERS_RECIVED	send()가 실행되고 headers, status 가능
3	LOADING	로딩 중(데이터 일부만 받음)
4	DONE	동작이 완료됨

XMLHttpRequest 객체의 status 속성(Property)

요청(request)에 대한 상태 코드를 나타냄

Value	Description
200	요청이 성공했을 경우 200
304	마지막 요청이후 요청한 페이지가 수정되지 않은 경우 304 Not Modified
4xx	클라이언트 오류 400 Bad Request 401 Unauthorized 404 Not Found
500	서버 오류 500 Internal Server Error

(출처 : <http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>)

XMLHttpRequest 객체의 onreadystatechange 이벤트

요청(request)에 대한 상태를 나타내는 앞서 언급한 readyState에 변화가 있을 때마다 동작하는 이벤트

```
var xhr = new XMLHttpRequest();
xhr.onreadystatechange = function () {
    if ((xhr.readyState == 4) && (xhr.status == 200)) {
        // 필요한 작업

    }
}

//요청(request) 설정
xhr.open("GET", "files/piechart.json", true);

//요청 보내기
xhr.send(null);
```



```
<title>Pie Chart with json</title>
<style>
  table, th, td{
    border:1px solid black;
    border-collapse:collapse;
    padding: 5px;
  }
  table > caption{font-weight:bolder; margin-bottom:5px;}
  thead{ background:#0026ff; color:white;}
  tbody{ text-align:right; }
  table{ width : 500px;
    margin-bottom:10px;
    box-shadow:4px 4px 10px blue;
  }

  #chartcanvas {
    /*display:none;*/
    border: 1px solid red;
    padding: 0;
  }
</style>
```

```
<body>
  <a href="#">Pie chart 나타내기</a>
  <table id="yourTable">
    <caption>전세계 웹브라우저 점유율 현황(2015년 3월 기준)</caption>
    <thead>
      <tr>
        <th>Chrome</th>
        <th>IE</th>
        <th>Safari</th>
        <th>Firefox</th>
        <th>Opera</th>
        <th>etc</th>
      </tr>
    </thead>
  </table>
  <canvas id="chartcanvas" width="500" height="400"> </canvas>
  <script src="js/json2.js"> </script>
  <script src="js/piechartwithajaxjson.js"> </script>
```

```
(function () {  
    "use strict";  
    var body = document.getElementsByTagName('body')[0];  
    var link = document.getElementsByTagName('a')[0];  
  
    //var thead = document.getElementsByTagName('thead')[0];  
    //thead.addEventListener('click', showChart, false);  
    //var canvas = document.getElementById('chartcanvas');  
    //canvas.addEventListener('click', hideCanvas, false);  
  
    var rates = [];  
  
    link.onclick = function () {  
  
    };  
  
    function hideCanvas() { this.style.display = "none"; }  
  
    function showChart() { }  
  
    function AddBrowserText(ctx, strBrowser, strRate, colors, text_xcoord, text_ycoord) { }  
  
})();
```

```

link.onclick = function () {
    var xhr = new XMLHttpRequest();
    xhr.onreadystatechange = function () {
        if ((xhr.readyState == 4) && (xhr.status == 200)) {
            var table = document.getElementsByTagName('table')[0];
            var json = xhr.responseText; //var json = JSON.parse(xhr.responseText);
            alert(json);
            var heading = json.heading;
            var browsers = json.browsers;
            var tbody = document.createElement('tbody');
            var tr = document.createElement('tr');

            for (var i = 0; i < browsers.length; i++) {
                var td = document.createElement('td');
                var tdText = document.createTextNode(browsers[i].rate+"%");
                td.appendChild(tdText);
                tr.appendChild(td);
                rates.push(browsers[i].rate);
            }
            tbody.appendChild(tr);
            table.appendChild(tbody);
            body.removeChild(link);
        }
    };
};

```

var thead =
document.getElementsByTagName('thead')[0];
thead.addEventListener('click', showChart, false);
var canvas =
document.getElementById('chartcanvas');
canvas.addEventListener('click', hideCanvas, false);

```
xhr.open("GET", "files/piechart.json", true); //요청 설정
xhr.send(null); //요청 보내기
return false;
}

function showChart() {
    var canvas = document.getElementById('chartcanvas');
    var ctx = canvas.getContext("2d");
    var sw = canvas.width;
    var sh = canvas.height;
    var PADDING = 100;

    //Browser별 색상 lawngreen, blue, deeppink, aquamarine3, magenta, gold
    var colors = ["#7cfc00", "#0000ff", "#ff1493", "#66CDAA", "#ff00ff", "#FFD700"];

    var center_X = sw / 2; //원의 중심 x 좌표
    var center_Y = sh / 2; //원의 중심 y 좌표

    // 두 계산값 중 작은 값을 원의 반지름으로 설정
    var radius = Math.min(sw - (PADDING * 2), sh - (PADDING * 2)) / 2;
    var angle = 0;
    var total = 0;
    for (var i in rates) { total += rates[i]; } //데이터(data)의 총합 계산
```

```
for (var i = 0; i < rates.length; i++) {  
    ctx.fillStyle = colors[i]; //생성되는 부분의 채울 색 설정  
    ctx.beginPath();  
    ctx.moveTo(center_X, center_Y); //원의 중심으로 이동  
    ctx.arc(center_X, center_Y, radius, angle, angle + (Math.PI * 2 * (rates[i] / total)));  
    ctx.lineTo(center_X, center_Y);  
    ctx.fill();  
    angle += Math.PI * 2 * (rates[i] / total);  
}  
  
var strMessage = "Web Browser Market Share March 2015";  
ctx.textAlign = "center";  
ctx.fillStyle = "black";  
ctx.font = "14pt Century Gothic";  
ctx.fillText(strMessage, sw / 2, 30);  
  
AddBrowserText(ctx, "Chrome", rates[0] + "%", colors[0], 280, 320);  
AddBrowserText(ctx, "Internet Explorer", rates[1] + "%", colors[1], 20, 170);  
AddBrowserText(ctx, "Firefox", rates[2] + "%", colors[2], 140, 70);  
AddBrowserText(ctx, "Safari", rates[3] + "%", colors[3], 260, 70);  
AddBrowserText(ctx, "Opera", rates[4] + "%", colors[4], 360, 130);  
AddBrowserText(ctx, "etc", rates[5] + "%", colors[5], 370, 170);  
canvas.style.display = "block";  
}
```

```
function AddBrowserText(ctx, strBrowser, strRate, colors, text_xcoord, text_ycoord) {  
    ctx.textAlign = "left";  
    ctx.fillStyle = colors;  
    ctx.font = "12pt Century Gothic";  
    ctx.fillText(strBrowser, text_xcoord, text_ycoord);  
    ctx.fillText(strRate, text_xcoord, text_ycoord + 20);  
}
```