

---

# rMate Chart for HTML5 사용 설명서

**Version 5.0**



---

정품을 구입하신 고객에게는 기술상담 및 지원을 제공합니다

서울시 영등포구 영신로 220, 611 호

(영등포동 8 가, KnK 디지털 타워)

(Tel : 02-2655-9767, [riamore@riamore.net](mailto:riamore@riamore.net))

# (주)리아모어소프트

데스크탑 용 데모 : <http://demo.riamore.net/HTML5demo/chart>

모바일 용 데모 : <http://demo.riamore.net/m/chart>

# 목 차

1. 개요.....	7
1.1. rMate Chart for HTML5 의 주요 특징 .....	7
1.2. 시스템 요구사항.....	7
1.3. 차트 시스템 구성 환경 설명.....	7
1.4. 제품의 구성요소 .....	8
1.5. 차트 리스트 .....	8
2. 기본적인 차트의 생성.....	10
2.1. rMate Chart for HTML5 라이선스 등록하기 .....	10
2.2. 기본적인 차트 생성하기.....	10
2.3. 차트 버전 정보 보기.....	15
3. 차트 데이터 형식.....	15
3.1. XML 형식의 데이터 작성 및 차트에 삽입하기.....	15
3.2. 배열 형식의 데이터 작성 및 차트에 삽입하기.....	17
4. 차트 설정 및 연동방식 .....	19
4.1. chartVars 설정하기.....	19
4.2. 차트에 접근 가능한 함수들.....	20
5. 차트와 레이아웃.....	22
5.1. 레이아웃 작성 방법에 대하여.....	22
5.2. 레이아웃의 rMateChart 노드 설정하기. ....	23
5.3. 레이아웃의 Options 노드 설정하기.....	24
5.3.1. 차트 제목(Caption), 부제목(SubCaption) 넣기. ....	24
5.3.2. 차트 범례(Legend) 넣기.....	24
5.3.3. 데이터 에디터 사용하기.....	25
5.4. 레이아웃의 Style 노드(CSS 적용) 설정하기.....	26
6. 차트 유형 별 레이아웃 설정하기.....	31
6.1. 차트의 공통적인 속성 알아보기.....	31
6.2. Cartesian 차트의 축에 대하여.....	32
6.2.1. CategoryAxis 와 LinearAxis 예제. ....	34
6.2.2. DateTimeAxis 와 LogAxis 예제.....	35
6.3. 칼럼 2D 차트 .....	35
6.4. 칼럼 3D 차트 .....	36
6.5. 실린더 3D 차트 .....	38
6.5.1. 실린더 3D 칼럼 차트.....	38
6.5.2. 실린더 3D 바차트.....	39
6.6. 바 2D 차트. ....	40

6.7.	바 3D 차트 .....	41
6.8.	파이, 도넛차트,.....	42
6.9.	버블 3D 차트 .....	46
6.10.	영역(Area) 차트.....	47
6.11.	플롯차트 .....	49
6.12.	라인차트.....	50
6.13.	점선(Dashed-Line) 차트.....	52
6.14.	콤비네이션 차트.....	53
6.15.	실시간 차트 .....	55
6.15.1.	데이터 개수를 기준으로 실시간 차트 표현(CategoryAxis 활용) .....	56
6.15.2.	정해진 시간을 기준으로 실시간 차트 표현(DateTimeAxis 활용) .....	57
6.15.3.	HttpServiceRepeater 로 일반 차트의 데이터를 실시간으로 바꾸기.....	58
6.16.	레이더(방사형) 차트 .....	59
6.17.	목표 대비 실적 차트.....	63
6.18.	스크롤 차트 .....	65
6.19.	브로큰 축(Broken Axis)차트 .....	68
6.20.	히스토리 차트 .....	69
6.21.	From-To 차트 .....	73
6.22.	매트릭스 차트 .....	75
6.23.	이미지 차트 .....	77
6.24.	윙 차트 .....	80
6.25.	실시간-프리미엄 차트 .....	81
6.26.	캔들스틱 차트 .....	89
6.27.	게이지 차트 .....	95
6.27.1.	Circular 게이지 .....	95
6.27.2.	Half-Circular 게이지 .....	97
6.27.3.	Cylinder 게이지.....	98
6.27.4.	Linear 게이지.....	99
6.28.	슬라이드 차트 기능 사용하기.....	100
6.29.	트리맵 차트 .....	103
6.30.	워드 클라우드 .....	104
6.31.	벡터 차트 .....	106
6.32.	에러 바 차트 .....	106
6.33.	박스 플롯 차트.....	108
6.34.	히스토그램 차트.....	109
6.35.	모션 차트 .....	110
7.	고급 사용자를 위한 rMate Chart 레이아웃 설정.....	111
7.1.	차트에 수치필드 표시하기.....	111

7.2.	차트 시리즈 아이템 각각에 컬러 지정하기.....	112
7.3.	축(Axis)에 대한 스타일 적용하기.....	113
7.3.1.	축 스타일 설명 및 예제.....	113
7.3.2.	축의 Visible 속성 사용하여 축의 유, 무 나타내기.....	116
7.3.3.	축의 위치 바꾸기.....	116
7.3.4.	세로축 2 개(듀얼축)로 각각의 데이터 표현하기.....	117
7.3.5.	수직, 수평 축의 수치에 3 자리 단위로 쉼표(,) 찍기.....	118
7.3.6.	수직, 수평 축에 통화단위 넣기.....	120
7.3.7.	수직, 수평 축을 DateTimeAxis 정의한 경우 날짜 포맷 변환하기.....	121
7.3.8.	축 제목(대표문자) 삽입하기.....	123
7.4.	차트 축을 기준으로 안쪽의 배경 꾸미기.....	124
7.4.1.	차트 안쪽 배경에 그리드 라인 삽입하기.....	124
7.4.2.	차트 안쪽 배경에 이미지 삽입하기.....	125
7.5.	차트 생성시 효과 적용하기.....	127
7.6.	마우스 오버 시 데이터팁(툴팁) 나타내기.....	128
7.7.	칼럼 차트 스택형 데이터간 연결선 잇기.....	129
7.8.	차트 아이템 클릭 시 불러지는 함수 설정하기.....	131
7.9.	사용자 정의 함수 설정하기.....	134
7.9.1.	데이터팁(툴팁) 함수 사용자 정의.....	135
7.9.2.	축 라벨 사용자 정의.....	138
7.9.3.	수치필드 사용자 정의.....	139
7.9.4.	채우기 색 사용자 정의.....	142
7.10.	차트 안 범위 지정 및 다수의 선 굵기.....	144
7.11.	확대/축소와 마우스 이동에 따른 십자가 표시하기.....	147
7.12.	차트 안에 메모 표시하기.....	150
7.13.	라인차트에서 수직선으로 데이터 출력하기.....	151
7.14.	동적으로 차트 레이아웃 또는 데이터 변경하기.....	152
7.15.	하나의 html 문서 안에 여러 개의 차트를 생성하기.....	156
7.16.	실시간 차트 예제 - 주식 모니터링 차트.....	159
7.17.	차트의 이미지 데이터 가져오기.....	161
7.18.	차트의 이미지 서버로 전송.....	162
7.19.	모바일에서의 이미지 데이터 가져오기.....	162
8.	장애인을 위한 기능.....	163
8.1.	시각 장애인을 위한 대체 텍스트.....	163
8.2.	색맹이나 색약이신 분을 위한 패턴지원.....	163
9.	테마 ( Theme ).....	164
9.1.	rMateChartH5 에서 제공하는 테마 등록하기.....	164
9.2.	제공되는 테마 적용하기.....	165

9.3.	기본 테마로 돌아가기.....	165
9.4.	테마 삭제하기 .....	165
10.	rMateChart for Flash 와 rMateChart for HTML5 의 차이점 설명.....	166
10.1.	지원하지 않는 기능.....	166
10.2.	플래쉬 차트의 리사이즈와 HTML5 차트의 리사이즈 차이점 .....	167
11.	제품 사용시 발생 할 수 있는 문제 해결 정보.....	169

## 1. 개요

### 1.1. rMate Chart for HTML5 의 주요 특징

rMate Chart for HTML5 는 ActiveX 기술을 배제한 순수 자바스크립트와 HTML5 의 Canvas 를 바탕으로 개발된 어플리케이션으로 사용자에게 데이터를 시각적으로 알기 쉽게 제시할 수 있는 솔루션을 제공합니다. 단순한 2 차원이 아닌 3D 와 그라데이션 효과를 넣은 2D 를 통해 수치 데이터를 표현하기 때문에 한 차원 업그레이드 된 UI 를 경험하실 수 있습니다.

<http://www.riamore.net> 의 데모 메뉴에서 강력하고 화려한 UI 의 차트를 직접 체험하실 수 있습니다.

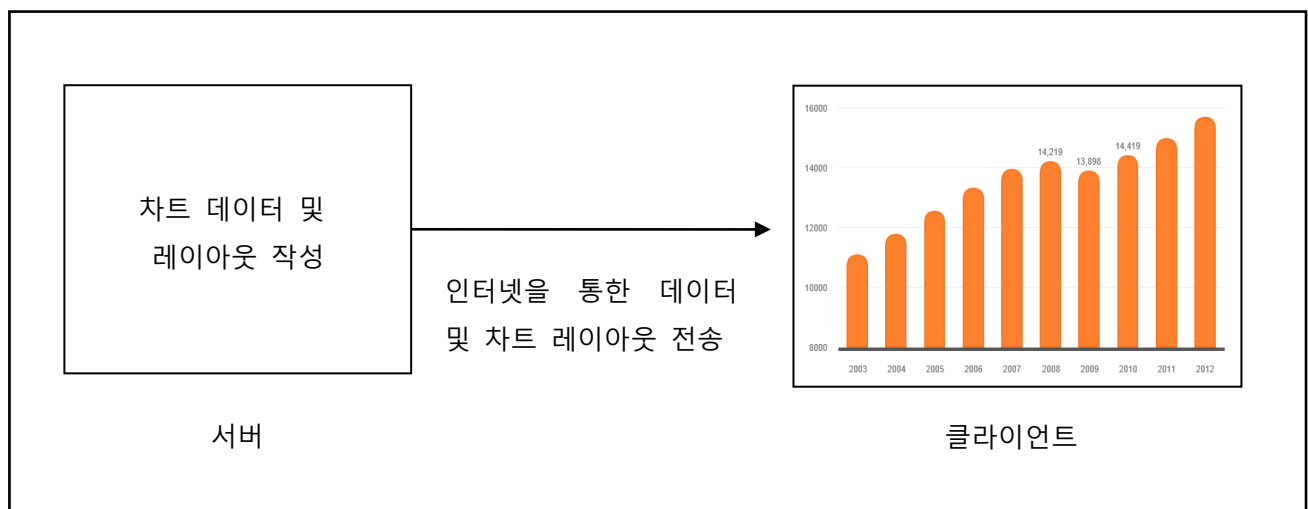
### 1.2. 시스템 요구사항.

- 서버 사이드 : 순수 자바스크립트와 HTML5 기술을 바탕으로 두기 때문에 톰캣, IIS, 웹로직, 웹스피어, 제우스, LCDS 등등 모든 WAS 서버에서 작동하며, **서버 스크립트 언어에 의존적이지 않습니다.**
- 클라이언트 사이드 : HTML5 의 Canvas 를 지원하는 브라우저가 필요합니다.

IE	파이어폭스	사파리	크롬	아이폰	안드로이드
9.0+	3.0+	3.0+	3.0+	1.0+	1.0+

단, 인터넷 익스플로러(IE)는 익스플로러캔버스(explorerCanvas) 라이브러리를 활용하여 IE 7, 8 에서 차트를 구현할 수 있으나 성능 저하 및 제약 사항이 발생합니다. 또한 브라우저의 특성에 의거하여 지원되는 기능 및 표시가 차이가 날 수 있습니다.

### 1.3. 차트 시스템 구성 환경 설명.



### <그림 1 서버와 클라이언트간의 데이터 흐름도>

rMate 차트를 생성하기 위해서는 차트 레이아웃과 수치 데이터가 필요합니다. 차트 레이아웃은 차트 유형의 선택을 시작으로 차트 컬러 변경, 차트 축 조작 등등 세밀한 부분을 제어하는 XML 파일입니다. 이에 대한 자세한 설명은 "5.차트와 레이아웃."을 참고하십시오.

차트 데이터는 **XML 형식과 배열 객체**를 지원합니다. 사용자의 환경과 편의에 따라 데이터를 작성하실 수 있습니다. 차트 데이터 작성에 대한 자세한 설명은 "3.차트 데이터 형식"을 참고하십시오.

## 1.4. 제품의 구성요소

제공된 rMate Chart for HTML5 CD 의 내용은 아래와 같습니다.

디렉토리 구조는 rMateChartH5, LicenseKey, Docs, Samples 으로 구성되어 있으며 각각 디렉토리 역할은 아래와 같습니다.

### 가. rMateChartH5

제품의 자바스크립트(rMateChartH5.js 등) 및 필요한 이미지가 위치하고 있으며, 실제 제품을 실행하는데 필요한 최소한의 파일이 들어 있습니다.

하위의 JS 디렉토리에는 제품에서 필요한 자바스크립트가 들어있으며, Assets 디렉토리에는 차트에서 사용하는 이미지와 그에 따른 CSS 가 들어 있습니다. 적용시 rMateChartH5 디렉토리를 전체로 서버에 올리고 해당 url 을 적용하시면 작업이 편리합니다.

### 나. LicenseKey

rMate Chart 라이선스가 있는 폴더입니다. (rMateChartH5License.js)

### 다. Docs

설명서는 일반 사용설명서와 레이아웃 API 두 가지 형태로 제공됩니다.

Docs->api 폴더에 레이아웃 작성을 위한 문서가 있습니다. API 는 Docs->api->index.html 을 실행하여 볼 수 있습니다.

### 라. Samples

각종 예제를 볼 수 있도록 작성된 html 과 xml 파일입니다.

## 1.5. 차트 리스트

아래 표는 rMateChartH5 의 차트 리스트를 나타냅니다. 표현하고자 하는 차트 및 데이터 유형에 따라 레이아웃 파일과 js 파일을 선택 하십시오. 제공된 레이아웃 파일은 기본 설정이며 사용자의 취향에 맞게 변경 하실 수 있습니다. 이에 대한 자세한 사항은 "차트와 레이아웃."을 참고하십시오.



- 제공된 차트 js 파일 및 레이아웃 파일은 구매자의 라이선스에 따라 상이한 부분이 있을 수 있습니다.

파일명	설 명
rMateChartH5.js	컬럼, 바, 파이, 라인, 영역 등의 기본 차트
rMateRadarChartH5.js	레이더 차트
rMateHistoryChartH5.js	히스토리 차트
rMateRealtimeChartH5.js	실시간 차트
rMateScrollChartH5.js	스크롤 차트
rMateMatrixChartH5.js	매트릭스 차트
rMateImageChartH5.js	이미지 차트
rMateCandleChartH5.js	캔들 차트
rMateWingChartH5.js	윙 차트
rMateBrokenChart.js	브로큰 차트
rMateRealtimePremiumH5.js	실시간-프리미엄 차트
rMateGaugeChartH5.js	게이지 차트
rMateVectorChartH5.js	벡터 차트
rMateErrorChartH5.js	에러 바 차트
rMateHistogramChartH5.js	히스토그램 차트
rMateMotionChartH5.js	모션 차트
rMateBoxPlotChartH5.js	박스 플롯 차트
rMateTreeMapChartH5.js	트리 맵 차트
rMateWordCloudChartH5.js	워드 클라우드 차트
rMateIntegrationH5.js	해당 라이선스에서 생성 가능한 모든 차트 내포(통합버전)

## 2. 기본적인 차트의 생성.

### 2.1. rMate Chart for HTML5 라이선스 등록하기

정상적인 차트를 생성하기 위해서는 rMate Chart for HTML5 라이선스를 등록하셔야 합니다. 제공된 시디의 다음경로에 있는 파일이 차트에 삽입할 라이선스입니다.

- /LicenseKey/rMateChartH5License.js

위 자바스크립트 파일을 <head> 태그 안에 삽입하시면 됩니다. 다른 작업을 하실 필요는 없습니다.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>

<!--rMateChartH5 라이선스 등록 완료 모습 -->
<script src="rMateChartH5License.js" type="text/javascript" language="javascript"></script>

...

...

...
```

<예제 1 rMate Chart 라이선스 등록 예제>

### 2.2. 기본적인 차트 생성하기

제공된 샘플을 참고하여 3D 칼럼 싱글 데이터 차트를 생성하여 보도록 하겠습니다.

제공된 제품에서 아래의 경로를 따라 각각의 파일을 복사하여 작업폴더 안에 복사하십시오.

(아래의 예제는 작업폴더안에 아래의 경로를 그대로 만들었다고 가정한 예제입니다.)

- 차트 JS 라이브러리 : rMateChartH5/JS/rMateChartH5.js
- 라이선스 파일 : LicenseKey/rMateChartH5License.js
- 샘플 HTML 파일 : Samples/Column\_3D.html
- 차트 CSS 파일 : rMateChartH5/Assets/Css/rMateChartH5.css

\* 정상적인 차트 생성을 위해 라이선스 파일은 반드시 포함시켜야 합니다.

다음은 Column\_3D.html 파일의 내용입니다.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<meta http-equiv="Content-Script-Type" content="text/javascript" />
<meta http-equiv="Content-Style-Type" content="text/css" />
<meta http-equiv="X-UA-Compatible" content="IE=edge" />

<link rel="stylesheet" type="text/css" href="./rMateChartH5Sample.css" />

<!-- IE7, 8 에서 차트 생성하고자 하는 경우 -->
<!--[if IE]> <script language="javascript" type="text/javascript" src="./rMateChartH5/JS/excanvas.js"
"> </script> <![endif]-->

<!-- rMateChartH5 라이브러리 라이선스 키입니다. 반드시 포함시키십시오. -->
<script language="javascript" type="text/javascript"
src="./LicenseKey/rMateChartH5License.js"> </script>

<!-- rMateChartH5 에서 사용하는 스타일 -->
<link rel="stylesheet" type="text/css" href="./rMateChartH5/Assets/Css/rMateChartH5.css"/>

<!-- 실제적인 rMateChartH5 라이브러리 -->
<script language="javascript" type="text/javascript"
src="./rMateChartH5/JS/rMateChartH5.js"> </script>

<script type="text/javascript">

// -----차트 설정 시작-----

// rMate 차트 생성 준비가 완료된 상태 시 호출할 함수를 지정합니다. <4.1 chartVars 설정하기
참고>
var chartVars = "rMateOnLoadCallFunction=chartReadyHandler";

// rMateChart 를 생성합니다.
// 파라미터 (순서대로)
// 1. 차트의 id ( 임의로 지정하십시오. )
// 2. 차트가 위치할 div 의 id (즉, 차트의 부모 div 의 id 입니다.)
```

```
// 3. 차트 생성 시 필요한 환경 변수들의 묶음인 chartVars
// 4. 차트의 가로 사이즈 (생략 가능, 생략 시 100%)
// 5. 차트의 세로 사이즈 (생략 가능, 생략 시 100%)
rMateChartH5.create("chart1", "chartHolder", chartVars, "100%", "100%");

// 차트의 속성인 rMateOnLoadCallFunction 으로 설정된 함수.
// rMate 차트 준비가 완료된 경우 이 함수가 호출됩니다.
// 이 함수를 통해 차트에 레이아웃과 데이터를 삽입합니다.
// 파라미터 : id - rMateChartH5.create() 사용 시 사용자가 지정한 id 입니다.
function chartReadyHandler(id) {
    document.getElementById(id).setLayout(layoutStr);
    document.getElementById(id).setData(chartData);
}

// 스트링 형식으로 레이아웃 정의. <5.차트와 레이아웃. 참고>
var layoutStr =
'<rMateChart backgroundColor="0xFFFFF" cornerRadius="12" borderStyle="solid">'
+'    <Options>'
+'        <Caption text="Annual Report"/>'
+'    </Options>'
+'    <NumberFormatter id="numFmt" precision="0"/>'
+'    <Column3DChart showDataTips="true">'
+'        <horizontalAxis>'
+'            <CategoryAxis categoryField="Month" />'
+'        </horizontalAxis>'
+'        <series>'
+'            <Column3DSeries labelPosition="inside" yField="Profit"
displayName="Profit">'
+'                <showDataEffect>'
+'                    <SeriesInterpolate/>'
+'                </showDataEffect>'
+'            </Column3DSeries>'
+'        </series>'
+'    </Column3DChart>'
+'</rMateChart>';

// 차트 데이터 <3.2. 배열 형식의 데이터 작성 및 차트에 삽입하기. 참고>
var chartData = [ {"Month": "Jan", "Profit": 10000},
```

```

    {"Month":"Feb", "Profit":15000},
    {"Month":"Mar", "Profit":12000},
    {"Month":"Apr", "Profit":30200},
    {"Month":"May", "Profit":28000},
    {"Month":"Jun", "Profit":12000},
    {"Month":"Jul", "Profit":22000},
    {"Month":"Aug", "Profit":13000},
    {"Month":"Sep", "Profit":22000},
    {"Month":"Oct", "Profit":29000},
    {"Month":"Nov", "Profit":18000},
    {"Month":"Dec", "Profit":30000} ];

// -----차트 설정 끝 -----

</script>
</head>
<body>

<div class="container">
    <div class="header">
        <p>rMate LineChart</p>
    </div>
    <div class="desc">
        Line Curve
    </div>
    <div class="content">
        <!-- 차트가 삽입될 DIV -->
        <div id="chartHolder" style="width:600px; height:400px;">
        </div>
    </div>
</div>
</body>
</html>

```

<예제 2 Html 문서에 차트 삽입하기>

사용자가 웹 페이지를 작성할 때 지켜야 할 사항은 다음과 같습니다.

1. HTML 의 doctype 을 지정하십시오. (다른 doctype 이어도 무방합니다)

```
<!DOCTYPE html>
```

2. rMateChartH5License.js 와 rMateChartH5.js , rMateChartH5.css 를 포함시켜 주십시오.

```
<script language="javascript" type="text/javascript"
src="../LicenseKey/rMateChartH5License.js"></script>
<script language="javascript" type="text/javascript"
src="../rMateChartH5/JS/rMateChartH5.js"></script>
<link rel="stylesheet" type="text/css" href="../rMateChartH5/Assets/rMateChartH5.css"/>
```

3. 차트가 생성될 Div 태그를 지정하여 영역을 확보해 주십시오.

```
<div id="chartHolder" style="width:600px; height:400px;"></div>
```

4. 차트의 레이아웃과 데이터를 작성하여 주십시오.

5. rMateChartH5.create() 함수 호출로 차트를 생성합니다.

```
// rMateChart 를 생성합니다.
// 파라미터 (순서대로)
// 1. 차트의 id ( 임의로 지정하십시오. )
// 2. 차트가 위치할 div 의 id (즉, 차트의 부모 div 의 id 입니다.)
// 3. 차트 생성 시 필요한 환경 변수들의 묶음인 chartVars
// 4. 차트의 가로 사이즈 (생략 가능, 생략 시 100%)
// 5. 차트의 세로 사이즈 (생략 가능, 생략 시 100%)
rMateChartH5.create("chart1", "chartHolder", chartVars, "100%", "100%");
```

6. IE 7, 8 에서 차트가 보여지길 원한다면 excanvas.js 파일을 포함시켜 주십시오.

IE 7, 8 에서는 HTML5 가 지원되지 않습니다. 그러나 excanvas.js 로 HTML5 의 canvas 를 비교적 가깝게 표현할 수 있습니다. (단, 성능 저하 및 제약 사항이 발생합니다.)

```
<!--[if IE]><script language="javascript" type="text/javascript"
src="../rMateChartH5/JS/excanvas.js"></script><![endif]-->
```

excanvas.js 에 대한 사항은 <http://excanvas.sourceforge.net/> 참고하십시오.

## 2.3. 차트 버전 정보 보기

rMateChart for HTML5 의 버전 정보는 향후 기술 지원이나 업그레이드 시 반드시 필요합니다. 보통 제공된 시디에 버전 정보가 함께 표시되나 이미 제작된 웹페이지의 차트 버전을 확인하고자 하는 경우 다음과 같이 확인 할 수 있습니다.

자바스크립트에서 rMateChartH5.create() 함수로 차트를 작성하였듯이 rMateChartH5.version 을 콘솔이나 alert 창으로 띄워 출력해 볼 수 있습니다.

```
alert(rMateChartH5.version);
```

## 3. 차트 데이터 형식

rMate Chart 의 수치 데이터는 **XML 형식과 배열 형식**을 지원합니다. 사용자는 몇 가지 규칙에 따라 차트 데이터를 작성할 필요가 있습니다. 규칙에 의거하여 단일 데이터(차트의 싱글 시리즈)와 다중 데이터(차트의 멀티 시리즈)를 XML 형식과 배열형식으로 각각 변환하여 어떻게 차트에 삽입하는지를 이번 장에서 설명합니다.

### 3.1. XML 형식의 데이터 작성 및 차트에 삽입하기.

XML 형식으로 데이터를 작성 할 때 반드시 준수해야 할 점은 한 개의 데이터를 감싸는 노드는 반드시 <item>으로 시작하여 </item>으로 끝을 맺어야 합니다. rMateChartH5 는 <item>의 개수만큼 데이터를 출력하게 됩니다. 데이터 XML 파일에서 <item></item>이 존재 하지 않는다면 차트는 데이터를 표현하지 않습니다.

단일 데이터(즉, 차트의 싱글 시리즈)는 표현하고자 하는 수치가 한 개일 때 유효하며, 다중 데이터(즉, 차트의 멀티 시리즈)는 표현하고자 하는 수치가 2 개 이상일 때 유효합니다. 매출(Revenue), 비용(Cost), 이윤(Profit)을 바탕으로 월간 보고서를 차트로 표현하기 위해 데이터를 작성해 보도록 하겠습니다.

월(Month)	매출(Revenue)	비용(Cost)	이윤(Profit)
Jan.	10,000	5,000	5,000
Feb.	15,000	7,000	8,000
Mar.	12,000	6,000	6,000
Apr.	30,200	4,000	26,200
May.	28,000	10,000	18,000
Jun.	12,000	5,000	7,000
Jul.	22,000	10000	12,000
Aug.	13,000	6,000	7,000

Sep	22,000	10,000	12,000
Oct.	29,000	8,000	11,000
Nov.	18,000	7,500	10,500
Dec.	30,000	12,000	28,000

### <XML 형식의 데이터>

```

<items>
  <item>
    <Month>Jan</Month>
    <Revenue>10000</ Revenue >
    <Cost>5000</Cost>
    <Profit>5000</Profit>
  </item>
  <item>
    <Month>Feb</Month>
    <Revenue>15000</Revenue>
    <Cost>7000</Cost>
    <Profit>8000</Profit>
  </item>
  .
  .
  .
  <item>
    <Month>Dec</Month>
    <Revenue>30000</Revenue>
    <Cost>12000</Cost>
    <Profit>18000</Profit>
  </item>
</items>
  
```

**XML 형식으로 데이터를 변환할 때 데이터 한 개의 시작과 끝은 반드시 item 이여야 합니다.**

### <예제 3 XML 형식으로 데이터 변환>

위와 같이 다중 데이터(멀티 시리즈)를 XML 유형으로 표현할 수 있습니다.



예를 들어 위에 작성한 XML 데이터를 "singleData.xml" 로 저장하였다면 아래와 같이 차트에 singleData.xml 의 경로(또는 URL)를 넘겨줘야 합니다.

```
document.getElementById("chart1").setDataURL(http://demo.riamore.net/HTML5demo/chart/Samples/DataXml/singleData.xml);
```

참고 : 차트의 setDataURL 은 RPC 이기 때문에 확장자가 xml 일 필요가 없습니다. 중요한 점은 응답으로 받는 XML 형식입니다. 따라서 서버 사이드에 맞게(jsp, php 등) XML 을 작성하는 모듈이 있다면 해당 모듈의 주소가 가능합니다.

### 3.2. 배열 형식의 데이터 작성 및 차트에 삽입하기.

rMateChartH5 는 사용자의 편의를 위하여 배열 형식의 데이터를 차트에 삽입할 수 있도록 하였습니다. 그 방법에 대한 설명입니다.

매월 이윤에 대해 자바스크립트에서 배열 형식으로 데이터를 작성해 보도록 하겠습니다.

월(Month)	이윤(Profit)
Jan.	10,000
Feb.	15,000
Mar.	12,000
Apr.	30,200
May.	28,000
Jun.	12,000
Jul.	22,000
Aug.	13,000
Sep	22,000
Oct.	29,000
Nov.	18,000
Dec.	30,000

```
var chartData = [{"Month": "Jan", "Profit": 10000},
  {"Month": "Feb", "Profit": 15000},
  {"Month": "Mar", "Profit": 12000},
  {"Month": "Apr", "Profit": 30200},
  {"Month": "May", "Profit": 28000},
  {"Month": "Jun", "Profit": 12000},
  {"Month": "Jul", "Profit": 22000},
  {"Month": "Aug", "Profit": 13000},
  {"Month": "Sep", "Profit": 22000},
  {"Month": "Oct", "Profit": 29000},
  {"Month": "Nov", "Profit": 18000},
  {"Month": "Dec", "Profit": 30000}];
```

<예제 4 배열형식으로 데이터 변환>

위와 같이 배열 형식으로 변환할 수 있습니다.

3.1 장에서 작성된 매출(Revenue), 비용(Cost), 이윤(Profit)을 바탕으로 작성된 다중 데이터를 배열형식으로 변환하면 다음과 같습니다.

```
var chartData = [{"Month": "Jan", "Revenue": 10000, "Cost": 5000, "Profit": 5000},
```

```
{ "Month": "Feb", "Revenue": 15000, "Cost": 7000, "Profit": 8000 },  
{ "Month": "Mar", "Revenue": 12000, "Cost": 6000, "Profit": 6000 },  
{ "Month": "Apr", "Revenue": 30200, "Cost": 4000, "Profit": 26200 },  
{ "Month": "May", "Revenue": 28000, "Cost": 10000, "Profit": 18000 },  
{ "Month": "Jun", "Revenue": 12000, "Cost": 5000, "Profit": 7000 },  
{ "Month": "Jul", "Revenue": 22000, "Cost": 10000, "Profit": 12000 },  
{ "Month": "Aug", "Revenue": 13000, "Cost": 6000, "Profit": 7000 },  
{ "Month": "Sep", "Revenue": 22000, "Cost": 10000, "Profit": 12000 },  
{ "Month": "Oct", "Revenue": 29000, "Cost": 8000, "Profit": 21000 },  
{ "Month": "Nov", "Revenue": 18000, "Cost": 7500, "Profit": 10500 },  
{ "Month": "Dec", "Revenue": 30000, "Cost": 12000, "Profit": 18000 } ];
```

<예제 5 배열형식으로 다중 데이터 변환>

이와 같이 작성된 차트 데이터를 차트에 삽입하는 방법은 아래와 같습니다.

```
document.getElementById("chart1").setData(chartData);
```

## 4. 차트 설정 및 연동방식

### 4.1. chartVars 설정하기

chartVars 변수는 rMateChartH5 의 데이터, 레이아웃 등을 설정할 수 있는 가장 기본적인 변수입니다. 이 변수는 스크립트 단에서 차트를 생성할 때 파라미터로 함께 삽입하게 됩니다.

chartVars 변수 설정은 다음과 같은 규칙을 따릅니다.

- chartVars 변수는 스트링입니다.
- 두 개 이상의 설정을 하기 위해 구분자 & 를 사용합니다. 예를 들어 데이터 XML URL 과 레이아웃 XML URL 을 설정하는 방법은 아래와 같습니다

```
var layoutURL = "/Column_3D_Layout.xml";
var chartVars = "layoutURL="+layoutURL;

var dataURL = "/singleData.xml";
chartVars += "&dataURL="+dataURL;
```

chartVars 로 설정 가능한 속성값은 다음과 같습니다.

속성명	유효값	설명
layoutURL	URL 주소	레이아웃 URL 경로
dataURL	URL 주소	데이터 URL 경로
rMateOnLoadCallFunction	자바스크립트 함수명	차트 준비가 완료된 경우 호출할 함수 설정 (최초 1번만 호출됨) 파라미터 : id - 차트 생성 시 사용자가 지정한 id
displayCompleteCallFunction	자바스크립트 함수명	차트 데이터가 모두 렌더링된 경우 호출할 함수 설정 파라미터 : id - 차트 생성 시 사용자가 지정한 id
useDataEditor	true, false	해당 페이지의 차트에 데이터 에디터를 사용할지 설정합니다.
usePattern	true, false	해당 페이지의 차트에 색맹, 색약을 가지고 계신 분들을 위한 패턴 이미지를 출력할지 설정합니다.

- rMateOnLoadCallFunction 과 displayCompleteCallFunction 의 차이점.

rMateOnLoadCallFunction 은 차트 준비가 완료된 후 호출되는 함수 입니다. 이 함수는 알메이트 차트 준비가 완료되었기에 데이터, 레이아웃을 삽입할 때 많이 쓰입니다.

그러나 displayCompleteCallFunction 은 차트에 주어진 데이터와 레이아웃을 바탕으로 차트 출력이 완전히 완료되었을 경우 호출되는 함수입니다. 따라서 이 함수가 호출되었을 경우에는 화면에 차트 출력이 끝난 것으로 볼 수 있습니다. 차트 출력이 마무리 되고 어떤 작업을 해야 할 경우 유용하게 쓸 수 있는 함수입니다.

- chartVars 사용 예.

```
var rMateOnLoadCallFunction = "rMateChartOnLoad";
chartVars = "rMateOnLoadCallFunction="+rMateOnLoadCallFunction;

function rMateChartOnLoad(id)
{
    ...
    ...
}
```

위와 같이 chartVars 를 설정한 후 실제로 차트를 생성하는 함수에, 설정한 chartVars 를 파라미터로 넣어줍니다.

```
rMateChartH5.create("chart1", "chartHolder", chartVars, 500, 500);
```

## 4.2. 차트에 접근 가능한 함수들

사용자의 편의를 위해 rMate Chart 는 다음과 같은 함수를 열어 놓았습니다.

1. setDataType() : 차트에 적용하는 데이터의 형태를 설정합니다.
2. setDataURL(value) : 데이터 XML URL 경로를 차트에 삽입합니다.
3. setLayoutURL (value): 레이아웃 XML URL 경로를 차트에 삽입합니다.
4. setData (value): 배열 형태 또는 XML 스트링 형태의 데이터를 차트에 삽입합니다.
5. setLayout (value): 스트링 형태의 레이아웃을 차트에 삽입합니다.
6. setSlideDataSet (value): 슬라이더 기능에서 사용할 데이터 셋을 설정합니다.
7. setSlideLayoutSet (value): 슬라이더 기능에서 사용할 레이아웃 셋을 설정합니다.
8. saveAsImage(): 차트 이미지를 이미지로 변환하려 할 때 차트의 스냅샷을 얻어 냅니다.
9. getSnapshot(): 차트 이미지 서버에 전송할 때 base64 로 인코딩된 형태의 스냅샷을 얻어냅니다.

10. `resize()`: 차트 전체의 크기를 변경하고 싶을 경우 호출 합니다.
11. `legendCheck(value)`: 레이아웃에서 Legend 를 `useVisibleCheck='true'`로 설정 했을 경우 `value` 에 해당하는 인덱스의 Series 를 선택/해제 할 수 있습니다.
12. `legendAllCheck(value)`: 레이아웃에서 Legend 를 `useVisibleCheck='true'`로 설정 했을 경우 범례의 전체선택(true)/전체해제(false) 를 제어 할 수 있습니다.
13. `showAdditionalPreloader()`: 프리로더를 강제로 표시합니다.
14. `removeAdditionalPreloader()`: 표시된 프리로더를 제거합니다.
15. `visibleItemSize()`: 스크롤 차트 출력 후 화면에 보여지는 아이템 개수를 제어할 수 있습니다.
16. `hasNoData()`: 데이터가 존재하지 않는다는 창을 차트 위에 출력합니다.
17. `changeScrollBarSize()`: 기본 스크롤바의 크기를 변경 할 수 있습니다.
18. `showDataEditor()`: 숨겨져 있는 데이터 에디터를 보여지도록 설정합니다.
19. `hideDataEditor()`: 출력되어져 있는 데이터 에디터를 숨깁니다.
20. `getCSVData()`: 출력되어져 있는 차트 데이터를 CSV 형태로 반환합니다.
21. `setDataDrillDown(value)`: 차트에 대해 드릴다운 데이터를 보여주려 할 때 데이터를 설정합니다.
22. `setDataDrillDownURL(url)`: 드릴 다운에 대한 데이터를 URL 형식으로 설정합니다.
23. `getDrillDownDepth()`: 현재 드릴 다운되어있는 depth 를 반환합니다.
24. `sliderPlay()`: 모션 차트일 경우 특정 버튼으로 슬라이더를 진행 하도록 합니다.
25. `sliderPause()`: 진행중인 슬라이더를 멈추도록 합니다.
26. `sliderReset()`: 슬라이더를 초기화 하도록 합니다.

위 함수들을 활용하여 rMate Chart 는 총 6 가지 방법으로 데이터와 레이아웃을 삽입할 수 있습니다. 이것을 정리한 표는 아래와 같습니다.

	레이아웃	데이터	사용 함수
첫 번째 방법	XML 경로	XML 경로	<code>setLayoutURL(경로)</code> , <code>setDataURL(경로)</code>
두 번째 방법	XML 경로	배열 형태	<code>setLayoutURL(경로)</code> , <code>setData(배열)</code>
세 번째 방법	XML 경로	XML 스트링	<code>setLayoutURL(경로)</code> , <code>setData(스트링)</code>
네 번째 방법	스트링 형태	XML 경로	<code>setLayout(스트링)</code> , <code>setDataURL(경로)</code>
다섯 번째 방법	스트링 형태	배열 형태	<code>setLayout(스트링)</code> , <code>setData(배열)</code>
여섯 번째 방법	스트링 형태	XML 스트링	<code>setLayout(스트링)</code> , <code>setData(스트링)</code>
일곱 번째 방법	스트링 형태	CSV 경로	<code>setType("csv")</code> <code>setLayout(스트링)</code> <code>setDataURL(경로)</code>

<표 1 차트에 레이아웃과 데이터를 삽입하는 방법을 나타낸 표>

위 함수를 이용하여 레이아웃을 동적으로 바꾸거나 변경된 데이터를 동적으로 삽입할 수 있습니다. 이에 대한 예제는 주어진 샘플의 Chart Properties -> **데이터 생성 및 연동방식** 을 참고하여 주십시오.

표 1에 해당하는 차트에 레이아웃과 데이터를 삽입 후 차트에 접근하여 legendAllCheck 함수를 실행할 경우에는 차트에 **displayCompleteCallFunction** 을 설정하여 해당 콜백 함수에서 차트에 접근하여 legendAllCheck 함수를 실행하도록 하십시오.

## 5. 차트와 레이아웃.

### 5.1. 레이아웃 작성 방법에 대하여.

rMate 차트의 모든 레이아웃은 <rMateChart>태그로 시작하여 </rMateChart>태그로 끝납니다. 레이아웃은 크게 Options 와 차트(칼럼, 바, 파이 등의 차트), Style 세 부분으로 구분됩니다. 사용자는 Options 태그에서 차트의 제목(Caption)과 부제목(SubCaption) 그리고 범례(Legend)를 삽입할 수 있습니다.

레이아웃을 작성하는 방법은 기본적으로 XML 형식입니다.

rMate 차트는 ID 에 의한 객체 접근을 지원합니다. 예를 들어 Column3DSeries 객체의 속성으로 fill 이 존재합니다. 이 fill 속성의 속성값으로 SolidColor 인스턴스를 할당합니다. 이를 표현한 일반적인 방법은 아래와 같습니다.

```
<rMateChart>
  <Column3DChart showDataTips="true">
    .
    .
    .
    <series>
      <Column3DSeries yField="Profit">
        <fill>
          <SolidColor color="0xFF0000">
        </fill>
      </Column3DSeries>
    </series>
  </Column3DChart>
</rMateChart>
```

<예제 6 일반적인 속성 값 할당 법>

그러나 사용자의 편의를 위해 <예제 7 ID 에 의한 속성 값 할당 법>과 같이 SolidColor 인스턴스를 미리 생성한 후 그 ID 를 통해 SolidColor 인스턴스를 활용할 수 있습니다. 이는 여러 객체들이 하나의 SolidColor 인스턴스에 접근하고자 할 때 불필요하게 SolidColor 인스턴스를 생성 하지 않고 미리 만들어진 하나의 SolidColor 에 접근하기 때문에 효율적입니다. 또한 어느 특정 객체를 반드시 참조해야 하는 경우에도 효율적입니다.

```

<rMateChart>
  <SolidColor id="color1" color="0xFF0000"/>
  <Column3DChart showDataTips="true">
    .
    .
    .
    <series>                                //객체 ID인 경우엔 시작과 끝을 중괄호({,})로 묶음
      <Column3DSeries yField="Profit" fill="{color1}"/>
      <Column3DSeries yField="Cost" fill="{color1}"/>
    </series>
  </Column3DChart>
</rMateChart>

```

<예제 7 ID 에 의한 속성 값 할당 법>

인스턴스 ID 에 의한 접근을 활용한 좋은 예는 수직 축 2 개를 생성하여 서로 다른 시리즈가 각각의 축을 참조하는 것입니다.

앞으로 레이아웃을 작성할 때(차트 종류에 관계없이) Chart 와 Series 가 자주 나오게 될 것입니다. 이에 대한 개념을 정리하면 아래와 같습니다.

	설 명	비고
차트(Chart)	실질적인 차트입니다. 차트는 외형을 담당합니다. 축, 배경, 차트사이즈 등등이 포함됩니다. 가장 중요한 역할로 시리즈의 위치(좌표)를 지정합니다.	ColumnChart, Column3DChart, PieChart, Pie3DChart, BarChart 등등
시리즈(Series)	실질적인 데이터를 표현합니다. 표현하고자 하는 수치데이터가 3개인 경우 반드시 시리즈를 3개 정의할 필요가 있습니다.	ColumnSeries, Column3DSeries, BarSeries, Pie3DSereis 등등

<표 2 차트와 시리즈 설명표>

## 5.2. 레이아웃의 rMateChart 노드 설정하기.

rMateChart 노드는 레이아웃의 시작과 끝을 나타냅니다. 또한 이 노드의 속성은 차트 전체의 꾸미기를 담당합니다.

아래는 rMateChart 노드의 속성 사용 예를 나타낸 것 입니다.

```
<rMateChart cornerRadius="12" borderStyle="solid" backgroundColor="0xFFFF77">
  <Options>
    <Caption text="Annual Report" />

    .....

</rMateChart>
```

<예제 8 rMateChart 노드의 속성 사용 예>

### 5.3. 레이아웃의 Options 노드 설정하기.

Options 는 레이아웃의 필수사항이 아닌 선택사항입니다. 차트에 제목이나 부제목, 범례(Legend)를 삽입하고자 할 때 작성하십시오.

```
<rMateChart>
  <Options>
    <Caption text="Annual Report"/> // 제목
    <SubCaption text="2008"/> //부제목
    <Legend/> //범례
  </Options>
  <Column3DChart showDataTips="true" width="100%" height="100%" >
    <series>
      <Column3DSeries yField="Profit" displayName="Profit">
        .
        .
        .
      </Column3DSeries>
    </series>
  </Column3DChart>
</rMateChart>
```

<예제 9 Options 태그 사용 예>

#### 5.3.1. 차트 제목(Caption), 부제목(SubCaption) 넣기.

<예제 9 Options 태그 사용 예>는 칼럼 3D 차트의 레이아웃입니다. 제목은 <Caption>태그의 text 속성에 원하는 문구를 넣으시면 됩니다.

#### 5.3.2. 차트 범례(Legend) 넣기.



차트의 범례(legend)는 보통 멀티시리즈(다중데이터 표현)인 경우 각 시리즈가 표현하는 데이터의 대표자를 표시하는 역할을 합니다.

범례(legend) 작성시에는 차트시리즈(즉, 칼럼 3D 차트인 경우 <Column3DSeries> 태그)의 속성인 displayName 을 정의 하여야 합니다. 여기서 정의한 문자열이 범례의 텍스트로 출력됩니다.

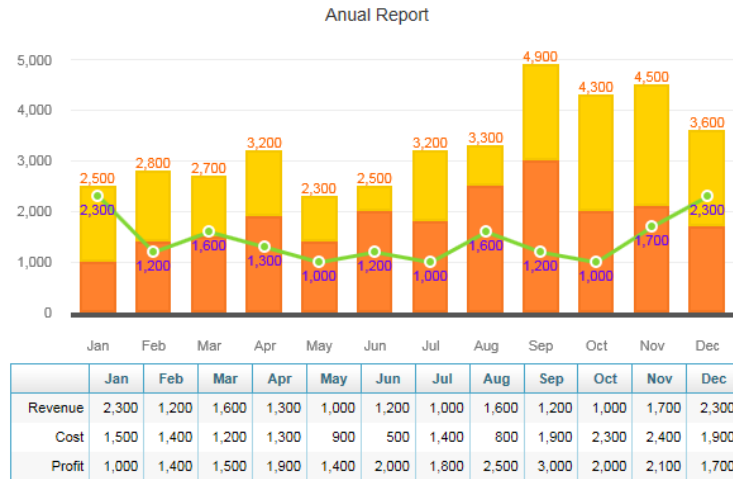
출 력 결 과			
적 용 속 성	position="right" direction="vertical" useVisibleCheck="true" vAlign="middle"	position="top" direction="horizontal" useVisibleCheck="true" hAlign="right"	position="left" direction="vertical" useVisibleCheck="false" vAlign="bottom"

<예제 10 다양한 범례 속성을 적용한 예>

### 5.3.3. 데이터 에디터 사용하기

데이터 에디터는 차트 하단에 출력되며, 에디터의 셀을 더블클릭하여 차트 데이터의 확인, 수정이 가능하게 합니다.

차트에서 데이터 에디터를 사용하는 법은 간단합니다. 차트 환경변수 chartVars 에 useDataEditor=true 를 추가하신 후 레이아웃에서 <Opionts> ~ </Options> 안에 <DataEditor./>를 설정해주시면 됩니다. 데이터 에디터 내의 속성은 제품 내의 Docs/api/inedxe.html 실행 후 DataEditor 부분을 살펴보시기 바랍니다.



< 그림 2 데이터 에디터 사용 모습 >

#### 5.4. 레이아웃의 Style 노드(CSS 적용) 설정하기.

<Style> 노드는 전반적인 스타일을 미리 정의하는 스타일시트입니다. <Style/>을 선언하는 방법에는 몇 가지 규칙이 있습니다.

첫째, <Style/> 노드는 반드시 <rMateChart> 노드의 자식 위치에 정의하십시오.

둘째, 스타일네임을 정의 할 때 시작은 도트(.)를 찍고 소문자로 시작하는 영문자로 표기하십시오.

셋째, 스타일네임을 정한 후 그에 따른 구체적인 스타일을 정의 할 때는 반드시 중괄호({})로 시작과 끝을 표시하십시오.

넷째, 속성명과 속성값(value)의 구분은 콜론(:)이며 끝은 세미콜론으로 나타냅니다.

다음은 올바른 예와 잘못된 예를 나타낸 것입니다.

| 올바른 예  | 잘못 작성 된 예  |
|--|--|
| <pre> &lt;rMateChart&gt;   &lt;Options&gt;    ...    &lt;Style&gt;     .rMateChartStyle     {       backgroundColor:0xFFFFFE; </pre> | <pre> &lt;rMateChart&gt;   &lt;Options&gt;    ...    &lt;Style&gt;     ChartStyle     [       backgroundColor:0xFFFFFE; </pre> |

<pre>borderColor:0x77EE9E; cornerRadius:12; borderThickness:3; borderStyle:solid; } &lt;Style&gt; &lt;/rMateChart&gt;</pre>	<pre>borderColor:"0x77EE9E"; cornerRadius:12; borderThickness:3; borderStyle:solid; } &lt;Style&gt; &lt;/rMateChart&gt;</pre>
---	---

<표 3 Style 작성 예>

위와 같이 rMateChartStyle 스타일을 정의하였습니다. 그렇다면 이 스타일을 적용하는 방법에 대해 보도록 하겠습니다.

<pre>&lt;rMateChart styleName="rMateChartStyle"&gt;   &lt;Options&gt;     &lt;Caption text="Annual Report"/&gt;   &lt;/Options&gt;    .....    &lt;Style&gt;     .rMateChartStyle     {       backgroundColor:0xFFFFFE;       borderColor:0x77EE9E;       cornerRadius:12;       borderThickness:3;       borderStyle:solid;     }   &lt;Style&gt;  &lt;/rMateChart&gt;</pre>
---

<예제 11 Style 적용 예>

<예제 11 Style 적용 예>를 보시면 rMateChartStyle 은 <rMateChart/> 노드의 스타일을 정의 한 것이고 적용법은 styleName="정의한 스타일이름" 즉, styleName="rMateChartStyle" 과 같습니다.

시리즈의 labelAlign 은 칼럼시리즈와 바시리즈에만 해당되는 속성입니다. 각각의 유효 속성값으로 칼럼시리즈인 경우 "top | middle | bottom" 바시리즈인 경우 "left | center | right" 입니다.

스타일을 활용한 예제와 출력 모습은 아래와 같습니다.

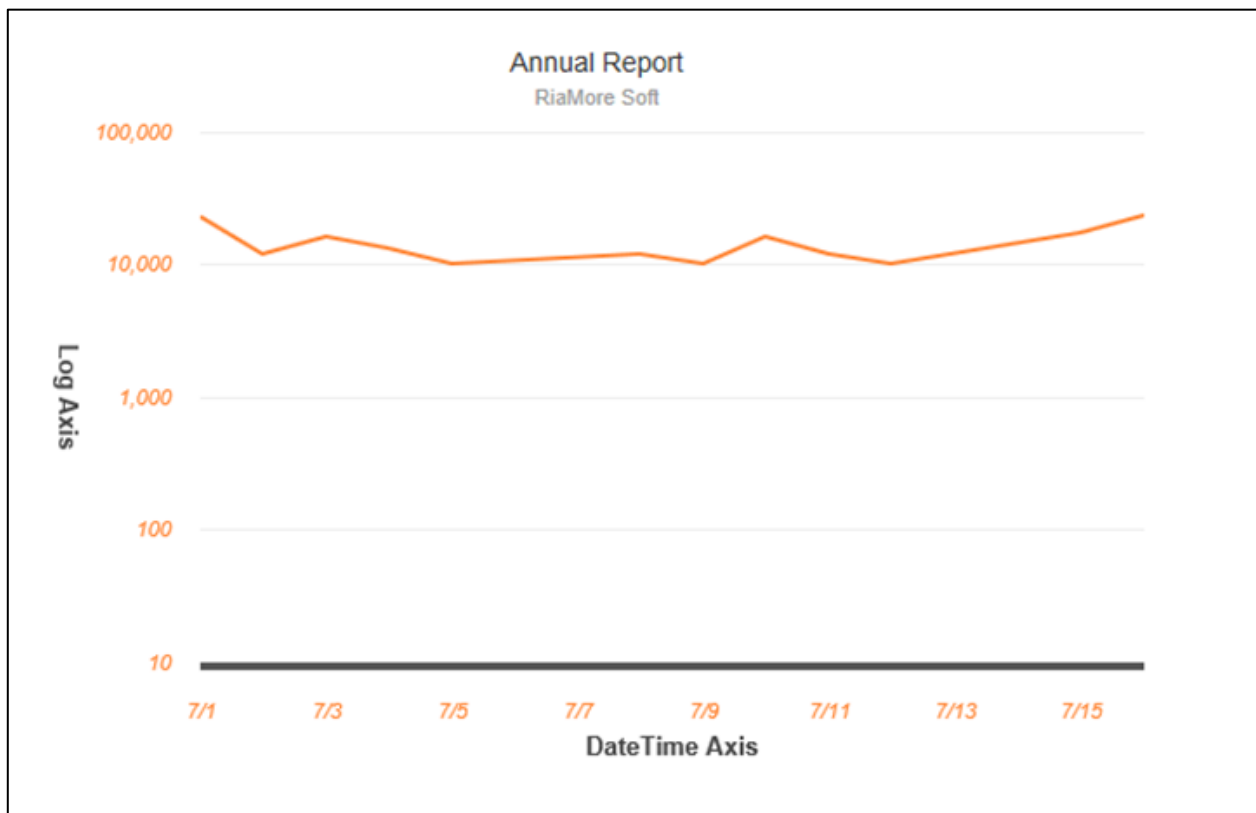
```
<rMateChart styleName= "rMateChartStyle">
  <Options>
    <Caption text= "Annual Report" styleName= "captionStyle"/>
    <SubCaption text= "RiaMore Soft" styleName= "subCaptionStyle"/>
  </Options>
  <DateFormatter id= "dateFmt" formatString= "M/D"/>
  <NumberFormatter id= "numFmt"/>
  <Line2DChart showDataTips= "true" styleName= "chartStyle">
    <horizontalAxis>
      <CategoryAxis id= "hAxis" categoryField= "Month" title= "Horizontal Axis"/>
    </horizontalAxis>
    <horizontalAxisRenderers>
      <Axis2DRenderer axis= "{hAxis}" axisTitleStyleName= "chartAxisStyle"/>
    </horizontalAxisRenderers>
    <series>
      <Line2DSeries yField= "Revenue" displayName= "Revenue">
        <showDataEffect>
          <SeriesInterpolate/>
        </showDataEffect>
      </Line2DSeries>
    </series>
  </Line2DChart>
  <Style>
    .rMateChartStyle {
      backgroundColor:#FFFFFFE;
      borderColor:#77EE9E;
      cornerRadius:12;
      borderThickness:3;
      borderStyle:solid;
    }
    .captionStyle {
      fontSize:12;
      fontFamily:Tahoma;
      fontWeight:bold;
      color:#777777;
    }
    .subCaptionStyle {
      fontSize:11;
```

```

        fontStyle:italic;
        color:#777777;
    }
    .chartStyle {
        fontSize:11;
        fontStyle:italic;
        color:#0000FF;
    }
    .chartAxisStyle {
        color : #4691E1;
        fontSize : 14;
        fontWeight : bold;
        fontStyle : italic;
    }
</Style>
</rMateChart>

```

<예제 12 Style 적용한 예제>



<그림 3 Style 적용 결과>



## 6. 차트 유형 별 레이아웃 설정하기.

### 6.1. 차트의 공통적인 속성 알아보기.

차트는 크게 축이 존재하는 Cartesian 차트와 축이 없는 Polar 차트로 나뉩니다. 축이 있는 차트의 기본적인 속성에 대한 설명은 아래와 같습니다.

속성 명	유효 값	설명
horizontalAxis	CategoryAxis, LinearAxis, DateTimeAxis, LogAxis 객체	수평축(X축)을 정의합니다.
verticalAxis	CategoryAxis, LinearAxis, DateTimeAxis, LogAxis 객체	수직축(Y축)을 정의합니다.
horizontalAxisRenderer	Axis3DRenderer, AxisRenderer	수평축의 렌더러를 정의합니다.
verticalAxisRenderer	Axis3DRenderer, AxisRenderer	수직축의 렌더러를 정의합니다.
series	차트의 시리즈 (예:Column3DSeries, Line2DSeries 등등)	차트 시리즈를 정의합니다.
annotationElements	GridLines, Image, CanvasElement, CrossRangeZoomer 등	차트 앞쪽(z-index 개념으로 상단)에 위치할 엘리먼트를 정의합니다.
backgroundElements	GridLines, Image, CanvasElement 등	차트 뒷배경에 위치할 엘리먼트를 정의합니다.
showDataTips	true   false	마우스오버시 데이터팁(툴팁)의 유무를 나타냅니다.
paddingTop	Number	위쪽 여백
paddingBottom	Number	아래 여백
paddingLeft	Number	왼쪽 여백
paddingRight	Number	오른쪽 여백
itemClickJsFunction	자바스크립트 함수명	아이템 클릭 시 차트가 호출할 함수를 나타냅니다.
dataTipJsFunction	자바스크립트 함수명	데이터팁(툴팁)을 사용자 정의할 함수를 나타냅니다.
gutterLeft	Number	차트 축을 기준으로 왼쪽 여백
gutterRight	Number	차트 축을 기준으로 오른쪽 여백
gutterTop	Number	차트 축을 기준으로 윗쪽 여백
gutterBottom	Number	차트 축을 기준으로 아래 여백

<예제 13 Cartesian 차트의 공통적인 속성 설명>

아래는 도넛, 파이차트와 같이 축이 없는 Polar 차트의 속성에 대한 설명입니다.

속성 명	유효 값	설명
innerRadius	0.0 ~ 1.0(기본값:0.0)	도넛차트와 같이 가운데 공간의 정도를 표현합니다. 1.0 에 가까울수록 가운데 공간이 커집니다.
showDataTips	true   false(기본값:false)	마우스오버시 데이터팁(툴팁)의 유무를 나타냅니다.
explodable	true   false(기본값:true)	마우스 클릭 시 도넛, 파이차트의 조각이 떨어져 나오는 효과를 나타냅니다.
series	PieGradationSeries, Pie3DSeries	차트의 시리즈를 정의합니다.
itemClickJsFunction	자바스크립트 함수명	아이템 클릭 시 차트가 호출할 함수를 나타냅니다.
dataTipJsFunction	자바스크립트 함수명	데이터팁(툴팁)을 사용자 정의할 함수를 나타냅니다.

<예제 14 Polar 차트의 공통적인 속성 설명>

## 6.2. Cartesian 차트의 축에 대하여.

파이차트, 도넛차트 외의 차트에는 X 축(수평축)과 Y 축(수직축)이 존재합니다. 축의 종류에는 크게 카테고리 유형과 수치 유형 2 개가 있습니다. 스트링 형태를 정의하는 축이 CategoryAxis 입니다. 이는 보통 차트에서 수치화 할 수 없지만 그룹화가 가능한 계열을 위한 것입니다. 예를 들면 경영부, 연구부, 회계부와 같이 수치화 될 수 없지만 일개 그룹으로 묶어 축 필드로 사용하고자 할 때 CategoryAxis 를 사용합니다.

수치 유형으로는 LinearAxis, LogAxis, DateTimeAxis 3 가지 Type 이 있습니다. LinearAxis 는 연속적인 데이터(즉, 일반적인 수치)를 위한 것이고, LogAxis 는 Log 함수에 의한 정의입니다. 마지막으로 DateTimeAxis 는 날짜, 시간과 관련이 있습니다.

아래 표는 각각의 축에 대한 속성 명과 유효 값에 대한 설명입니다.

Axis 명	속성 명	유효 값	설명
CategoryAxis	categoryField	데이터의 특정 필드 명 (예:Month)	카테고리축이 참조할 데이터의 특정 필드명을 입력하세요. 카테고리축을 정의하였을 때 반드시 입력하셔야 합니다.
	displayName	String(문자열)	데이터팁(툴팁)에 나타날 축을 대표하는 문자열입니다.



	title	String(문자열)	축의 제목(대표 문자) 입니다.
	labelJsFunction	자바스크립트 함수명	축 라벨 텍스트를 사용자 정의할 함수를 나타냅니다.
LinearAxis	interval	Number	축에 나타나는 라벨의 간격을 나타냅니다.
	displayName	String(문자열)	데이터팁(툴팁)에 나타날 축을 대표하는 문자열입니다.
	minimum	Number	축 라벨의 최소치를 나타냅니다.
	maximum	Number	축 라벨의 최대치를 나타냅니다.
	title	String(문자열)	축의 제목(대표 문자) 입니다.
	labelJsFunction	자바스크립트 함수명	축 라벨 텍스트를 사용자 정의할 함수를 나타냅니다.
DateTimeAxis	dataUnits	milliseconds, seconds, minutes, hours, days, weeks, months, years	차트가 출력할 데이터의 표시에 사용하는 단위를 지정합니다
	labelUnits	milliseconds, seconds, minutes, hours, days, weeks, months, years	라벨에 나타날 단위를 지정합니다.
	title	String(문자열)	축의 제목(대표 문자) 입니다.
	interval	Number	축에 나타나는 라벨의 시간, 날짜 간격을 나타냅니다. (라벨이 나타날 공간이 작을 경우 지정된 interval 값은 무시될 수 있습니다.)
	dataInterval	Number	dataUnits 로 지정된 그래프내의 데이터간의 간격을 지정합니다. 예를 들어 dataUnits="seconds"로 설정하고 데이터 간격은 3초 단위라면 dataInterval은 4로 줄 필요가 있습니다. 단위가 초단위이므로 차트는 그 자리를 마련합니다. dataInterval="4"는 3초 단위를 하나로 본다는 뜻으로 4번째 자리부터 그래프를 렌더링합니다. (차트 type 에 따라 무시되기도 합니다.)
	displayName	String(문자열)	데이터팁(툴팁)에 나타날 축을 대표하는 문자열입니다.
	displayLocalTime	false true(기본값:false)	true 로 설정되었을 경우는, DateTimeAxis 에 의해, 모든 데이터치가 어플리케이션을 실행하는 클라이언트 머신의 타임 존에 있다고 보여집니다. false 로 설정되었을 경우, 모든 값은 세계 표준시 (그리니치

			표준시)가 됩니다.
	labelJsFunction	자바스크립트 함수명	축 라벨 텍스트를 사용자 정의할 함수를 나타냅니다.
LogAxis	interval	10의 승수	10의 승수 단위로 라벨에 표시합니다.
	minimum	Number	축 라벨의 최소치를 나타냅니다.
	maximum	Number	축 라벨의 최대치를 나타냅니다.
	title	String(문자열)	축의 제목(대표 문자) 입니다.
	displayName	String(문자열)	데이터팁(툴팁)에 나타날 축을 대표하는 문자열입니다.
	labelJsFunction	자바스크립트 함수명	축 라벨 텍스트를 사용자 정의할 함수를 나타냅니다.

<표 4 각각의 Axis 속성 명과 유효 값 설명>

### 6.2.1. CategoryAxis 와 LinearAxis 예제.

아래 예제를 보면 수평축으로 CategoryAxis 를 정의하였습니다. CategoryAxis 의 categoryField 는 반드시 차트 데이터와 일치시켜야 합니다. 데이터의 Month 필드를 수평축의 카테고리로 활용한다는 뜻이 됩니다. 만약 축을 정의 하지 않으면 기본값으로 LinearAxis 가 됩니다.

```

<rMateChart backgroundColor="0xFFFFEE" cornerRadius="12" borderStyle="solid">
  <Options>
    <Caption text="Annual Report"/>
  </Options>
  <Column3DChart showDataTips="true">

    <horizontalAxis> //수평축으로 카테고리축 정의
      <CategoryAxis categoryField="Month" title="Category Axis" />
    </horizontalAxis>

    <verticalAxis> // 수직축으로 선형축 정의
      <LinearAxis maximum="3500" title="Linear Axis"/>
    </verticalAxis>
    <series>
      <Column3DSeries yField="Profit" displayName="Profit">
        <showDataEffect>
          <SeriesInterpolate />
        </showDataEffect>
      </Column3DSeries>
    </series>
  </Column3DChart>
</rMateChart>
  
```

<예제 15 수평축으로 카테고리축을, 수직축으로 리니어축을 정의한 예>

### 6.2.2. DateTimeAxis 와 LogAxis 예제.

```
<rMateChart backgroundColor="0xFFFFEE" cornerRadius="12" borderStyle="solid">
  <Options>
    <Caption text="Annual Report"/>
  </Options>
  <Line2DChart showDataTips="true">

    <horizontalAxis> // 수평축으로 DateTime축 정의
      <DateTimeAxis dataUnits="days" labelUnits="days" title="Date Time Axis" interval="3"
        displayName="Date" displayLocalTime="true"/>
    </horizontalAxis>

    <verticalAxis> //수직축으로 로그축 정의
      <LogAxis title="Log Axis" interval="10" minimum="10" maximum="10000" />
    </verticalAxis>
    <series>
      <Line2DSeries xField="Date" yField="Profit" displayName="Profit"/>
    </series>
  </Line2DChart>
</rMateChart>
```

<예제 16 수평축으로 DateTime 축을, 수직축으로 로그축을 정의한 예제>

**\*중요** : **DateTimeAxis** 를 사용한 경우 반드시 시리즈 **DateTime Field** 를 명시해 줘야 합니다. 예를 들어 칼럼차트의 수평축에 DateTimeAxis 를 정의 한 경우 xField 를 정의해야 합니다.(바차트의 수직축이 DateTimeAxis 인 경우 yField) 이 xField 는 데이터의 시간영역에 해당되는 필드명을 나타냅니다. 이것은 CategoryAxis 의 cateogoryField 와 동일한 의미이지만 DateTimeAxis 의 속성이 아닌 시리즈의 xField(또는 yField)에 해당 필드를 정의함으로써 DateTimeAxis 는 데이터 시간에 맞게 표현합니다.

## 6.3. 칼럼 2D 차트

칼럼 2D 차트는 <Column2DChart> 노드로 시작과 끝을 나타냅니다. <Column2DChart/> 노드의 자식 노드로 수평 축(horizontalAxis), 수직 축(verticalAxis), 시리즈(series), 배경엘레멘츠(backgroundElements) 등을 정의합니다.

아래 예제는 싱글 데이터(싱글 시리즈) 칼럼 2D 차트 예제입니다.

<Column2DChart>의 <series> 자식 노드로 <Column2DSeries>를 정의 한 것을 볼 수 있습니다. 이는 데이터 표현을 칼럼시리즈로 한다는 뜻입니다. **모든 차트의 데이터 표현은 시리즈가 담당하고 있습니다.**

그래서 시리즈의 속성을 잘 정의 하는 것이 중요합니다. 시리즈의 부모 노드 꺾인 Chart(예를 들면 Column2DChart, Bar2DChart 등등)는 시리즈의 자리배치와 축의 위치, 배경 등 외형을 담당합니다.

```
<rMateChart backgroundColor="0xFFFFEE" cornerRadius="12" borderStyle="solid">
  <Options>
    <Caption text="Anual Report"/>
  </Options>
  <Column2DChart showDataTips="true">
    <horizontalAxis>
      <CategoryAxis categoryField="Month"/>
    </horizontalAxis>
    <series>
      <Column2DSeries yField="Profit" itemRenderer="SemiCircleColumnItemRenderer">
        <showDataEffect>
          <SeriesInterpolate/>
        </showDataEffect>
        <fill>
          <SolidColor color="0xFF0000" alpha="0.5"/>
        </fill>
        <stroke>
          <Stroke color="0xFFFF00" weight="1"/>
        </stroke>
      </Column2DSeries>
    </series>
  </Column2DChart>
</rMateChart>
```

<예제 17 칼럼 차트 예제>

<Column2DSeries> 노드의 itemRenderer 는 차트의 시리즈가 어떻게 데이터를 그래픽적으로 표현할지를 결정합니다. ColumnSeries 노드의 itemRenderer 의 유효 속성값으로는 두개가 존재합니다.

- SemiCircleColumnItemRenderer
- GradientColumnItemRenderer



## 6.4. 칼럼 3D 차트

칼럼 3D 차트는 <Column3DChart> 노드로 시작과 끝을 나타냅니다. 칼럼 3D 차트의 시리즈는 <Column3DSeries>입니다.

아래는 rMate 멀티시리즈 칼럼 3D 차트 레이아웃 예제입니다.

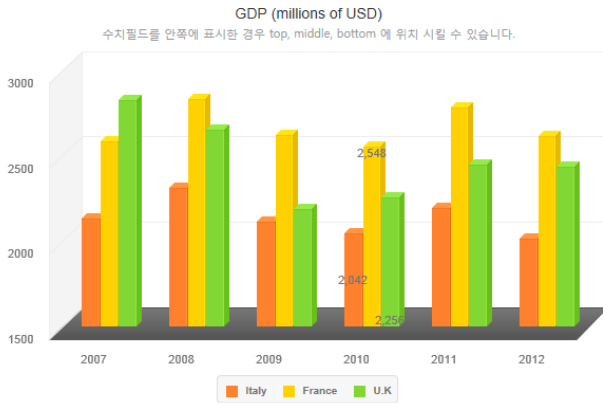
```
<rMateChart>
  <Column3DChart showDataTips="true" type="clustered">
    //칼럼차트 생성, 툴팁 보이기 true
    <horizontalAxis//X축 설정, X축은 카테고리이며 기준 필드는 Month
      <CategoryAxis categoryField="Month"/>
    </horizontalAxis>
    <series> //시리즈는 데이터의 표현을 말합니다.
      // 칼럼시리즈가 2개 반복 → Profit, Cost 데이터의 표현이 한쌍임.
      <Column3DSeries yField="Profit">
        <showDataEffect> //차트 생성시 이펙트 정의
        <SeriesInterpolate/>
        </showDataEffect>
      </Column3DSeries>
      <Column3DSeries yField="Cost"> // 차트데이터 값 중 Cost를 y축필드에 대응시킵니다.
        <showDataEffect>
        <SeriesInterpolate/>
        </showDataEffect>
      </Column3DSeries>
    </series> </Column3DChart> </rMateChart>
```

<표 5 칼럼 3D 차트 레이아웃 예제>

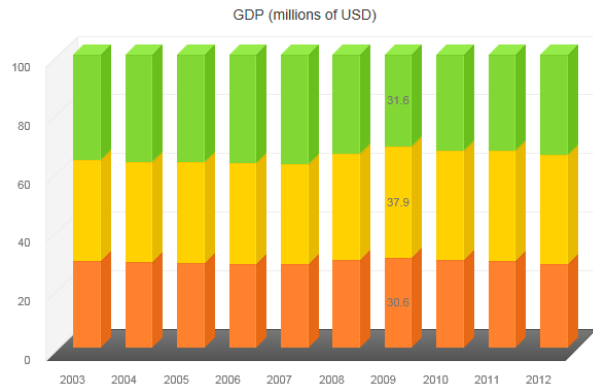
칼럼 차트의 프로퍼티 중 type 은 차트의 데이터 출력 방식을 결정합니다. type 에는 4 가지가 존재합니다.

- clustered : 일반적인 다중데이터(차트의 멀티시리즈) 방식으로 데이터를 표현합니다. (기본값)
- stacked : 데이터를 위에 쌓아 올린 방식으로 표현합니다.
- overlaid : 수치 데이터 값을 겹쳐서 표현 합니다. 주로 목표 위치와 현재 위치를 나타낼 때 많이 쓰입니다.
- 100% : 차트의 수치 데이터를 퍼센티지로 계산 후 값을 퍼센티지로 나타냅니다.

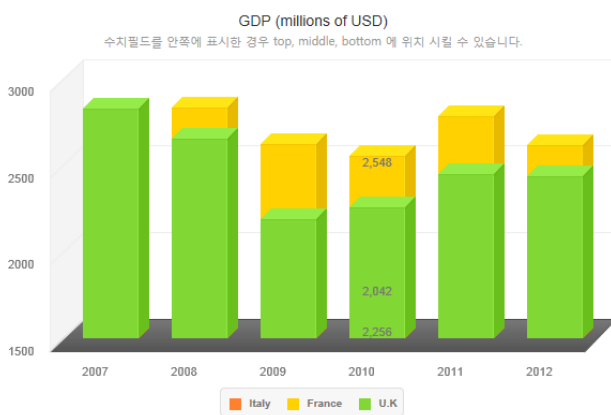
다음은 이에 대한 출력 화면입니다.



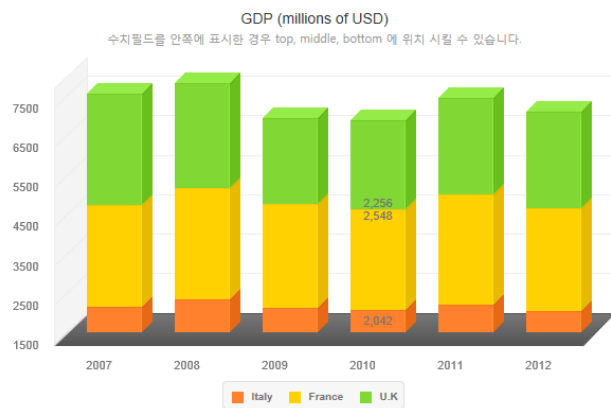
<type="clustered">



<type="100%">



<type="overlaid">



<type="stacked">

<그림 4 칼럼차트 Type 별 출력 결과>

## 6.5. 실린더 3D 차트

### 6.5.1. 실린더 3D 칼럼 차트

실린더 3D 칼럼 차트는 칼럼 3D 차트와 같습니다. 다만, 큐브 형태의 칼럼이 아닌 3D 원통 모양으로 데이터를 표현합니다.

모든 속성 및 레이아웃 작성은 3D 칼럼 차트와 동일 합니다.

작성된 3D 칼럼 차트에서 시리즈의 속성인 itemRenderer 속성에 CylinderItemRenderer 를 값으로 할당하면 실린더 3D 차트가 생성됩니다.

다음은 <표 5 칼럼 3D 차트 레이아웃 예제> 레이아웃에 아이템렌더러 속성을 정의한 예제입니다.

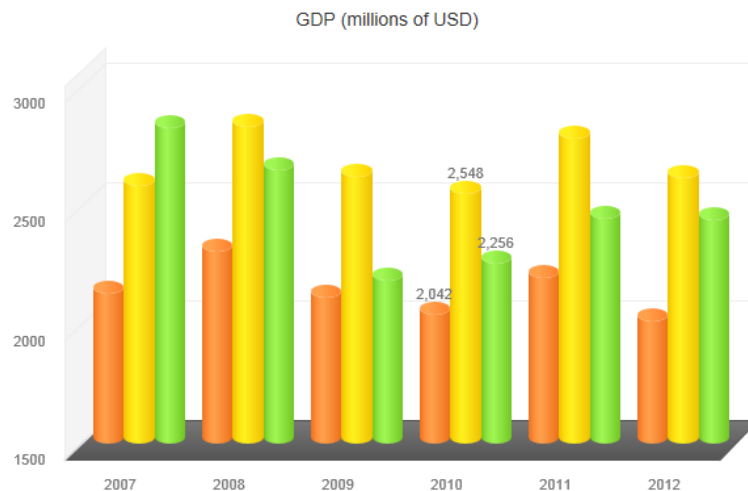
```
<rMateChart>
  <Column3DChart showDataTips="true" type="clustered">
    //칼럼차트 생성, 툴팁 보이기 true
```

```

<horizontalAxis//X축 설정, X축은 카테고리이며 기준 필드는 Month
    <CategoryAxis categoryField="Month"/>
</horizontalAxis>
<series> //시리즈는 데이터의 표현을 말합니다.
    // 칼럼시리즈가 2개 반복 → Profit, Cost 데이터의 표현이 한쌍임.
    <Column3DSeries yField="Profit" itemRenderer="CylinderItemRenderer">
        <showDataEffect> //차트 생성시 이펙트 정의
            <SeriesInterpolate/>
        </showDataEffect>
    </Column3DSeries>
    <Column3DSeries yField="Cost"// 차트데이터 값 중 Cost를 y축필드에 대응시킵니다.
        itemRenderer="CylinderItemRenderer">
            <showDataEffect>
                <SeriesInterpolate/>
            </showDataEffect>
        </Column3DSeries>
</series> </Column3DChart> </rMateChart>

```

<예제 18 실린더 3D 차트 레이아웃 예제>



<그림 5 실린더 3D 차트 결과 모습>

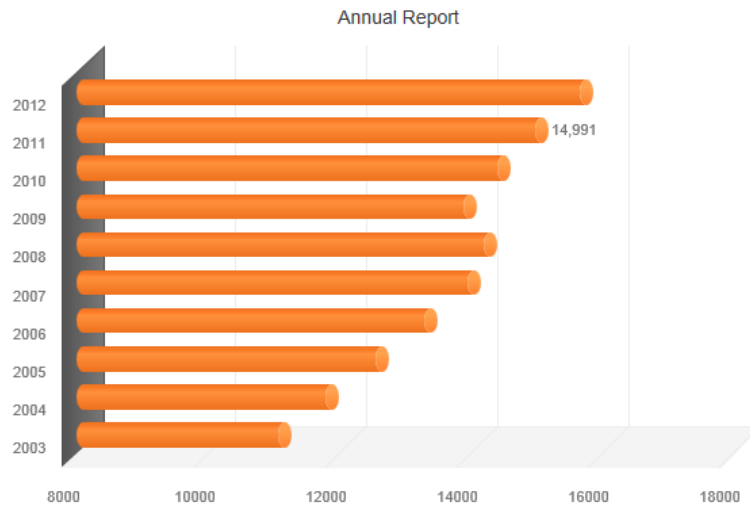
### 6.5.2. 실린더 3D 바차트

실린더 3D 바 차트는 바 3D 차트와 같습니다. 다만, 큐브 형태의 칼럼이 아닌 3D 원통 모양으로 데이터를 표현합니다.

모든 속성 및 레이아웃 작성은 3D 바 차트와 동일 합니다.

작성된 3D 바 차트에서 시리즈의 속성인 itemRenderer 속성에 **BarCylinderItemRenderer** 를 값으로 할당하면 실린더 3D 차트가 생성됩니다

레이아웃 작성은 "6.5.1 실린더 3D 칼럼 차트" 를 참고하여 주십시오.



<그림 6 실린더 3D 바차트 출력 모습>

## 6.6. 바 2D 차트.

바 차트는 칼럼차트와 비슷합니다. 다만 수치 데이터를 X 축(수평축)에 할당해야 합니다.

반드시 verticalAxis(수직축)을 정의 하여 그룹핑에 해당되는 데이터를 수직축 필드로 입력하고, 수치데이터에 해당되는 값은 수평축 필드에 할당합니다.

바 2D 차트는 <Bar2DChart> 노드로 시작과 끝을 나타냅니다. Bar 2D 차트의 itemRenderer 속성값으로 2 개의 유효값이 존재합니다.

- SemiCircleBarItemRenderer
- GradientBarItemRenderer



그 외 속성과 속성값은 제품내에 포함된 API(/Docs/api)를 참고하십시오.

```
<rMateChart backgroundColor="0xFFFFEE" cornerRadius="12" borderStyle="solid">
  <Options>
    <Caption text="Annual Report"/>
  </Options>
  <Bar2DChart showDataTips="true">
    <verticalAxis>
      <CategoryAxis categoryField="Month"/>
    </verticalAxis>
    <series>
      <Bar2DSeries xField="Profit" itemRenderer="SemiCircleBarItemRenderer">
```

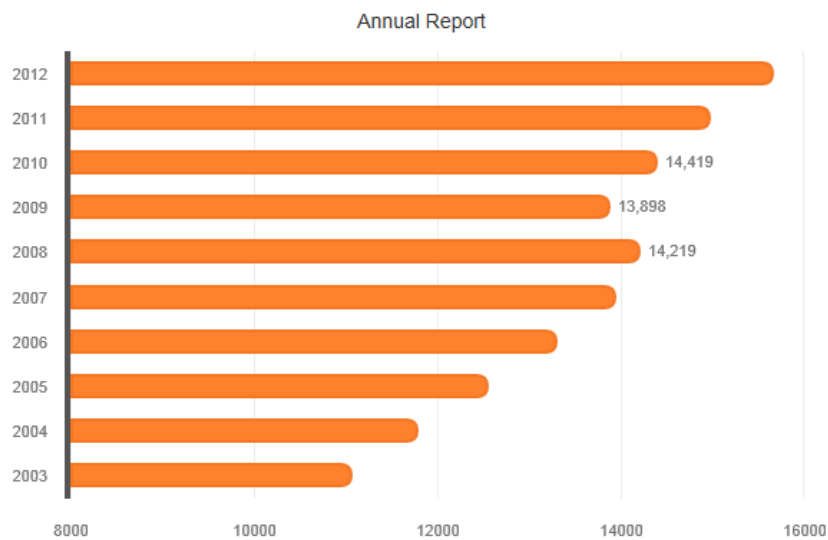


```

        <showDataEffect>
            <SeriesInterpolate/>
        </showDataEffect>
    </Bar2DSeries>
</series>
</Bar2DChart>
</rMateChart>

```

<예제 19 바차트 예제>



<그림 7 바차트 출력 결과>

## 6.7. 바 3D 차트.

바 3D 차트는 <Bar3DChart> 노드로 시작과 끝을 나타냅니다.

속성 및 유효값은 제품내에 포함된 API(/Docs/api)를 참고하십시오.

```

<rMateChart>
  <Options>
    <Caption text="Annual Report"/>
  </Options>
  <Bar3DChart showDataTips="true">
    <verticalAxis> //수직축을 설정하고 CateogryAxis 로 그룹핑에 해당되는 Month정의
      <CategoryAxis categoryField="Month" />
    </verticalAxis>
    <series>
      <Bar3DSeries xField="Profit"> //xField에 수치데이터 Profit
        <showDataEffect>
          <SeriesInterpolate />
        </showDataEffect>
      </Bar3DSeries>
    </series>
  </Bar3DChart>
</rMateChart>

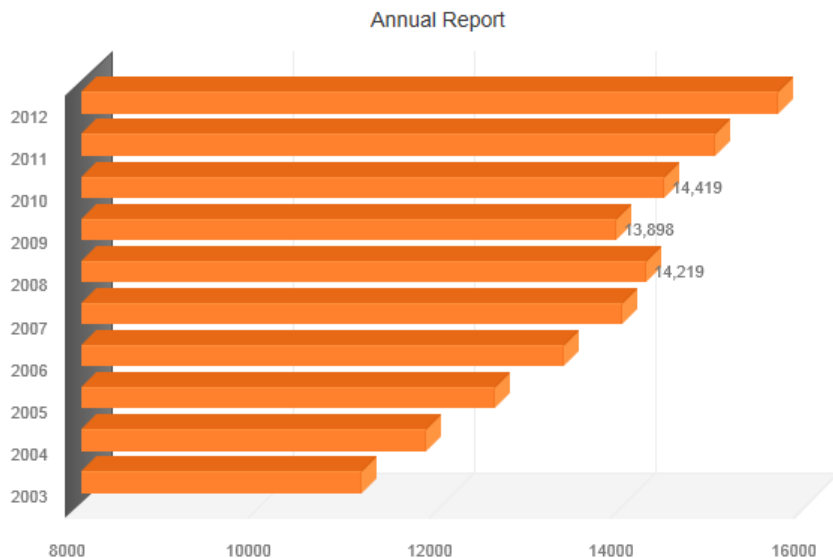
```

```

</showDataEffect>
</Bar3DSeries>
</series>
</Bar3DChart>
</rMateChart>

```

<예제 20 바 3D 차트 예제>



<그림 8 바 3D 차트 출력 결과>

## 6.8. 파이, 도넛차트,

2D 는 <Pie2DChart> 노드로 시작과 끝을 나타내고, 3D 는 <Pie3DChart>노드로 시작과 끝을 나타냅니다. 2D 와 3D 차트 각각의 시리즈는 <Pie2DSeries/> 와 <Pie3DSeries/> 노드로 데이터를 표현합니다.

파이차트와 도넛차트의 레이아웃은 단 한가지를 제외하고 모두 같습니다.

아래 레이아웃의 innerRadius 프로퍼티 값에 의해 파이차트와 도넛차트를 표현합니다. innerRadius 의 범위는 0 ~ 1 까지이며 0 값이 파이차트이며 1 로 가까이 갈수록 도넛차트의 가운데 공간이 커집니다.

```

<rMateChart>
  <Pie2DChart innerRadius="0" showDataTips="true">
    <series>
      <Pie2DSeries
        nameField="Month" //툴팁에 표시할 필드 이름. <그림 9 파이차트와 도넛차트 출력 결과> 툴팁의
November에 해당
        field="Profit" // 파이차트가 참고할 데이터 영역.
        depth="0.05" //2D 차트의 그라데이션 깊이(범위 0 ~ 0.2)
        labelPosition="callout">

```

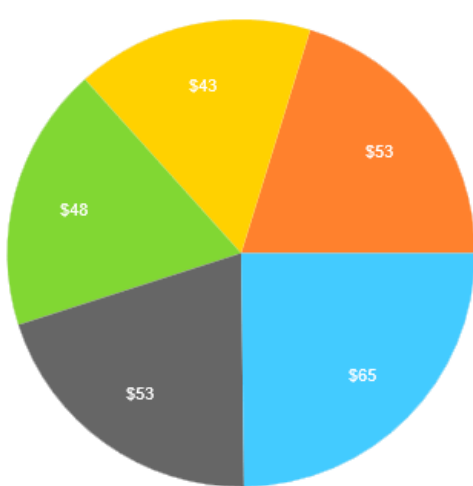
```

<showDataEffect> <!--데이터 출력시 SeriesInterpolate 효과 표시-->
    <SeriesInterpolate/>
</showDataEffect>

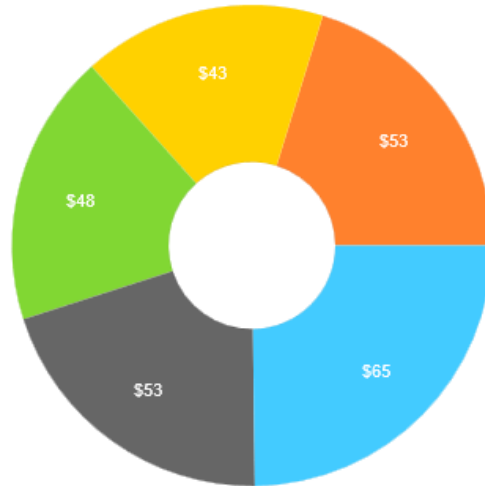
<fills> <!--시리즈에 2가지 색을 지정하여 2가지색이 반복되고 있습니다. -->
    <SolidColor color="0xff0000"/>
    <SolidColor color="0xfffff"/>
</fills>
</Pie2DSeries>
</series>
</Pie2DChart>
<rMateChart>

```

<예제 21 파이차트 예제>



<innerRadius = 0 인 경우>



<innerRadius = 0.33 인 경우>

<그림 9 파이차트와 도넛차트 출력 결과>

위와 같이 파이차트와 도넛차트는 innerRadius 프로퍼티 값에 따라 달라집니다.

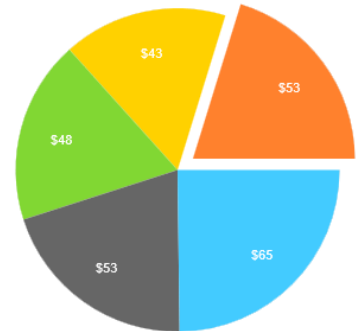
다음으로 labelPosition 에 대한 설명입니다.

labelPosition 은 차트 데이터의 값을 따로 표시할 때 그 위치를 정의합니다. 할당 가능한 값과 설명은 아래와 같습니다.

- none : 데이터 값을 표시하지 않습니다.
- inside : 차트 안에 표시합니다.
- outside : 차트 밖에 표시합니다.
- callout : 위 예제와 같이 따로 선을 긋고 표시합니다.
- insideWithCallout : 기본적으로 안쪽에 표시하나 크기가 작아 충분한 공간이 없을 때 따로 선을 긋고 callout 형태로 표시합니다.

파이, 도넛차트의 데이터영역을 클릭한 경우 해당 영역을 다음과 같이 데이터 영역을 빠져 나오게 할 수 있습니다.(explodable 은 true 가 기본값입니다.)

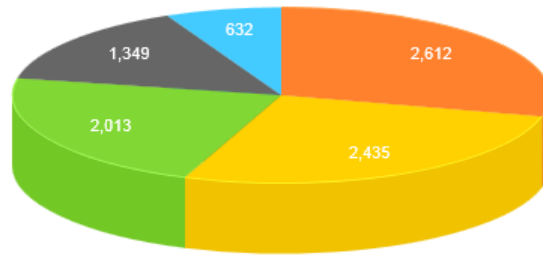
```
<rMateChart>
  <Options>
    <Caption text="Annual Report"/>
    <SubCaption text="2008"/>
  </Options>
  <Pie2DChart explodable="true" innerRadius="0"
showDataTips="true">
    .....
</rMateChart>
```



<예제 22 파이차트 해당 영역 클릭한 경우 예제와 출력모습>

```
<rMateChart backgroundColor="0xFFFFEE" cornerRadius="12" borderStyle="solid">
  <Options>
    <Caption text="Annual Report"/>
    <SubCaption text="2008"/>
  </Options>
  <Pie3DChart showDataTips="true" explodable="false">
    <series>
      <Pie3DSeries nameField="Month" field="Profit" labelPosition="inside">
        <showDataEffect>
          <SeriesZoom/>
        </showDataEffect>
      </Pie3DSeries>
    </series>
  </Pie3DChart>
</rMateChart>
```

<예제 23 파이 3D 차트 예제>

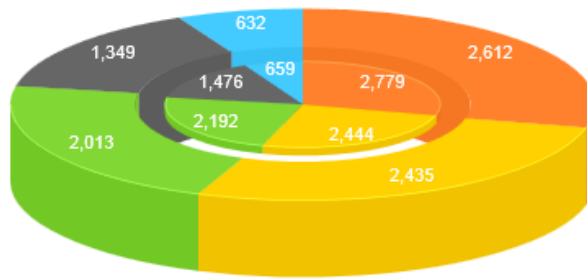


<그림 10 파이 3D 차트 출력 결과>

파이 Stacked 3D 차트는 시리즈 2 개로 표현이 가능합니다. 아래는 그 예제와 실행화면입니다.

```
<rMateChart cornerRadius="12" borderStyle="solid">
  <Options>
    <Caption text="Annual Report" />
    <SubCaption text="2008" />
  </Options>
  <Pie3DChart showDataTips="true" explodable="false">
    <series>
      <Pie3DSeries nameField="Month" field="Profit" displayName="Profit" labelPosition="inside">
        <showDataEffect>
          <SeriesZoom />
        </showDataEffect>
      </Pie3DSeries>
      <Pie3DSeries nameField="Month" field="Cost" displayName="Cost" labelPosition="inside">
        <showDataEffect>
          <SeriesZoom />
        </showDataEffect>
      </Pie3DSeries>
    </series>
  </Pie3DChart>
</rMateChart>
```

<예제 24 파이 Stacked 3D 차트 예>



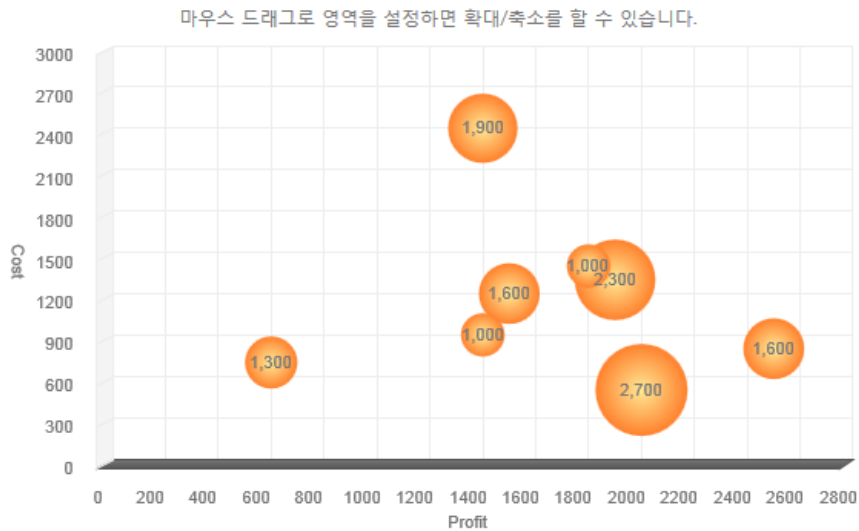
<그림 11 파이 3D Stacked 차트 출력 결과>

## 6.9. 버블 3D 차트.

버블 3D 차트는 <Bubble3DChart> 노드로 시작과 끝을 나타내고 데이터 표현이라 할 수 있는 시리즈는 <Bubble3DSeries> 노드로 표현합니다. 버블 3D 차트는 수치데이터를 X, Y, Radius 3 개의 필드에 표현합니다. X는 X 축에 대응하는 값, Y는 Y 축, 그리고 Radius는 버블의 크기를 결정하는 값입니다. 실질적으로 데이터를 표현하는 것은 시리즈이기 때문에 <Bubble3DSeries> 노드의 속성으로 xField, yField, RadiusField에 각각의 수치데이터를 입력하십시오. 다음은 예제와 실행화면입니다.

```
<rMateChart>
  <Bubble3DChart showDataTips="true">
    <series>
      <Bubble3DSeries xField="Profit" yField="Cost" radiusField="Revenue">
        <showDataEffect>
          <SeriesInterpolate />
        </showDataEffect>
        <fills> <!--6가지 색을 지정하여 6가지색이 반복되고 있습니다. -->
          <SolidColor color="0xFF0000"/>
          <SolidColor color="0x00FF00"/>
          <SolidColor color="0x0000FF"/>
          <SolidColor color="0xFF00FF"/>
          <SolidColor color="0xFFFF00"/>
          <SolidColor color="0xFFFFFF"/>
        </fills>
      </Bubble3DSeries>
    </series>
  </Bubble3DChart>
</rMateChart>
```

<예제 25 버블 3D 차트 예제>

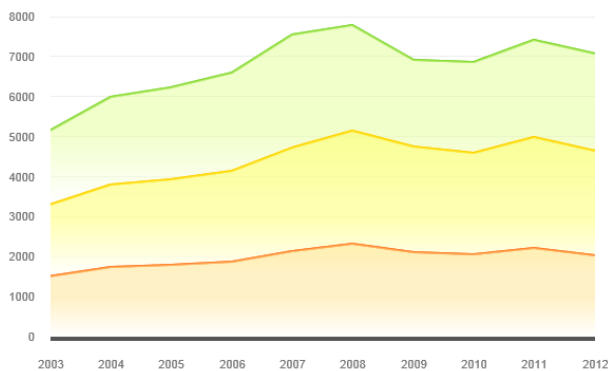


<그림 12 버블 3D 차트 출력 결과>

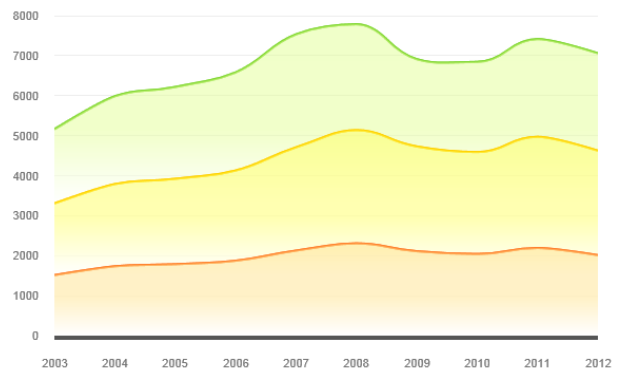
## 6.10. 영역(Area) 차트.

영역차트의 type 에는 overlaid, stacked, 100% 3 가지가 있습니다. 이에 대한 기능은 칼럼차트와 같고 영역차트 type 의 기본값은 overlaid 입니다.(칼럼 차트는 clustered)

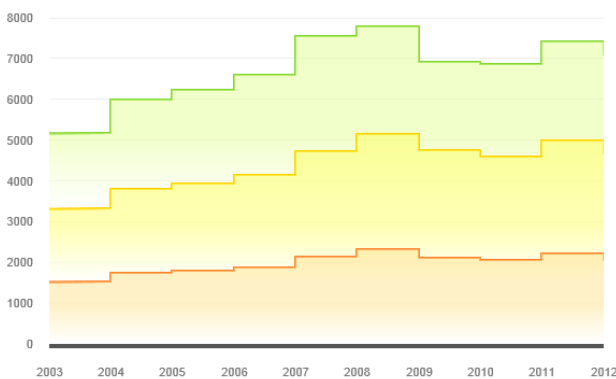
Area2DSeries 프로퍼티 중 form 은 데이터를 어떻게 표현할지를 명시합니다. 아래 그림을 참고하십시오.



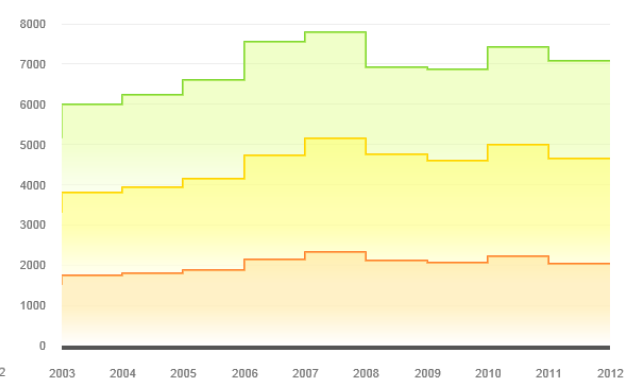
<form="segment" 기본값>



<form="curve">



<form="step">

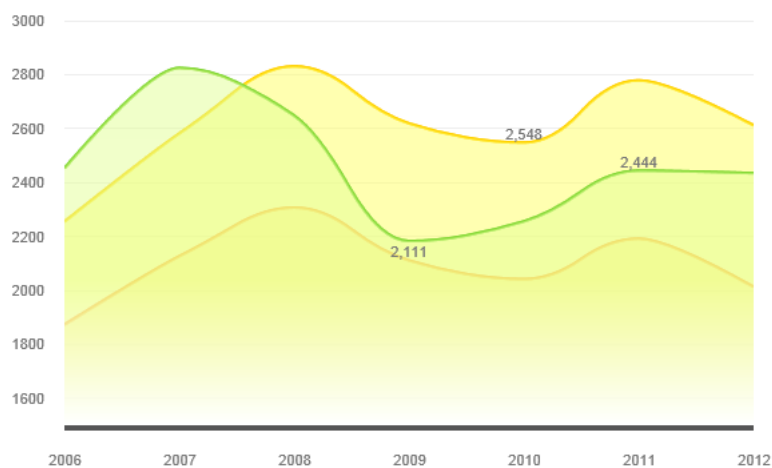


<form="reverseStep">

<그림 13 영역차트 form 별 출력 결과>

```
<rMateChart>
  <Area2DChart showDataTips="true" type="stacked">
    <horizontalAxis>
      <CategoryAxis categoryField="Month" />
    </horizontalAxis>
    <series>
      <Area2DSeries yField="Profit" form="curve" displayName="Profit">
        <showDataEffect>
          <SeriesInterpolate />
        </showDataEffect>
      </Area2DSeries>
      <Area2DSeries yField="Cost" form="curve" displayName="Cost">
        <showDataEffect>
          <SeriesInterpolate />
        </showDataEffect>
      </Area2DSeries>
      <Area2DSeries yField="Revenue" form="curve" displayName="Revenue">
        <showDataEffect>
          <SeriesInterpolate />
        </showDataEffect>
      </Area2DSeries>
    </series>
  </Area2DChart>
</rMateChart>
```

<예제 26 영역차트 예제>



<그림 14 영역차트 출력 결과>



## 6.11. 플롯차트

플롯차트는 수평 축 상의 위치를 나타내는 값과 수직 축 상의 위치를 나타내는 값을 가지는 데이터 포인트를 직교좌표로 나타내는 경우에 사용합니다.

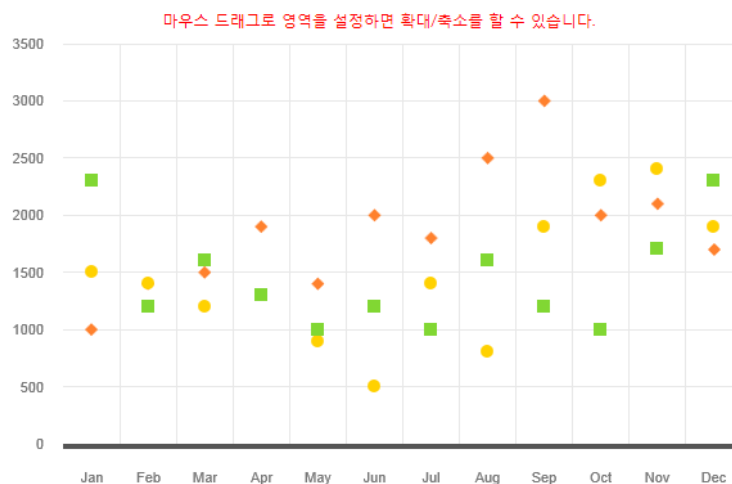
플롯차트를 작성하려면 `<Plot2DChart>`로 시작하여 `</Plot2DChart>`로 끝냅니다. 시리즈는 `<Plot2DSeries>`입니다.

**주의:** `Plot2DChart`에서는, 각각의 `PlotSeries`에 `xField` 프로퍼티와 `yField` 프로퍼티의 양쪽 모두를 지정해야 합니다.

예제를 보도록 하겠습니다.

```
<rMateChart backgroundColor="0xFFFFEE" cornerRadius="12" borderStyle="solid">
  <Plot2DChart showDataTips="true"> //플롯차트 생성
    <verticalAxis>
      <LinearAxis maximum="3500" />
    </verticalAxis>
    <horizontalAxis>
      <LinearAxis maximum="2800" />
    </horizontalAxis>
    <series>
      <Plot2DSeries xField="Cost" yField="Profit" radius="5" displayName="Cost/Profit"/>
      <Plot2DSeries xField="Revenue" yField="Profit" radius="5" displayName="Revenue/Profit"/>
      <Plot2DSeries xField="Cost" yField="Revenue" radius="5" displayName="Cost/Revenue"/>
    </series>
  </Plot2DChart>
</rMateChart>
```

<예제 27 플롯차트 예제>



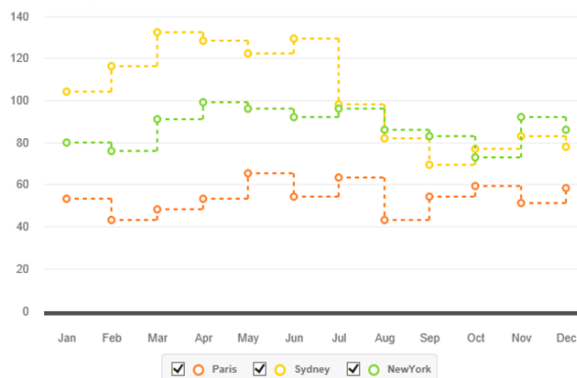
<그림 15 플롯차트 출력 결과>

## 6.12. 라인차트.

라인차트는 다른 Cartesian 차트와 비교하여 type 속성이 없습니다. 기본적으로 overlaid 속성입니다. 그리고 라인차트의 시리즈인 Line2DSeries 의 속성으로 form 이 존재합니다. 라인시리즈의 form 속성은 영역차트의 시리즈인 Area2DSeries 의 form 속성과 같이 segment, curve, step, reverseStep 이 존재합니다. 위 Area2DSeries 의 form 속성을 표현한 그림을 참고하십시오.

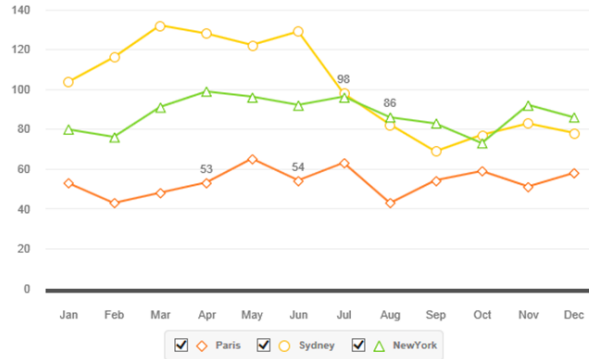
```
<rMateChart cornerRadius="12" borderStyle="solid">
  <Options>
    <Caption text="Anual Report" />
  </Options>
  <Line2DChart showDataTips="true">
    <horizontalAxis>
      <CategoryAxis categoryField="Month" />
    </horizontalAxis>
    <series>
      <Line2DSeries yField="Profit" form="step">
        <showDataEffect>
          <SeriesInterpolate />
        </showDataEffect>
      </Line2DSeries>
      <Line2DSeries yField="Cost" form="step"/>
      <Line2DSeries yField="Revenue" form="step"/>
    </series>
  </Line2DChart>
</rMateChart>
```

<예제 28 라인차트 예제>



<그림 16 라인차트 출력 결과>

라인차트에서 X 축과 Y 축이 만나는 지점에 itemRenderer 를 적용하여 특수하게 출력할 수 있습니다. 출력결과와 예제는 아래와 같습니다.



<그림 17 라인차트에 itemRenderer 적용한 경우 출력 결과>

```
<rMateChart cornerRadius="12" borderStyle="solid" paddingTop="10" paddingBottom="20" paddingRight="20"
paddingLeft="20">
  <Options>
    <Caption text="Anual Report" />
  </Options>
  <Line2DChart showDataTips="true">
    <horizontalAxis>
      <CategoryAxis categoryField="Month"/>
    </horizontalAxis>
    <series>
      1 <Line2DSeries yField="Profit" radius="10" fill="0xFF0000" itemRenderer="
DiamondItemRenderer"/>
      2 <Line2DSeries yField="Cost" radius="10" fill="0x00FF00" itemRenderer="CircleItemRenderer"/>
      3 <Line2DSeries yField="Revenue" radius="10" fill="0xFFFF00"
itemRenderer="TriangleItemRenderer"/>
    </series>
  </Line2DChart>
</rMateChart>
```

<예제 29 라인차트에 itemRenderer 적용한 예제>





Line2DSeries 에 itemRenderer 를 적용할 때 따르는 주요 속성은 아래와 같습니다.

- radius : X 축과 Y 축이 만나는 지점에 도형의 크기를 결정합니다.
- fill : 도형(itemRenderer)의 안쪽 컬러를 지정합니다.

- stroke : 도형의 테두리 선을 지정합니다.

라인차트 전체 라인의 색과 두께 조정은 lineStroke 속성을 이용합니다. 이에 대한 설명은 <예제 51 라인차트의 선(stroke)를 변경한 경우> 를 참고하십시오.

라인차트의 itemRenderer 속성값으로 다음 유효값이 존재합니다.

- DiamondItemRenderer – 위 예제 1 번에 해당.
- CircleItemRenderer – 위 예제 2 번에 해당.
- TriangleItemRenderer – 위 예제 3 번에 해당.
- CrossItemRenderer - 십자가 모양 
- XShapeItemRenderer - X 자 모양 
- IShapeItemRenderer – I 자 모양 
- RectangleItemRenderer – 네모 모양 

### 6.13. 점선(Dashed-Line) 차트

점선 차트(Dashed-Line Chart) 는 라인차트의 모든 기능을 포함하고, 레이아웃 작성법 또한 같습니다. 라인 차트와 비교하여 점선 차트의 고유 속성은 다음과 같습니다.

속성 명	유효 값	설 명
lineStyle	"normal", "dashLine"	라인차트에서 라인의 type 을 나타냅니다. 점선차트를 생성하기 위해서는 dashLine 으로 값을 할당하십시오.
dashLinePlacement,	"before", "after"	dashLineSeperatePos 을 설정한 경우 점선을 dashLineSeperatePos 기준으로 전에 나타낼지 후에 나타낼지를 결정합니다.
dashLineSeperatePos	데이터의 인덱스 (즉, uint)	점선과 실선을 같이 사용하고자 할 때의 경계점을 나타냅니다. 이 점은 데이터의 인덱스 입니다.
dashLinePattern	Number(픽셀단위) (기본값 : 10)	표현할 점선의 길이를 지정합니다. 예를 들어 5 를 지정 시 5픽셀에 해당되는 선을 그린 후 5픽셀에 해당되는 공백을 생성합니다. 이 패턴을 반복합니다. 길이를 크게 할 수록 성능은 향상됩니다. 차트 렌더링이 느리다면 패턴 길이를 길게 잡으십시오.

<표 6 점선 차트 속성 유효값 및 설명>

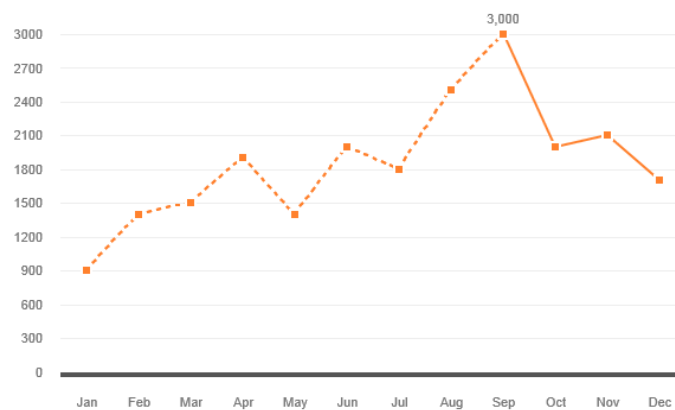
```
<rMateChart backgroundColor= "0xFFFFF" cornerRadius= "12" borderStyle= "solid">
```

```

<Line2DChart showDataTips="true" >
  <horizontalAxis>
    <CategoryAxis categoryField="Month" padding="0.5" />
  </horizontalAxis>
  <series>
    <Line2DSeries labelPosition="up"
    yField="Profit" radius="4
    lineStyle="dashLine"           // 점선 차트로 표현
    dashLineSeperatePos="5"       // 인덱스5(즉, 6번째 데이터) 에서 점실선 나눔
    dashLinePlacement="before"    // 점선은 앞에 나타남
    itemRenderer="RectangleItemRenderer">
    </Line2DSeries>
  </series>
</Line2DChart>
</rMateChart>

```

<표 7 점선 차트 예제>



<그림 18 점선 차트 출력 모습>

#### 6.14. 콤비네이션 차트.

콤비네이션 차트는 서로 다른 유형의 차트가 함께 있는 것을 말합니다. 아래 예제는 칼럼 차트와 라인차트가 함께 데이터를 표현하는 콤비네이션 차트입니다.

- 칼럼차트라면 series 프로퍼티에 2D 인 경우 Column2DSeries, 3D 인 경우 Column3DSeries 로 데이터를 표현하였지만 콤비네이션 차트에서는 칼럼차트와 라인차트 두 차트의 시리즈(데이터 표현)가 존재하므로 칼럼시리즈를 묶을 수 있는 Column2DSet 또는 Column3DSet 을 정의하십시오.
- 2D 를 표현하기 위해 Column2DSet 을 정의 한 후 그 자식으로 Column2DSeries 를 원하는 데이터 표현 만큼 정의하십시오.

- Column2DSet(또는 Column3DSet) 의 type 은 일반 칼럼차트의 type 과 동일합니다. <그림 4 칼럼차트 Type 별 출력 결과> 참고. (그러나 기본값은 overlaid)
- Column2DSet 과 형제인 위치에 Line2DSeries 를 정의하십시오.

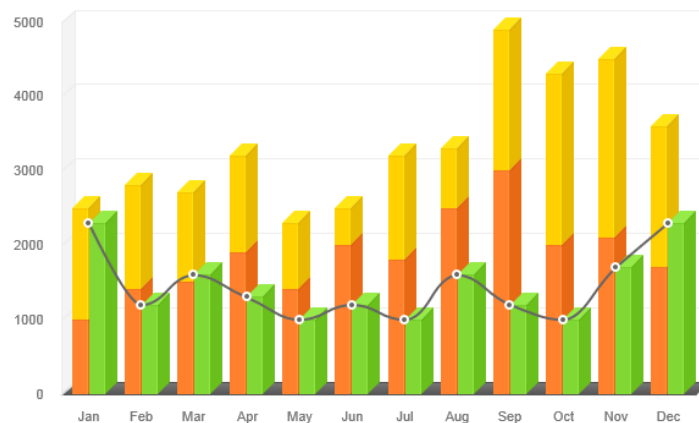
위의 설명을 구체화한 코드입니다.

```
<rMateChart>
  <Combination2DChart showDataTips="true">
    <horizontalAxis>
      <CategoryAxis categoryField="Month"/>
    </horizontalAxis>

    <series>
      <Column2DSet type="clustered">
        <series>
          <Column2DSeries yField="Profit" displayName="Profit"/>
          <Column2DSeries yField="Revenue" displayName="Revenue"/>
        </series>
      </Column2DSet>
      <Line2DSeries yField="Cost" displayName="Cost">
        <lineStroke> //라인시리즈의 라인의 두께, 색을 지정합니다.
          <Stroke color="0xFFCC00" weight="4"/>
        </lineStroke>
      </Line2DSeries>
    </series>
  </Combination2DChart>
</rMateChart>
```

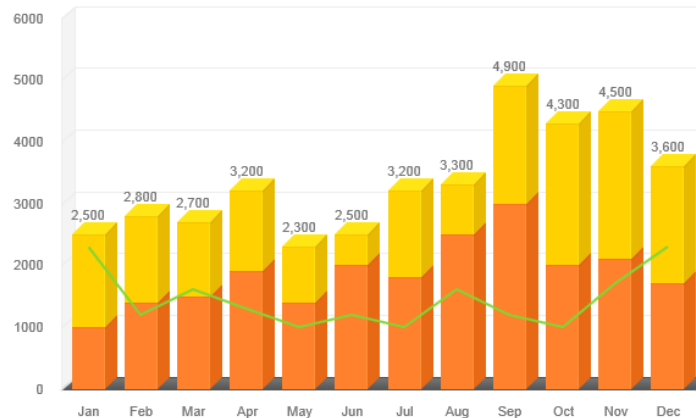
<예제 30 콤비네이션 차트 예제>

다음은 위의 예제 출력화면입니다.



<그림 19 Column2DSet type="clustered" 설정한 콤비네이션 차트 결과>

칼럼차트 type 을 스택으로 표현하는 콤비네이션을 원하다면 <ColumnSet type="stacked">를 정의하십시오.



<그림 20 Colum2DnSet type="stacked" 설정한 콤비네이션 차트 결과>

## 6.15. 실시간 차트

실시간 차트는 주기적으로 변하는 데이터를 그대로 표현합니다. 차트는 폴링 방식으로 데이터를 얻도록 설계되었습니다.

실시간 차트는 <RealTimeChart/> 노드로 시작과 끝을 지시합니다. 실시간 차트의 시리즈(series) 속성으로는 칼럼시리즈, 라인시리즈, Area 시리즈 등을 갖습니다. 그리고 서버에서 생성되는 데이터를 갖고 오기 위해 주기적으로 원격 호출(RPC, Remote Procedure Call)이 이루어 집니다. 그 역할을 담당하는 것이 HttpServiceRepeater 입니다. 실시간으로 데이터를 표현하기 위해서 반드시 레이아웃에 정의해야 합니다. 실시간 차트 레이아웃을 작성할 때 주의점은 아래와 같습니다.

첫째, HttpServiceRepeater 는 반드시 <rMateChart/> 노드의 자식 위치에, 그리고 <RealTimeCartesianChart/>의 아래 형제 위치에 놓여야 합니다.

둘째, target 속성의 속성값으로 차트(즉, 실시간차트)의 id 값을 지시하십시오.

셋째, HttpServiceRepeater 의 url 은 차트 데이터가 있는 XML URL 경로입니다. 이 XML 데이터 형식은 반드시 아래와 같은 구조로 작성되어야 합니다.

```
<?xml version="1.0" encoding="utf-8" ?>
<items>                                //XML의 루트는 반드시 items 로 시작
  <item>                                // 한 개의 데이터 셋은 반드시 item 으로 묶여야 합니다.
    <Time>13:8:27</Time> // 이곳은 사용자 정의하십시오.
    <Volume>5527</Volume>
    <Price>309</Price>
  </item>
</items>
```

### <예제 31 실시간 차트 데이터 양식>

실시간 차트를 HTML 문서에 임베딩 시 차트 초기의 데이터를 삽입할 필요는 없습니다.

**\*참고 : 예를 들어 차트가 데이터를 갱신하는 주기가 5 초라고 한다면, 차트는 매 5 초마다 서버에서 풀링 방식으로 데이터를 가져와 표현합니다. 그러나 실제로 여러 가지 요인(선로 상태, 통신 상태, 트래픽 수 등등)에 따라 서버와의 통신에서 오차는 발생할 수 있습니다. 서버사이드는 이 오차를 고려하여 차트가 가져갈 데이터를 갱신 할 필요가 있습니다.**

실시간 차트는 두 가지 유형으로 데이터를 표현합니다.

1. 데이터 개수를 기준으로 표현합니다. 서버나 기타 경로를 통하여 데이터를 받으면 차트는 즉시 데이터를 표현합니다. displayDataSize 의 값을 20 으로 설정하였다면 한 화면에 20 개의 데이터를 표현한 후 다음 데이터부터 차례로 화면을 갱신하며 데이터를 출력합니다.
2. 시간을 기준으로 데이터를 표현합니다. 예를 들어 timePeriod 값을 3600 초(1 시간)으로 설정하고 interval 값을 3 으로 설정했다면 매 3 초마다 차트는 1 시간동안 데이터를 현재 화면에 출력하고 1 시간 이후의 다음 데이터부터 차례로 화면을 갱신하며 데이터를 출력합니다.(interval 값은 HttpServiceRepeater 의 interval 과 같아야 정상적입니다.)

#### 6.15.1. 데이터 개수를 기준으로 실시간 차트 표현(CategoryAxis 활용)

현재 차트가 표현하는 데이터의 개수를 생각한다면 CategoryAxis 를 활용하여 차트 레이아웃을 작성하여야 합니다.

CategoryAxis 는 연속적이지 않고, 데이터의 카테고리 필드를 개별적으로 판단합니다. 예를 들어 데이터 필드 중 시간이나 날짜와 관련된 필드가 있을 때 이것은 형식상 시간, 날짜이지만 차트의 카테고리축은 그렇게 판단하지 않고 단순 문자열(String)로 판단합니다. 연속적이지 않기 때문에 중복데이터를 그대로 표현하는 단점이 있습니다.

```
<rMateChart cornerRadius="12" borderStyle="solid">
  <RealTimeChart id="chart" dataDisplayType="dataSize" displayDataSize="15" showDataTips="true">
    <horizontalAxis>
      <CategoryAxis id="hAxis" categoryField="Time"/>
    </horizontalAxis>
    <series>
      <Column2DSeries yField="Volume" displayName="Trading Volume"
itemRenderer="GradientColumnItemRenderer">
      <fill> //채우기 색 정의
```



```

        <SolidColor color="0xB0C759" />
      </fill>
    </Column2DSeries>
  </series>
</RealTimeChart>
<HttpServiceRepeater url="http://demo.riamore.net/chartTest/data.jsp" target="{chart}" interval="3"
method="get" /> //3초 단위로 폴링방식으로 데이터 가져옴
</rMateChart>

```

<예제 32 CategoryAxis 활용한 실시간 차트 예제>

#### 6.15.2. 정해진 시간을 기준으로 실시간 차트 표현(DateTimeAxis 활용)

최근 데이터를 기준으로 정해진 시간 동안 들어온 데이터를 표현하고자 할 때 DateTimeAxis 를 활용하여 레이아웃을 작성하십시오. DateTimeAxis 는 연속적인 시간을 표현하기 때문에 반복적인 데이터는 무시합니다. 그래서 어떠한 오차로 인해 차트가 서버의 갱신된 데이터를 가져오지 못했을 경우 즉, 이전의 중복된 데이터를 가져왔을 경우 무시하게 됩니다. 또한 오차로 인해 다음 데이터를 얻지 못하고 다다음 데이터를 얻었을 경우 얻지 못한 데이터에 대해 그 시간 영역에 해당되는 공백을 남겨 표시합니다.

일반적인 실시간 차트는 DateTimeAxis 를 활용하십시오.

```

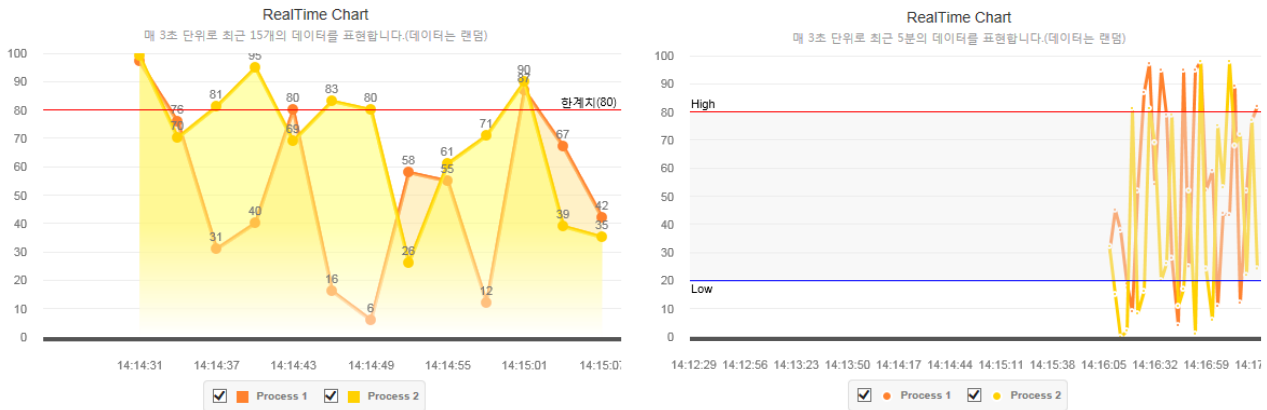
<rMateChart cornerRadius="12" borderStyle="solid">
  <RealTimeChart id="chart" dataDisplayType="time" timePeriod="60" interval="3" showDataTips="true">
//60초동안 3초 간격으로 데이터 표현(시간 기준)
    <horizontalAxis>
      //데이터와 축라벨은 모두 초단위, 데이터가 3초단위로 들어오므로 dataInterval=3, 축라벨 간격은
      9(초), GMT가 아닌 Local시간 표시 true
      <DateTimeAxis dataUnits="seconds" labelUnits="seconds" dataInterval="3" interval="9"
displayLocalTime="true"/>
    </horizontalAxis>
    <series> //DateTimeAxis 이므로 xField 정의 필수.
      <Column2DSeries xField="Time" yField="Volume" displayName="Trading Volume"
itemRenderer="GradientColumnItemRenderer">
        <fill>
          <SolidColor color="0xB0C759" />
        </fill>
      </Column2DSeries>
    </series>
  </RealTimeChart>
  <HttpServiceRepeater url="http://demo.riamore.net/chartTest/data4.jsp" target="{chart}" interval="3"
method="get" />

```

</rMateChart>

<예제 33 DateTimeAxis 활용한 실시간 차트 예제>

**\*참고 : RealTimeChart 의 interval, DateTimeAxis 의 interval, HttpServiceRepeater 의 interval, 이 3 개의 interval 을 똑같이 맞추으로써 원하는 모양의 차트가 나옵니다.**



<그림 21 Category 축 실시간차트(좌), DateTime 축 실시간 차트(우)>

6.15.3. HttpServiceRepeater 로 일반 차트의 데이터를 실시간으로 바꾸기.

**HttpServiceRepeater 는 반드시 실시간 차트에만 국한적이지 않습니다.**

예를 들어 칼럼 3D 차트를 생성하였습니다. 이 때 매 1 분 마다 칼럼 3D 차트의 전체 데이터를 갱신하고자 할 때 HttpServiceRepeater 는 유용합니다.

칼럼 3D 차트 레이아웃을 그대로 활용하고 아래와 같이 HttpServiceRepeater 를 레이아웃에 넣어주시면 됩니다.

```
<rMateChart cornerRadius="12" borderStyle="solid">
  <Options>
    <Caption text="Annual Report" />
  </Options>
  <Column3DChart id="chart" showDataTips="true">
    <horizontalAxis>
      <CategoryAxis categoryField="Month" />
    </horizontalAxis>
    <series>
      <Column3DSeries yField="Profit" displayName="Profit">
        <showDataEffect>
          <SeriesInterpolate />
        </showDataEffect>
      </Column3DSeries>
    </series>
  </Column3DChart>
</rMateChart>
```

```

        </Column3DSeries>
    </series>
</Column3DChart>

<HttpServiceRepeater url="http://demo.riamore.net/chartTest/singleData.jsp" target="{chart}"
    interval="60"/>
</rMateChart>

```

<예제 34 칼럼차트에 HttpServiceRepeater 적용>

## 6.16. 레이더(방사형) 차트

레이더(방사형) 차트는 <RadarChart/> 노드로 시작과 끝을 지시합니다. 레이더 차트의 데이터 표현을 나타내는 레이더 라인 즉, 차트 시리즈는 <RadarSeries> 입니다.

레이더 차트는 radialAxis 와 angularAxis 두 개의 축이 존재합니다. radialAxis 는 레이더 차트 원점에서 테두리까지를 잇는 축으로 각 데이터의 수치를 표현하는 역할을 합니다.

angularAxis 는 레이더 차트 테두리에 출력되는 축으로 레이더 차트 데이터의 카테고리에 해당되는 데이터를 표현합니다.

연간 가계 지출 현황이라는 데이터를 갖고 레이더 차트로 표현하기 위해 일단 데이터를 정의해 보도록 하겠습니다.

다음은 일반적인 데이터를 갖고 차례로 XML, 배열 형태로 데이터를 바꾼 형태입니다.

### 1. 일반적인 데이터

	2005년	2006년	2007년	2008년
<b>Food</b>	100	100	180	150
<b>Health Care</b>	80	120	200	210
<b>Transportation</b>	70	115	100	240
<b>Clothing</b>	80	120	140	210
<b>Education</b>	90	160	130	200
<b>Shelter</b>	100	180	165	140
<b>Leisure</b>	76	120	130	170
<b>Others</b>	80	140	140	190

### 2. 일반적인 데이터를 배열 형태로 바꾼 결과.

```

var chartData = [{ "catName": "Food", "year2005": 100, "year2006": 100, "year2007": 180, "year2008": 150 }
    , { "catName": "Health Care", "year2005": 80, "year2006": 120, "year2007": 200, "year2008": 210 }
    , { "catName": "Transportation", "year2005": 70, "year2006": 115, "year2007": 100,
    "year2008": 240 }

```

```
,{"catName":"Clothing", "year2005":80, "year2006":120, "year2007":140, "year2008":210}
,{"catName":"Education", "year2005":90, "year2006":160, "year2007":130, "year2008":200}
,{"catName":"Shelter", "year2005":100, "year2006":180, "year2007":165, "year2008":140}
,{"catName":"Leisure", "year2005":76, "year2006":120, "year2007":130, "year2008":170}
,{"catName":"Others", "year2005":80, "year2006":140, "year2007":140, "year2008":190}];
```

### 3. 일반적인 데이터를 XML 형태로 바꾼 결과.

```
<items>
  <item>
    <catName>Food</catName>
    <year2005>100</year2005>
    <year2006>100</year2006>
    <year2007>180</year2007>
    <year2008>150</year2008>
  </item>
  <item>
    <catName>Health Care</catName>
    <year2005>80</year2005>
    <year2006>120</year2006>
    <year2007>200</year2007>
    <year2008>210</year2008>
  </item>
  .
  .
  .
  <item>
    <catName>Others</catName>
    <year2005>80</year2005>
    <year2006>140</year2006>
    <year2007>140</year2007>
    <year2008>190</year2008>
  </item>
</items>
```

위 데이터를 바탕으로 작성한 레이더 차트 레이아웃 예제입니다.

```
<rMateChart backgroundColor="0xFFFFEE" cornerRadius="12" borderStyle="solid">
  <Options>
    <Caption text="연도별 가계 지출 품목 분석"/>
    <SubCaption text="(2005~2008)"/>
    <Legend useVisibleCheck="true"/>
  </Options>
  <RadarChart id="chart1"
```

```

isSeriesOnAxis="false"
type="polygon"
showAllDataTips="false"
showDataTips="true">
<radialAxisRenderers>
  <AxisRenderer horizontal="true" visible="true" styleName="hangingCategoryAxis" >
  </AxisRenderer>
  <AxisRenderer horizontal="false" visible="false" styleName="hangingCategoryAxis" >
  </AxisRenderer>
</radialAxisRenderers>
<radialAxis>
  <LinearAxis/>
</radialAxis>
<angularAxis>
  <CategoryAxis id="aAxis" categoryField="catName"
displayName="Category" />
  </angularAxis>
<angularAxisRenderers>
  <AngularAxisRenderer axis="{aAxis}" axisTitleStyleName="axisTitle"
styleName="hangingCategoryAxis"/>
</angularAxisRenderers>
<series>
  <RadarSeries field="year2005"
    displayName="2005년">
    <showDataEffect>
      <SeriesInterpolate/>
    </showDataEffect>
    <stroke> <!-- 데이터 포인트에 찍힌 도형의 외곽 테두리 선 스타일 -->
    <Stroke color="0xE48701" weight="2"/>
    </stroke>
    <lineStroke> <!-- 레이더 라인 스타일 -->
      <Stroke color="0xE48701" weight="2"/>
    </lineStroke>
    <areaFill> <!-- 레이더 안쪽 영역 채우기 색 -->
      <SolidColor color="0xE48701" alpha="0.1"/>
    </areaFill>
    <fill> <!-- 데이터 포인트에 찍힌 도형의 안쪽 채우기 색 -->
      <SolidColor color="0xFFFFFFFF"/>
    </fill>
  </RadarSeries>

  <RadarSeries field="year2006"
    displayName="2006년">
    <showDataEffect>

```

```

        <SeriesInterpolate/>
    </showDataEffect>
    <stroke>
        <Stroke color="0xB0C759" weight="2"/>
    </stroke>
    <lineStroke>
        <Stroke color="0xB0C759" weight="2"/>
    </lineStroke>
    <areaFill>
        <SolidColor color="0xB0C759" alpha="0.1"/>
    </areaFill>
    <fill>
        <SolidColor color="0xFFFFFFFF"/>
    </fill>
</RadarSeries>
<RadarSeries field="year2007"
    displayName="2007년">
    <showDataEffect>
        <SeriesInterpolate/>
    </showDataEffect>
    <stroke>
        <Stroke color="0x0A80C4" weight="2"/>
    </stroke>
    <lineStroke>
        <Stroke color="0x0A80C4" weight="2"/>
    </lineStroke>
    <areaFill>
        <SolidColor color="0x0A80C4" alpha="0.1"/>
    </areaFill>
    <fill>
        <SolidColor color="0xFFFFFFFF"/>
    </fill>
</RadarSeries>
<RadarSeries field="year2008"
    displayName="2008년">
    <showDataEffect>
        <SeriesInterpolate/>
    </showDataEffect>
    <stroke>
        <Stroke color="0xCC99CC" weight="2"/>
    </stroke>
    <lineStroke>
        <Stroke color="0xCC99CC" weight="2"/>
    </lineStroke>

```

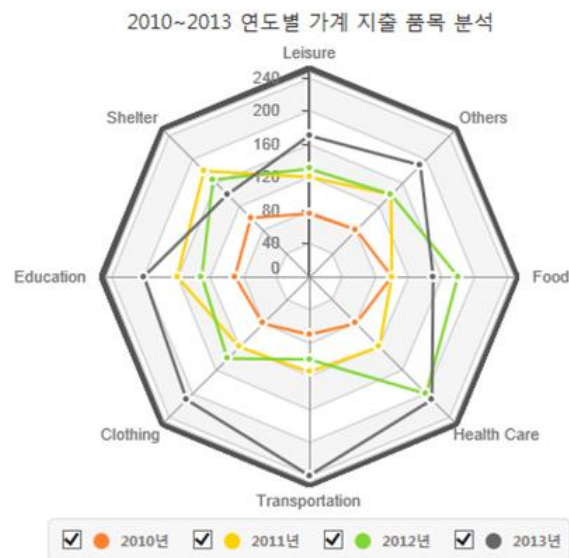
```

<areaFill>
    <SolidColor color="0xCC99CC" alpha="0.1"/>
</areaFill>
<fill>
    <SolidColor color="0xFFFFF"/>
</fill>
</RadarSeries>
</series>
</RadarChart>
</rMateChart>

```

<예제 35 레이더 차트 레이아웃>

정의한 데이터와 위 레이아웃으로 출력한 차트는 다음과 같습니다.



<그림 22 레이더 차트 출력 화면>

## 6.17. 목표 대비 실적 차트.

목표 대비 실적 차트는 목표치와 달성치 두 데이터를 표현하는 차트로 2D 리니어 유형과 3D 실린더 유형이 있습니다.

목표 대비 실적 차트는 2D 의 경우 <Combination2DChart/> 노드로, 3D 의 경우 <Combination3DChart/> 노드로, 차트의 시작과 끝을 지시합니다. 목표 대비 실적 차트 시리즈(series) 속성 값으로는 단 2 개의 시리즈만 넣을 수 있습니다. 목표에 해당되는 시리즈와, 실적에 해당되는 시리즈입니다.

기본적으로 모든 속성은 Cartesian 공통 속성과 같습니다. 이에 대한 설명은 <예제 13 Cartesian 차트의 공통적인 속성 설명> 을 참고하십시오.

시리즈를 정의 할 때 반드시 지켜야 할 점은 실적에 해당되는 시리즈를 먼저 정의 한 후 목표에 해당되는 시리즈를 정의 해야 합니다.

	2D Linear Type	3D Cylinder Type	3D Cylinder-Bar Type
차트 노드	Combination2DChart	Combination3DChart	Combination3DChart
실적에 해당되는 시리즈 노드	Target2DResultSeries	Target3DResultSeries	HTarget3DResultSeries
목표에 해당되는 시리즈 노드	Target2DGoalSeries	Target3DGoalSeries	HTarget3DResultSeries

<표 8 목표 대비 실적 차트 노드 정의표>

아래는 2D 목표 대비 실적 차트 레이아웃 예제입니다.

```

<rMateChart backgroundColor="0xFFFFEE" cornerRadius="12" borderStyle="solid">
  <Options>
    <Caption text="목표 대비 실적 차트"/>
    <SubCaption text="Linear Type"/>
  </Options>
  <Combination2DChart showDataTips="true">
    <horizontalAxis>
      <CategoryAxis categoryField="Month" />
    </horizontalAxis>
    <series>
      <!-- 실적에 해당 필히 순서 준수-->
      <Target2DResultSeries yField="Result" displayName="Result">
        <showDataEffect>
          <SeriesInterpolate />
        </showDataEffect>
      </Target2DResultSeries>
      <!--목표에 해당 -->
      <Target2DGoalSeries yField="Goal" displayName="Goal">
        <showDataEffect>
          <SeriesInterpolate />
        </showDataEffect>
      </Target2DGoalSeries>
    </series>
  </Combination2DChart>
</rMateChart>

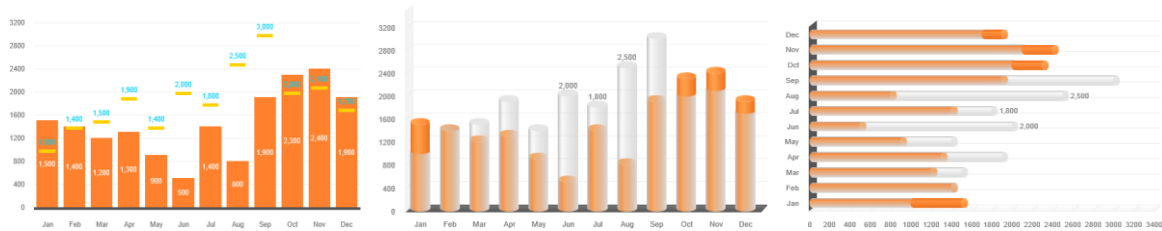
```

<예제 36 2D 목표 대비 실적 차트(Linear Type) 레이아웃 예제>



3D 실린더 형 목표 대비 실적 차트 레이아웃은 위 2D 목표 대비 실적 차트 레이아웃에서 2D 라는 글자를 3D 라는 글자로 바꿔주기만 하면 됩니다.

목표 대비 실적 차트 출력 모습은 다음과 같습니다.



<그림 23 목표 대비 실적 차트 출력 결과(2D, 3D)>

## 6.18. 스크롤 차트

스크롤 차트는 데이터양이 많은 경우 지정된 데이터 개수만을 표시하고 다음 또는 이전 데이터는 스크롤을 통해 데이터를 표시하게끔 하는 차트입니다. 스크롤 차트는 현재 2D 차트에 한해 지원하고 있습니다. 스크롤 차트를 생성하기 위해서는 기존 2D 차트 레이아웃에서 <ScrollableAxisRenderer/> 와 <CategoryLinearAxis/> 를 추가 작성할 필요가 있습니다.

CategoryLinearAxis 는 LinearAxis 를 상속받아 만든 클래스으므로 LinearAxis 의 모든 속성을 상속 받았습니다.

스크롤 차트 레이아웃을 작성하기 위해 다음 단계를 차례로 밟으십시오.

- 기존 2D 차트 레이아웃 작성법 대로 레이아웃을 작성합니다. 예를 들어 칼럼 2D 차트를 스크롤 가능한 차트로 만들어 보겠습니다.
- 칼럼 차트는 스크롤 방향이 가로 방향이므로 가로축(horizontalAxis) 속성에 CategoryLinearAxis 를 value 로 할당합니다.(바차트의 경우엔 세로축 속성에 할당)
- 가로축렌더러(horizontalAxisRenderers) 속성 값으로 ScrollableAxisRenderer 를 할당합니다.

위 방법처럼 기존 차트를 스크롤 차트로 변경하는 방법은 해당 축과 해당 축 렌더러에 할당하는 값을 조금만 수정하면 됩니다.

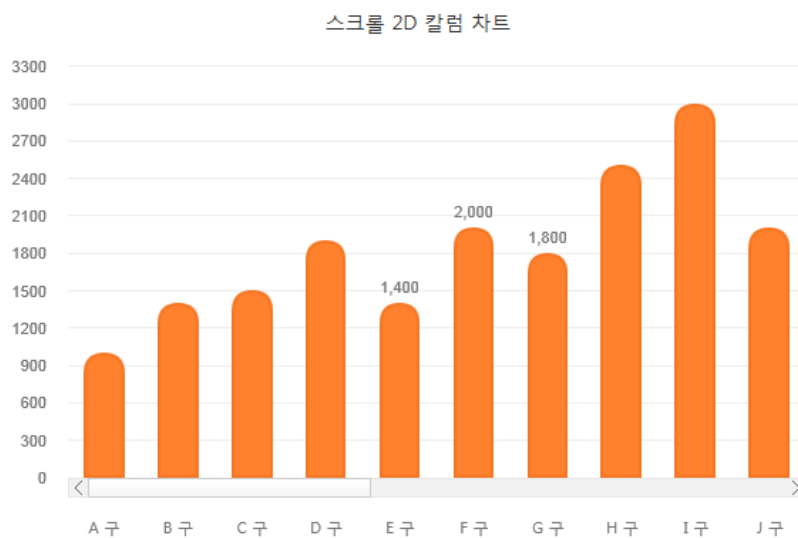
```
<rMateChart backgroundColor="0xFFFFEE" cornerRadius="12" borderStyle="solid">
  <Options>
    <Caption text="스크롤 2D 칼럼 차트"/>
  </Options>
  <Column2DChart showDataTips="true" gutterRight="10">
    <series>
      <Column2DSeries id="cs1" yField="Data1" displayName="Data1" itemRenderer="
```

```

SemiCircleColumnItemRenderer"/>
    </series>
    <horizontalAxis>
        <CategoryLinearAxis id="hAxis" categoryField="Gu"/>
    </horizontalAxis>
    <horizontalAxisRenderers>
        <ScrollableAxisRenderer axis="{hAxis}" visibleItemSize="10" scrollSensitivity="10"/>
    </horizontalAxisRenderers>
</Column2DChart>
</rMateChart>

```

<예제 37 스크롤 2D 칼럼 차트 예제>



<그림 24 스크롤 2D 칼럼 차트 출력 모습>

- 스크롤 이미지 CSS 적용 시키기.

스크롤 차트에서 스크롤에 해당되는 이미지는 HTML의 CSS에서 적용합니다.

```

<style>
/* 스크롤 바 전체 Div */
.rMateChartH5__ScrollBar {
    border : 1px solid #78CBDF;
}

/* 수평 스크롤 바 트랙 Div */
.rMateChartH5__HScrollTrack {
    background : url('./Assets/h_scroll_track.png') repeat-x;
}

```

```
/* 수평 스크롤 바 썸 Div */
.rMateChartH5_HScrollThumb {
    background:url('./Assets/h_scroll_thumb.png') repeat-x;
    border : 1px solid #78CBDF;
    cursor : pointer;
}

/* 수평 스크롤 바 썸 Div - hover */
.rMateChartH5_HScrollThumb:hover{
    background:url('./Assets/h_scroll_thumb_hover.png') repeat-x;
}

/* 수평 스크롤 바 오른쪽 화살표 Div */
.rMateChartH5_HScrollUpArrow {
    background:url('./Assets/right_scroll_arrow.png') no-repeat;
    cursor : pointer;
}

/* 수평 스크롤 바 왼쪽 화살표 Div */
.rMateChartH5_HScrollDownArrow {
    background:url('./Assets/left_scroll_arrow.png') no-repeat;
    cursor : pointer;
}

/* 수직 스크롤 트랙 Div */
.rMateChartH5_VScrollTrack {
    background : url('./Assets/v_scroll_track.png') repeat-y;
}

.rMateChartH5_VScrollThumb {
    background:url('./Assets/v_scroll_thumb.png') repeat-y;
    border : 1px solid #78CBDF;
    cursor : pointer;
}

.rMateChartH5_VScrollThumb:hover{
    background:url('./Assets/v_scroll_thumb_hover.png') repeat-y;
```

```

}

.rMateChartH5_VScrollUpArrow {
    background:url('./Assets/up_scroll_arrow.png') no-repeat;
    cursor : pointer;
}

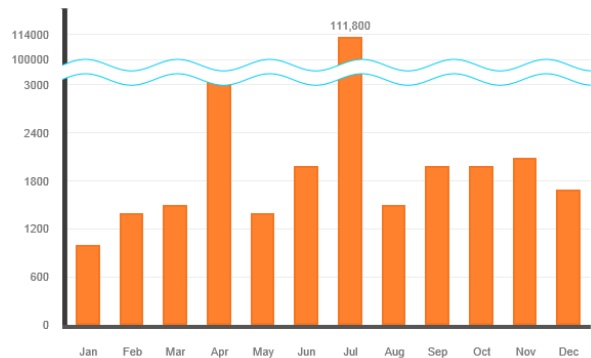
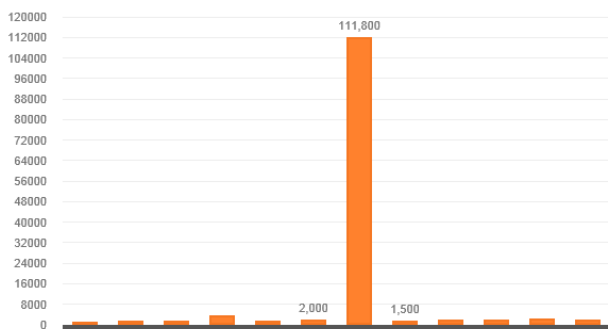
.rMateChartH5_VScrollDownArrow {
    background:url('./Assets/down_scroll_arrow.png') no-repeat;
    cursor : pointer;
}

</style>

```

## 6.19. 브로큰 축(Broken Axis)차트

브로큰 축 차트는 데이터간의 값 편차가 심할 경우 사용합니다. 예를 들어 대부분의 데이터가 3000미만 인데 특정 데이터는 10만이 넘어가는 데이터와 같은 경우 차트는 이를 모두 표현하기 위하여 y축에대한 값을 10만을 넘는 값을 설정하게 됩니다. 이로 인하여 3000미만의 데이터들의 변동에 대한 것을 한눈에 알아 볼 수 없습니다.



일반 차트( 데이터 3000미만과 10만의 데이터 )

브로큰 축 차트( 데이터 3000미만과 10만의 데이터 )

< 그림 25 브로큰 축 차트 출력 모습 >

브로큰 축 차트는 위와 같은 데이터 값 편차가 클 경우 사용하면 데이터 변동에 대한 것을 한눈에 보기 좋게 됩니다.

브로큰 축 차트 레이아웃은 아래와 같이 설정 합니다.

```

<?xml version="1.0" encoding="utf-8"?>
<rMateChart backgroundColor="0xFFFFF" borderStyle="solid" cornerRadius="5">

```

```

<Options>
  <Caption text= "Annual Report"/>
</Options>
<Column2DChart showDataTips= "true">
  <horizontalAxis>
    <CategoryAxis categoryField= "Month"/>
  </horizontalAxis>
  <verticalAxis>
    <!-- BrokenAxis를 설정 하시려면 해당 축이 설정될 곳에 BrokenAxis를 설정하십시오. -->
    <BrokenAxis id= "vAxis" brokenMinimum= "3000" brokenMaximum= "110000"
brokenRatio= "0.8"/>
    <!-- brokenMinimum - Broken축이 시작될 값 입니다. -->
    <!-- brokenMaximum - Broken축이 끝날 값 입니다. -->
    <!-- brokenRatio - Broken축이 그려질 위치 값입니다. 0 ~ 1이 유효값이며 0에 가까울수록 축의 최소값에 -->
    <!-- 가까워지며 1에 가까워질수록 축의 최대값에 가까워집니다. -->
  </verticalAxis>
  <verticalAxisRenderers>
    <BrokenAxis2DRenderer axis= "{vAxis}"/>
    <!-- BrokenAxis를 사용할 경우에 BrokenAxis2DRenderer를 설정 합니다. -->
    <!-- 이 외의 렌더러를 설정할 경우 올바르게 표현이 되지 않습니다. -->
  </verticalAxisRenderers>
  <series>
    <Column2DSeries yField= "Profit" displayName= "Profit">
      <showDataEffect>
        <SeriesInterpolate/>
      </showDataEffect>
    </Column2DSeries>
  </series>
  <annotationElements>
    <CrossRangeZoomer enableZooming= "false"/>
    <!-- BrokenAxis는 확대/축소를 지원하지 않습니다. -->
  </annotationElements>
</Column2DChart>
</rMateChart>

```

< 예제 38 브로큰 축 차트 예제 >

주의 사항 -

브로큰 축 차트는 Line,Area,Column,Bar차트 이외의 차트는 지원하지 않습니다.

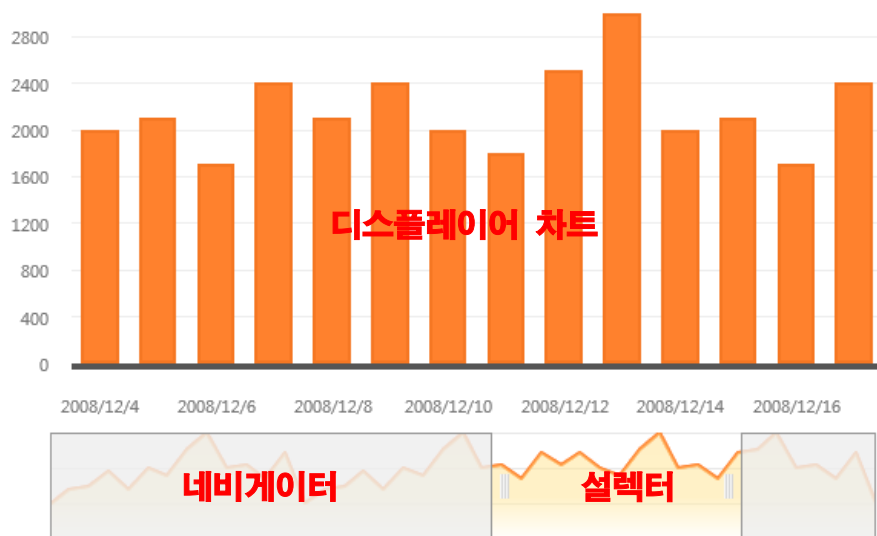
브로큰 축 차트는 CrossRangeZoomer 의 확대 / 축소, HistoryChart 를 지원하지 않습니다.

## 6.20. 히스토리 차트

히스토리 차트의 기본적인 사상은 스크롤 차트와 같습니다. 그러나 히스토리 차트는 더 많은 데이터의 경

우 전체 데이터의 흐름을 나타내는 네비게이터(또는 overview)가 있어 특정 데이터 영역을 자유롭게 넘나들며 특정 데이터의 개수가 아닌 유동적으로 데이터의 개수를 사용자가 조정하여 표현할 수 있습니다. 히스토리 차트는 <HistoryChart/> 노드로 시작과 끝을 지시합니다. 히스토리 차트의 구성은 디스플레이어, 네비게이터, 선택터로 구성되어 있습니다.

- 디스플레이어(displayer)  
디스플레이어 차트는 사용자가 선택된 영역만이 표시하는 차트입니다. 실제로 사용자가 보고자 하는 부분의 데이터를 표시합니다.
- 네비게이터(navigator)  
네비게이터는 전체 데이터를 표현하여 데이터의 흐름을 전체적으로 볼 수 있습니다. 이는 곧 오버뷰 역할을 합니다.
- 선택터(selector)  
선택터는 네비게이터에서 사용자가 특정 영역을 선택하게끔 하는 역할을 합니다. 선택터에서 선택된 영역이 곧 디스플레이어 차트에 표현됩니다.



<그림 26 히스토리 차트 구성도>

HistoryChart의 속성 및 유효 값 설명에 대한 표입니다.

속성 명	유효 값	설 명
displayerChart	Displayer	히스토리 차트의 디스플레이어차트를 정의합니다.
navigator	Navigator	네비게이터를 정의합니다.
selector	HistoryRangeSelector	선택터를 정의합니다.

### <표 9 히스토리 차트 속성 및 유효값 설명>

<HistoryChart/>의 속성으로 각각 displayerChart 와 navigator 의 유효 값인 Displayer 와 Navigator 는 각각 히스토리에서만 적용되는 차트입니다. Displayer 는 Cartesian 공통 속성을 모두 갖고 있으며 Navigator 는 Area2DChart 의 모든 속성을 갖고 있습니다.

히스토리 차트 레이아웃 작성 예입니다.

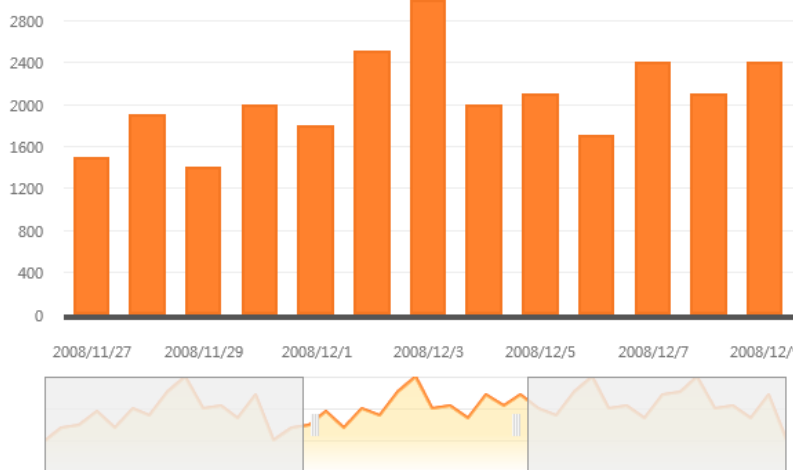
```
<rMateChart backgroundColor="0xFFFFEE" cornerRadius="12" borderStyle="solid">
  <Options>
    <Caption text="히스토리 2D 차트"/>
  </Options>
  <HistoryChart>
    <displayerChart> <!-- 디스플레이 차트 정의 -->
      <Displayer id="chart1" showDataTips="true" width="100%" height="100%">
        <horizontalAxis>
          <CategoryAxis id="mainHAxis" categoryField="Date"/>
        </horizontalAxis>
        <series>
          <Column2DSeries id="columnSeries"
            yField="Data1" fill="0xB0C759"
            displayName="Data1">
            <showDataEffect>
              <SeriesInterpolate duration="1000"/>
            </showDataEffect>
          </Column2DSeries>
        </series>
      </Displayer>
    </displayerChart>
    <navigator>
      <Navigator id="navi" width="100%" height="100">
        <horizontalAxis>
          <CategoryAxis categoryField="Date" id="naviHAxis"/>
        </horizontalAxis>
        <horizontalAxisRenderers>
          <AxisRenderer axis="{naviHAxis}" visible="false"/>
        </horizontalAxisRenderers>
        <verticalAxis>
          <LinearAxis id="vAxis"/>
        </verticalAxis>
        <verticalAxisRenderers>
          <AxisRenderer axis="{vAxis}" visible="false"/>
        </verticalAxisRenderers>
      </Navigator>
    </navigator>
  </HistoryChart>
</rMateChart>
```

```

        <backgroundElements>
        <GridLines direction="horizontal"/>
        </backgroundElements>
        <series>
        <Area2DSeries name="A" yField="Data1"/>
        </series>
    </Navigator>
</navigator>
<selector><!-- 시작 시 보여지는 차트는 총 데이터의 약 30%이며 센터 영역입니다. -->
    <HistoryRangeSelector width="100%" startingRange="center" visibleItemSize="30"/>
</selector>
</HistoryChart>
</rMateChart>

```

<예제 39 히스토리 차트 레이아웃 예제>



<그림 27 히스토리 차트 출력 모습>

- 히스토리 차트의 네비게이터의 손잡이(Thumb) 이미지 CSS 적용시키기

히스토리 차트의 네비게이터에 해당되는 이미지는 HTML 의 CSS 에서 적용합니다.

```

<style>
/* 히스토리 차트 범위 선택 Thumb */
.rMateChartH5_ChartRangeSelector_Thumb {
    background:url('./Assets/selector.png') no-repeat;
}
</style>

```



## 6.21. From-To 차트

From-To 차트는 칼럼 차트 계열과 바 차트 계열에 시작점 데이터와 끝 점 데이터를 삽입하여 데이터 영역을 표현하는 차트입니다.

이 데이터 영역을 활용하여 waterfall 차트, step 차트 생성이 가능합니다.

칼럼 시리즈(Column2DSeries, Column3DSeries) 와 바 시리즈(Bar2DSeries, Bar3DSeries) 그리고 영역 시리즈(Area2DSeries) 의 추가 속성으로 minField 가 시작점 데이터에 해당됩니다.

From-To 차트의 모든 속성은 각각 칼럼, 바, 영역 차트와 동일합니다.

이를 활용하여 칼럼 2D 차트로 waterfall 차트를 생성해 보도록 하겠습니다.

먼저 minField 에 해당하는 수치필드를 포함한 데이터를 정의합니다.

배열 형태로 정의하면 아래와 같습니다.

```
var data= [
    {"cat": "1월", "from": 1000, "to": 5000},
    {"cat": "2월", "from": 5000, "to": 12000},
    {"cat": "3월", "from": 12000, "to": 16000},
    {"cat": "4월", "from": 16000, "to": 10000},
    {"cat": "5월", "from": 10000, "to": 100},
    {"cat": "6월", "from": 100, "to": -1000},
    {"cat": "7월", "from": -1000, "to": 3000},
    {"cat": "8월", "from": 3000, "to": 8000},
    {"cat": "9월", "from": 8000, "to": 12000},
    {"cat": "10월", "from": 12000, "to": 14000},
    {"cat": "11월", "from": 14000, "to": 7500},
    {"cat": "12월", "from": 7500, "to": 2500},
    {"cat": "내년 목표", "from": 0, "to": 20000}
];
```

<예제 40 From-To 차트 데이터 정의 예제>

위 데이터는 한해 매출 증감을 나타낸 예시 데이터입니다.

이 데이터를 차트로 표현하기 위해 작성된 레이아웃은 아래와 같습니다.

```
<rMateChart backgroundColor= '0xFFFFEE' cornerRadius= '12' borderStyle= 'solid'>
    <Options>
        <Caption text= '2009 월별 매출액 증감 비교 (Waterfall)'/>
        <SubCaption text= '단위 : 만원' textAlign= 'right' fontSize= '11' paddingRight= '20'>
    </Options>
    <NumberFormatter id= 'fmt'>
```

```

<Column2DChart showDataTips='true' dataTipJsFunction='dataTipFunc' fontSize='11'>
  <horizontalAxis>
    <CategoryAxis categoryField='cat'/>
  </horizontalAxis>
  <verticalAxis>
    <LinearAxis formatter='{fmt}'/>
  </verticalAxis>
  <series>
    <Column2DSeries id='series1' minField='from' yField='to' fillJsFunction='fillFunc'
itemRenderer='BoxItemRenderer'>
      <showDataEffect>
        <SeriesSlide direction='up' duration='1000'/>
      </showDataEffect>
      <stroke>
        <Stroke weight='1' color='0x999999' alpha='0.7'/>
      </stroke>
    </Column2DSeries>
  </series>
</Column2DChart>
</rMateChart>

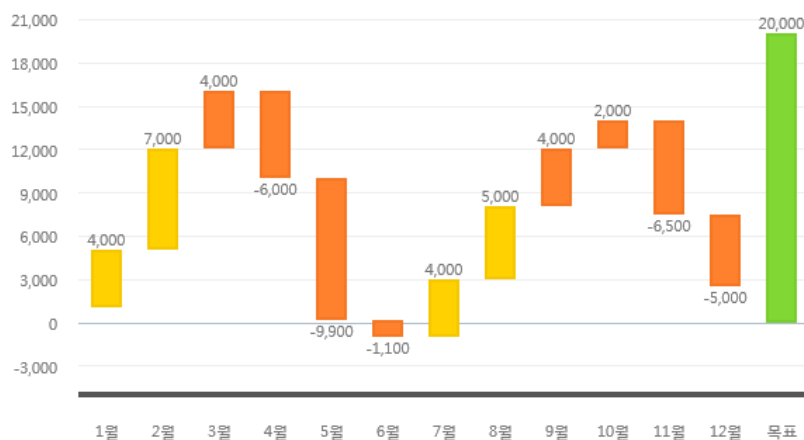
```

<예제 41 From-To 차트 레이아웃 예제>

위 레이아웃 예제에서 minField 속성에 데이터의 from 필드를 삽입한 것을 볼 수 있습니다. 따라서 from 필드는 시리즈 아이템들의 시작점이 됩니다.

또한 fillJsFunction 과 dataTipJsFunction 을 정의하여 데이터팁과 채우기색을 사용자 정의 한 것을 볼 수 있습니다. 데이터팁과 채우기색의 사용자 정의 함수에 대한 설명은 "7.9 사용자 정의 함수 설정하기"를 참고하십시오.

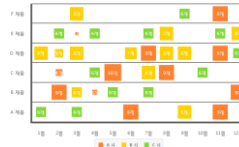

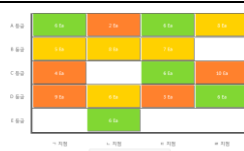

위 예제를 출력한 결과는 아래와 같습니다.



<그림 28 From-To 차트(waterfall) 출력화면>

## 6.22. 매트릭스 차트

매트릭스 차트는 x,y 값으로 위치를 정하고 z 값으로 크기를 정하여 표현하는 차트입니다.  
 매트릭스 차트의 표현 유형으로는 아래와 같습니다.

매트릭스 차트의 유형(type)	설 명	예제 모습
renderer	x,y로 위치를 잡고 z로 도형 크기를 정하여 표현합니다.	
image	x,y로 위치를 잡고 z로 이미지 크기를 정하여 표현합니다.	
fill	x,y 로 위치를 잡으며 z값은 무시되어 해당 칸을 색채움으로 표시합니다.	
plot	x,y로 위치를 잡으며 z값은 무시되어 해당칸에 plot형태로 표현합니다.	

<표 10 매트릭스 차트의 유형 표>

매트릭스 차트 유형을 결정하는 방법은 아래와 같습니다.

<code>&lt;Matrix2DChart showDataTips="true" type="renderer" selectionMode="single"&gt;</code>
---

Matrix2DChart 의 속성으로는 아래와 같습니다.

속성 명	유효 값	설 명
type	renderer, image, fill, plot (default : renderer)	매트릭스가 그려질 때의 표현 방법입니다.
drawType	radius, area (default : radius)	매트릭스 차트가 그려질 때 어떤 기준으로 그릴지 정합니다. radius : 반지름 기준, area : 면적 기준

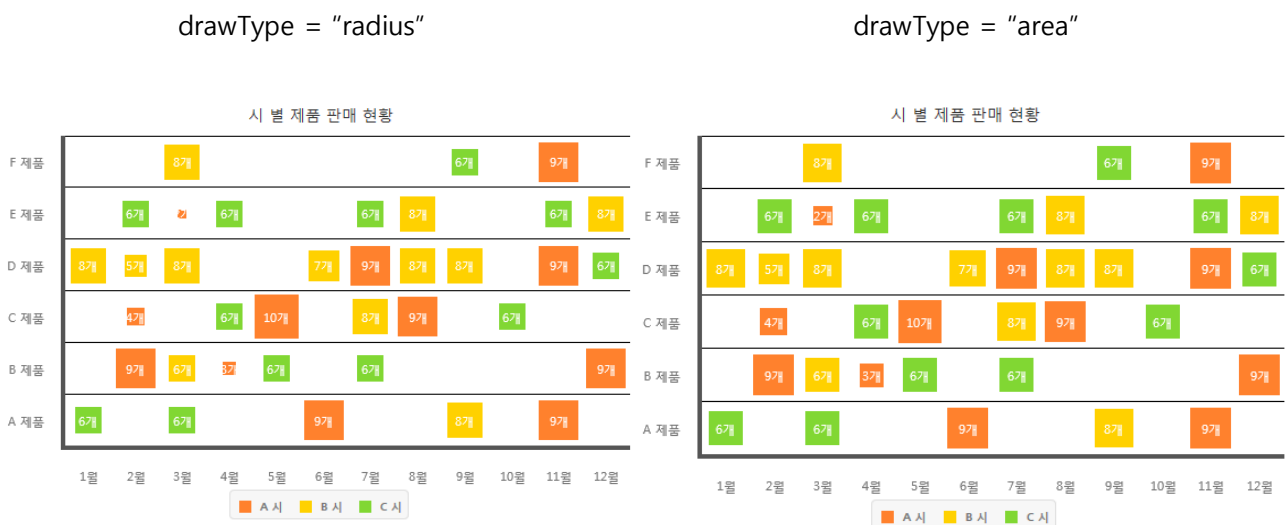
		( fill, plot은 무시됩니다 )
--	--	-----------------------

<표 11 매트릭스 차트 속성 표>

Matrix2DChart 의 type 이 renderer, image 일 때 어떤 기준으로 그릴지 정하는 방법은 아래와 같습니다.

```
<Matrix2DChart showDataTips="true" type="renderer" drawType="radius" selectionMode="single">
```

아래 그림은 drawType="radius" 와 "area"의 비교 모습입니다.



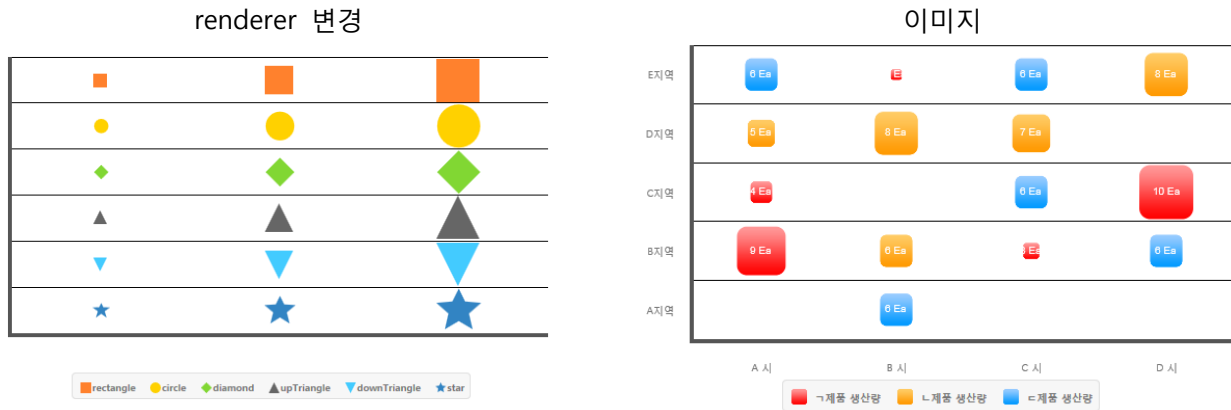
매트릭스 차트의 기본도형( renderer )을 변경하고 싶으시다면 아래와 같이 하십시오.

```
<Matrix2DSeries ..... renderer="circle" // renderer="star" .....>
```

매트릭스 차트를 이미지로 표현하고 싶으시다면 imageSource 에 해당 이미지 주소를 넣어주십시오.

```
<Matrix2DSeries ..... imageSource="/Images/icon1.png" .....>
```

위 속성들을 변경 후의 모습입니다.




## 6.23. 이미지 차트

이미지 차트는 칼럼 형태의 차트를 특정 이미지로 표현하는 차트입니다. 이미지 차트의 유형을 결정하는 요소는 크게 두가지로 나뉩니다.

- 첫째, 이미지 고유의 비율에 따라 이미지를 표현할지의 여부
- 둘째, 데이터 하나를 표현할 때 단일 이미지 또는 다중 이미지를 사용할지의 여부

위 두가지 옵션에 따라 결정되는 이미지 차트의 유형은 아래와 같습니다.

이미지 차트 유형 (imageDisplayType)	이미지 고유 비율 (true,false)	설 명	예제 모습
single	True(정비율)	데이터를 단일 이미지로 표현하되 이미지 고유 사이즈 유지, 남은 여백은 막대를 세웁니다.	
	False(차등비율)	데이터를 단일 이미지로 표현하되 이미지 고유 사이즈가 아닌 차트가 결정한 사이즈대로 표현합니다.	
singleRepeat	True(정비율)	데이터를 단일 이미지로 표현하되 이미지 고유 사이즈 유지, 남은 여백은 <b>이미지의 반복</b> 으로 처리	
	False(차등비율)	singleRepeat 유형에서의 차등비율은 존재하지 않습니다.	X
multiple	True(정비율)	Multiple 유형에서의 차등비율은 존재하지 않습니다.	X

	False(차등비율)	데이터를 다중 이미지로 표현합니다. 각각의 이미지는 고유 값을 갖습니다. 이 값은 차트의 사이즈에 의해 계산되어 이미지로 전달됩니다.	
--	-------------	--	---

<표 12 이미지 차트 유형 표>

기본적으로 이미지 차트의 모든 속성은 칼럼차트와 동일합니다. 이미지 차트 고유 속성은 이미지 시리즈의 imgSource 입니다.

imgSource 속성은 이미지의 경로와 어떻게 표현할지를 결정하는 속성입니다.

```
<ImageSeries yField= "Data1" imageDisplayType= "singleRepeat" displayName= "분양수" styleName= "seriesStyle"
formatter= "{numFmt}">
  <imgSource>
    <ImageSourceItem url= "../Samples/Images/10000.png"/>
  </imgSource>
  <showDataEffect>
    <SeriesSlide duration= "1000" direction= "up"/>
  </showDataEffect>
</ImageSeries>
```

위 예제는 단일 시리즈를 정의한 것입니다. imageDisplayType 을 singleRepeat 로 설정하여 하나의 이미지를 반복적으로 처리하도록 하였습니다.

그리고 imgSource 노드에서 이미지의 경로를 정의합니다.

imgsource 노드의 자식 노드로 설정가능한 유효값은 ImageSourceItem 노드입니다.

ImageSourceItem 노드는 이미지의 경로와 이미지를 정비율로 표현 할지 여부를 나타냅니다.

속성명	유효값	설 명
maintainAspectRatio	True   false(기본값 : true)	이미지 고유 비율대로 표현할지 여부
url	이미지 주소(URL)	이미지 파일의 주소입니다.
Value	Number	이미지가 갖을 고유 value 입니다.(multiple 에 서만 해당됩니다.)

<표 13 이미지 차트 ImageSourceItem 속성 설명>

3 개의 이미지가 각각 고유 값을 갖는 multiple 이미지 차트 레이아웃을 작성하면 아래와 같습니다.

```
<ImageChart id= "chart" showDataTips= "true" gutterLeft= "20" gutterRight= "20" showLabelVertically= "true">
  <!--X축 정의 -->
  <horizontalAxis>
    <CategoryAxis id= "hAxis" categoryField= "Region"/>
```

```

    </horizontalAxis>
<!--Y축 정의 -->
    <verticalAxis>
        <LinearAxis id= "vAxis"/>
    </verticalAxis>
    <series>
<!--단일 이미지 시리즈 정의 -->
    <ImageSeries yField= "Data1" imageDisplayType= "multiple" styleName= "seriesStyle"
formatter= "{numFmt}">
        <imgSource>

            <!--value 100 을 갖는 첫번째 이미지 -->
            <ImageSourceItem maintainAspectRatio= "false" url= "../Samples/Images/3-1.png"
value= "100"/>

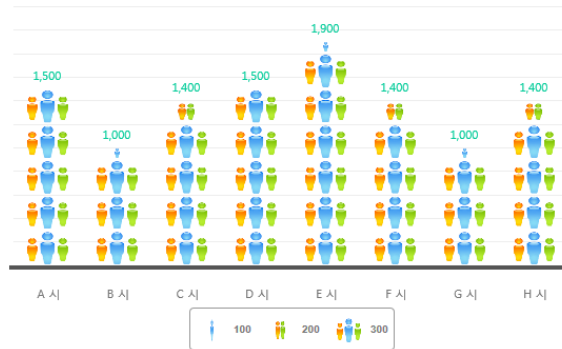
            <!--value 200 을 갖는 두번째 이미지 -->
            <ImageSourceItem maintainAspectRatio= "false" url= "../Samples/Images/3-2.png"
value= "200"/>

            <!--value 300 을 갖는 세번째 이미지 -->
            <ImageSourceItem maintainAspectRatio= "false" url= "../Samples/Images/3-3.png"
value= "300"/>

        </imgSource>
    </ImageSeries>
</series>
<horizontalAxisRenderers>
    <AxisRenderer axis= "{hAxis}" fontSize= "11"/>
</horizontalAxisRenderers>
<!--Y 축은 visible off -->
    <verticalAxisRenderers>
        <AxisRenderer axis= "{vAxis}" visible= "false" includeInLayout= "false"/>
    </verticalAxisRenderers>
</ImageChart>

```

위 레이아웃을 실행한 모습은 아래와 같습니다.

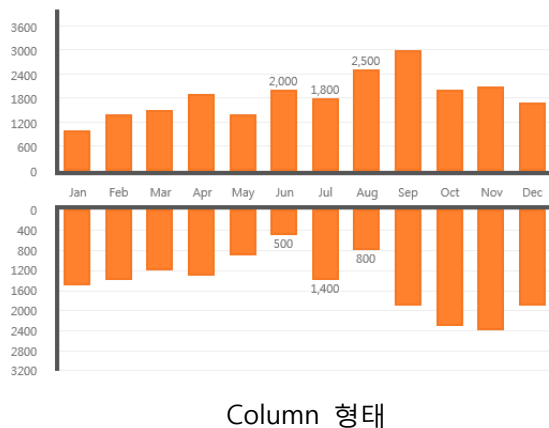
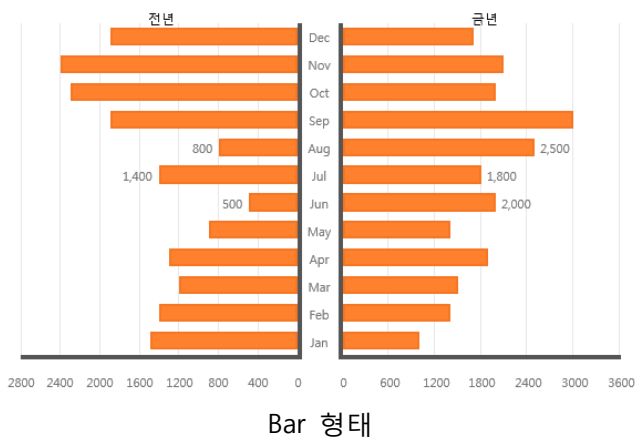


<그림 29 이미지 차트 실행화면>

## 6.24. wing 차트

wing 차트는 기존 차트와는 달리 양옆 혹은 위아래로 출력되는 형태의 차트 입니다. 예를 들어 해당 차트는 특정 제품에 대하여 여성 이용자와 남성 이용자의 구매 현황 같은 것을 시각적으로 보기 쉽게 표현하는 차트 입니다.

현재 wing 차트에는 두가지의 형태만 존재합니다. 해당 형태는 Column 과 Bar 형 두가지만 존재합니다.



위 그림들을 살펴보면 양옆, 위아래로 해당 월에 대한 값들을 한눈에 비교해 보실 수 있습니다. wing 차트의 컬럼 형태를 출력하고 싶으시다면 레이아웃을 아래와 같이 작성 하십시오.

```
<?xml version="1.0" encoding="utf-8"?>
<rMateChart backgroundColor="0xFFFFFFFF" borderStyle="solid" cornerRadius="5">
  <Options>
    <Caption text="Annual Report"/>
  </Options>
  <!-- wing 컬럼 2D 차트를 생성 하실 경우에는 Column2DWingChart를 정의 하십시오. -->
  <Column2DWingChart showDataTips="true">
    <horizontalAxis>
      <CategoryAxis categoryField="Month"/>
    </horizontalAxis>
  </Column2DWingChart>
</rMateChart>
```



```

</horizontalAxis>
<series>
<!-- Column2DWingChart의 series속성으로는 Column2DWingSeries를 설정 하여야 합니다. -->
<!-- 일반 Column2D,3DChart에서는 데이터 필드로 yField만 존재 하였지만 Wing Chart에서는-->
<!-- yFieldOpp란 데이터 필드가 존재하므로 해당 데이터 필드도 설정 해주셔야 합니다. -->
<!-- yField는 위쪽 영역, yFieldOpp는 아래쪽 영역 입니다. -->
    <Column2DWingSeries yField= "Profit" yFieldOpp= "Cost" labelPosition= "inside">
        <showDataEffect>
            <WingSeriesInterpolate/>
        </showDataEffect>
    </Column2DWingSeries>
</series>
</Column2DWingChart>
</rMateChart>

```

위와 같이 작성하시면 Wing차트의 Column 형태를 출력 하실 수 있습니다.

## 6.25. 실시간-프리미엄 차트

실시간-프리미엄 차트는 6.15 실시간 차트를 업그레이드 한 차트입니다. 기본적으로 실시간 차트와 비교하여 통신 방법에 의한 변화는 없습니다. HTTPService 를 통하여 RPC 요청을 하고 응답으로 차트에 표현하게 됩니다. 그러나 실시간-프리미엄 차트만의 특징은 아래와 같습니다.

1. 차트에 표현되는 시리즈 각각에 해당 URL 을 통하여 주기를 다르게 표현할 수 있습니다. 예를 들어 3 개의 시리즈에 각각 5 초, 10 초, 1 분 단위로 개별 주기를 줄 수 있습니다.
2. 기존 실시간 차트는 오른쪽 방향에서 왼쪽 방향으로 특정 시간(또는 데이터량) 만큼 출력을 한 반면 실시간-프리미엄 차트는 왼쪽에서 오른쪽 방향으로 서버 사이드에서 결장한 시간 범위만큼 출력할 수 있습니다. 개별 시리즈의 주기가 다르므로 개별 시리즈가 표현될 시간 범위를 개별적으로 줄 수 있습니다.
3. 초기 데이터를 클라이언트의 차트에 표현해야 할 경우 차트가 로딩 완료 된 시점에 초기 데이터를 받아와 출력할 수 있습니다.
4. 실시간-프리미엄 차트의 **X 축 정의는 항상 DateTimeAxis 로 정의**하여야 합니다. 즉, 기존 실시간 차트의 데이터량 만큼 출력시키는 CategoryAxis 는 지원하지 않습니다.

실시간-프리미엄 차트를 생성하기 위해서는 기존 실시간 차트처럼 <RealTimeChart/> 노드로 레이아웃을 작성하지 않습니다. 실시간-프리미엄 차트는 어떤 종류의 rMate 차트도 가능합니다.

단, 실시간-프리미엄 차트를 가능케 하는 <HttpMultiServiceRepeater /> 노드를 차트 아래에 정의하여야 합니다.

HttpMultiServiceRepeater 는 원격 호출 할 오퍼레이션들을 갖습니다. 이 오퍼레이션은 시리즈 개수만큼 정의되어 각 주기마다 RPC 요청하여 응답으로 해당 시리즈에 값을 넘겨 출력하게끔 합니다.

HttpMultiServiceRepeater 노드의 속성에 대한 설명과 유효값은 아래와 같습니다.

속성명	유효값	설 명
baseUrl	URL 주소	RPCItem 속성 'url'의 기본 URL에 해당됩니다. 즉, RPCItem 의 url 은 "baseUrl + RPCItem의 url" 이 됩니다.
method	get   post (기본값 :get)	HTTP 프로토콜 메소드입니다. Get방식인지 Post 방식인지를 결정.
requestTimeout	Second(초)	요청 후 응답 대기 시간입니다.
targetController	차트 id	RPCItem 의 target 의 컨트롤러에 해당됩니다. 즉, 언제나 차트가 됩니다.
showErrorMessage	true false(기본값:true)	RPC 요청 시 실패 또는 에러 발생 시 Alert 메시지를 띄울 지를 나타냅니다. 만약 showErrorMessage 속성을 false 로 설정한다면 어떤 메시지도 출력하지 않습니다.

<표 14 HttpMultiServiceRepeater 속성 및 유효값 설명표>

다음은 HttpMultiServiceRepeater 노드의 RPCList 값에 해당되는 RPCItem 노드의 속성 및 유효값에 대한 설명입니다.

속성명	유효값	설 명
name	String	RPCItem 의 name 입니다. 반드시 정의하십시오.(임의의 스트링)
url	String	RPCItem 속성 'url'의 기본 URL에 해당됩니다. 즉, RPCItem 의 url 은 "baseUrl + RPCItem의 url" 이 됩니다.
method	get   post (기본값 :get)	HTTP 프로토콜 메소드입니다. Get방식인지 Post 방식인지를 결정합니다.
target	시리즈의 id	RPCItem 해당 url 로 요청하여 받은 데이터를 표현할 시리즈를 나타냅니다. 각각의 RPCItem 은 각각의 시리즈가 됩니다
interval	Seconds(초)	RPC 요청 주기를 나타냅니다. 최초 차트 로딩시 요청을 보내고 주어진 interval 간격으로 다음 요청이 이루어집니다. 예를 들어 5로 설정한 경우 5초 간격으로 요청을 하게 됩니다. Interval을 설정하지 않은 경우 주기적으로 요청이 이루어지지 않습니다. 이런 경우는 비교 데이터 즉, 갱신만 하고자 할 때

		사용합니다.
concurrency	multiple   single   last(기본값:multiple)	<p>HTTP 동일 서비스 발생 시 처리 방법을 나타냅니다.</p> <p>multiple : 기존 요구를 취소하지 않고 모든 요청을 보냅니다.</p> <p>single : 한번에 1개의 요청만 가능합니다. 중복 발생 시 에러 Alert 메시지를 띄웁니다.</p> <p>last : 기존의 요청을 모두 취소하고, 마지막 요청만을 보냅니다.</p>
retryCount	Number(기본값:30)	RPC 요청 시 실패 응답이 온 경우 재시도 횟수를 나타냅니다.

<표 15 RPCItem 노드 속성 및 유효값 설명표>

예로 컴비네이션 2D 차트를 통하여 실시간-프리미엄 차트를 생성해 보겠습니다.

```

<rMateChart backgroundColor= "0xFFFFF" cornerRadius= "12" borderStyle= "solid">
  <Options>
    <Caption text= "다른 주기를 갖는 데이터 실시간 표현"/>
    <SubCaption text= "본 샘플은 랜덤 데이터입니다." textAlign= "right" fontSize= "11"/>
    <Legend fontSize= "11" useVisibleCheck= "true"/>
  </Options>

  // 각종 포메터 정의
  <DateFormatter id= "dateOrgFmt" formatString= "YYYY/MM/DD HH:NN:SS"/>
  <DateFormatter id= "dateFmt" formatString= "HH:NN:SS"/>
  <DateFormatter id= "dateFmt2" formatString= "HH:NN"/>
  <NumberFormatter id= "numFmt"/>

  // 칼럼시리즈와 라인 시리즈 2개를 갖는 컴비네이션 2D 차트 생성
  <Combination2DChart id= "chart" showDataTips= "true" dataTipMode= "multiple">
    <series>

      <!-- 5 초 주기 라인 시리즈 -->
      <Line2DSeries id= "lineSeries" xField= "date" yField= "data5" displayName= "Data(5
Sec)">

        <lineStroke>
          <Stroke color= "0xFF6666" weight= "2" alpha= "1"/>
        </lineStroke>
        <horizontalAxis>
          <DateTimeAxis id= "hAxis2" displayLocalTime= "true"
labelUnits= "minutes" dataUnits= "seconds" interval= "1" formatter= "{dateOrgFmt}" displayName= "Time"/>
        </horizontalAxis>
      </Line2DSeries>
    </series>
  </Combination2DChart>
</rMateChart>
  
```

```

        <verticalAxis>
            <LinearAxis id= "vAxis2" minimum= "0" maximum= "150"/>
        </verticalAxis>
    </Line2DSeries>

    <!-- 3 초 주기 라인 시리즈 -->
    <Line2DSeries id= "lineSeries2" xField= "date" yField= "data3" displayName= "Data(3 Sec)"
verticalAxis= "{vAxis2}" horizontalAxis= "{hAxis2}">
        <lineStroke>
            <Stroke color= "0x339966" weight= "1" alpha= "1"/>
        </lineStroke>
    </Line2DSeries>

    <!-- 누적량 -->
    <Column2DSeries id= "columnSeries" xField= "date" yField= "data60"
displayName= "누적량" itemRenderer= "BoxItemRenderer">
        <horizontalAxis>
            <DateTimeAxis id= "hAxis" displayLocalTime= "true"
labelUnits= "hours" dataUnits= "minutes" interval= "3" dataInterval= "10" formatter= "{dateOrgFmt}"
displayName= "Time"/>
        </horizontalAxis>
        <verticalAxis>
            <LinearAxis id= "vAxis" minimum= "0" maximum= "999"/>
        </verticalAxis>
        <fill>
            <SolidColor color= "0x6666FF"/>
        </fill>
    </Column2DSeries>
</series>
<horizontalAxisRenderers>
    <AxisRenderer axis= "{hAxis}" placement= "bottom" formatter= "{dateFmt2}"
tickLength= "30" minorTickLength= "0" tickPlacement= "inside" showLine= "false">
        <axisStroke>
            <Stroke weight= "1" color= "0x999999"/>
        </axisStroke>
        <tickStroke>
            <Stroke weight= "1" color= "0x6666FF" alpha= "0.5"/>
        </tickStroke>
    </AxisRenderer>
    <AxisRenderer axis= "{hAxis2}" placement= "bottom" formatter= "{dateFmt}">
        <axisStroke>
            <Stroke weight= "1" color= "0x999999"/>
        </axisStroke>
    </AxisRenderer>

```

```

    </horizontalAxisRenderers>
    <verticalAxisRenderers>
        <AxisRenderer axis="{vAxis}" placement="right" formatter="{numFmt}"/>
        <AxisRenderer axis="{vAxis2}" placement="left" formatter="{numFmt}"/>
    </verticalAxisRenderers>
</Combination2DChart>

// 실시간-프리미엄 차트 생성을 위한 HttpMultiServiceRepeater 정의.
    <HttpMultiServiceRepeater baseUrl="http://demo.riamore.net/realtimeSample/" targetController="{chart}"
requestTimeout="30">
        <RPCList>
            <RPCItem name="rpc1" url="data3Interval.jsp" target="{lineSeries2}" interval="3"
concurrency="last" retryCount="30"/>
            <RPCItem name="rpc2" url="data5Interval.jsp" target="{lineSeries}" interval="5"
concurrency="last" retryCount="30"/>
            <RPCItem name="rpc3" url="data23ToCurrent2.jsp" target="{columnSeries}"
interval="600" concurrency="last" retryCount="30"/>
        </RPCList>
    </HttpMultiServiceRepeater>
</rMateChart>

```

#### <예제 42 실시간-프리미엄 차트 레이아웃 예제>

위 샘플은 라인 2 개와 칼럼을 실시간으로 표현합니다. 각각의 주기는 3 초, 5 초, 10 분 입니다.

처음 차트 구동 시 초기 데이터를 위해 RPC 호출을 하여 출력한 후 주기마다 반복적으로 RPC 호출을 하여 데이터를 표현합니다.

라인 차트 2 개의 초기 데이터는 없는 상태(서버 사이드에서 이와 같이 작업함)로 차트가 구동 시부터 10 분 영역의 데이터를 실시간으로 뿌린 후 10 분이 지나면 뿌려진 데이터를 모두 삭제하고 다시 10 분 영역의 데이터를 각 주기 간격으로 출력합니다.

아래 칼럼 차트는 10 분동안 쌓여진 데이터의 누적량을 표현하는 예제로 10 분마다 RPC 호출하여 칼럼 하나씩을 더하게 됩니다.

칼럼 차트의 경우 설정된 알람 시간(매일 23 시 59 분)이 되면 출력된 데이터를 모두 갱신하게 됩니다

각 서버 스크립트 예제 파일은 제공된 시디의 다음 폴더 경로에서 확인해 볼 수 있습니다.

시디/Samples/RealtimeServerSamples/

위 예제를 위한 서버사이드 샘플을 살펴보도록 하겠습니다.

리얼 타임 차트 서버사이드 예제.

<http://demo.riamore.net/realtimeSample/hourDataToday.jsp>

---

#### 예제 작동 방식 설명

---

// 본 예제는 DB 데이터가 아닌 임의의 랜덤 데이터를 기반으로 합니다.

requestAllData = true 인 경우 현재 시(hour):00 부터 현재 시:분 까지 5초 간격 데이터를 생성합니다.

requestAllData = false 인 경우 현재 시:분에 해당하는 데이터 하나를 생성하게 됩니다.

차트 레이아웃에서 <RPCItem> 의 속성으로 interval 을 5로 주어 5초마다 하나의 데이터를 서버에서 가져와 기존 데이터와 adding 을 하게 됩니다.

<nextInitDate> 값에 설정된 시간이 되면 모든 데이터를 삭제하고 다시 요청하게 됩니다.(refresh)

아래 URL을 각각 복사하여 브라우저에 출력해 보십시오.

<http://demo.riamore.net/realtimeSample/hourDataToday.jsp?requestAllData=true>

<http://demo.riamore.net/realtimeSample/hourDataToday.jsp?requestAllData=false>

---

#### 파라미터 설명.

---

**requestAllData** : 차트가 모든 데이터를 요청하는 플래그입니다.

즉, 현재 차트에 뿌려진 데이터를 갱신하고자 할 때 사용합니다.

requestAllData=true 로 설정되어 호출하는 순간은 다음과 같습니다.

1. 처음 차트가 로딩되어 처음 RPC 요청할 때.
2. 현재 시간이 <nextInitDate> 시간을 넘은 경우 true 로 설정되어 RPC 요청합니다.  
requestAllData=true 로 설정된 경우 서버사이드는 다음과 같이 작업을 할 필요가 있습니다.  
이는 요청 후 응답으로 받은 데이터가 정확하다는 서로간의 통신법입니다.  
<infoMsg> 태그 밑에 <isInitData>true</isInitData> 로 설정.  
requestAllData=false 인 경우 <isInitData>false</isInitData> 여야 합니다.

**index** : 차트가 최근에 받은 자료의 index를 나타냅니다.

이는 서버사이드에서 응답으로 전달해준 자료입니다.

index 를 통하여 서버와 클라이언트 간 자료 중복을 줄일 수 있습니다.

다시 말해 차트가 이전에 받은 자료의 index를 다음 호출 때 파라미터로 넘겨줌으로써 서버는 index 다음 자료를 쉽게 넘겨줄 수 있습니다.

**dummy** : 이 값은 IE 의 캐시 문제를 피하기 위해 넣은 더미값입니다.

경우에 따라서는 유용하게 쓰일 수 있습니다.

1970년 1월 1일 00시 부터 현재 차트가 RPC 를 보낸 순간까지의 밀리섹컨드를 나타냅니다.

이 값은 서버의 시간 기준이 아닌 클라이언트 시간 기준입니다.

-----  
 XML 작성 규칙.  
 -----

클라이언트가 응답으로 가져갈 XML 은 다음 규칙을 따라야 합니다.

규칙 1. 루트 노드 다음으로 <infoMsg> 노드 정보 및 서브 노드 정보는 반드시 필요합니다. (이름 변경 불가)

규칙 2. 차트가 뿌릴 데이터에 해당되는 노드는 반드시 <item> 이어야 합니다. (이름 변경 불가)

-----  
 예제 XML 설명.  
 -----

현재 예제는 랜덤으로 값을 나타냅니다. (DB 데이터 값이 아님)

requestAllData 파라미터의 true/false 에 따라 다르게 XML 이 출력됩니다.

1. requestAllData = false 인 경우

데이터 한개에 해당되는 XML 를 생성합니다.

이 데이터는 차트가 기존 데이터에 adding 을 합니다.

2. requestAllData = true 인 경우.

차트가 처음 로딩 시 requestAllData=true 파라미터로 요청을 합니다. 이 때 서버사이드에서 적당한 초기 데이터를 차트에 넘기십시오.

현재 샘플에선 현재 시간의 00분 부터 현재 분(minutes)까지 누적 데이터를 생성합니다.

requestAllData = true 는 초기 데이터이므로 차트가 뿌려줄 누적 데이터를 모두 select 하는 것이 좋습니다.

infoMsg 노드의 <nextInitDate> 값 시간에 다시 true 로 설정되어 요청합니다.

그 외는 모두 false로 요청.

3. <infoMsg> 노드에 대한 설명.

1. <index> 노드.

현재 샘플에선 index 파라미터를 사용하여 샘플이 작성되지 않았습니다.

그러나 이 노드는 현재 서버가 작성한 데이터의 고유값을 삽입하면 유리합니다.

차트는 항상 RPC 요청 시 차트가 최근에 응답으로 받아간 index를 갖고 요청을 합니다.

그렇기 때문에 이 파라미터를 이용하면 다음 데이터를 쉽게 전달해 줄 수 있습니다.

처음 차트 로딩 시(즉, 이전에 받아간 데이터 없을 경우)엔 init 임.

2. <timeNow> 노드.

서버의 현재 시간을 나타냅니다. 이는 데이터를 갱신(초기화)할 때 사용됩니다.

모든 클라이언트의 데이터를 일괄적으로 갱신(초기화) 하기를 원할 때 서버의 현재 시간을 삽입하여 주십시오.

이 노드를 생략한다면 모든 클라이언트의 로컬 시간을 기준으로 endDate 와 비교하여 갱신됩니다.

즉, <timeNow> 가 없다면 갱신되는 시간은 클라이언트마다 다를 수 있습니다.

3. <isInitData> 노드.  
 requestAllData = true 로 파라미터를 받은 경우 isInitData 노드를 true 로 설정하세요.

이는 서버와 클라이언트간 규약입니다.

4. <startDate> 노드.  
 차트에 표시되는 시간의 초기 시간입니다. 이 노드가 없다면 처음 데이터를 기준으로 처음 시간이 뿌려집니다.

5. <endDate> 노드.  
 차트에 표시할 시간의 마지막입니다. 이는 데이터가 앞으로 표현될 시간입니다.(현재 시간과 무관, 단순 데이터의 시간임.)

6. <nextInitDate> 노드.  
 이 값보다 현재 시간이 커진다면 차트는 갱신을 시도합니다.  
 이 말은 현재 차트에 뿌려진 데이터를 모두 지우고  
 새로 서버에 초기 데이터를 요청하는 requestAllData=true 파라미터로 요청을 합니다.

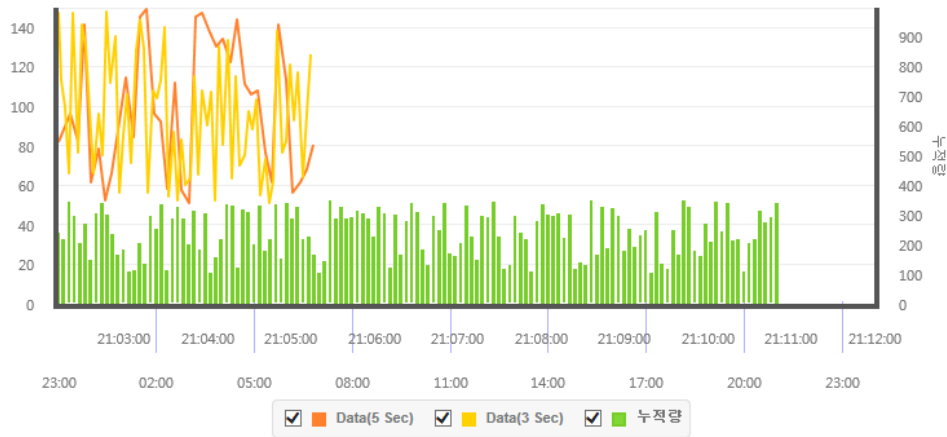
이 값은 실제 현재 시간과 관련이 있습니다. 또한 timeNow 노드 값과 깊은 관련이 있습니다.(timeNow 참고)

예를 들어 한달전 데이터를 현재 10분 주기로 보여주고 있는데 특정 시점에서 차트의 <endDate> 값까지 출력한 경우  
 이 시간을 참조하여 초기화를 요청하게 됩니다. (초기화는 requestAllData=true 파라미터 설정임).

<예제 43 실시간-프리미엄 서버사이드 샘플 설명>

위 레이아웃과 서버 샘플을 실행한 화면은 아래와 같습니다.





<그림 30 실시간-프리미엄 차트 실행화면>

## 6.26. 캔들스틱 차트

캔들스틱 차트는 전문적인 주식차트의 축소 버전입니다. 일반적인 주식차트와 같이 시가, 종가, 고가, 저가 거래량의 데이터를 설정하여 출력 할 수 있습니다. 그러나 전문적인 주식차트와 같이 마우스로 특정 지점에 선을 긋거나 특정 기호를 삽입하는 기능은 지원하지 않고 있지 않습니다.

아래는 캔들차트에서 지원가능한 기능들입니다.

- 캔들차트 안에서의 최소, 최대값을 표현합니다.
- 최소, 최대값을 사용자가 원하는 형태로 출력할 수 있습니다.
- 특정 데이터에 공시에 대한 데이터가 존재한다면 공시기호를 표현할 수 있습니다.
- 공시기호를 일반 itemRenderer 로 표현가능하며 div 객체를 이용하여 특정 문자, 이미지를 표현할 수 있습니다.
- div 로 된 공시기호는 마우스 클릭 이벤트를 받을 수 있습니다.
- 시가와 종가에따라 선 색상, 막대 테두리, 막대 배경색상등을 변경 할 수 있습니다.
- 차트에 보여지는 아이템의 갯수를 변경 할 수 있습니다.



### <그림 31 캔들차트 기본 화면 >

```

<rMateChart backgroundColor="0xFFFFFF" cornerRadius="12" borderStyle="solid">
  <Options>
    <Caption text="Riamore CandleChart"/>
  </Options>
  // 각종 포메터 정의
  <NumberFormatter id="nft" precision="0"/>
  // 한 화면에 두개의 차트를 설정 할 수 있는 DualChart 설정
  <DualChart leftGutterSyncEnable="true" rightGutterSyncEnable="true">
    // mainChart와 subChart의 왼쪽, 오른쪽 여백을 동기화 시킵니다.
    <mainChart>
      <Candlestick2DChart showDataTips="true" paddingBottom="0">
        <horizontalAxis>
          <CategoryAxis id="hAxis" categoryField="date"/>
        </horizontalAxis>
        <verticalAxis>
          <LinearAxis baseAtZero="false"/>
        </verticalAxis>
        <series>
          // openField : 시가 closeField : 종가 highField : 고가 lowField : 저가
          // showMaxValueLabel : 가장 큰 값 수치를 표현합니다.
          // showMinValueLabel : 가장 작은 값 수치를 표현합니다.
          <Candlestick2DSeries openField="openprc" closeField="closeprc"
            highField="high" lowField="low" showMinValueLabel="true" showMaxValueLabel="true"
            maxLabelJsFunction="maxLabelFunc" minLabelJsFunction="minLabelFunc">
            // 시가 보다 종가가 높을 경우 막대의 색상 입니다.
            <fill>
              <SolidColor color="#ff0000" alpha="0.5"/>
            </fill>
            // 시가 보다 종가가 높을 경우 고가와 저가를 잇는 선의
            <stroke>
              <Stroke color="#ff0000"/>
            </stroke>
            // 시가 보다 종가가 높을 경우 막대의 테두리 색상입니다.
            <boxStroke>
              <Stroke color="#ff0000"/>
            </boxStroke>
            // 종가 보다 시가가 높을 경우 막대의 배경 색상입니다.
            <declineFill>
              <SolidColor color="#0000ff" alpha="0.5"/>
            </declineFill>
            // 종가 보다 시가가 높을 경우 고가와 저가를 잇는 선의
  
```

색상 입니다..

```

        <declineStroke>
            <Stroke color= "#0000ff"/>
        </declineStroke>
        // 종가 보다 시가가 높을 경우 막대의 테두리 색상입니다.
        <declineBoxStroke>
            <Stroke color= "#0000ff"/>
        </declineBoxStroke>
    </Candlestick2DSeries>
</series/>
<horizontalAxisRenderers>
    <Axis2DRenderer placement= "bottom" axis= "{hAxis}"
tickLength= "0"/>
    </horizontalAxisRenderers>
    <annotationElements>
// syncCrossRangeZoomer : mainChart의 십자선과 동기화 시킬 subChart의 CrossRangeZoomer id를 설정합니다.
    <CrossRangeZoomer id= "candleCRZ" enableZooming= "false" syncCrossRangeZoomer= "{columnCRZ}"
zoomType= "both" horizontalLabelFormatter= "{nft}"/>
    </annotationElements>
</Candlestick2DChart>
</mainChart>
<subChart>
    <Column2DChart showDatatips= "true" height= "20%" paddingTop= "0"
paddingBottom= "0" gutterTop= "4">
        <horizontalAxis>
            <CategoryAxis id= "hAxis2" categoryField= "date"/>
        </horizontalAxis>
        <verticalAxis>
            <LinearAxis baseAtZero= "false"/>
        </verticalAxis>
        <series>
            <Column2DSeries yField= "trdvolume"
itemRenderer= "BoxItemRenderer">
                <fill>
                    <SolidColor color= "#eca614"/>
                </fill>
            </Column2DSeries>
        </series>
        <horizontalAxisRenderers>
            <Axis2DRenderer axis= "{hAxis2}" showLabels= "false"
tickLength= "0"/>
        </horizontalAxisRenderers>
        <annotationElements>
// syncCrossRangeZoomer : subChart의 십자선과 동기화 시킬 mainChart의 CrossRangeZoomer id를 설정합니다.

```

```

<CrossRangeZoomer id="columnCRZ" enableZooming="false"
syncCrossRangeZoomer="{candleCRZ}" horizontalLabelFormatter="{nft}" verticalLabelPlacement="top"/>
</annotationElements>
</Column2DChart>
</subChart>
<dataSelector>
  <DualScrollBar inverted="true" visibleItemSize="50"/>
</dataSelector>
</DualChart>
</rMateChart>

```

<예제 44 캔들차트 기본 예제 >



<그림 32 캔들차트 공시 표현 화면 >

```

<rMateChart backgroundColor="0xFFFFFF" cornerRadius="12" borderStyle="solid">
  <Options>
    <Caption text="Riamore CandleChart"/>
  </Options>
  // 각종 포메터 정의
  <NumberFormatter id="nft" precision="0"/>
  // 한 화면에 두개의 차트를 설정 할 수 있는 DualChart 설정
  <DualChart leftGutterSyncEnable="true" rightGutterSyncEnable="true">
    // mainChart와 subChart의 왼쪽, 오른쪽 여백을 동기화 시킵니다.
    <mainChart>
      <Candlestick2DChart showDataTips="true" paddingBottom="0">
        <horizontalAxis>
          <CategoryAxis id="hAxis" categoryField="date"/>
        </horizontalAxis>
        <verticalAxis>

```

```

    <LinearAxis baseAtZero="false"/>
  </verticalAxis>
  <series>
    // openField : 시가 closeField : 종가 highField : 고가 lowField : 저가
    // showMaxValueLabel : 가장 큰 값 수치를 표현합니다.
    // showMinValueLabel : 가장 작은 값 수치를 표현합니다.
    // symbolField : 공시 데이터를 설정하는 데이터 필드 입니다.
    // symbolRenderer : 공시 데이터의 존재유무를 알려주는 렌더러입니다.
    <Candlestick2DSeries openField="openprc" closeField="closeprc"
    highField="high" lowField="low" showMinValueLabel="true" showMaxValueLabel="true"
    maxLabelJsFunction="maxLabelFunc" minLabelJsFunction="minLabelFunc" symbolField="gongsi"
    symbolRenderer="UpArrowItemRenderer">

      // 시가 보다 종가가 높을 경우 막대의 색상 입니다.
      <fill>
        <SolidColor color="#ff0000" alpha="0.5"/>
      </fill>
      // 시가 보다 종가가 높을 경우 고가와 저가를 잇는 선의
      색상 입니다.

      <stroke>
        <Stroke color="#ff0000"/>
      </stroke>
      // 시가 보다 종가가 높을 경우 막대의 테두리 색상입니다.
      <boxStroke>
        <Stroke color="#ff0000"/>
      </boxStroke>
      // 종가 보다 시가가 높을 경우 막대의 배경 색상입니다.
      <declineFill>
        <SolidColor color="#0000ff" alpha="0.5"/>
      </declineFill>
      // 종가 보다 시가가 높을 경우 고가와 저가를 잇는 선의
      색상 입니다..

      <declineStroke>
        <Stroke color="#0000ff"/>
      </declineStroke>
      // 종가 보다 시가가 높을 경우 막대의 테두리 색상입니다.
      <declineBoxStroke>
        <Stroke color="#0000ff"/>
      </declineBoxStroke>
    </Candlestick2DSeries>
  </series/>
  <horizontalAxisRenderers>
    <Axis2DRenderer placement="bottom" axis="{hAxis}"
    tickLength="0"/>
  </horizontalAxisRenderers>

```

```

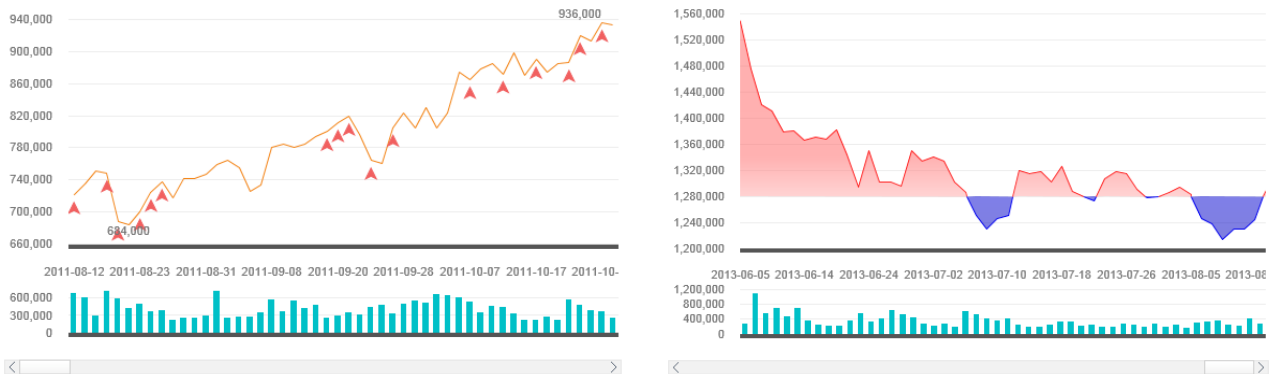
        <annotationElements>
// syncCrossRangeZoomer : mainChart의 십자선과 동기화 시킬 subChart의 CrossRangeZoomer id를 설정합니다.
        <CrossRangeZoomer id= "candleCRZ" enableZooming= "false" syncCrossRangeZoomer= "{columnCRZ}"
zoomType= "both" horizontalLabelFormatter= "{nft}"/>
        </annotationElements>
    </Candlestick2DChart>
</mainChart>
<subChart>
    <Column2DChart showDatatips= "true" height= "20%" paddingTop= "0"
paddingBottom= "0" gutterTop= "4">
        <horizontalAxis>
            <CategoryAxis id= "hAxis2" categoryField= "date"/>
        </horizontalAxis>
        <verticalAxis>
            <LinearAxis baseAtZero= "false"/>
        </ verticalAxis>
        <series>
            <Column2DSeries yField= "trdvolume"
itemRenderer= "BoxItemRenderer">
                <fill>
                    <SolidColor color= "#eca614"/>
                </fill>
            </Column2DSeries>
        </series>
        <horizontalAxisRenderers>
            <Axis2DRenderer axis= "{hAxis2}" showLabels= "false"
tickLength= "0"/>
        </horizontalAxisRenderers>
        <annotationElements>
// syncCrossRangeZoomer : subChart의 십자선과 동기화 시킬 mainChart의 CrossRangeZoomer id를 설정합니다.
            <CrossRangeZoomer id= "columnCRZ" enableZooming= "false"
syncCrossRangeZoomer= "{candleCRZ}" horizontalLabelFormatter= "{nft}" verticalLabelPlacement= "top"/>
            </annotationElements>
        </Column2DChart>
    </subChart>
    <dataSelector>
        <DualScrollBar inverted= "true" visibleItemSize= "50"/>
    </dataSelector>
</DualChart>
</rMateChart>

```

<예제 45 캔들차트 공시표현 예제 >

위 캔들차트 기본형외에 아래 그림과 같이 캔들라인차트, 캔들영역차트로 출력을 할 수 있습니다.

<Candlestick2D 부분을 <CandleLine2D, <CandleArea2D 로 변경하시고 데이터 필드명만 변경하시면 아래와 같은 차트를 출력 하실 수 있습니다. 더욱 자세한 내용은 샘플페이지와 api 를 참고하십시오



<그림 33 캔들 차트 기본형외에 캔들라인, 캔들영역 차트 >

## 6.27. 게이지 차트

### 6.27.1. Circular 게이지

Circular 게이지의 시작과 끝 노드는 <CircularGauge/> 입니다. 0~360 도 범위를 갖는 원형 게이지를 생성하게 되는데 디폴트 0 도의 위치는 시계 3 시 방향이며 시계 방향으로 각도는 진행됩니다.

Circular 게이지는 크게 배경 프레임, 바늘, 바늘 커버 3 개로 구성되어 있습니다. 이 3 개를 각각 디자인함으로써 다양한 모습의 게이지 작성이 가능합니다.

게이지는 기본적으로 디폴트 속성을 갖고 있습니다.

그러나 반드시 사용자가 정의 해야 할 속성은 아래와 같습니다.

- startAngle
- minimumAngle
- maximumAngle
- value
- minimum
- maximum

```
<rMateChart>
  <Options>
    <Caption text= "게이지 예제 - Red"/>
    <SubCaption text= "게이지, 색상, 그래데이션 등을 변경 할 수 있습니다." textAlign= "right"
paddingRight= "10" fontSize= "11" />
  </Options>
  <CurrencyFormatter id= "numFmt" precision= "0" currencySymbol= "%" alignSymbol= "right"/>
  <CircularGauge width= "300" height= "300" valueChangeFunction= "valueChangeFunc"
```

```

startAngle= "270" minimumAngle= "0" maximumAngle= "360"
minimum= "0" maximum= "100" value= "28"
interval= "10" minorInterval= "2"
formatter= "{numFmt}"
padding= "10"
labelGap= "10"
tickLabelStyleName= "tickText"
valueLabelStyleName= "valueText"
editMode= "true" liveDragging= "true"
showDataTip= "true"
tickColor= "0xCCCCCC"
minorTickColor= "0x932108"
coverRadiusRatio= "0.1"
hideTickLabel= "first"
needleThickness= "10"
pointThickness= "5"
needleLengthRatio= "0.6"
needlePointStyle= "steep"
needleBackLengthRatio= "0"
isValueTop= "false"
valueYOffset= "50">
<frameStroke>
  <Stroke color= "0xCCCCCC" weight= "10"/>
</frameStroke>
<frameFill>
  <SolidColor color= "0x932108"/>
</frameFill>
<needleFill>
  <LinearGradient angle= "90">
    <entries>
      <GradientEntry color= "0xEEEEEE" ratio= "0" alpha= "1"/>
      <GradientEntry color= "0x555555" ratio= "1" alpha= "1"/>
    </entries>
  </LinearGradient>
</needleFill>
<needleCoverFill>
  <RadialGradient>
    <entries>
      <GradientEntry color= "0xFFFFFF" ratio= "0" alpha= "1"/>
      <GradientEntry color= "0xB0B0B0" ratio= "1" alpha= "1"/>
    </entries>
  </RadialGradient>
</needleCoverFill>
</CircularGauge>

```



```
<Style>
    .valueText {
        fontSize:12;
        fontFamily:Myriad;
        textAlign:center;
        borderColor:0x999999;
        backgroundColor:0xFFFFFFFF;
        backgroundAlpha:1;
        paddingTop:2;
        borderThickness:1;
        borderAlpha:1;
        borderStyle:solid;
        color:0xFF0000;
    }
    .tickText {
        fontFamily:Myriad;
        fontSize:18;
        color:0xFFFFFFFF;
    }
</Style>
</rMateChart>
```

<예제 46 Circular 게이지 레이아웃 예제>

위 레이아웃을 실행시킨 화면은 아래와 같습니다.



<그림 34 Circular 게이지 실행 화면>

## 6.27.2. Half-Circular 게이지

Half-Circular 게이지는 <HalfCircularGauge/> 노드로 시작과 끝을 정의 합니다.  
Half-Circular 게이지에서 시작 각도(startAngle) 은 무효한 속성입니다.

minimumAngle 은 초기값이 180 도입니다. 즉, Half-Circular 게이지의 특성 상 9 시 방향이 초기점이 됩니다. 9 시방향은 180 도에 해당됩니다. 만약 180 도보다 작은 각도를 설정시 Half 게이지는 180 도로 강제 설정하게 됩니다. 또한 maximumAngle 은 360 도보다 커질 수 없습니다.

따라서 Half-Circular 게이지가 표현할 수 있는 범위는 9 시 방향에서 3 시 방향 사이입니다.

그 외 모든 속성 및 스타일은 Circular 게이지와 같습니다

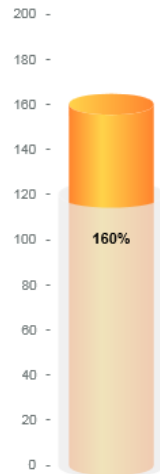
### 6.27.3. Cylinder 게이지

실린더 게이지는 수직형태와 수평 형태가 존재합니다. 수직 형태의 실린더 게이지를 생성하기 위해서는 <HCylinderGauge/> 노드로 시작과 끝을 결정하고, 수평 형태의 실린더 게이지를 생성하기 위해서는 <HCylinderGauge/> 노드로 시작과 끝을 정의합니다.

다음은 수직 실린더 게이지의 샘플 레이아웃을 작성한 모습과 실행화면입니다.

```
<rMateChart backgroundColor= "0xFFFFF" cornerRadius= "12" borderStyle= "solid">
  <Options>
    <Caption text= "게이지"/>
  </Options>
  <VCylinderGauge id= "cy1" width= "30%" height= "180"
    minimum= "0" maximum= "160"
    labels= "[0, 20, 40, 60, 80, 100, 120, 140, 160]"
    value= "50"
    targetMark= "150"
    snapInterval= "1"
    tickThickness= "2"
    tickLength= "2"
    tickInterval= "20"
    liveDragging= "true"
    tickColor= "0x000000"
    valueLabelXOffset= "10"
    valueLabelStyleName= "valueLabel"
  />
  <Style>
    .valuelabel{
      fontWeight:bold;
    }
  </Style>
</rMateChart>
```

<예제 47 VCylinderGauge 레이아웃 예제>



<그림 35 VCylinderGauge 실행화면>

#### 6.27.4. Linear 게이지

리니어 게이지는 세로 방향의 막대 모양과 가로 방향의 막대 모양으로 나뉩니다. 세로 방향은 <VLinearGauge/> 로 시작과 끝을 정의하고, 가로 방향은 <HLinearGauge/> 노드로 정의합니다.

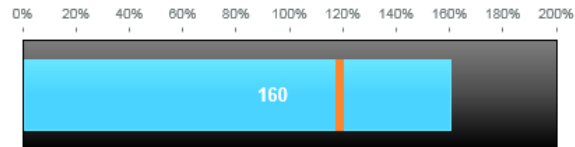
리니어 게이지는 크게 값(value) 을 표현하는 막대와 목표치를 나타내는 목표선(target line)으로 구성되어 있습니다.

다음은 수평 리니어 게이지의 샘플 레이아웃과 실행 화면입니다.

```
<rMateChart backgroundColor= "0xFFFFFF" cornerRadius= "12" borderStyle= "solid">
  <Options>
    <Caption text= "게이지"/>
  </Options>
  <HLinearGauge id= "hl" width= "300" height= "100"
    minimum= "0" maximum= "100"
    liveDragging= "true"
    labels= "[0, 25, 50, 75, 100]"
    value= "50"
    targetMark= "90"
    tickThickness= "1"
    tickLength= "4"
    tickInterval= "5"
    snapInterval= "1"
    linearColumnWidth= ".6"
    tickColor= "#000000"
    valueLabelFontWeight= "bold"
  />
  <Style>
    .valuelabel {
```

```
fontWeight:bold;
}
</Style>
</rMateChart>
```

<예제 48 Horizontal Linear Gauge 레이아웃 예제>



<그림 36 Horizontal Linear Gauge 실행 화면>

## 6.28. 슬라이드 차트 기능 사용하기

슬라이드 차트 기능은 여러 페이지의 차트를 생성하여 프리젠테이션 형식과 같이 차트를 보여주는 기능입니다.

슬라이드 차트에 삽입가능한 차트에는 제한이 없습니다. 알메이트 차트에서 작성 가능한 차트는 모두 슬라이드 차트로 표현 가능합니다.

슬라이드 차트는 반드시 다음 규칙을 따라야 합니다.

1. 레이아웃은 setSlideLayoutSet() 함수로 삽입해야 합니다.
2. 데이터는 setSlideDataSet() 함수로 삽입해야 합니다.

위 규칙대로 레이아웃과 데이터를 삽입하면 슬라이드 차트가 생성됩니다.

예를 들어 총 5 페이지의 차트 슬라이드를 구성한다면 데이터와 레이아웃은 각각 5 개의 쌍으로 이루어져야 합니다. 따라서 레이아웃과 데이터는 배열 형태가 되며 배열 요소는 각각 스트링 형태의 레이아웃과 배열 형태의 데이터가 됩니다.

다음은 그에 대한 예제입니다.

```
<!-- 사용자 정의 설정 시작 -->
<script language="JavaScript" type="text/javascript">

// -----차트 chartVars 설정 시작-----
// rMate 차트 생성 준비가 완료된 상태 시 호출할 함수를 지정합니다.
```

```

var chartVars = "rMateOnLoadCallFunction=rMateChartOnLoad";
// -----차트 chartVars 설정 끝-----

// rMate 차트 준비가 완료된 경우 이 함수가 호출됩니다.
function rMateChartOnLoad()
{
    var layout1 = getCartesianLayout("Column2D","칼럼 차트로 표현",["Profit"]);
    var layout2 = getCartesianLayout("Line2D","라인 차트로 표현",["Profit"]);
    var layout3 = getCartesianLayout("Column3D","칼럼 3D 멀티 데이터 표현",["Profit","Cost"]);

    ● 매뉴얼은 같은 레이아웃과 같은 데이터로 슬라이드 차트를 표현하고 있습니다.

    // 슬라이드에 넣을 데이터와 레이아웃들.
    layoutSet = [layout1, layout2, layout3, layout1, layout2];
    dataSet = [chartData, chartData, chartData, chartData, chartData];

    // 슬라이드에서 표현할 레이아웃들 삽입.
    document.getElementById("chart").setSlideLayoutSet(layoutSet);

    // 슬라이드에서 표현할 데이터들 삽입.
    document.getElementById("chart").setSlideDataSet(dataSet);
}

//레이아웃을 반환하는 함수를 작성한 것입니다.
//해당 레이아웃은 스트링 형태의 조합이므로 개발자에 의해 다음과 같은 함수 작성이 가능합니다.
//파라미터 설명
//type : 차트 type
//title : 차트 Caption
//dataField : 시리즈가 표현할 실데이터의 필드명 배열
function getCartesianLayout(type, title, dataField)
{
    var layout="<rMateChart borderStyle='solid'>"
        + "<Options><Caption text='"+ title +'"/></Options>"
        + "<" + type + "Chart showDataTips='true'>"
        + "<series>";

    for(var i=0; i<dataField.length; ++i) {
        layout += "<" + type + "Series yField='" + dataField[i] + "' displayName='" + dataField[i]
+ "'/>"
    }

    layout += "</series>"

```

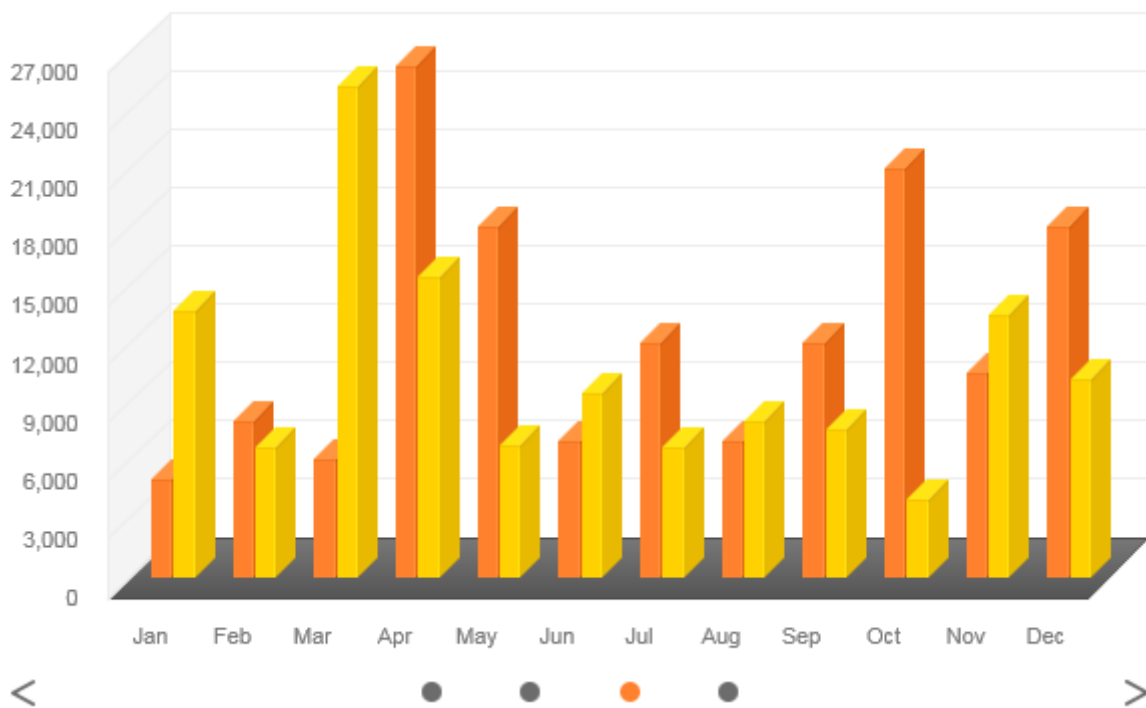
```

+ "<horizontalAxis>"
+       "<CategoryAxis categoryField='Month'/>"
+ "</horizontalAxis>"
+ "</" + type + "Chart>"
+ "</rMateChart>";

return layout;
}
//배열 데이터로 정의
var chartData = [{"Month": "Jan", "Profit": 13000},
  {"Month": "Feb", "Profit": 12000},
  {"Month": "Mar", "Profit": 15000},
  {"Month": "Apr", "Profit": 22200},
  {"Month": "May", "Profit": 18000},
  {"Month": "Jun", "Profit": 15000},
  {"Month": "Jul", "Profit": 22000},
  {"Month": "Aug", "Profit": 14000},
  {"Month": "Sep", "Profit": 26000},
  {"Month": "Oct", "Profit": 22000},
  {"Month": "Nov", "Profit": 28000},
  {"Month": "Dec", "Profit": 34000}];
<!-- 사용자 정의 설정 끝 -->

```

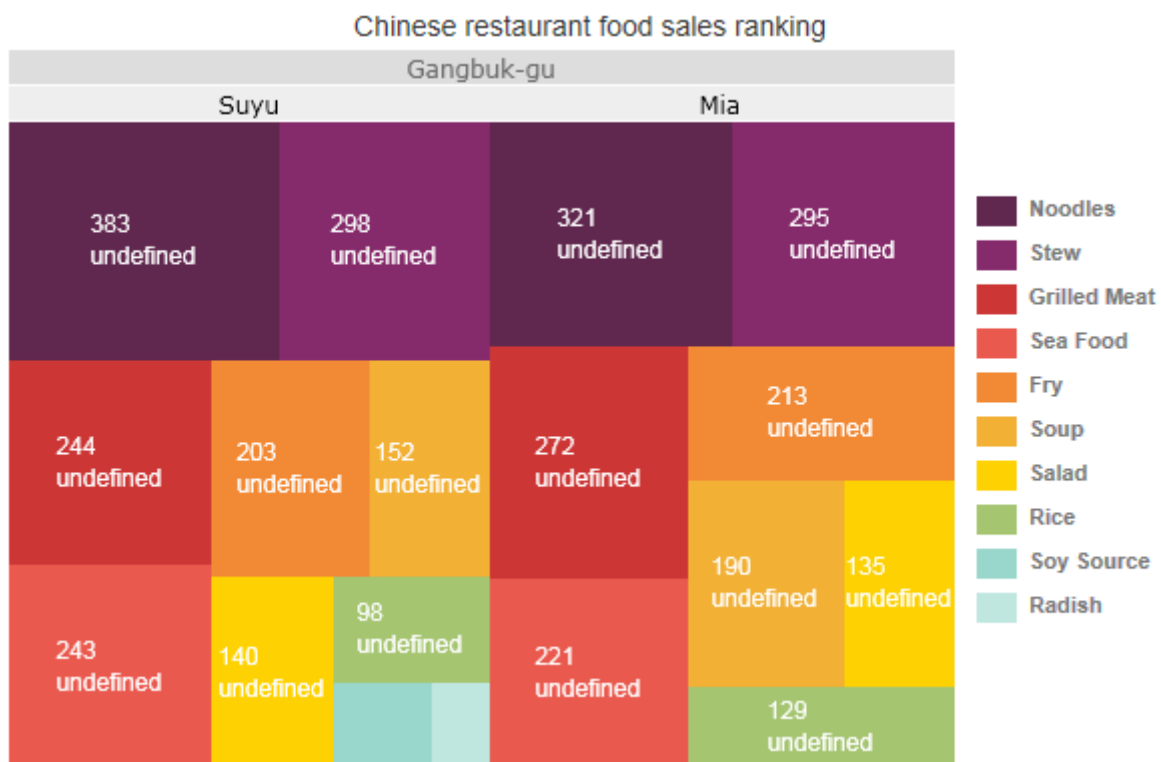
<예제 49 슬라이드 차트 작성 예제>



<그림 37 슬라이드 차트 출력 모습>

## 6.29. 트리맵 차트

트리 맵 차트는 계층적 데이터를 설정하여 자식 부모 관계의 데이터를 사각형의 모양으로 출력하는 차트 입니다. 자식에 해당하는 데이터는 부모의 사각형안에 부모보다 작은 크기로 출력이 되어집니다. 트리 구조로 박스 크기와 색상으로 해당 데이터들의 연관성을 출력합니다. 동일한 카테고리의 데이터가 지역에 따라 혹은 상황에 따라 달라지거나 각 지역별로 투표율에 대한 내용 등 이러한 데이터들을 한눈에 보기 쉽게 출력해주는 차트입니다.



< 그림 38 트리맵 차트 >

트리 맵 차트의 데이터는 아래와 같은 구조를 가지고 있어야 합니다.

```
// 차트 데이터
var chartData = [
  {
    "name": "master",
    "items": [
      {
        "name": "ABCD",
        "items": [
          {"Month": "Jan", "Vancouver": 80},

```

```

    {"Month":"Feb", "Vancouver":90},
    {"Month":"Mar", "Vancouver":383},
    {"Month":"Apr", "Vancouver":81},
    {"Month":"Test text width", "Vancouver":87},
    {"Month":"Jun", "Vancouver":89},
    {"Month":"Jul", "Vancouver":86},
    {"Month":"Aug", "Vancouver":192},
    {"Month":"Sep", "Vancouver":88},
    {"Month":"Oct", "Vancouver":84},
    {"Month":"Nov", "Vancouver":87},
    {"Month":"Dec", "Vancouver":90}
  ]
}, {
  "name" : "CDEF",
  "items" : [
    {"Month":"Jun", "Vancouver":189},
    {"Month":"Jul", "Vancouver":186},
    {"Month":"Aug", "Vancouver":192},
    {"Month":"Sep", "Vancouver":188},
    {"Month":"Oct", "Vancouver":184},
    {"Month":"Nov", "Vancouver":187},
    {"Month":"Dec", "Vancouver":190}
  ]
}
]
}
];

```

위 데이터에서 최상위 부모는 "master"라고 정의된 데이터이며 해당 "master" 데이터 자식 데이터로 "ABCD", "CDEF" 데이터가 존재하며 각 "ABCD", "CDEF" 에게도 자식데이터들이 존재하게 됩니다. 트리 맵 차트는 위 부모 자식간의 데이터를 가지고 < 그림 38 트리맵 차트 > 와 같이 각 데이터 별로 사각형 형태로 출력이 되어집니다.

### 6.30. 워드 클라우드

워드 클라우드 차트는 각 단어들의 중요도나 인기도의 데이터를 시각적으로 표현해주게 됩니다. 각 단어들은 중요도나 인기의 수치에 따라 글자 색상이 변하거나 크기가 변경되며 출력됩니다. 워드 클라우드 차트의 데이터는 두 가지 형태로 나눌 수 있습니다. 하나는 기존 형태로 어떤 데이터를 출력 할지 설정하는 것입니다. ( ex / yField = "field 명" xField = "field 명" ) 또 하나의 방법은 배열안에



출력 할 단어 데이터들만을 가지고 있는 형태입니다. ( ex / data = ["가", "나", "다", "라", "가", "다" ... ] )  
이 데이터 형태를 설정할 경우 차트는 해당 데이터들을 처음부터 끝까지 검색하여 같은 단어가 몇 개나 존재하는지 계산하여 데이터를 새로 구성하게 됩니다. 이 시간이 존재함으로 데이터의 양이 많아질 경우 첫 번째 방법과 출력 속도차이가 달라질 수 있습니다.



< 그림 39 워드 클라우드 >

워드 클라우드 데이터 설정 첫 번째 방법

```
var chartData = ["Lorem", "ipsum", "dolor", "sit", "amet", "consectetuer",  
                "s adipscing", "elit", "sed", "ipsum" .....];
```

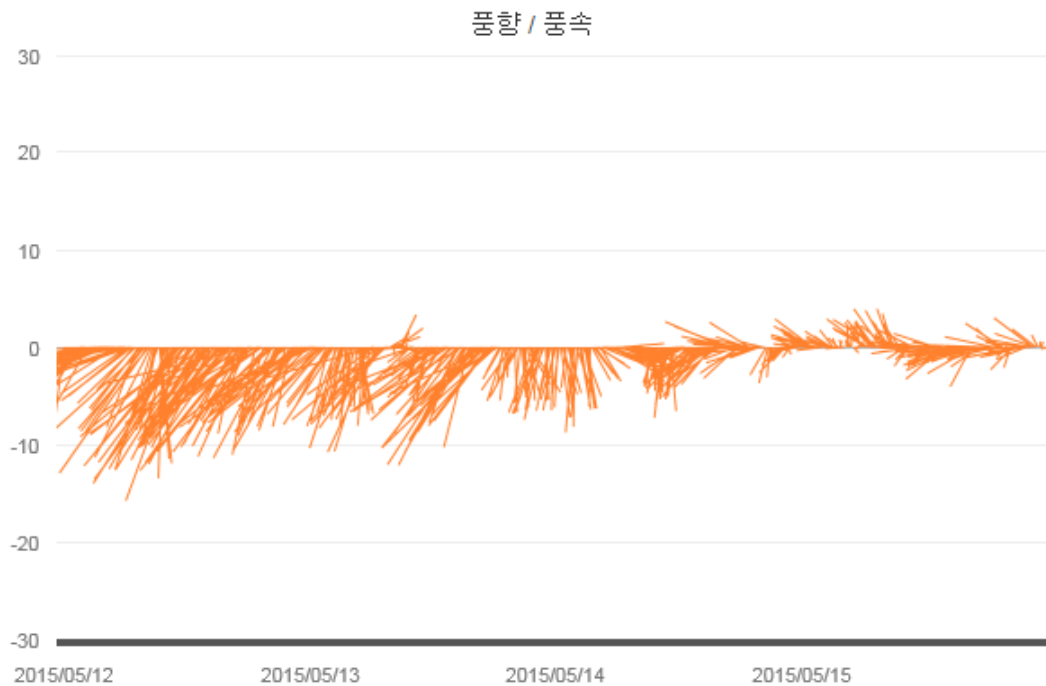
워드 클라우드 데이터 설정 두 번째 방법

```
var chartData = [{"text": "강남구", "value": 12}, {"text": "강동구", "value": 8}, {"text": "광진구", "value": 9}, {"text": "강서구", "value": 3}, {"text": "관악구", "value": 3}, {"text": "강북구", "value": 1}, ...];
```

위 설명과 같이 워드 클라우드 데이터 설정은 두 가지 방법이 존재합니다.  
가져온 데이터는 가장 큰 값을 시작으로 작은 값까지 순서대로 출력되어집니다.

브라우저의 폰트, 클라이언트의 폰트 환경에 따라 설치가 안되어있거나 지원하지 않는 폰트라면 올바르게 출력이 되지 않거나 글자가 깨져서 나올 수 있습니다.

### 6.31. 벡터 차트



< 그림 40 벡터 차트 >

벡터 차트는 각도 데이터와 크기 데이터를 가지고 각 데이터를 선으로 표현하는 차트입니다.

그림 40 을 살펴보면 x 축으로는 각 날짜를 표현하며 해당 날짜에 대하여 어느 방향으로 얼마만큼의 세기로 바람이 부는지 표현하고 있습니다.

풍향 데이터가 아닌 바다의 유속 데이터 혹은 그 외의 데이터에 관하여 각도와 크기를 가지고 있다면 벡터 차트로 표현이 가능합니다.

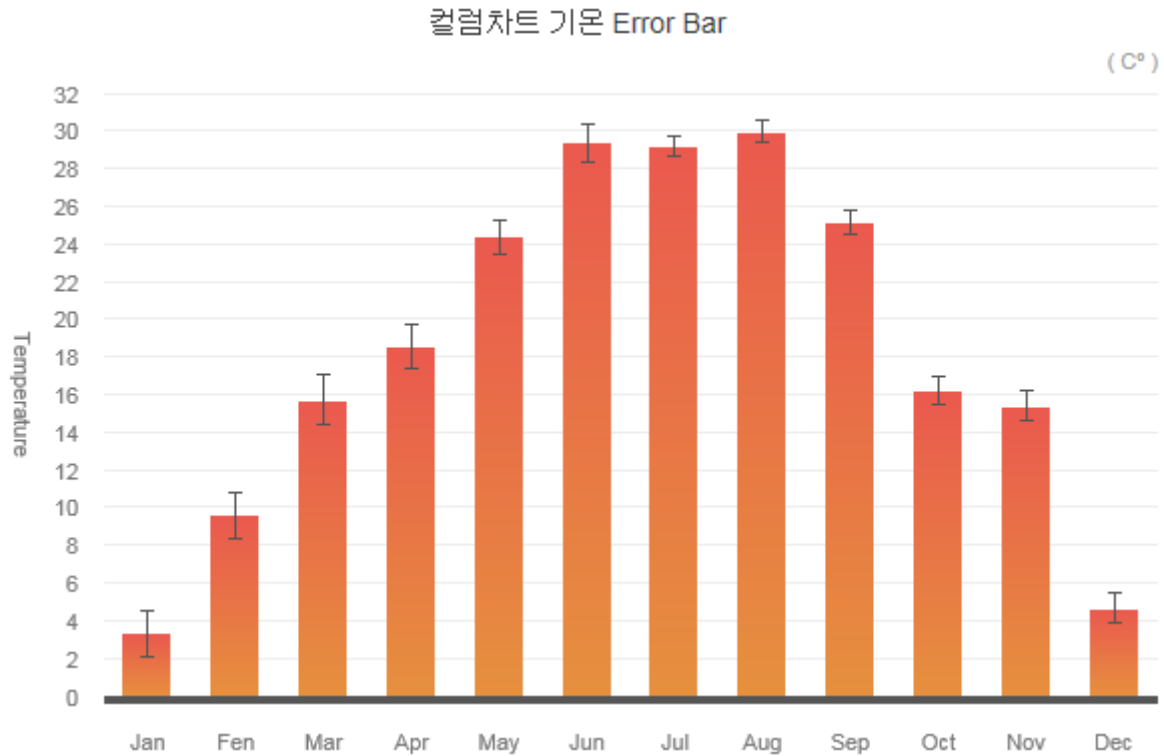
### 6.32. 에러 바 차트

에러 바 차트는 다량의 데이터들을 가지고 오차 혹은 확실하지 않은 측정 값의 범위를 출력해주도록 합니다. 차트에 설정된 데이터들의 평균을 구하고 표준편차를 구한 후 표준오차를 출력해내어 각 데이터들의 평균 값에서 오차 범위를 나타냅니다.

에러 바 차트의 경우의 데이터 구조는 아래와 같습니다.

```
var chartData = [{"Month": "Jan", "temperature": [6.8, 8.9, 14.39, ...]},
                  {"Month": "Feb", "temperature": [9.39, 5.1, 7.2, ...]},
                  {"Month": "Mar", "temperature": [1.0, 6.1, 2.15, 6.1, ...]},
                  {"Month": "Apr", "temperature": [19.16, 7.4, 14.39, 17.28, ...]},
                  ....];
```

위 데이터와 같이 ErrorBar 로 출력할 데이터는 temperature 필드에 배열 형태의 데이터를 설정합니다.  
차트 내에서는 해당 필드의 데이터 집합들을 가져와 출력하게 됩니다.



< 그림 41 에러 바 차트 >

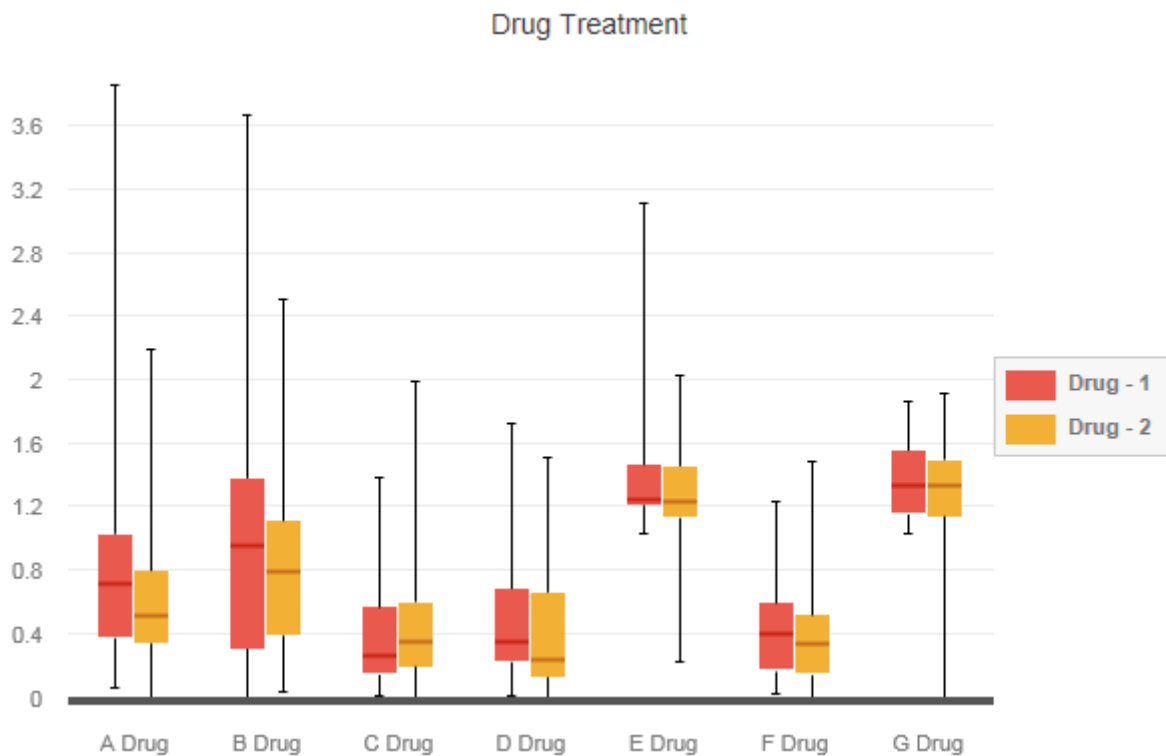
```
....
+'<Column2DChart showDataTips="true" dataTipFormatter="{cft}">'
    ....
    +'<series>'
        +'<Column2DSeries yField="temperature" showErrorBar="true" errorBarDirection="both"/>'
    +'</series>'
+'</Column2DChart>'
....
```

위 레이아웃에서 yField 로 차트에 설정한 데이터들 중 temperature 필드를 설정하도록 합니다.  
설정한 showErrorBar="true" 속성으로 오차 범위에 대한 선을 출력하도록 합니다.  
errorBarDirection 으로 오차 범위에 대한 선의 방향을 설정합니다.  
선의 방향은 아래와 같이 3 가지를 설정 할 수 있습니다.

- both – 위, 아래로 선을 그리도록 합니다.
- up – 위에만 선을 그리도록 합니다.
- down – 아래만 선을 그리도록 합니다.

### 6.33. 박스 플롯 차트

박스 플롯 차트는 다량의 데이터들을 최소 값, 최대 값, 2 사분위, 3 사분위로 나타냅니다.  
차트에 설정된 데이터들을 최소 값에서부터 최대 값까지 25% 씩 나누어 표현합니다.  
최소 값에서부터 2 사분위, 3 사분위에서 최대 값까지를 선으로 표현하며 2 사분위에서 3 사분위까지를 Box 형태로 출력합니다.



< 그림 42 박스 플롯 차트 >

아래는 위 그림의 샘플 데이터 입니다.

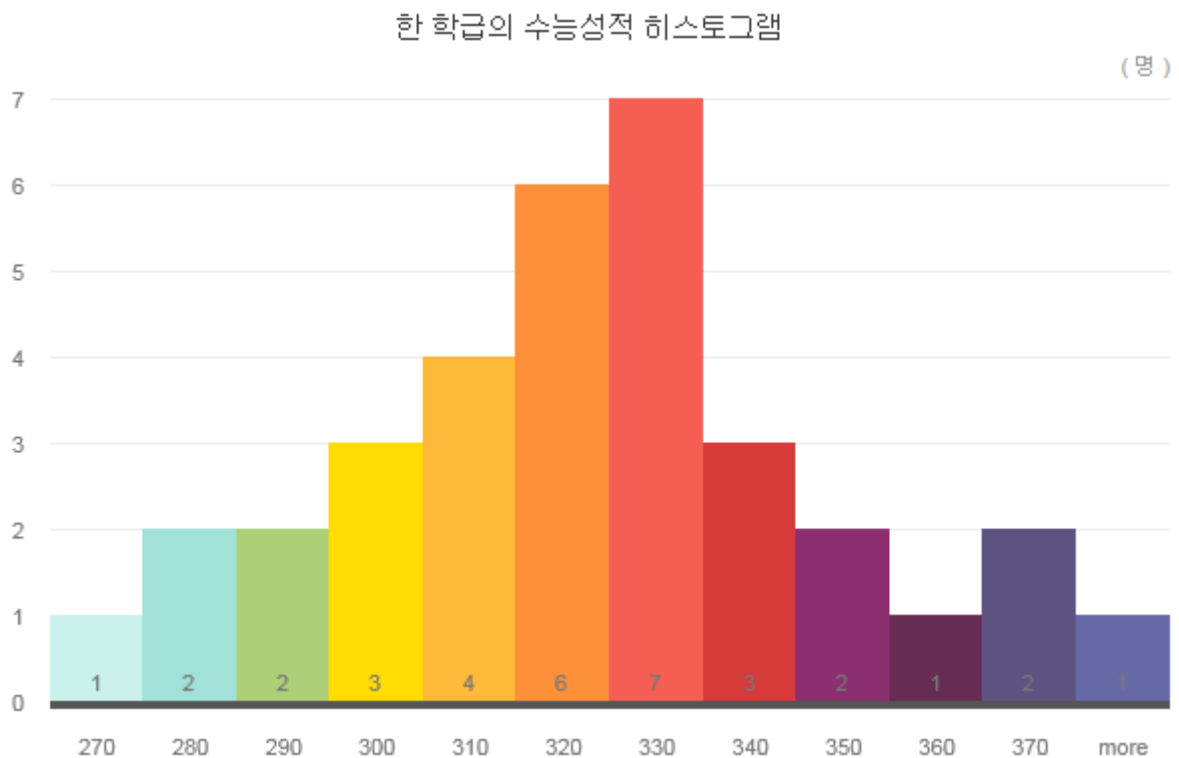
```
// 차트 데이터
var chartData = [{"category": "A Drug", "Drug1": [1.05198, 1.00211, 0.63396, 0.70381, 0.23767, 0.70693, 0.12827, 0.82422, 1.0682, 0.23055, 0.78603, 0.54711, 0.786, 0.48393, 0.46068, 1.06331, 0.48065, 1.21493, 1.04961, 0.11145, 0.54881, 0.7029, 0.90385, 0.77662, 1.1578, 0.50103, 0.36086, 0.9577, 0.32817, 0.26668, 0.28617, 0.28907, 1.16461, 0.05326, 1.05258, 0.10204, 0.76055, 0.95481, 1.96947, 1.03068, 3.8701], "Drug2": [0.97755, 1.19132, 0.53781, 0.8197, 0.62809, 0.89745, 0.37964, 0.85656, 0.2872, 0.77542, 0.45259, 2.2, 0.46686, 0.45078, 0.2503, 0, 0.40992, 0.64416, 0.97218, 0.80421, 0.48289, 0.42668, 1.0634, 0.27623, 0.11335, 0.61329, 0.50746, 0.63754, 0.50025, 0.45166, 0.55219, 0.1492, 0.92092, 0.51731, 0.05138, 0.33631, 0.72907, 0.3213, 0.05734, 0.14734, 1.1033]},
...
];
```

## 6.34. 히스토그램 차트

히스토그램 차트는 도수분포표를 그래프로 나타내는 차트입니다.

차트에 설정된 데이터들을 가지고 사용자가 설정한 구간을 기준으로 데이터들에 대한 그룹을 정하고 해당 그룹에 대한 데이터를 출력하는 차트입니다.

그림 43의 샘플은 특정 학급에 대하여 수능성적에 대한 히스토그램 차트이며 각 점수에 대한 구간을 설정합니다. 설정한 구간에 해당되는 데이터들의 개수를 추출하여 추출된 데이터를 가지고 차트로 출력합니다.



< 그림 43 히스토그램 차트 >

// 스트링 형식으로 레이아웃 정의.

**var** layoutStr =

'<rMateChart backgroundColor="0xFFFFF" borderStyle="none">'

  +'<Histogram2DChart binRange="[270,280,290,300,310,320,330,340,350,360,370]" .... >'

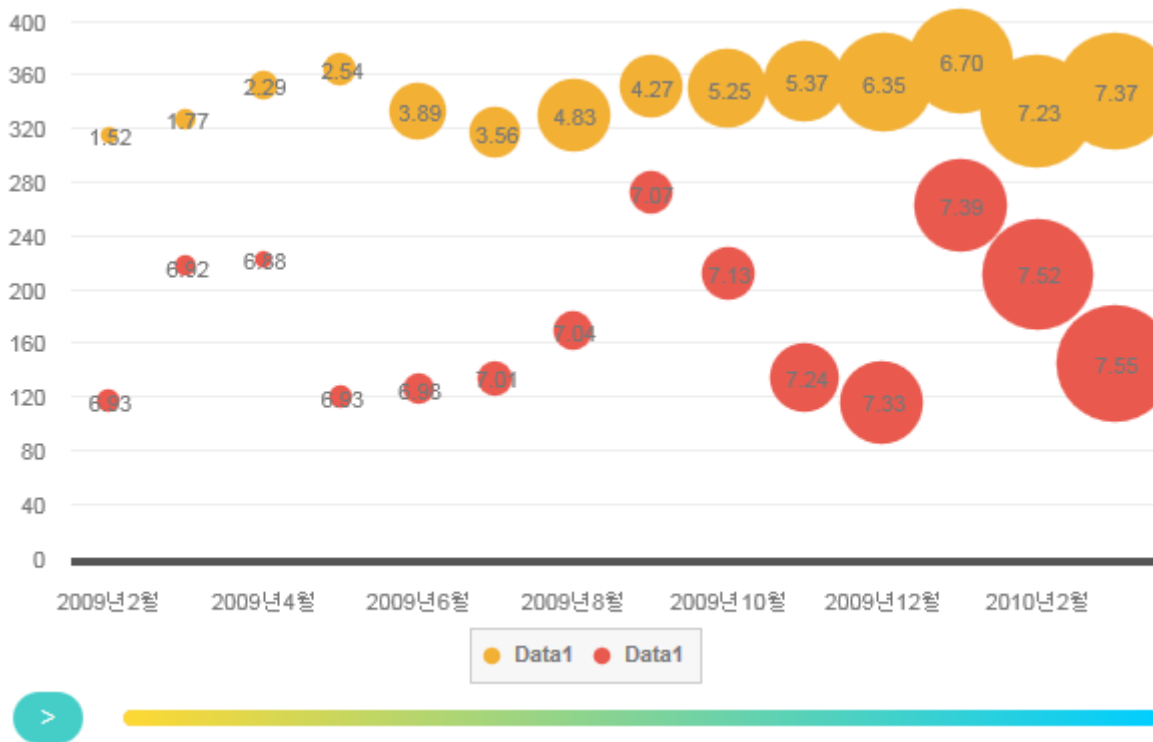
  ....

위 히스토그램 차트 레이아웃에서 <Histogram2DChart의 노드로 binRange를 설정하도록 합니다.

binRange로 270 점에서부터 10 점마다 총 370 까지의 구간을 설정하였습니다. 각 구간에 해당하는 데이터들의 개수를 세어 출력하며 설정된 구간을 넘어가는 데이터에 관하여서는 "more"와 같이 출력하도록 합니다.

### 6.35. 모션 차트

모션차트는 데이터들이 변화되는 과정을 차트 아래부분의 슬라이더로 변경하며 확인할 수 있는 차트 입니다. 재생 버튼을 실행하면 현재 출력되어있는 데이터부터 마지막의 데이터까지 순차적으로 진행되어 출력되고 슬라이더 부분을 중간 중간 클릭하여 사용자가 원하는 부분에 대한 데이터를 출력하게 합니다.



< 그림 44 모션 차트 >

모션차트는 버블, 컬럼, 라인 세가지의 차트만을 지원하고 있으며 각 차트의 특성에 따라 레이아웃과 데이터의 설정 부분이 달라지게 됩니다.

버블, 컬럼
<pre>// 차트 데이터 var chartData = [ {"Month":"2009년2월","Data1":314.4017076,"Data2":1.523997507,"Data3":316,"Data4":6.938824349}, {"Month":"2009년7월","Data1":316.29,"Data2":3.565551631,"Data3":319,"Data4":7.015549976}, ... ];</pre>
라인
<pre>// 차트 데이터 var chartData = [ [ {"Month":"Jan", "Data1":312},</pre>

```

        {"Month":"Feb", "Data1":300},
        ....
    ],[
        {"Month":"Jan", "Data1":612},
        {"Month":"Feb", "Data1":600},
        ....
    ],[
        ....
    ]
];

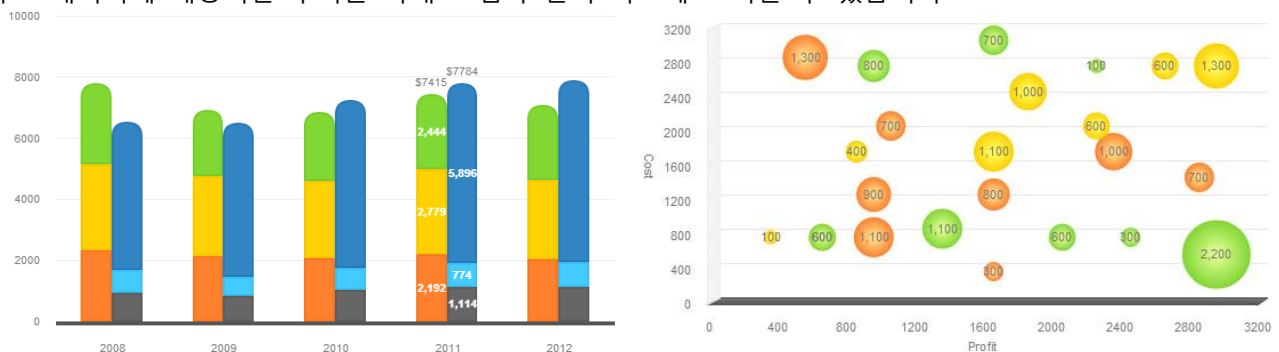
```

버블과 컬럼은 차트에 설정하던 기존 방법과 같은식으로 데이터를 설정하는 반면  
라인은 버블과 컬럼에 넣는 데이터 형식을 묶은 배열데이터를 차트에 설정하게 됩니다.

## 7. 고급 사용자를 위한 rMate Chart 레이아웃 설정.

### 7.1. 차트에 수치필드 표시하기

차트 데이터에 해당되는 수치를 아래 그림과 같이 차트에 표시할 수 있습니다.



이는 차트의 시리즈 속성으로 각각의 시리즈에 대한 속성은 labelPosition 입니다. 시리즈에 따라 labelPosition 의 유효값이 다릅니다. 아래 표는 labelPosition 의 유효값에 대한 설명입니다.

대표 시리즈 명	유효값	설 명
ColumnSeries(2D, 3D), BarSeries(2D, 3D), ImageSeries	inside, outside, both, none	수치필드를 안쪽에 표시할지 바깥쪽에 표시할지 또는 양쪽에 표시할지를 지정합니다.
Line2DSeries, Area2DSeries	up, down, both, none	수치필드를 데이터 포인트를 기준으로 위에 표시할지 아래에 표시할지 또는 양쪽에 표시할지를 지정합니다.

Bubble3DSeries	inside,none	버블 안쪽에 수치필드를 표시할지 여부를 나타냅니다.
PieSeries(2D,3D)	inside, outside, callout, insideWithCallout	파이 안쪽, 바깥쪽 또는 선을 근거 바깥쪽에 표시할지를 지정합니다. insideWithCallout 을 설정한 경우 파이의 크기가 충분히 커 라벨이 표시될 공간이 있는 경우 inside 에 공간이 부족한 경우 callout 형태로 출력합니다.

<표 16 시리즈 별 labelPosition 유효 값 및 설명>

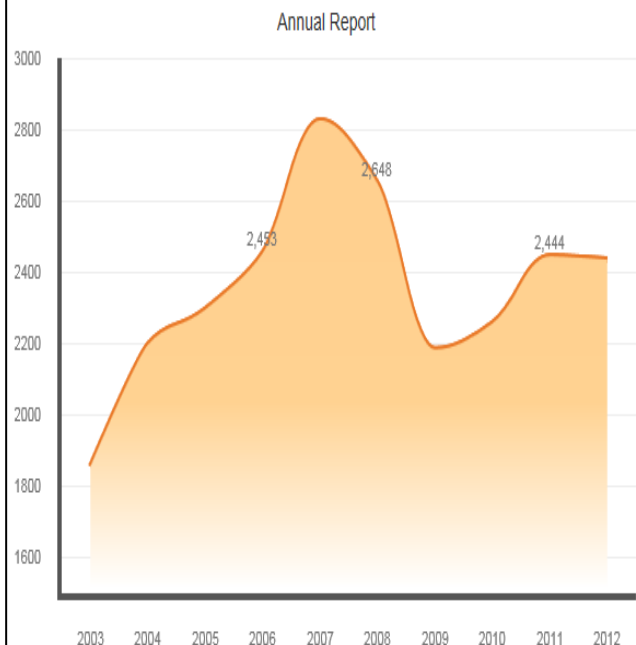
## 7.2. 차트 시리즈 아이템 각각에 컬러 지정하기.

차트 시리즈 아이템의 컬러는 선 색깔과 채우기 색으로 나뉩니다.

```

<series>
  <Area2DSeries yField="Profits"
    form="curve"
    displayName="Profits">
    <areaFill>
      <SolidColor
        color="0x00FF99"/>
      </areaFill>
      <areaStroke>
        <Stroke color="0xFF0000"
          weight="3"/>
        </areaStroke>
      </Area2DSeries>
    </series>
  
```

- 영역차트는 areaFill, areaStroke 를 사용한  
다는 점에 주의.



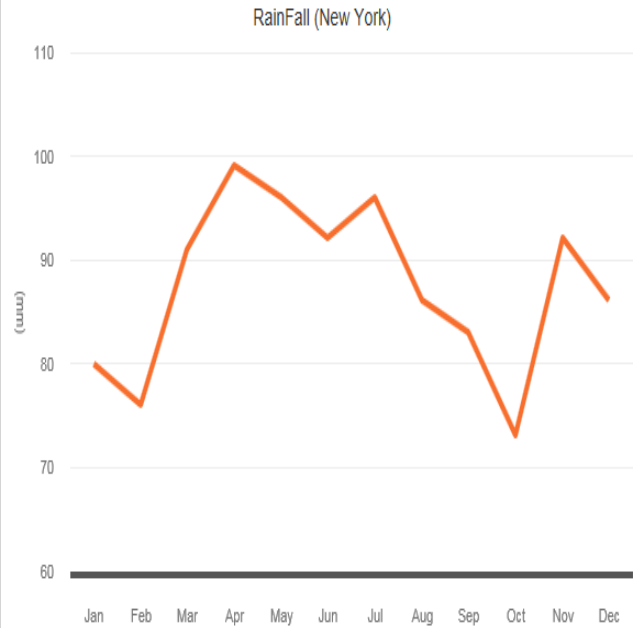
<예제 50 영역차트에 areaFill 과 stroke 를 한 경우>



```
<Line2DChart>
  <horizontalAxis>
    <CategoryAxis
      categoryField="Month"/>
    </horizontalAxis>
    <series>
      <Line2DSeries yField="Profit"
        form="curve" displayName="Profit">
        <lineStroke>
          <Stroke color="0xF0FF00"
            weight="4"/>
        </lineStroke>
      </Line2DSeries>
    </series>
  </Line2DChart>
```

// color는 선색깔을 나타냅니다.  
// weight 은 선의 두께를 나타냅니다.

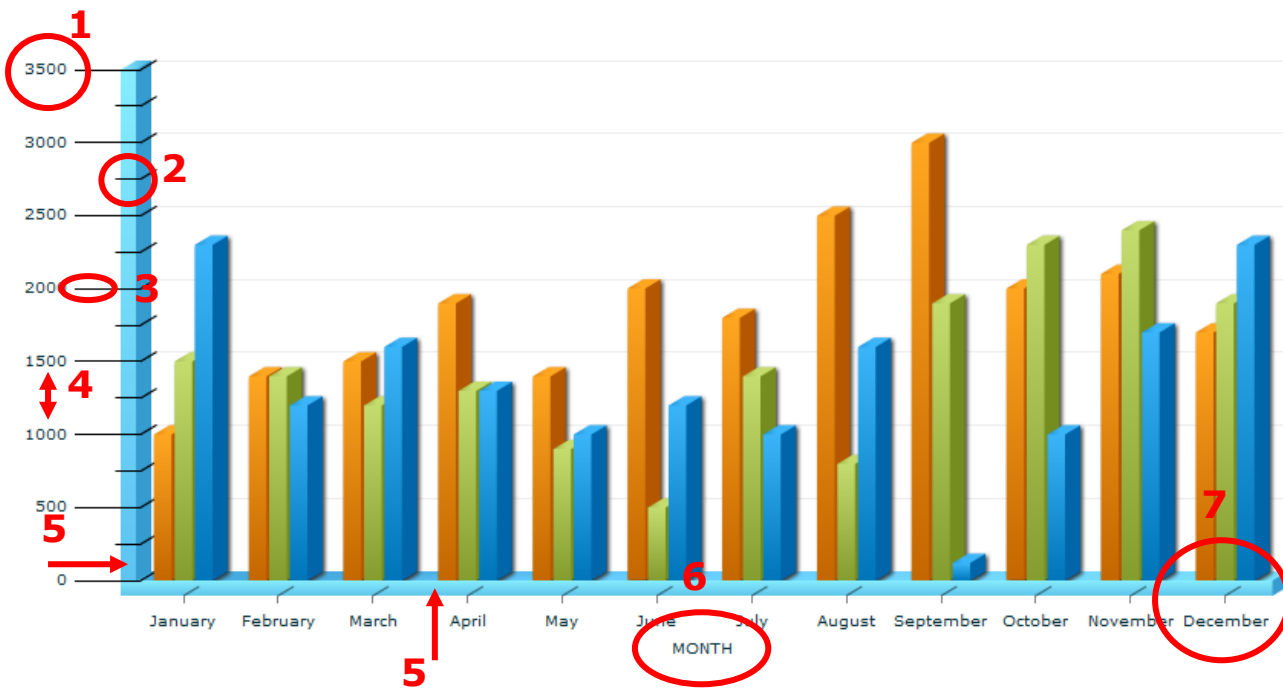
- 라인차트는 lineStroke 를 사용한다는 점에 주의.



<예제 51 라인차트의 선(stroke)를 변경한 경우>

### 7.3. 축(Axis)에 대한 스타일 적용하기.

#### 7.3.1. 축 스타일 설명 및 예제.



<그림 45 축 꾸미기를 표현한 그림>

위는 수평축과 수직축을 사용자 정의한 화면입니다. 위에 표시된 번호의 설명은 아래와 같습니다.

1. 수직축의 Maximum 값을 나타냅니다. 이는 오직 LinearAxis 에서만 유효합니다.
2. minorTick 을 나타냅니다. minorTick 의 길이와 Stroke 스타일을 정의할 수 있습니다.
3. tick 을 나타냅니다. tick 길이와 tick 의 위치, Stroke 스타일을 정의할 수 있습니다.
4. 축에 표시된 숫자들의 간격을 의미합니다. 현재는 500 입니다. LinearAxis 에서만 유효합니다.
5. 수직축과 수평축의 전체적인 Stroke 스타일입니다. 현재 하늘색의 축을 정의하였습니다.
6. 축의 대표 제목을 붙일 수 있습니다.
7. tick 과 라벨과의 관계입니다. 라벨과 라벨 사이에 tick 존재하는 경우와 라벨 위에 tick 존재하는 경우로 나뉩니다. 현재는 라벨 위에 tick 이 존재하는 경우입니다.

위에서 지시하고 설명한 부분을 레이아웃에서 찾아보면 다음과 같습니다.

```

<Column3DChart>
  6 <horizontalAxis>
    <CategoryAxis categoryField="Month" ticksBetweenLabels="false" title="MONTH"
displayName="Month"/>
  </horizontalAxis>
  <verticalAxis>
    <LinearAxis interval="500" baseAtZero="true" 1 maximum="3500"/>
  </verticalAxis>
  <horizontalAxisRenderers> <!-- 수평축-->
    3 <Axis3DRenderer visible="true"
      tickLength="5" 2
      minorTickLength="5"
      tickPlacement="outside"
      placement="bottom" <!--수평축의 위치 나타냄 top 또는 bottom
      canDropLabels="false"
      showLabels="true"
      labelAlign="center">
      5
    <axisStroke>
      <Stroke weight="10" color="0x66CCFF" caps="none"/>
    </axisStroke>
    3 <tickStroke>
      <Stroke weight="1" color="0x000000" alpha="0.5" />
    </tickStroke> 2
    <minorTickStroke>
      <Stroke weight="1" color="0x000000" caps="square"/>
    </minorTickStroke>
  </horizontalAxisRenderers>
</Column3DChart>
  
```

```

    </Axis3DRenderer>
  </horizontalAxisRenderers>

  <verticalAxisRenderers> <!--수직축 꾸미기는 수평축과 같습니다. 수평축을 참고하여
                           주십시오.-->
    <Axis3DRenderer visible="true"
      tickLength="30"
      minorTickLength="3"
      tickPlacement="left"
      placement="left" <!--수직축의 위치, left 또는 right-->
      canDropLabels="false"
      showLabels="true"
      labelAlign="center">
      <axisStroke>
        <Stroke weight="10" color="0x66CCFF" caps="none"/>
      </axisStroke>
      <tickStroke>
        <Stroke weight="1" color="0x000000" />
      </tickStroke>
      <minorTickStroke>
        <Stroke weight="1" color="0x000000" caps="square"/>
      </minorTickStroke>
    </Axis3DRenderer>
  </verticalAxisRenderers>

  .....

</Column3DChart>

```

<예제 52 축 꾸미기 설정한 예제>

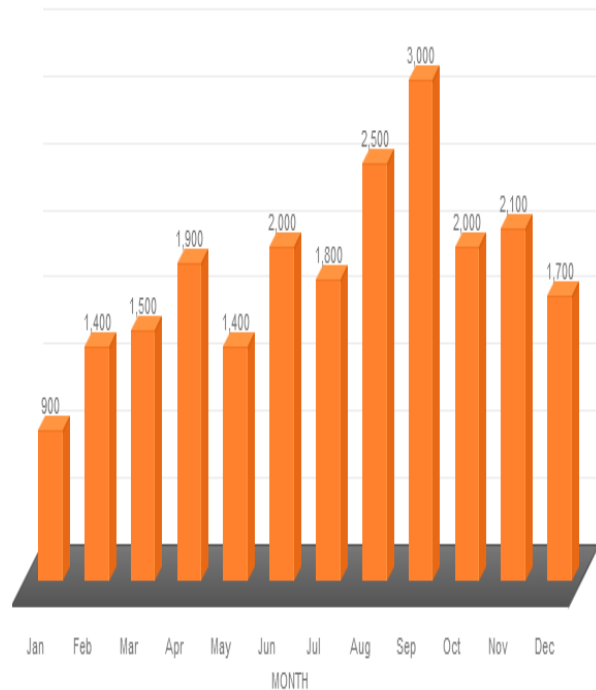
### 7.3.2. 축의 Visible 속성 사용하여 축의 유, 무 나타내기.

```
<Column3DChart>
    .....

    <horizontalAxisRenderers>
        <Axis3DRenderer
            placement="bottom"
            showLabels="true"
            labelAlign="center">
        </Axis3DRenderer>
    </horizontalAxisRenderers>

    <verticalAxisRenderers>
        <Axis3DRenderer
            visible="false">
        </Axis3DRenderer>
    </verticalAxisRenderers>

    .....
</Column3DChart>
```



<예제 53 수직축 보이지 않게 설정한 예제>

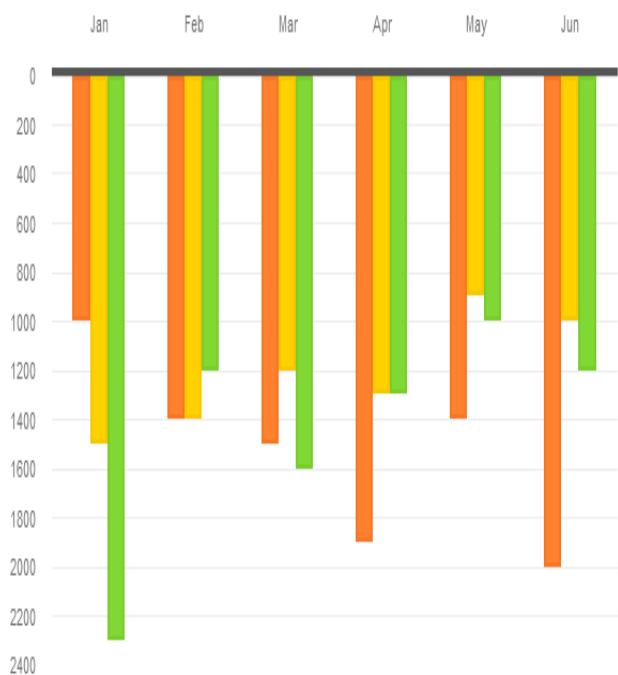
### 7.3.3. 축의 위치 바꾸기.

```
<rMateChart>
    <Column2DChart>
        .....

        <horizontalAxisRenderers>
            <Axis2DRenderer
                placement="top"
                showLabels="true">
            </Axis2DRenderer>
        </horizontalAxisRenderers>

        <verticalAxisRenderers>
            <Axis2DRenderer
                placement="left"
                showLabels="true">
            </Axis2DRenderer>
        </verticalAxisRenderers>

        .....
    </Column2DChart>
</rMateChart>
```



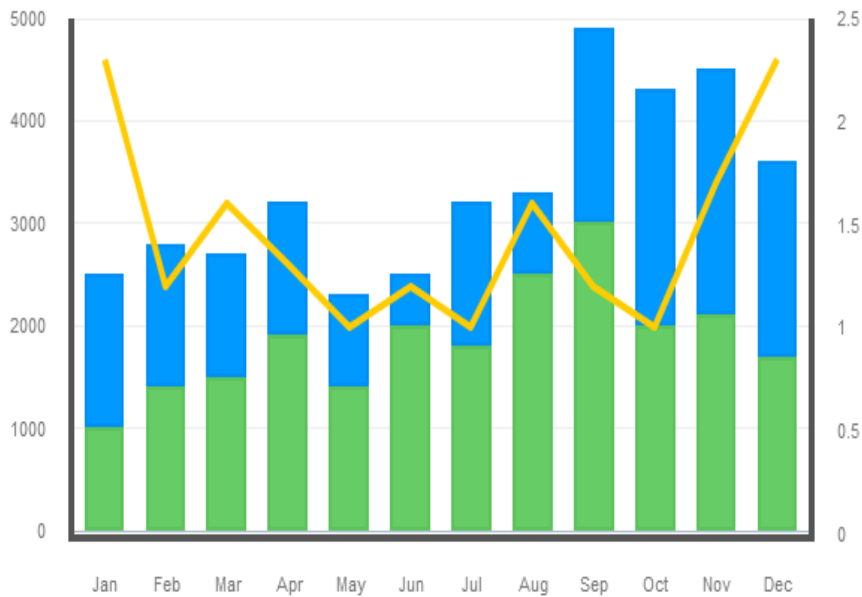
<예제 54 수평축을 위쪽으로, 수직축을 오른쪽으로... 수직축 라벨을 감춘 예제>

#### 7.3.4. 세로축 2 개(듀얼축)로 각각의 데이터 표현하기.

```
<rMateChart cornerRadius="12" borderStyle="solid">
  <Options>
    <Caption text="Annual Report" />
    <SubCaption text="2008" />
    <Legend />
  </Options>
  <Combination2DChart showDataTips="true">
    <horizontalAxis>
      <CategoryAxis categoryField="Month" />
    </horizontalAxis>
    <verticalAxis> // 칼럼시리즈가 참고할 수직축을 정의합니다.
      <LinearAxis id="axis1"/>
    </verticalAxis>
    <series>
      <Column2DSeries yField="Profit" displayName="Profit">
        <showDataEffect>
          <SeriesInterpolate />
        </showDataEffect>
        <fill>
          <SolidColor color="0x66CC66" />
        </fill>
      </Column2DSeries>
      <Line2DSeries selectable="true" yField="Cost" displayName="Cost">
        <verticalAxis> // 라인시리즈가 참고할 수직축을 정의합니다.
          <LinearAxis id="axis2"/>
        </verticalAxis>
        <showDataEffect>
          <SeriesInterpolate />
        </showDataEffect>
        <lineStroke>
          <Stroke color="0xFFCC00" weight="3" />
        </lineStroke>
      </Line2DSeries>
    </series>
    <verticalAxisRenderers> //차트에서 각각의 시리즈가 만든 축을 참조하도록 설정합니다.
      <Axis2DRenderer axis="{axis1}" />
      <Axis2DRenderer axis="{axis2}" />
    </verticalAxisRenderers>
  </Combination2DChart>
```

</rMateChart>

<예제 55 수직축 2 개 설정한 예>

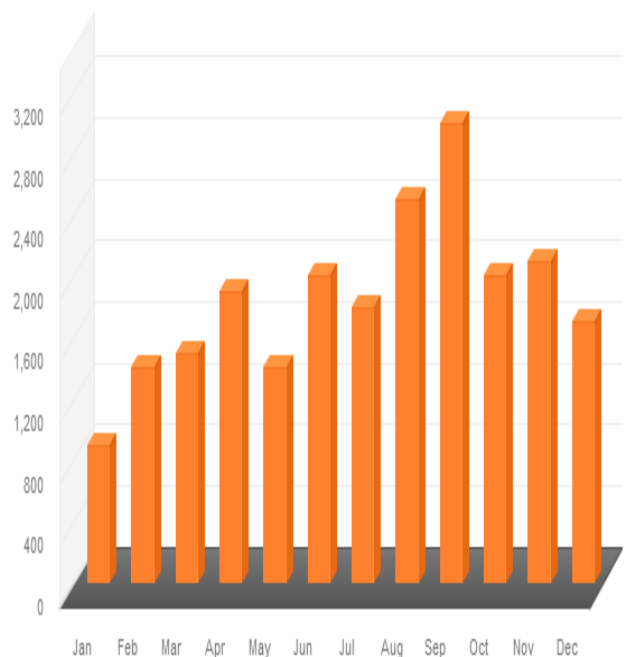


<그림 46 수직축 2 개 설정한 경우 결과>

### 7.3.5. 수직, 수평 축의 수치에 3 자리 단위로 콤마(,) 찍기

LinearAxis 로 정의된 수직 축 또는 수평 축은 출력 결과가 숫자입니다. 수의 단위가 천 단위를 넘었을 경우 콤마 구분자를 삽입하여 읽기 쉽게 표현할 수 있습니다.

```
<rMateChart cornerRadius="12">
  <Options>
    <Caption text="Anual Report"/>
  </Options>
  <NumberFormatter id="numfmt"
    useThousandsSeparator="true"/>
  <Column3DChart>
    <horizontalAxis>
      <CategoryAxis
        categoryField="Month"/>
    </horizontalAxis>
    <verticalAxis>
      <LinearAxis
        formatter="{numfmt}"/>
    </verticalAxis>
    <series>
      <Column3DSeries yField="Profit"
        displayName="Profit"/>
    </series>
  </Column3DChart>
</rMateChart>
```



<예제 56 수직축 수치에 쉼표 구분자 넣기>

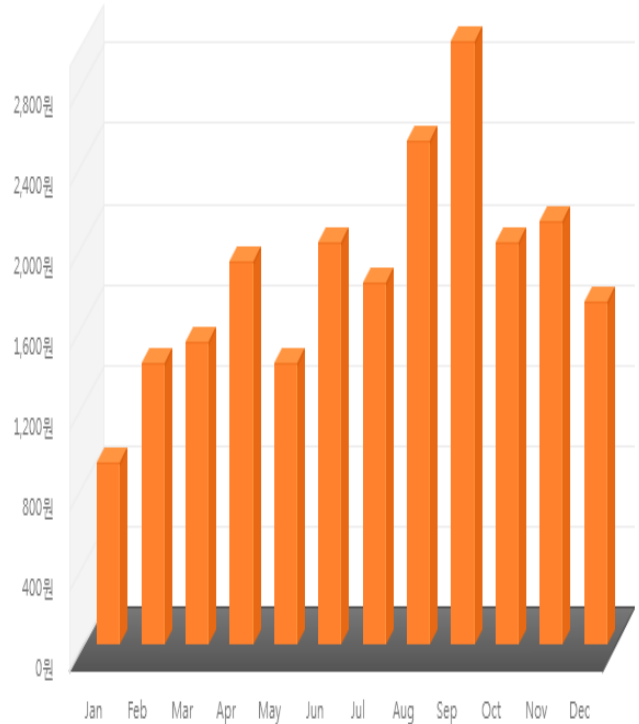
속성 명	유효 값	설명
useThousandsSeparator	True   false (기본값:true)	true 의 경우, 수치는 3 자리수 마다 정해진 구분자로 구분됩니다.
useNegativeSign	true   false (기본값:true)	true 인 경우 음수 앞에 마이너스 부호(-)를 붙여 출력합니다. false 인 경우 음수를 괄호로 출력합니다.
precision	Number (기본값:-1)	출력 할 소수의 자릿수를 나타냅니다. precision를 -1 로 설정하면 precision를 무효로 할 수 있습니다. 값이 -1 의 경우, 자릿수를 변경하지 않습니다.
thousandsSeparatorTo	구분 문자 (기본값: 쉼표(','))	출력 숫자의 자릿수 구분 기호로서 사용하는 캐릭터를 나타냅니다.
decimalSeparatorTo	구분 문자 (기본값:도트('.'))	소수의 값을 출력할 경우에 사용하는 소수점의 단락 캐릭터를 나타냅니다.
rounding	down", "nearest", "up", "none"(기본값:"none")	정해진 소수점 위치에서 올림, 반올림 등을 결정합니다.

<표 17 NumberFormatter 속성 설명표>

### 7.3.6. 수직, 수평 축에 통화단위 넣기

수직축 또는 수평축이 금액을 나타내는 경우 통화 단위를 넣을 필요가 있습니다. 아래 예제는 대한민국 통화단위인 "원"을 오른쪽에 삽입한 예제입니다

```
<rMateChart>
  <Options>
    <Caption text="Anual Report"/>
  </Options>
  <CurrencyFormatter id="fmt"
    currencySymbol="원"
    alignSymbol="right"/>
  <Column3DChart showDataTips="true"
    fontSize="11">
    <horizontalAxis>
      <CategoryAxis
        categoryField="Month"/>
    </horizontalAxis>
    <verticalAxis>
      <LinearAxis
        formatter="{fmt}"/>
    </verticalAxis>
    <series>
      <Column3DSeries
        yField="Profit"/>
    </series>
  </Column3DChart>
</rMateChart>
```



<예제 57 수직 축에 통화단위 "원"을 삽입한 예제>

속성 명	유효 값	설명
currencySymbol	문자(기본값:\$)	포맷의 대상이 되는 수치의 통화 기호로서 사용되는 캐릭터를 나타냅니다
alignSymbol	left   right (기본값:left)	통화 기호의 위치를, 포맷이 끝난 수치의 좌측 또는 우측으로 설정합니다.
useThousandsSeparator	True   false (기본값:true)	true 의 경우, 수치는 3 자리수 마다 정해진 구분자로 구분됩니다.
useNegativeSign	true   false (기본값:true)	true 인 경우 음수 앞에 마이너스 부호(-)를 붙여 출력합니다. false 인 경우 음수를 괄호로 출력합니다.
precision	Number (기본값:-1)	출력 할 소수의 자릿수를 나타냅니다.



		precision를 -1 로 설정하면 precision를 무효로 할 수 있습니다. 값이 -1 의 경우, 자릿수를 변경하지 않습니다.
thousandsSeparatorTo	구분 문자 (기본값: 콤마(,))	출력 숫자의 자릿수 구분 기호로서 사용하는 캐릭터를 나타냅니다.
decimalSeparatorTo	구분 문자 (기본값: 도트(.))	소수의 값을 출력할 경우에 사용하는 소수점의 단락 캐릭터를 나타냅니다.

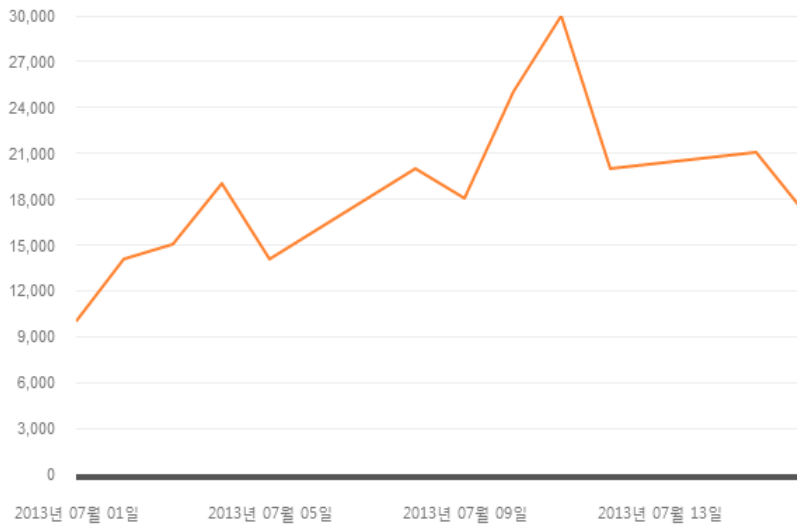
<표 18 CurrencyFormatter 속성 설명표>

### 7.3.7. 수직, 수평 축을 DateTimeAxis 정의한 경우 날짜 포맷 변환하기.

라인차트에서 수평축을 DateTimeAxis 로 정의한 경우 수평축은 날짜(시간)에 따라 데이터가 출력됩니다. 예를 들어 "days"에 따라 데이터가 표현될 때 수평축의 라벨은 "2009/07/10", "2009/07/11" 과 같이 표현됩니다. 수평축에 표현된 날짜의 포맷을 "07/10/2009" 처럼 "월/일/년" 또는 "2009 년 7 월 10 일" 처럼 사용자 정의 형식으로 표현하고 할 때 DateFormatter 를 사용합니다.

아래 예제는 수평축의 라벨을 "2009 년 7 월 10 일" 로 표현한 것 입니다.

```
<rMateChart>
  <Options>
    <Caption text="Annual Report"/>
  </Options>
  <DateFormatter id="dateFmt" formatString="YYYY 년 MM 월 DD일" />
  <NumberFormatter id="curFmt" />
  <Line2DChart showDataTips="true" fontSize="11">
    <horizontalAxis>
      <DateTimeAxis dataUnits="days" labelUnits="days" interval="4"
        formatter="{dateFmt}" alignLabelsToUnits="false" displayLocalTime="true"/>
    </horizontalAxis>
    <verticalAxis>
      <LinearAxis formatter="{curFmt}"/>
    </verticalAxis>
    <series>
      <Line2DSeries xField="Date" yField="Profit" displayName="Profit"/>
    </series>
  </Line2DChart>
</rMateChart>
```



<예제 58 DateTime 축의 날짜 포맷 변환 예>

DateFormatter 를 정의한 후 임의의 id 를 부여합니다. 그리고 DateTimeAxis 의 formatter 속성에 DateFormatter 의 id 를 할당하십시오. **주의할 점은 id 는 일반 String 객체가 아닌 고유 객체를 나타내는 스트링이므로 중괄호로 묶습니다.** (<예제 7 ID 에 의한 속성 값 할당 법> 참고)

DateFormatter 의 속성인 formatString 표현하는 패턴을 정의하는 속성입니다.

예를 들어 “2009/12/01” 과 같이 표현하고자 할 때는 “YYYY/MM/DD”로 정의하십시오. 이에 대한 설명은 아래 표와 같습니다.

패턴 캐릭터	설 명	표현 예
Y	연도(year)입니다	YY = 05 YYYY = 2005 YYYYYY = 02005
M	월(month)입니다.	M = 7 MM= 07 MMM=Jul MMMM= July
D	일(day)입니다.	D = 4 DD = 04 DD = 10
H	시(Hour)입니다.	H = 1 HH = 01
N	분(Minute)입니다.	N = 1 NN = 01

S	초(Second)입니다.	SS = 30
---	---------------	---------

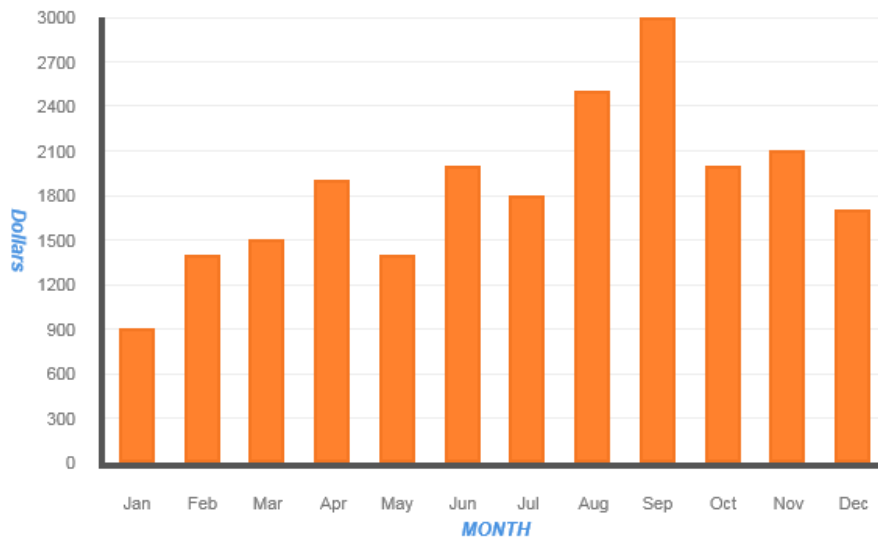
<표 19 formatString 패턴 캐릭터 설명>

### 7.3.8. 축 제목(대표문자) 삽입하기.

수직 축 또는 수평 축이 현재 표현하는 데이터의 이름을 넣고자 할 때가 있습니다. 예를 들어 수직 축에 "Profit" 이라는 제목을 수평 축에 "Month" 라는 제목을 넣고자 할 때 예제는 아래와 같습니다.

```
<rMateChart>
  <Options>
    <Caption text="Annual Report"/>
  </Options>
  <Column2DChart showDataTips="true">
    <horizontalAxis>
      <CategoryAxis categoryField="Month" title="Month"/> //가로축의 제목은 Month
    </horizontalAxis>
    <verticalAxis>
      <LinearAxis title="Profit"/> //세로축의 제목은 Profit
    </verticalAxis>
    <horizontalAxisRenderers> //가로축을 꾸미기 위해 정의하였습니다.
      <Axis2DRenderer axisTitleStyleName="title"/>
    </horizontalAxisRenderers>
    <verticalAxisRenderers> //세로축을 꾸미기 위해 정의하였습니다.
      <Axis2DRenderer axisTitleStyleName="title"/>
    </verticalAxisRenderers>
    <series>
      <Column2DSeries yField="Profit"/>
    </series>
  </Column2DChart>
  <Style> //축의 제목(대표문자)를 꾸미기 위해 정의한 스타일 노트입니다.
    .title //12포인트, 파랑의 볼드체로 축 제목을 꾸밉니다.
    {
      fontSize:12;
      color:0x0000FF;
      fontWeight:"bold";
    }
  </Style>
</rMateChart>
```

<예제 59 축에 제목(대표 문자)를 삽입한 예제>

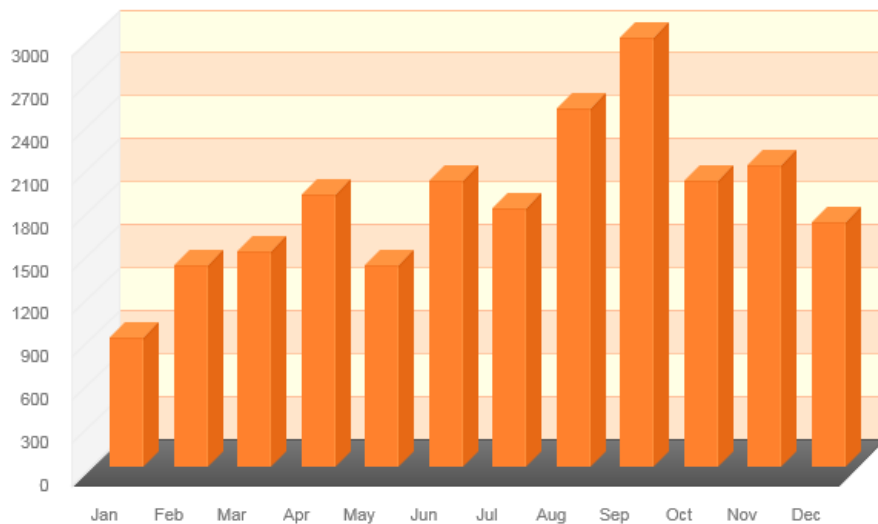


<그림 47 축에 제목(대표 문자)를 삽입한 결과화면>

#### 7.4. 차트 축을 기준으로 안쪽의 배경 꾸미기.

##### 7.4.1. 차트 안쪽 배경에 그리드 라인 삽입하기.

X 축과 Y 축을 갖는 모든 차트는 속성으로 `backgroundElements` 를 갖습니다. 이 `backgroundElements` 는 차트에서 축을 기준으로 안쪽에 해당되는 영역의 배경을 의미합니다. 이곳을 꾸미기 위해 `backgroundElements` 속성을 이용합니다.



<예제 60 차트의 축 안쪽 배경에 스타일 적용한 경우>

위 예제는 수평선만을 정의한 경우입니다. 이에 대한 옵션 설명은 다음과 같습니다.

- `direction = "horizontal", "vertical", "both"` : 수평선 정의, 수직선 정의, 수평,수직 모두 정의

- horizontalChangeCount = "1" : 1 줄 단위로 fill 과 alternateFill 색이 변합니다.
- horizontalStroke : fill 과 alternateFill 사이의 선을 정의합니다.
- horizontalFill : 최초 시작하는 채우기 색입니다.(예제의 옅은 오렌지색)
- horizontalAlternateFill : fill 의 상대적인 채우기 색입니다.(예제의 오렌지색)

아래 레이아웃 예제에서 direction = "vertical" 로 변경하면 수직선이 나옵니다. 수직선 정의 부분은 수평선과 같습니다. 참고하십시오. (레이아웃 예제의 회색 강조 부분이 수직선을 의미합니다.

```
<rMateChart>
  <Column3DChart>
    <backgroundElements>
      <GridLines direction="horizontal" verticalChangeCount="1" horizontalChangeCount="1">
        <horizontalStroke>
          <Stroke color="0xFF9966" alpha="1" weight="2"/>
        </horizontalStroke>
        <horizontalFill>
          <SolidColor color="0xFFCC99" alpha="0.5"/>
        </horizontalFill>
        <horizontalAlternateFill>
          <SolidColor color="0xFFFFCC" alpha="0.5"/>
        </horizontalAlternateFill>
      </GridLines>
    </backgroundElements>
  </Column3DChart>
</rMateChart>
```

<예제 61 차트의 축 안쪽 배경에 스타일 적용한 예제>

#### 7.4.2. 차트 안쪽 배경에 이미지 삽입하기.

차트 안쪽 배경에 외부 이미지를 삽입하기 위해서는 차트의 backgroundElements 속성 또는 annotationElements 속성을 이용합니다. backgroundElements 속성은 그래프가 뿌려진 레이어를 기준으로 뒷면을 의미하고 annotationElements 속성은 그래프가 뿌려진 레이어를 기준으로 앞면을 의미합니다. 다음은 backgroundElements 속성을 이용하여 이미지를 삽입하는 예제입니다.

```
<rMateChart backgroundColor="0xFFFFEE" cornerRadius="12" borderStyle="solid">
  <Line2DChart showDataTips="true">
    <horizontalAxis>
      <CategoryAxis categoryField="Month"/>
    </horizontalAxis>
    <series>
      <Line2DSeries yField='Profit' displayName='Profit' itemRenderer=CircleItemRenderer'
radius='5'>
```

```

        <showDataEffect>
            <SeriesInterpolate />
        </showDataEffect>
        <lineStroke> <Stroke color='0xFF3366' weight='2' /> </lineStroke>
        <stroke> <Stroke color='0xFF3366' weight='2' /> </stroke>
    </Line2DSeries>
</series>
<backgroundElements>
    <Image source="http://www.riamore.net/image/riamoreMain.jpg"
maintainAspectRatio="false" alpha="1" />
</backgroundElements>
</Line2DChart>
</rMateChart>

```

<예제 62 차트 배경에 이미지 삽입 예제>



<그림 48 차트 배경에 이미지 삽입한 화면>

차트 앞면에 이미지를 삽입하는 방법은 <예제 62 차트 배경에 이미지 삽입 예제>에서 다음을 수정해 주십시오.

```

<annotationElements>
    <Image source="http://www.riamore.net/image/riamoreMain.jpg" maintainAspectRatio="false" alpha="0.25" />
</annotationElements>

```

<예제 63 차트 앞면에 이미지 삽입한 예제>

Image 의 투명도가 1 이라면 그래프가 전혀 보이지 않기 때문에 alpha 값을 적절히 조절해 주십시오.

다음은 Image 객체의 옵션 설명입니다.

속성명	설명
source	실제 이미지가 있는 URL 경로입니다.
maintainAspectRatio	이미지 원본 비율을 유지할지 여부를 나타냅니다. 만약 이 속성이 true 라면 차트 고유 비율을 유지하려합니다. 하지만 가로이든 세로이든 차트에 맞춰진 모습을 나타냅니다. false인 경우 차트에 딱맞춰진 사이즈를 갖습니다.
alpha	이미지의 투명도를 나타냅니다.(유효값 : 0~1)

<표 20 Image 객체 속성 설명>

## 7.5. 차트 생성시 효과 적용하기.

rMateChartH5 에서 차트 생성시 출력되는 이펙트는 SeriesInterpolate, SeriesSlide, SeriesZoom, SeriesClip 네 가지가 존재합니다.

- SeriesInterpolate : 초기값에서 목표값까지의 일련의 데이터 변동을 이펙트로 나타냅니다.
- SeriesSlide : 차트의 시리즈가 한 방향에서 반대방향으로 미끄러지듯 나타는 이펙트입니다.
- SeriesZoom : 차트의 한 기준점을 잡고서는 해당 기준점에서 확대되어지는 이펙트입니다.
- SeriesClip : 차트의 위, 아래, 왼쪽, 오른쪽을 기준으로 커져나가보이는 이펙트입니다.

■ **Line2DSeries, Area2DSeries 만 지원하며 IE7, 8 은 지원하지 않습니다.**

속성 명	유효 값	설명	적용가능한 이펙트
duration	밀리 세컨드(기본값:500)	effect 전체를 완료하는데 필요한 시간을 밀리 세컨드 단위로 지정합니다.	공통
elementOffset	밀리 세컨드(기본값 : 20)	effect를 지연 시키는 시간을 밀리 세컨드 단위로 지정합니다.	
minimumElementDuration	밀리 세컨드(기본값 : 0)	각 엘리먼트의 최소 지속 시간을 설정합니다. 계열에 포함되는 어느 엘리먼트에 대해서도, 이 값(밀리 세컨드)보다 짧은 시간에 effect가 실행되지 않습니다.	
offset	밀리 세컨드(기본값 : 0)	effect 개시의 지연 시간을 밀리 세컨드 단위로 지정합니다	
direction	left   right   up   down (기본값 : left)	슬라이딩해 오는 방향을	SeriesSlide

		설정합니다.	
horizontalFocus	center   left   right (기본값 : center)	줌을 시작하는 수평의 초점을 정합니다.	SeriesZoom
verticalFocus	center   top   bottom (기본값 : center)	줌을 시작하는 수직의 초점을 정합니다.	
relativeTo	chart   series (기본값 : chart)	줌의 초점을 위한 경계를 정합니다.	

<표 21 차트 생성시 효과(Effect) 속성과 유효값 설명>

다음과 같이 효과의 설정이 가능합니다.

```

<rMateChart>
  <Column3DChart>
    <Column3DSeries yField="Profit">
      <showDataEffect>
        <SeriesSlide duration="1000" //구체적으로 이펙트 설정한 경우..
          direction="down"
          minimumElementDuration="20"
          elementOffset="30"/>
      </showDataEffect>
    </Column3DSeries>
    <Column3DSeries yField="Cost">
      <showDataEffect>
        <SeriesSlide /> // 기본값만 적용하여 이펙트 설정한 경우.
      </showDataEffect>
    </Column3DSeries>
    .....
    .....
  </rMateChart>
  
```

<예제 64 SeriesSlide 로 이펙트 설정한 예제>

위에 작성한 예제의 이펙트는 다음과 같이 실행됩니다.


각 엘리먼트는 전의 엘리먼트의 30 밀리 세컨드 후에 개시해, 각각의 슬라이드를 완료하는데 20 밀리 세컨드가 걸립니다. effect 전체에서는, 데이터 계열이 모두 슬라이드할 때까지 1 초(1000 밀리 세컨드) 걸립니다. 슬라이딩해 오는 방향은 아래입니다.

## 7.6. 마우스 오버 시 데이터팁(툴팁) 나타내기.



데이터팁(툴팁)은 모든 차트(칼럼, 바, 영역, 라인 등등)에서 마우스 오버 시 나타나게 할 수 있습니다. 차트를 나타내는 노드의 속성 `showDataTips` 를 정의 한 후 속성값으로 `true` 설정을 합니다. 아래 예는 칼럼 3D 차트의 마우스 오버시 나타나는 데이터팁(툴팁)에 관한 예제입니다. `showDataTips` 속성은 모든 차트에 동일한 속성입니다

```
<rMateChart>
  <Options>
    <Caption text="Annual Report"/>
  </Options>
  <Column3DChart showDataTips="true">
    <horizontalAxis>
      <CategoryAxis categoryField="Month" title="Month"/>
    </horizontalAxis>
    <series>
      <Column3DSeries yField="Profit" displayName="Profit">
        <showDataEffect>
          <SeriesInterpolate/>
        </showDataEffect>
      </Column3DSeries>
      .....
    </series>
    .....
  </Column3DChart>
</rMateChart>
```



<마우스 오버시 모습>

<예제 65 데이터팁(툴팁) 예제>

`showDataTips` 를 `true` 로 설정하고 차트의 시리즈 속성 `displayName` 에 텍스트를 입력하면 차트에 마우스 오버 시 위와 같이 텍스트가 함께 출력됩니다. 예제는 `Profit` 이라는 텍스트를 입력하였기 때문에 `Profit` 이 출력되었습니다.

## 7.7. 칼럼 차트 스택형 데이터간 연결선 잇기.

칼럼 2D, 3D 차트를 스택형으로 생성한 경우 각 데이터간 연결선을 이을 수 있습니다. 이는 `Column2DSeries` 또는 `Column3DSeries` 의 한 속성인 `lineToEachItems` 를 `true` 로 설정함으로써 가능합니다.

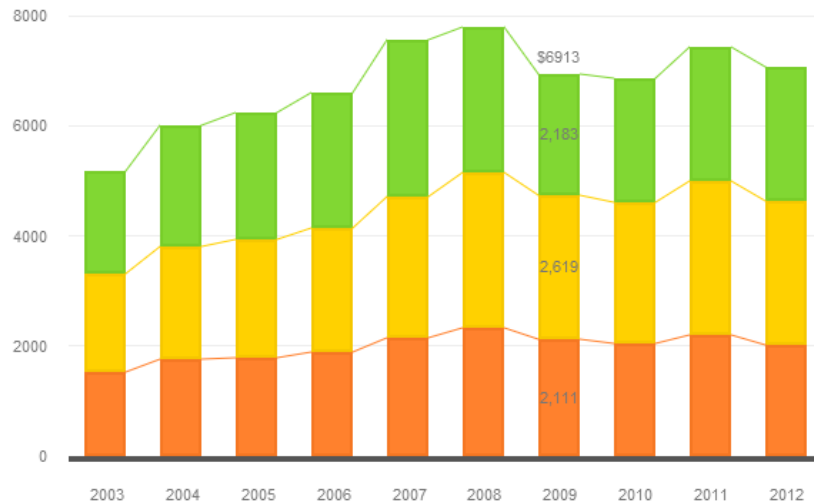
**alwaysShowLines** : 데이터 0 에 대한 선을 그릴지 그리지 않을지 정의합니다.

True - (기본값) 데이터가 0 이더라도 선을 그립니다.

False - 데이터가 0 이라면 선을 그리지 않습니다.

```
<Column2DChart showDataTips="true" type="stacked"> //스택형으로 정의
  <horizontalAxis>
    <CategoryAxis categoryField="Month" />
  </horizontalAxis>
  <series>
    //연결선 잇기 true 설정
    <Column2DSeries yField="Profit" lineToEachItems="true" alwayShowLines="true">
      <linkLineStroke> //연결선의 스타일 정의
        <Stroke weight="2" color="0x000000" caps="none" />
      </linkLineStroke>
      <showDataEffect>
        <SeriesInterpolate />
      </showDataEffect>
    </Column2DSeries>
    <Column2DSeries yField="Cost" lineToEachItems="true" alwayShowLines="true">
      <linkLineStroke>
        <Stroke weight="2" color="0x000000" caps="none" />
      </linkLineStroke>
      <showDataEffect>
        <SeriesInterpolate />
      </showDataEffect>
    </Column2DSeries>
    <Column2DSeries yField="Revenue" lineToEachItems="true" alwayShowLines="true">
      <linkLineStroke>
        <Stroke weight="2" color="0x000000" caps="none" />
      </linkLineStroke>
      <showDataEffect>
        <SeriesInterpolate />
      </showDataEffect>
    </Column2DSeries>
  </series>
</Column2DChart>
```

<예제 66 칼럼 2D 차트 스택형 데이터간 연결선 잇기 예제>



<그림 49 칼럼 2D 차트 스택형 연결선 이은 모습>

## 7.8. 차트 아이템 클릭 시 불러지는 함수 설정하기.

차트의 아이템을 클릭 한 경우 해당 아이템의 정보를 스크립트 단으로 보낼 수 있습니다. 이는 자바스크립트의 함수를 설정하여 그 함수를 호출하는 방식으로 다양한 작업을 가능케 합니다.

먼저, 차트에게 호출할 함수 명을 지정해 줘야 합니다. 레이아웃 XML 에서 차트의 속성으로 `itemClickJsFunction` 을 설정하고, 자바스크립트 함수 명을 속성값으로 할당합니다. 칼럼 3D 차트인 경우 `<Column3DChart/>` 이며, 바차트인 경우 `<Bar3DChart/>` 의 속성입니다.

파이차트를 예로 들면 아래와 같습니다. 이는 파이차트의 파이 한 조각(파이차트 아이템)을 클릭 한 경우 "chartClick" 이라는 함수를 호출하겠다는 이야기입니다. 이 "chartClick" 함수는 자바스크립트에서 반드시 정의되어야 합니다.

```
<rMateChart cornerRadius="12" borderStyle="solid">
  <Options>
    <Caption text="Anual Report" />
  </Options>
  <Pie2DChart showDataTips="true" itemClickJsFunction="chartClick">
    <series>
      <Pie2DSeries field="Profit" nameField="Month" displayName="Profit">
        <showDataEffect>
          <SeriesInterpolate/>
        </showDataEffect>
      </Pie2DSeries>
    </series>
  </Pie2DChart>
</rMateChart>
```

<예제 67 차트 클릭 시 자바스크립트 함수 호출하기>

<예제 67 차트 클릭 시 자바스크립트 함수 호출하기> 과 같이 차트 레이아웃 XML 에서 아이템 클릭 시 호출 할 함수를 지정하였다면 이제 자바스크립트에서 이 함수명과 동일한 함수를 정의하여야 합니다. 예제는 아래와 같습니다.

```
<html lang="ko">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=euc-kr" />
<script src="rMateChartH5.js" language="javascript"></script>

<!-- 사용자 정의 설정 시작 -->
<script language="JavaScript" type="text/javascript">

// ----- chartVars 설정 시작-----
var chartVars;
// 차트 레이아웃 URL 경로.
var layoutURL = encodeURIComponent("chartLayout.xml");
chartVars += "&layoutURL="+layoutURL;

// 데이터를 URL 경로를 통해 가져올 경우 설정하십시오.
// 배열형태로 데이터를 삽입할 경우 주석처리나 삭제하십시오.
// 배열형태와 같이 사용할 경우, 우선순위는 배열형태 데이터 삽입입니다.
var dataURL =encodeURIComponent("singleData.xml");
chartVars += "&dataURL="+dataURL;

rMateChartH5.create("chart1", "chartHolder", chartVars, "100%", "100%");

// ----- chartVars 설정 끝-----

// -----차트에서 item클릭시 불러지는 함수 설정 -----

/*
// layout XML 에서 Chart 속성을 넣을 때 itemClickJsFunction을 주고,만든 javascript 함수명을 넣어줍니다
// 예) <Column3DChart showDataTips="true" itemClickJsFunction="chartClick">
//
// 파라미터 설명
// seriesId : layout XML에서 부여한 series의 id가 있을 경우, 해당 id를 보내줍니다.
// displayText : 화면상에 보여주는 dataTip(마우스 오버 시 보여주는 박스-tooltip)의 내용
// index : 클릭된 item(막대나 파이조각등)의 index 번호 - 맨 왼쪽 또는 맨 위 것이 0번입니다
// data : 해당 item의 값을 표현하기 위해 입력된 data(입력된 데이터중 해당 row(레코드) 자료입니다)
// values : 해당 item의 값 (배열로 전달되며, 차트 종류에 따라 아래와 같이 들어옵니다)
Bar2D(3D)Series      0:x축값 1:y축값
```

```

Column2D(3D)Series  0:x축값 1:y축값
Area2DSeries        0:x축값 1:y축값
Bubble3DSeries      0:x축값 1:y축값 2:radius값
Line2DSeries        0:x축값 1:y축값
Pie2DSeries         0:값
Candlestick2DSeries 0:x축값 1:open값 2:close값 3:high값 4:low값
Bar2D(3D)WingSeries 0:x축값 1:y축값 2:xOpp값 3:yOpp값
TreeMapSeries       0:weight값 1:text값
BoxPlotSeries       0:upper값 1:median값 2:lower값 3:min값 4:hidden값
Vector2DSeries      0:degree값 1:velocity 값, 2:meter값
WordCloudSeries     0:weight값 1:text값

*/
function chartClick(seriesId, displayText, index, data, values)
{
    alert("seriesId:" + seriesId + "WndisplayText:" + displayText + "Wnindex:" + index + "Wndata:" + data + "Wnvalues:" + values);
}

<!-- 사용자 정의 설정 끝 -->
</script>
</head>

<body>
<div class="content">
    <!-- 차트가 삽입될 DIV -->
    <div id="chartHolder" style="width:600px; height:400px;">
    </div>
</div>
</body>
</html>

```

<예제 68 차트 클릭 시 호출하는 함수 자바스크립트에서 정의하기>

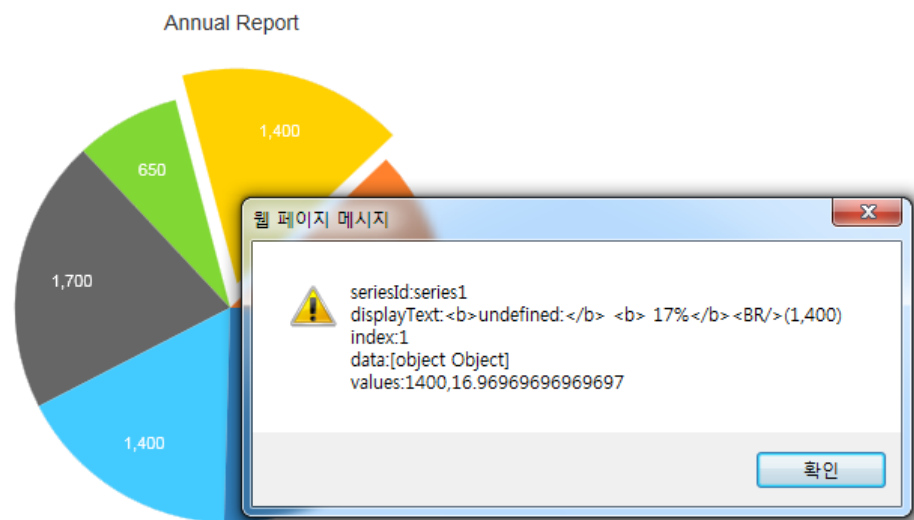
<예제 67 차트 클릭 시 자바스크립트 함수 호출하기> 를 보시면 자바스크립트에서 정의한 함수의 파라미터는 5 개가 있습니다.

파라미터 명	파라미터 설명
seriesId	레이아웃 XML에서 부여한 시리즈(Series)의 id가 있을 경우, 해당 id 입니다.
displayText	화면 상에 보여주는 데이터팁(툴팁, 마우스 오버시 나타나는 팁) 입니다.
index	클릭한 아이템(막대나 파이차트의 조각)의 인덱스 번호입니다. 맨 왼쪽, 맨 위쪽의 값이 0 번입니다.
data	해당 item의 값을 표현하기 위해 입력된 data 입니다. (입력된 데이터중 해당 row(레코드))

	자료입니다)
values	해당 item의 값 입니다. (배열로 전달되며, 차트 종류에 따라 아래와 같이 들어옵니다.)
	Bar2DSeries(Bar3DSeries) 0 : x축 값, 1 : y축 값
	Column2DSeries(Column3DSeries) 0 : x축 값, 1 : y축 값
	Area2DSeries 0 : x축 값, 1 : y축 값
	Bubble3DSeries 0 : x축 값, 1 : y축 값, 2: radius 값
	Line2DSeries 0 : x축 값, 1 : y축 값
	Pie2DSeries(Pie3DSeries) 0 : 값
	Bar2DWingSeries(Column2DWingSeries) 0 : x축 값, 1 : y축 값, 2 : xOpp 값, 3 : yOpp값
	Histogram2DSeries(Histogram3DSeries) 0 : x축 값, 1 : y축 값
	TreeMapSeries 0 : weight값, 1 : name값
	BoxPlotSeries 0 : upper 값, 1 : median 값, 2 : lower 값, 3 : min값, 4 : hidden 값
	Vector2DSeries 0 : degree값, 1 : velocity값, 2 : meter 값
	WordCloudSeries 0 : weightr값, 1 : text 값

<표 22 차트 클릭 시 호출하는 함수의 파라미터 설명>

지금까지 파이차트의 한 아이템을 클릭 한 경우 자바스크립트의 "chartClick" 함수를 호출하게 설정하였습니다. 이에 대한 실행화면으로 자바스크립트에서 알림창을 띄워 차트로부터 넘어온 파라미터를 보여주고 있습니다.



<그림 50 파이차트 클릭 시 함수 호출 실행화면>

## 7.9. 사용자 정의 함수 설정하기

사용자 정의 함수를 설정할 수 있는 경우는 총 3 가지가 존재합니다.

1. 데이터팁 사용자 정의
2. 축 라벨 사용자 정의(축에 직접정의, 축렌더러에 정의)
3. 칼럼, 바, 파이계열 차트의 수치필드 사용자 정의

자바스크립트 함수를 정의하고 레이아웃에 그 함수명을 지시하여 차트가 해당 함수를 호출하는 원리로 사용자 정의 함수가 적용됩니다.

**축 라벨, 수치필드(데이터팁 제외)를 사용자 정의할 때 반드시 formatter 속성과 함께 사용하지 마십시오.** formatter 는 기본적인 사용자 정의 함수입니다. formatter 속성과 labelJsFunction 속성을 함께 정의한 경우 정상적인 차트를 생성할 수 없습니다.

#### 7.9.1. 데이터팁(툴팁) 함수 사용자 정의.

차트에서 showDataTips="true" 설정 후 마우스 오버 시 보이는 데이터팁을 사용자 정의할 수 있습니다. 차트에서 마우스 오버 이벤트가 발생하면 레이아웃에서 미리 지정된 dataTipJsFunction 함수를 호출하여 그 함수가 리턴하는 값을 데이터팁으로 표시하게 됩니다.

레이아웃 XML 에서 dataTipJsFunction 속성을 추가하고 속성값으로 자바스크립트 함수명을 지정하십시오. 이 자바스크립트 함수는 차트 아이템에 마우스 오버 시 rMate 차트가 호출할 콜백함수입니다.

다음 예제는 칼럼 3D 차트의 데이터팁을 사용자 정의한 레이아웃 예제입니다.

```
<rMateChart backgroundColor="0xFFFFEE" cornerRadius="12" borderStyle="solid">
  <Options>
    <Caption text="Annual Report"/>
  </Options>
  <!-- dataTipFunc 는 자바스크립트 함수명 입니다. -->
  <Column3DChart showDataTips="true" dataTipJsFunction="dataTipFunc">
    <horizontalAxis>
      <CategoryAxis categoryField="Month" displayName="날짜"/>
    </horizontalAxis>
    <series>
      <Column3DSeries id="series1" yField="Profit" displayName="Profit">
        <showDataEffect>
          <SeriesInterpolate />
        </showDataEffect>
      </Column3DSeries>
      <Column3DSeries id="series2" yField="Cost" displayName="Cost">
        <showDataEffect>
          <SeriesInterpolate />
        </showDataEffect>
      </Column3DSeries>
      <Column3DSeries id="series3" yField="Revenue" displayName="Revenue">
```

```

        <showDataEffect>
            <SeriesInterpolate />
        </showDataEffect>
    </Column3DSeries>
</series>
</Column3DChart>
</rMateChart>

```

*<예제 69 데이터팁 사용자 정의 레이아웃 예제>*

레이아웃에서 자바스크립트 콜백함수를 등록한 후 rMate 차트를 임베딩하는 파일, 예를 들면 html, jsp, php 파일에서 다음과 같이 콜백함수를 실제로 정의하십시오.

```

// 차트에서 showDataTips="true" 설정 후 마우스 오버 시 보이는 데이터팁 정의
// layout XML 에서 Chart 속성을 넣을때 dataTipJsFunction를 주고, 만든 javascript 함수명을 넣어줍니다
// 예) <Column3DChart showDataTips="true" dataTipJsFunction를="dataTipFunc">
// 파라미터 설명
// seriesId : layout XML에서 부여한 series의 id가 있을 경우, 해당 id를 보내줍니다.
// seriesName : 시리즈의 displayName 으로 정의한 시리즈 이름을 보내줍니다.
// xName : X 축에 displayName 을 정의하였다면 X축의 displayName을 보내줍니다.
// yName : Y 축에 displayName 을 정의하였다면 Y축의 displayName을 보내줍니다.
// data : 해당 item의 값을 표현하기 위해 입력된 data(입력된 데이터중 해당 row(레코드) 자료입니다)
// values : 해당 item의 값 (배열로 전달되며, 차트 종류에 따라 아래와 같이 들어옵니다.)
    Bar2D(3D)Series      0:x축값 1:y축값
    Column2D(3D)Series   0:x축값 1:y축값
    Area2DSeries         0:x축값 1:y축값
    Bubble3DSeries       0:x축값 1:y축값 2:radius값
    Line2DSeries         0:x축값 1:y축값
    Pie2D(3D)Series      0:값 1:퍼센트값 2:nameFiled명
    Candlestick2DSeries  0:x축값 1:open값 2:close값 3:high값 4:low값
    Bar2D(3D)WingSeries  0:x축값 1:y축값 2:xOpp값 3:yOpp값
    TreeMapSeries        0:weight값 1:text값
    BoxPlotSeries        0:upper값 1:median값 2:lower값 3:min값 4:hidden값
    Vector2DSeries       0:degree값 1:velocity 값, 2:meter값
    WordCloudSeries      0:weight값 1:text값
*/
function dataTipFunc(seriesId, seriesName, index, xName, yName, data, values){
    return "<font color='#CC3300'>데이터팁 사용자 정의</font>#seriesId:"+seriesId
        + "<br/><font color='#0000FF'>현재 데이터 : </font>"
        + "<b>" + seriesName + "</b>" + "#nitemIndex:" + index
        + "<br/><font color='#0000FF'>xDisplayName : </font>" + "<b><font size='11'>" + xName + "</font></b>"
        + "<br/><font color='#0000FF'>yDisplayName : </font>" + "<b><font size='11'>" + yName + "</font></b>"
        + "<br/>data:" + data + "<br/>values:" + values;
}

```



```
}

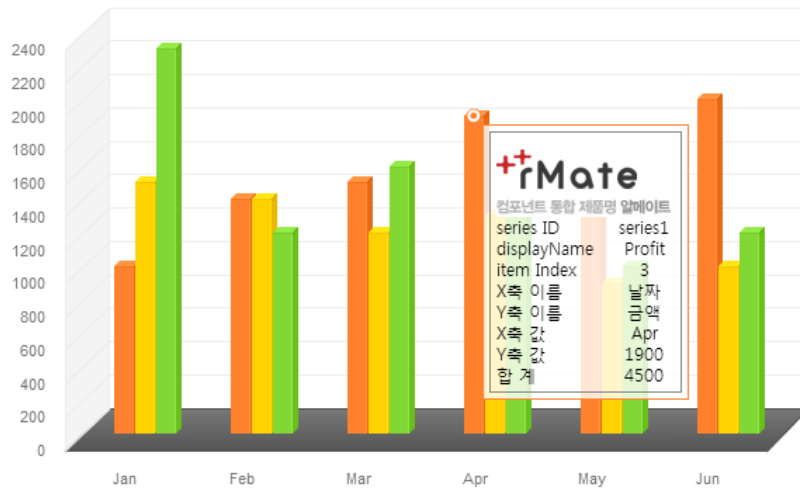
```

<예제 70 자바스크립트 데이터팁 콜백함수 예제>

다음은 자바스크립트 콜백함수를 정의할 때 함께 정의되어야 할 함수 파라미터에 대한 설명입니다.

파라미터 명	파라미터 설명																								
seriesId	레이아웃 XML에서 부여한 시리즈(Series)의 id가 있을 경우, 해당 id 입니다.																								
seriesName	화면 상에 보여주는 데이터팁(툴팁, 마우스 오버시 나타나는 팁) 입니다.																								
index	클릭한 아이템(막대나 파이차트의 조각)의 인덱스 번호입니다. 맨 왼쪽, 맨 위쪽의 값이 0 번입니다.																								
xName	X축에 displayName을 정의하였을 경우 나타나는 X축 이름입니다.																								
yName	Y축에 displayName을 정의하였을 경우 나타나는 Y축 이름입니다.																								
data	해당 item의 값을 표현하기 위해 입력된 data 입니다. (입력된 데이터중 해당 row(레코드) 자료입니다)																								
values	<p>해당 item의 값 입니다. (배열로 전달되며, 차트 종류에 따라 아래와 같이 들어옵니다.)</p> <table> <tr> <td>Bar2DSeries(Bar3DSeries)</td><td>0 : x축 값, 1 : y축 값</td></tr> <tr> <td>Column2DSeries(Column3DSeries)</td><td>0 : x축 값, 1 : y축 값</td></tr> <tr> <td>Area2DSeries</td><td>0 : x축 값, 1 : y축 값</td></tr> <tr> <td>Bubble3DSeries</td><td>0 : x축 값, 1 : y축 값, 2: radius 값</td></tr> <tr> <td>Line2DSeries</td><td>0 : x축 값, 1 : y축 값</td></tr> <tr> <td>Pie2DSeries(Pie3DSeries)</td><td>0 : 값 1:퍼센트 값 2:nameField</td></tr> <tr> <td>Bar2DWingSeries(Column2DWingSeries)</td><td>0 : x축 값, 1 : y축 값, 2 : xOpp 값, 3 : yOpp값</td></tr> <tr> <td>Histogram2DSeries(Histogram3DSeries)</td><td>0 : x축 값, 1 : y축 값</td></tr> <tr> <td>TreeMapSeries</td><td>0 : weight값, 1 : name값</td></tr> <tr> <td>BoxPlotSeries</td><td>0 : upper 값, 1 : median 값, 2 : lower 값, 3 : min값, 4 : hidden 값</td></tr> <tr> <td>Vector2DSeries</td><td>0 : degree값, 1 : velocity값, 2 : meter 값</td></tr> <tr> <td>WordCloudSeries</td><td>0 : weighr값, 1 : text 값</td></tr> </table>	Bar2DSeries(Bar3DSeries)	0 : x축 값, 1 : y축 값	Column2DSeries(Column3DSeries)	0 : x축 값, 1 : y축 값	Area2DSeries	0 : x축 값, 1 : y축 값	Bubble3DSeries	0 : x축 값, 1 : y축 값, 2: radius 값	Line2DSeries	0 : x축 값, 1 : y축 값	Pie2DSeries(Pie3DSeries)	0 : 값 1:퍼센트 값 2:nameField	Bar2DWingSeries(Column2DWingSeries)	0 : x축 값, 1 : y축 값, 2 : xOpp 값, 3 : yOpp값	Histogram2DSeries(Histogram3DSeries)	0 : x축 값, 1 : y축 값	TreeMapSeries	0 : weight값, 1 : name값	BoxPlotSeries	0 : upper 값, 1 : median 값, 2 : lower 값, 3 : min값, 4 : hidden 값	Vector2DSeries	0 : degree값, 1 : velocity값, 2 : meter 값	WordCloudSeries	0 : weighr값, 1 : text 값
Bar2DSeries(Bar3DSeries)	0 : x축 값, 1 : y축 값																								
Column2DSeries(Column3DSeries)	0 : x축 값, 1 : y축 값																								
Area2DSeries	0 : x축 값, 1 : y축 값																								
Bubble3DSeries	0 : x축 값, 1 : y축 값, 2: radius 값																								
Line2DSeries	0 : x축 값, 1 : y축 값																								
Pie2DSeries(Pie3DSeries)	0 : 값 1:퍼센트 값 2:nameField																								
Bar2DWingSeries(Column2DWingSeries)	0 : x축 값, 1 : y축 값, 2 : xOpp 값, 3 : yOpp값																								
Histogram2DSeries(Histogram3DSeries)	0 : x축 값, 1 : y축 값																								
TreeMapSeries	0 : weight값, 1 : name값																								
BoxPlotSeries	0 : upper 값, 1 : median 값, 2 : lower 값, 3 : min값, 4 : hidden 값																								
Vector2DSeries	0 : degree값, 1 : velocity값, 2 : meter 값																								
WordCloudSeries	0 : weighr값, 1 : text 값																								

<표 23 데이터팁 사용자 정의 콜백함수 파라미터 설명>



<그림 51 데이터팁 사용자 정의 실행모습>

### 7.9.2. 축 라벨 사용자 정의

X, Y 축이 존재하는 데카르트좌표계열 차트의 축에 표시되는 라벨값 사용자 정의는 X 축, Y 축에 해당되는 해당 축 노드에 `labelJsFunction` 속성을 정의하고 자바스크립트 콜백 함수명을 지정합니다.

축 라벨 사용자 정의는 축에 직접 하는 방법과 축렌더러에 정의하는 방법이 있습니다. 적용방법은 두가지가 같습니다. 축렌더러에 정의한 콜백함수가 늦게 호출됩니다.

다음은 X 축 라벨을 사용자 정의하기 위해 설정한 레이아웃 예제입니다.

```
<rMateChart backgroundColor="0xFFFFEE" cornerRadius="12" borderStyle="solid">
  <Options>
    <Caption text="Annual Report"/>
  </Options>
  <Column3DChart showDataTips="true">
    <horizontalAxis>
      <!-- axisLabelFunc 는 자바스크립트 함수명 입니다. -->
      <CategoryAxis categoryField="Month" displayName="날짜"
labelJsFunction="axisLabelFunc"/>
    </horizontalAxis>
    <verticalAxis>
      <LinearAxis displayName="금액"/>
    </verticalAxis>
    ...
    ...
    ...
  </Column3DChart>
```

</rMateChart>

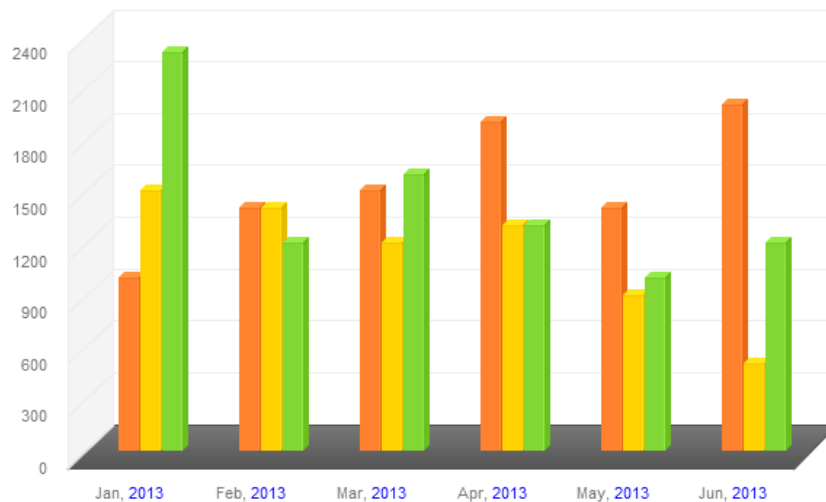
<예제 71 축 라벨 사용자 정의 레이아웃 예제>

```

/*
// ----- X축 라벨 사용자 정의 함수 -----
// X, Y 축이 존재하는 데카르트 좌표계열 차트에서 축 라벨을 사용자 정의 합니다.
// layout XML 에서 축 렌더러 속성을 넣을때 labelJsFunction 주고, 만든 javascript 함수명을 넣어줍니다
// 예))<horizontalAxis>
//      <CategoryAxis id="hAxis" categoryField="Month" labelJsFunction="axisLabelFunc" />
//    </horizontalAxis>
//
// 파라미터 설명
// id : 축의 ID
// value : 현재 아이템에 맞는 축의 라벨 값
*/
function axisLabelFunc(id, value)
{
    return value + ", 2009";
}

```

<예제 72 축 라벨 자바스크립트 콜백함수 정의 예제>



<예제 73 축 라벨 사용자 정의 실행화면>

### 7.9.3. 수치필드 사용자 정의

수치필드를 표시할 수 있는 칼럼, 바, 파이차트 계열에서 수치필드를 사용자 정의하고자 하는 경우 해당 Series 노드에 labelJsFunction 속성을 추가한 후 자바스크립트 함수명을 속성값으로 지시하십시오.

다음은 칼럼 2D 차트에 수치필드를 사용자 정의한 레이아웃 예제입니다.

```
<series>
  <!-- seriesLabelFunc 는 자바스크립트 함수명 입니다. -->
  <Column2DSeries yField="Profit" displayName="Profit" labelPosition="outside" styleName="seriesStyle"
  outsideLabelJsFunction="seriesLabelFunc">
    <showDataEffect>
      <SeriesInterpolate/>
    </showDataEffect>
  </Column2DSeries>
</series>
```

<예제 74 수치필드 사용자 정의 레이아웃 예제>

```
/*
// ----- 수치 필드 사용자 정의 함수 -----// 칼럼, 바 차트에서 labelPosition
속성을 설정한 경우 수치 필드를 사용자 정의하는 함수입니다.
// layout XML 에서 Series 속성을 넣을때 labelJsFunction 주고, 만든 javascript 함수명을 넣어줍니다
//
// 예) <Column2DSeries yField="Profit" labelPosition="outside" outsideLabelJsFunction="seriesLabelFunc">
//
// 파라미터 설명
// seriesId : 해당 시리즈의 id
// index : 해당 아이템의 인덱스.
// data : 해당 item의 값을 표현하기 위해 입력된 data(입력된 데이터중 해당 row(레코드) 자료입니다)
// values : 해당 item의 값 (배열로 전달되며, 차트 종류에 따라 아래와 같이 들어옵니다.)
    Bar2D(3D)Series      0:x축값 1:y축값
    Column2D(3D)Series   0:x축값 1:y축값
    Pie2D(3D)Series      0:값 1:퍼센트값
    Bubble3DSeries        0:x축값 1:y축값 2:z축값
    Candlestick2DSeries   0:x축값 1:open값 2:close값 3:high값 4:low값
    Bar2D(3D)WingSeries   0:x축값 1:y축값 2:xOpp값 3:yOpp값
    TreeMapSeries         0:weight값 1:text값
    BoxPlotSeries         0:upper값 1:median값 2:lower값 3:min값 4:hidden값
    Vector2DSeries        0:degree값 1:velocity 값, 2:meter값
    WordCloudSeries       0:weight값 1:text값

// 참고 : 수치필드 사용자 정의 시엔 <br/>태그와 같은 html형식의 코딩 삽입이 불가능합니다.
*/
function seriesLabelFunc(seriesId, index, data, values)
{
    return "$"+values[1];
}
```

<예제 75 수치필드 자바스크립트 콜백함수 예제>

시리즈 명	수치 사용자 정의 함수 명
Column2DSeries Column3DSeries, Bar2DSeries, Bar3DSeries, Column2DWingSeries, Bar2DWingSeries	insideLabelJsFunction, outsideLabelJsFunction
Line2DSeries, Area2DSeries	upLabelJsFunction, downLabelJsFunction
Bubble3DSeries	insideLabelJsFunction
Pie2DSeries, Pie3DSeries	labelJsFunction

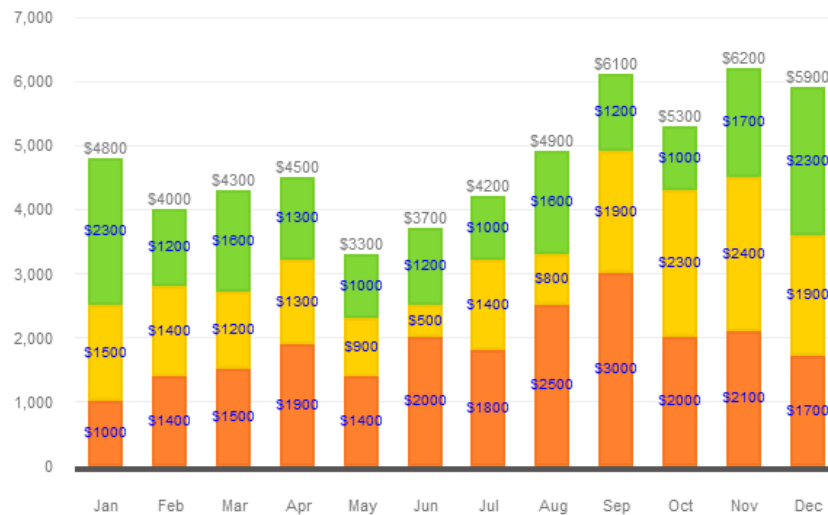
<표 24 시리즈 별 labelJsFunction 명>

**위 labelJsFunction 은 labelPosition 의 영향을 받습니다.**

다음은 자바스크립트 콜백함수를 정의할 때 함께 정의되어야 할 함수 파라미터에 대한 설명입니다.

파라미터 명	파라미터 설명																		
seriesId	해당 시리즈의 id 입니다.																		
index	아이템(막대나 파이차트의 조각)의 인덱스 번호입니다. 맨 왼쪽, 맨 위쪽의 값이 0 번 입니다.																		
data	해당 item의 값을 표현하기 위해 입력된 data 입니다. (입력된 데이터중 해당 row(레코드) 자료입니다)																		
values	<p>해당 item의 값 입니다. (배열로 전달되며, 차트 종류에 따라 아래와 같이 들어옵니다.)</p> <table border="1"> <tr> <td>BarSeries(Bar3DSeries)</td><td>0 : x축 값, 1 : y축 값</td></tr> <tr> <td>ColumnSeries(Column3DSeries)</td><td>0 : x축 값, 1 : y축 값</td></tr> <tr> <td>Pie2DSeries(Pie3DSeries)</td><td>0 : 값 1:퍼센트 값</td></tr> <tr> <td>Candlestick2DSeries</td><td>0 : x축 값, 1 : open값, 2 : close값 3 : high 값, 4 : low값</td></tr> <tr> <td>Column2D(Bar2D)WingSeries</td><td>0 : x축 값, 1 : y축 값, 2 : xOpp 값, 3 : yOpp 값</td></tr> <tr> <td>TreeMapSeries</td><td>0 : weight 값, 1 : text 값</td></tr> <tr> <td>BoxPlotSeries</td><td>0 : upper 값, 1 : median 값, 2 : lower 값 3 : min 값, 4 : hidden 값</td></tr> <tr> <td>Vector2DSeries</td><td>0 : degree 값, 1 : velocity 값, 2 : meter 값</td></tr> <tr> <td>WordCloudSeries</td><td>0 : weight 값, 1 : text 값</td></tr> </table>	BarSeries(Bar3DSeries)	0 : x축 값, 1 : y축 값	ColumnSeries(Column3DSeries)	0 : x축 값, 1 : y축 값	Pie2DSeries(Pie3DSeries)	0 : 값 1:퍼센트 값	Candlestick2DSeries	0 : x축 값, 1 : open값, 2 : close값 3 : high 값, 4 : low값	Column2D(Bar2D)WingSeries	0 : x축 값, 1 : y축 값, 2 : xOpp 값, 3 : yOpp 값	TreeMapSeries	0 : weight 값, 1 : text 값	BoxPlotSeries	0 : upper 값, 1 : median 값, 2 : lower 값 3 : min 값, 4 : hidden 값	Vector2DSeries	0 : degree 값, 1 : velocity 값, 2 : meter 값	WordCloudSeries	0 : weight 값, 1 : text 값
BarSeries(Bar3DSeries)	0 : x축 값, 1 : y축 값																		
ColumnSeries(Column3DSeries)	0 : x축 값, 1 : y축 값																		
Pie2DSeries(Pie3DSeries)	0 : 값 1:퍼센트 값																		
Candlestick2DSeries	0 : x축 값, 1 : open값, 2 : close값 3 : high 값, 4 : low값																		
Column2D(Bar2D)WingSeries	0 : x축 값, 1 : y축 값, 2 : xOpp 값, 3 : yOpp 값																		
TreeMapSeries	0 : weight 값, 1 : text 값																		
BoxPlotSeries	0 : upper 값, 1 : median 값, 2 : lower 값 3 : min 값, 4 : hidden 값																		
Vector2DSeries	0 : degree 값, 1 : velocity 값, 2 : meter 값																		
WordCloudSeries	0 : weight 값, 1 : text 값																		

<표 25 수치필드 콜백함수 파라미터 설명>



<그림 52 수치필드 사용자 정의 화면>

#### 7.9.4. 채우기 색 사용자 정의

차트의 채우기 색(fill) 을 어떤 조건에 따라 또는 기호에 맞게 사용자 정의하고 하는 경우 해당 Series 노드에 fillJsFunction 속성을 추가한 후 자바스크립트 함수명을 속성값으로 지시하십시오.

다음은 칼럼 3D 차트의 채우기색을 사용자 정의한 레이아웃 예제입니다.

```
<Column3DChart showDataTips="true" gutterLeft="0" showLabelVertically="true">
  <series>
    <Column3DSeries yField="Profit"
      fillJsFunction="fillJsFunc" styleName="seriesStyle">
      <showDataEffect>
        <SeriesSlide direction="up" duration="1000"/>
      </showDataEffect>
    </Column3DSeries>
  </series>
  ...
  ...
  ...
</Column3DChart>
```

<예제 76 채우기 색 사용자 정의 레이아웃>

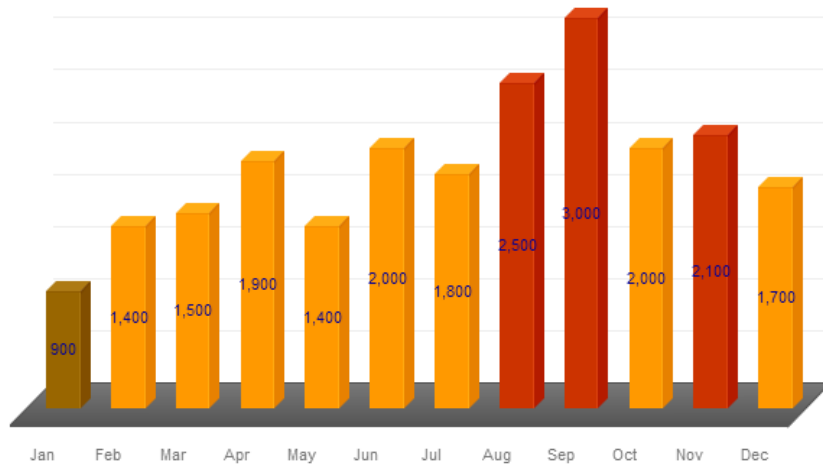
다음은 html, jsp, php, asp 등의 스크립트 파일에서 정의한 자바스크립트 예제입니다.

```
/*
//----- 채우기 색 사용자 정의 함수 -----
//
//차트의 채우기 색을 특정 조건에 따라 지정하는 사용자 정의 함수입니다.
```

```
//layout XML 에서 Series 속성을 넣을 때 fillJsFunction 을 주고, 만든 javascript 함수명을 넣어줍니다.
//
//예) <Pie2DSeries field="Profit" fillJsFunction="fillJsFunc">
//
//파라미터 설명
//seriesId : 해당 시리즈의 id
//index : 해당 아이템의 인덱스.
//data : 해당 item의 값을 표현하기 위해 입력된 data(입력된 데이터중 해당 row(레코드) 자료입니다)
//values : 해당 item의 값 (배열로 전달되며, 차트 종류에 따라 아래와 같이 들어옵니다.)
    Bar2D(3D)Series      0:x축값 1:y축값
    Column2D(3D)Series   0:x축값 1:y축값
    Area2DSeries         0:x축값 1:y축값
    Bubble3DSeries       0:x축값 1:y축값 2:radius값
    Line2DSeries         0:x축값 1:y축값
    Pie2D(3D)Series      0:값 1:퍼센트값 2:nameField명
    Column2D(Bar2D)WingSeries 0 : x축 값, 1 : y축 값, 2 : xOpp값, 3 : yOpp값
    Candlestick2DSeries  0 : x축 값, 1 : open값, 2 : close값 3: high값 4 : low값
    TreeMapSeries        0 : weight값, 1 : text 값
    BoxPlotSeries         0 : upper값, 1 : median값, 2 : lower값, 3 : min값, 4 : hidden값
    Vector2DSeries        0 : degree값 1 : velocity 값 2 : meter 값
    WordCloudSeries       0 : weight 값 1 : text 값

// From-To Chart 에서 minField 를 지정했다면 values 의 마지막 인덱스 값에 minField 값이 들어옵니다.
*/
function fillJsFunc(seriesId, index, data, values)
{
    if(values[1] > 2000)
        return "0xFF3366";
    else if(values[1] > 1000)
        return "0xFFFF33";
    else
        return "0xFF9999";
}
```

<예제 77 채우기색 자바스크립트 콜백함수 예제>

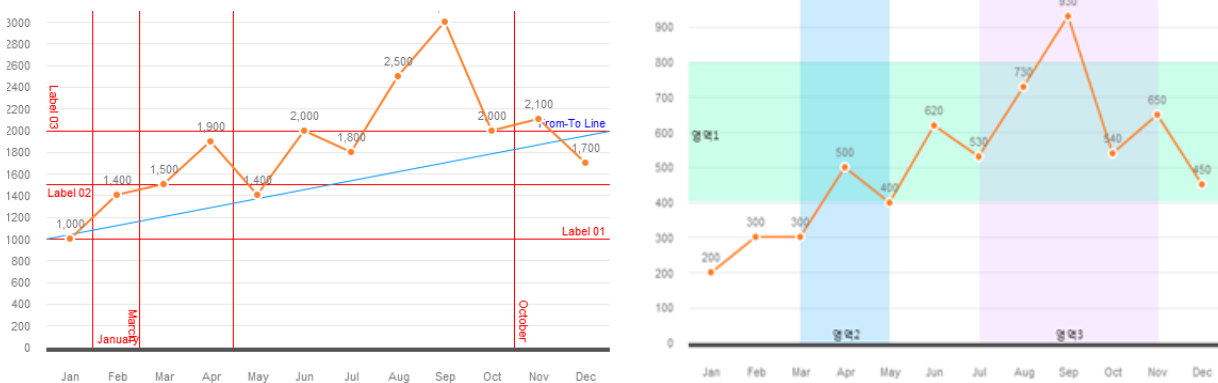


<그림 53 채우기색 사용자 정의 화면>

## 7.10. 차트 안 범위 지정 및 다수의 선 긋기

상하한선은 min/max 를 잡아 영역을 칠하고 선을 긋는 반면 차트 안 범위 지정 및 선 긋기는 범위의 지정 및 선을 긋는데 제한이 없습니다.

예로 아래 그림처럼 다수의 선을 긋고 영역 표시를 할 수 있습니다.



<그림 54 차트 안 범위 지정 및 다수의 선 긋기>

위를 표현하기 위한 노드는 다음과 같습니다.

차트의 backgroundElements 또는 annotationElements 속성	선이나 범위를 차트 아래에 표현하고자 하는 경우는 backgroundElements, 차트 위에 나타내고자 하는 경우 annotationElements 속성을 사용합니다.
AxisMarker	AxisLine 과 AxisRange 를 표현할 수 있는 대표자입니다.



AxisLine	라인을 표현할 수 있습니다.
AxisRange	영역을 표현할 수 있습니다.

실제로 차트에 영역과 라인을 표현하는 레이아웃 샘플입니다.

```

<rMateChart backgroundColor= "0xFFFFF">
  <Stroke id= "stroke1" color= "0xFF0000" weight= "1"/>
  <Line2DChart showDataTips= "true">

    .....
    // 차트의 배경에 표시하고 있음.
  <backgroundElements>
    <GridLines/> // 선과 영역포함하여 차트 그리드도 출력하기 위함
    <AxisMarker> // 선과 영역을 표현할 수 있는 대표자 선언

    <lines> // 라인을 긋고자함
    // AxisLine 을 3개 정의하여 3개의 라인을 그음.
    <AxisLine value= "1000" label= "Label 01" stroke= "{stroke1}" labelUpDown= "up"
color= "0xFF0000"/>

    <AxisLine value= "1500" label= "Label 02" stroke= "{stroke1}" labelAlign= "left"
labelUpDown= "down" color= "0xFF0000"/>

    <AxisLine value= "2000" label= "Label 03" stroke= "{stroke1}" labelUpDown= "up" labelAlign= "left"
labelRotation= "90" color= "0xFF0000"/>
  </lines>

  <ranges> // 영역을 표현하고자함.
  //AxisRange 2개 정의하여 2개의 범위 지정
  <AxisRange startValue= "1000" endValue= "2000" label= "영역1" fontSize= "11"
labelHorizontalAlign= "left" color= "0xFF00FF">
    <fill>
      <SolidColor color= "0x00FF99" alpha= "0.2"/>
    </fill>
  </AxisRange>
  <AxisRange startValue= "Mar" endValue= "May" label= "영역2" fontSize= "11"
labelVerticalAlign= "bottom" color= "0x0066FF" horizontal= "false">
    <fill>
      <SolidColor color= "0x0099FF" alpha= "0.2"/>
    </fill>
  </AxisRange>
  </ranges>
  </Line2DChart>
</rMateChart>
  
```

```

    </AxisRange>
  </ranges>

  AxisMarker>
</backgroundElements>
</Line2DChart>
</rMateChart>

```

다음은 해당 노드들의 속성 및 유효값 설명입니다.

속성명	유효값	설명
lines	AxisLine 노드	차트에 표현할 라인들을 정의합니다.
ranges	AxisRange 노드	차트에 표현할 영역들을 정의합니다.

AxisRange 와 AxisLine 은 기본적으로 라벨을 출력시킵니다. 따라서 Caption 의 속성을 모두 포함하고 있습니다. 아래 표는 Caption 의 속성 부분은 제외합니다. 이에 대한 속성 설명은 "5.3.1 차트 제목(Caption), 부제목(SubCaption) 넣기." 를 참고하여 주십시오.

AxisLine -

속성명	유효값	설명
labelAlign	center,left,right (기본값:right)	라인에 표시할 라벨의 수평정렬. 속성 horizontal이 false일 경우 left는 top, right는 bottom의 기능을 합니다
labelUpDown	up,down (기본값:up)	라벨을 라인의 위에 표시할지 아래에 표시할지 여부. 속성 horizontal이 false일 경우 up은 우측, down은 좌측에 위치 시킵니다.
labelRotation	0~360(기본값:0)	표시할 라벨의 회전. Embedded폰트를 사용할 경우에는 모든 각도 가능하나 시스템 폰트를 사용할 경우에는 0만 가능합니다.
linePosition	center, left, right	CategoryAxis의 경우 라인을 표시할 위치. CategoryAxis에서는 한개의 값에 대한 선을 그을 수 있는 위치가 여러개가 가능하여 선택할 수 있도록 합니다.
stroke	Stroke 객체	라인의 스타일을 설정합니다.
startValue	Number,String	표시 하는 선이 사선일 경우 라인이 시작되는 위치 값. value값이 설정되면 이 값은 무시됩니다.
endValue	Number,String	표시하려는 선이 사선일 경우 라인이 끝나는 위치

		값. value 값이 설정되면 이 값은 무시됩니다.
Value	Number,String	표시하려는 선이 사선이 아닌경우 라인이 표시될 값. 이 값이 설정되면 startValue, endValue값은 무시됩니다.
label	String	선에 표시할 텍스트를 정의하십시오.

<표 26 AxisLine 속성 및 유효값 설명>

AxisRange -

속성 명	유효 값	설 명
labelHorizontalAlign	String( center,left,right )	범위에 표시할 라벨의 수평정렬입니다.
labelVerticalAlign	String( top,middle,bottom )	범위에 표시할 라벨의 수직정렬입니다.
labelRotation	Number	표시할 라벨의 회전 각도입니다.  Embedded폰트를 사용할 경우에는 모든 각도 가능하나 시스템 폰트를 사용할 경우에는 0, 90만 가능합니다.
fill	Uint	범위안을 채울 색상을 지정합니다.
startValue	Number, String	표시하려는 범위의 위치 시작 값.
endValue	Number, String	표시하려는 범위의 위치 종료 값.
horizontal	Boolean( true, false )	수평범위 여부.  수평범위(true)이면 세로축을 기준으로 범위를 그리게 되며 false이면 수직범위를 그리게 됩니다.  true이면 수직좌표의 값이 들어가야 하며 false이면 수평좌표의 값이 들어가야 합니다.
label	String	범위 안에 넣을 텍스트를 정의하십시오.

<표 27 AxisRange 속성 및 유효값 설명>

## 7.11. 확대/축소와 마우스 이동에 따른 십자가 표시하기

확대/축소 및 십자기 표시 기능은 아래 리스트를 제외한 모든 차트에서 사용 가능합니다.

- 레이더 차트
- 파이, 도넛 차트
- 히스토리, 스크롤 차트
- 게이지 차트

확대/축소와 십자가 표시는 CrossRangeZoomer 노드의 정의로 사용이 가능합니다.

다음과같이 레이아웃을 작성하고 차트의 annotationElements 속성에 CrossRangeZoomer 를 설정하십시오.

```
<rMateChart>
<Combination2DChart showDataTips="true">
  <series>
    <Line2DSeries yField="Profit" displayName="Profit"/>
  </series>
  <annotationElements>
    <CrossRangeZoomer zoomType="horizontal" fontSize="11" color="0x000000"
      verticalLabelPlacement="bottom" enableZooming="true" enableCrossHair="true">
      <zoomRangeStroke>
        <Stroke weight="1" color="0xFF0000" alpha="0.3"/>
      </zoomRangeStroke>
      <zoomRangeFill>
        <SolidColor color="0x0000FF" alpha="0.2"/>
      </zoomRangeFill>
    </CrossRangeZoomer>
  </annotationElements>
</Combination2DChart>
</rMateChart>
```

<예제 78 확대/축소 및 십자가 표시 예제>

속 성	유효값	설 명
horizontalStroke	Stroke 객체	십자가의 가로선을 지정합니다.
verticalStroke	Stroke 객체	십자가의 세로선을 지정합니다.
zoomRangeStroke	Stroke 객체	줌 영역의 선 스타일을 지정합니다.
zoomRangeFill	SolidColor 객체	줌 영역의 채우기 색을 지정합니다.
horizontalLabelFormatter	포맷터 id	십자가의 가로선에 나타나는 라벨에 적용할 포맷터.
horizontalLabelOppFormatter	포맷터 id	십자가의 가로선, 오른쪽에 나타나는 라벨에 적용할 포맷터(useDualCrossXLabel="true" 인 경우 유효)
horizontalLabelPlacement	left  right(기본값:left)	십자가의 가로선 라벨의 위치를 나타냅니다. left일 경우 Chart 왼쪽에 right일 경우 오른쪽에

		좌표값이 표시됩니다.
showValueLabels	true  false (기본값:true)	십자가의 라벨에 표시되는 좌표 값 출력 여부입니다. (현재 마우스가 위치한 값 표시여부.)
verticalLabelFormatter	포맷터 id	십자가의 세로선에 나타나는 라벨에 적용할 포맷터.
verticalLabelOppFormatter	포맷터 id	십자가의 세로선, 왼쪽에 나타나는 라벨에 적용할 포맷터(useDualCrossYLabel="true" 인 경우 유효)
verticalLabelPlacement	top bottom (기본값:bottom)	십자가의 세로선 라벨의 위치를 나타냅니다. top일 경우 Chart 위에 bottom일 경우 아래에 좌표값이 표시됩니다.
useDualCrossXLabel	false true (기본값:false)	수직축을 2개 이상 설정한 경우 2개의 축에 대하여 라벨을 표시할지 여부를 나타냅니다. 즉, useDualCrossXLabel 를 true 로 설정한 경우, 수직축에 해당되는 라벨이 축 위치에 맞게 표시됩니다. (horizontalLabelPlacement 속성은 무시됨) 예를 들어 수직축을 3개 설정하였다면 앞에서 2개의 축에 대한 라벨만 표시됩니다.
useDualCrossYLabel	false true (기본값:false)	수평축을 2개 이상 설정한 경우 2개의 축에 대하여 라벨을 표시할지 여부를 나타냅니다. 즉, useDualCrossYLabel 를 true 로 설정한 경우, 수평축에 해당되는 라벨이 축 위치에 맞게 표시됩니다. (verticalLabelPlacement 속성은 무시됨) 예를 들어 수평축을 3개 설정하였다면 앞에서 2개의 축에 대한 라벨만 표시됩니다.
enableCrossHair		마우스 포인터를 기준으로 십자가 사용 여부를 나타냅니다.
enableZooming		Chart의 Zoom 기능 사용 여부를 나타냅니다.
zoomType	horizontal, vertical, both (기본값:both)	Chart의 Zoom 기능 사용 시 zoom 기능을 수직축, 수평축, 모두에 적용할지를 지정합니다. 예를 들어 zoomType 을 "horizontal" 로 지정하고 zoom in 을 수행했다면 수직축에 대해서는 zoom in 을 적용하지 않고 수평축에 대해서만 기능을 수행합니다.
resetMode	initial, auto(기본값:auto)	줌 기능에서 Restore 실행 방식을 결정합니다. initial 인 경우 사용자에게 의해 결정된 최초의 minimum, maximum 값을 유지하고, auto 인 경우 차트에 의해 minimum, maximum 이 정해집니다. 개발자에 의해 축의 minimum, maximum 설정 값

		이 유지되도록 하기 위해서는 initial 값을 할당하세요.
--	--	-----------------------------------

<표 28 CrossRangeZoomer 속성 및 유효값 설명표>

## 7.12. 차트 안에 메모 표시하기.

차트 안에 메모표시는 특정 데이터 기준이 아닌 차트 안쪽의 x,y 좌표계를 이용하여 메모를 표시하게 됩니다. 차트 위에 메모를 출력하려면 차트의 annotationElements 속성을, 차트 밑에 메모를 출력하려면 backgroundElements 속성을 사용하십시오.

본문의 예제는 차트 위에 메모를 표시해 보도록 하겠습니다.

```
<rMateChart>
  <Line2DChart showDataTips="true">
    <series>

    .... 라인 시리즈 설정 부분 중략

  </series>
  <annotationElements>

    <CanvasElement> <!--그룹으로 설정할 라벨들을 묶습니다. -->

      <!--라벨 1-->
      <Label left="0" top="20" width="200" height="20" color="0x0000FF" fontSize="15" textAlign="center"
        text="텍스트 쓰기 1(왼쪽)" backgroundAlpha="0" borderStyle="solid" borderColor="0xFF0000">
      </Label>
      <!--라벨 2-->
      <Label width="200" height="40" color="0x0000FF" fontSize="15" textAlign="center" verticalCenter="0"
        horizontalCenter="0" text="텍스트 쓰기 2&#xa;(차트 중앙 표시)" backgroundColor="0xFFFF00"
        backgroundAlpha="0.2" borderStyle="solid" borderColor="0xFF0000">
      </Label>

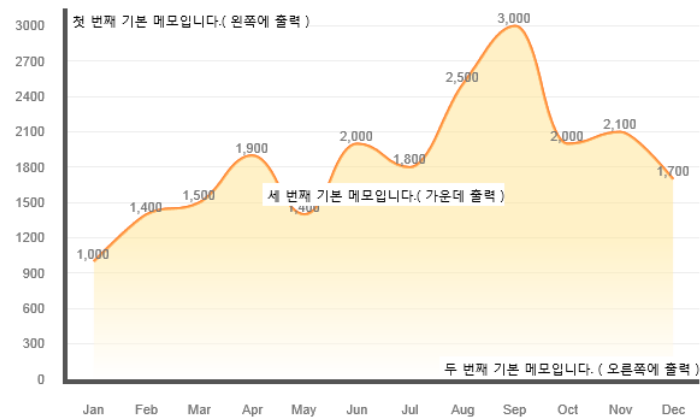
      <!--라벨 3-->
      <Label right="0" bottom="0" height="40" color="0xFF0000" fontSize="15" textAlign="center"
        text="텍스트 쓰기 3(오른쪽)" backgroundAlpha="0" borderStyle="none">
      </Label>

      <Label left="50" top="100" height="40" color="0x009966" fontSize="12" textAlign="center" text="임의의/
        X,Y 좌표(50,100)에 표시" backgroundAlpha="0" borderStyle="none">
      </Label>
    </CanvasElement>
```

```
</annotationElements>
</Line2DChart>
</rMateChart>
```

<예제 79 차트 안에 메모 표시 예제>

위처럼 정의한 메모들이 출력된 모습은 아래와 같습니다.



<그림 55 차트 안에 메모 표시한 모습>

- 라벨의 속성 및 유효값에 대한 설명은 “5.3.1 차트 제목(Caption), 부제목(SubCaption) 넣기.”를 참고하여 주십시오.

다음은 라벨의 고유 속성(즉, 제목, 부제목 속성이 아닌 라벨 속성)을 나타낸 표입니다.

속 성	유효값	설 명
backgroundColor	RGB 컬러	바탕의 색을 지정합니다.
backgroundAlpha	0~1 실수	바탕색 투명도를 지정합니다.
borderColor	RGB 컬러	테두리 선 색깔을 지정합니다.
borderAlpha	0~1 실수	테두리 선 투명도를 지정합니다.
borderThickness	Number(픽셀 단위)	테두리 두께를 지정합니다.
cornerRadius	Number	테두리 코너를 둥그렇게 합니다.
borderStyle	"solid", "none"	테두리 선을 넣을지 결정합니다.

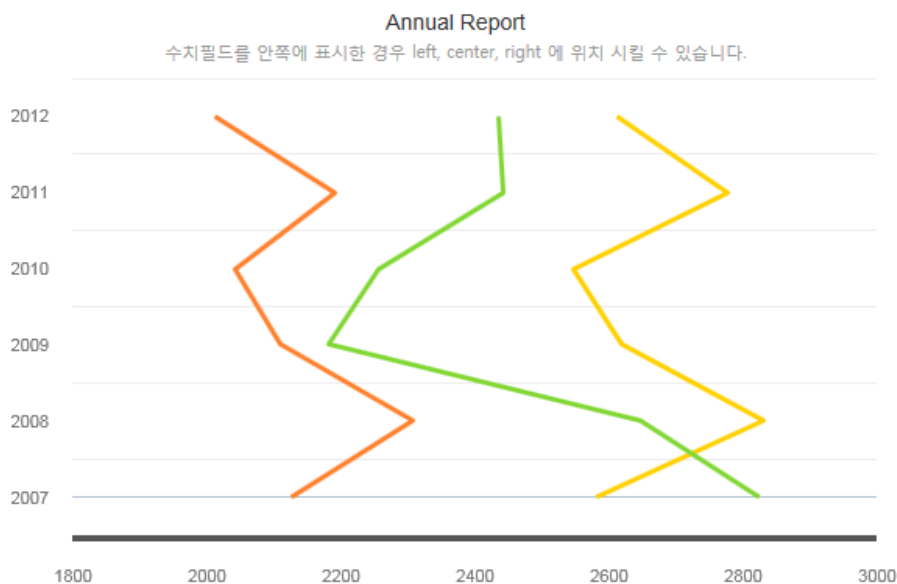
<표 29 차트에 라벨 표시 속성 유효값 및 설명>

### 7.13. 라인차트에서 수직선으로 데이터 출력하기.

라인차트는 기본적으로 수평축을 수치데이터로 삼고 수평방향으로 선 그래프가 진행되는 방식입니다. 라인차트의 선이 수직방향으로 수치 데이터를 표현하고자 할 때는 다음과 같이 하십시오.

- Line2DChart 의 horizontalAxis 의 카테고리축을 verticalAxis 로 변경하십시오.
- Line2DSeries 의 xField 와 yField 의 값을 반대로 하십시오.
- Line2DSeries 의 sortOnXField 를 false 로 설정합니다.

```
<rMateChart>
  <Line2DChart>
    <verticalAxis>
      <CategoryAxis categoryField="Month"/>
    </verticalAxis>
    <series>
      <Line2DSeries xField="Profits" yField="Month" sortOnXField="false" displayName="Profits">
      </Line2DSeries>
      <Line2DSeries xField="Costs" yField="Month" sortOnXField="false" displayName="Costs">
      </Line2DSeries>
      <Line2DSeries xField="Revenue" yField="Month" sortOnXField="false" displayName="Revenue">
      </Line2DSeries>
    </series>
  </Line2DChart>
</rMateChart>
```



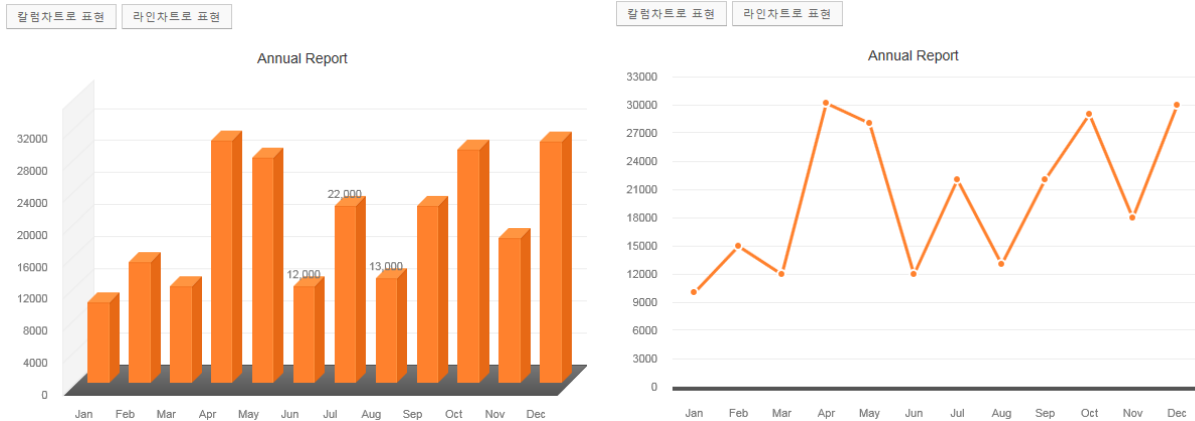
<예제 80 라인차트에 데이터 수직으로 출력 예제>

#### 7.14. 동적으로 차트 레이아웃 또는 데이터 변경하기.

때때로 이미 출력된 차트의 데이터를 동적으로 변경시키거나 칼럼차트로 표현된 데이터를 라인차트나 바차트로 표현해야 할 경우가 있습니다. 이런 경우 rMate 차트는 웹 페이지를 갱신하지 않고 동적으로 차트에 데이터나 레이아웃을 변경시킬 수 있습니다. 아래 예제는 기본적으로 데이터 1 을 라인차트로 출력합니다. Html 문서 안의 스크립트에서 만든 버튼으로 칼럼차트로 데이터 1 을 표현하거나 데이터 2 를 표



현하는 방법을 나타낸 예제입니다. rMateChartH5 js 파일은 생성하고 하는 차트에 맞는 js 파일을 선택해야 합니다. 그러나 이와 같이 동적으로 레이아웃을 변경하였을 때 그에 맞는 차트를 생성해야 하기 때문에 rMateIntegrationH5.js 를 활용합니다. 통합 차트는 모든 차트를 생성할 수 있습니다.



<그림 56 동적으로 레이아웃을 바꿔 출력한 결과>

동적으로 바뀌는 차트를 작성하는 방법을 제시한 html 문서는 아래와 같습니다.

```
<html lang="ko">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=euc-kr" />
<script src="rMateChartH5.js" language="javascript"></script>
<!-- 사용자 정의 설정 시작 -->
<script language="JavaScript" type="text/javascript">

// -----차트 chartVars 설정 시작-----
// rMate 차트 생성 준비가 완료된 상태 시 호출할 함수를 지정합니다.
var chartVars = "&rMateOnLoadCallFunction=rMateChartOnLoad";
// -----차트 chartVars 설정 끝-----

// rMateChart 를 생성합니다.
rMateChartH5.create("chart1", "chartHolder", chartVars, "100%", "100%");

// rMate 차트 준비가 완료된 경우 이 함수가 호출됩니다.
// 사용자는 이 함수를 chartVars에 반드시 등록해야 합니다.
// 여기서 작업을 하십시오.
// 차트 콜백함수 4개 존재합니다.
// 1. setLayout - 스트링으로 작성된 레이아웃 XML을 삽입합니다.
// 2. setData - 배열로 작성된 데이터를 삽입합니다.
// 3. setLayoutURL - 레이아웃 XML 경로를 지시합니다.
// 4. setDataURL - 데이터 XML 경로를 지시합니다.
```

// 아래 예제는 가능한 방법을 제시한 것 입니다.

// 현재 2번 차트는 레이아웃을 스트링 형태로 받아들이고 데이터는 배열 형태로 받아들입니다.

```
function rMateChartOnLoad()
{
    //예제1. 레이아웃 스트링 형태,데이터 배열 형태
    chart.setLayout(layoutStr);
    chart.setData(chartData);

    //예제2. 레이아웃 XML URL경로, 데이터 배열 형태
    //chart.setLayoutURL(layoutURL);
    //chart.setData(chartData);

    //예제3. 레이아웃 스트링 형태, 데이터 XML URL경로
    //chart.setLayout(layoutStr);
    //chart.setDataURL(dataURL);

    //예제4. 레이아웃 URL, 데이터 URL
    //chart.setLayoutURL(layoutURL);
    //chart.setDataURL(dataURL);
}

// 동적으로 할당할 레이아웃 정의.
var layoutURL = "./LayoutXml/Column_3D_Layout.xml";
var layoutStr = '<rMateChart cornerRadius="12">'
    + '<Options>'
    + '<Caption text="Annual Report"/>'
    + '</Options>'
    + '<NumberFormatter id="numfmt" useThousandsSeparator="true"/>'
    + '<Line2DChart showDataTips="true">'
    + '<horizontalAxis>'
    + '<CategoryAxis categoryField="Month"/>'
    + '</horizontalAxis>'
    + '<verticalAxis>'
    + '<LinearAxis interval="3000" formatter="{numfmt}"/>'
    + '</verticalAxis>'
    + '<series>'
    + '<Line2DSeries yField="Profit" displayName="Profit" '
    + 'itemRenderer="CircleItemRenderer" radius="5">'
    + '<showDataEffect>'
    + '<SeriesInterpolate/>'
    + '</showDataEffect>'
    + '</Line2DSeries>'
    + '</series>'
    + '</Line2DChart>'
```

```

+ '</rMateChart>'

var chartData = [{"Month": "Jan", "Revenue": 10000, "Cost": 5000, "Profit": 5000},
  {"Month": "Feb", "Revenue": 15000, "Cost": 7000, "Profit": 8000},
  {"Month": "Mar", "Revenue": 12000, "Cost": 6000, "Profit": 6000},
  {"Month": "Apr", "Revenue": 30200, "Cost": 4000, "Profit": 26200},
  {"Month": "May", "Revenue": 28000, "Cost": 10000, "Profit": 18000},
  {"Month": "Jun", "Revenue": 12000, "Cost": 5000, "Profit": 7000},
  {"Month": "Jul", "Revenue": 22000, "Cost": 10000, "Profit": 12000},
  {"Month": "Aug", "Revenue": 13000, "Cost": 6000, "Profit": 7000},
  {"Month": "Sep", "Revenue": 22000, "Cost": 10000, "Profit": 12000},
  {"Month": "Oct", "Revenue": 29000, "Cost": 8000, "Profit": 21000},
  {"Month": "Nov", "Revenue": 18000, "Cost": 7500, "Profit": 10500},
  {"Month": "Dec", "Revenue": 30000, "Cost": 12000, "Profit": 18000} ];

// 레이아웃 변경 함수
function changeLayout()
{
  chart.setLayoutURL(layoutURL);
}

// 레이아웃 변경 함수.
function changeLayout3()
{
  chart.setLayout(layoutStr);
}

</script>
<!-- 사용자 정의 설정 끝 -->
</head>

<body>
<table>
  <tr>
    <td>
      <div class="button button_top" onclick="changeLayout()">컬럼차트로 표현</div>
      <div class="button button_top" onclick="changeLayout3()">라인차트로 표현</div>
    </td>
  </tr>
  <tr>
    <td>
      <div class="content">
        <!-- 차트가 삽입될 DIV -->
        <div id="chartHolder" style="width:600px; height:400px;">

```

```

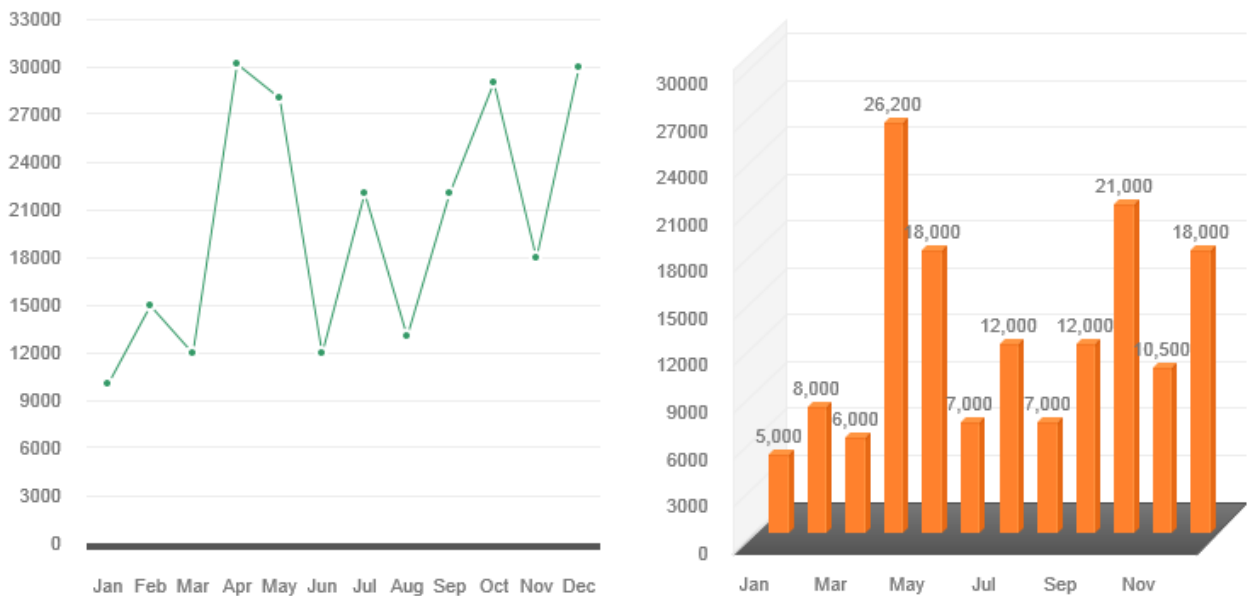
        </div>
    </div>
</td>
</tr>
</table>
</body>
</html>

```

<예제 81 동적으로 차트 레이아웃과 데이터를 변경시키는 예제>

## 7.15. 하나의 html 문서 안에 여러 개의 차트를 생성하기

Html 문서 한 페이지에 여러 개의 차트를 생성하고자 할 때는 아래 예제를 참고하세요. 아래 예제는 한 페이지에 2 개의 차트를 생성시키는 방법을 제시합니다.



<그림 57 하나의 html 문서에서 2 개의 차트 생성 결과>

이 예제는 라인차트와 칼럼차트를 생성 시키는 예제입니다. 모든 차트를 생성시킬 수 있는 rMateIntegration.js 를 활용하십시오. 라인과 칼럼 차트는 기본적인 차트이므로 rMateChart.js 를 사용하셔도 무관합니다.

```

<html lang="ko">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=euc-kr" />

<script src="rMateChartH5.js" language="javascript"></script>

```

```

<!-- 사용자 정의 설정 시작 -->
<script language="JavaScript" type="text/javascript">

// -----첫번째 차트 chartVars 설정 시작-----
var chartVars = "&rMateOnLoadCallFunction=rMateChartOnLoad";
// -----첫번째 차트 chartVars 설정 끝-----

// -----두번째 차트 chartVars 설정 시작-----
var chartVars2 = "&rMateOnLoadCallFunction=rMateChartOnLoad2";
// -----두번째 차트 chartVars 설정 끝-----

// rMateChart 를 생성합니다.
rMateChartH5.create("chart1", "chartHolder", chartVars, "100%", "100%");
rMateChartH5.create("chart2", "chartHolder2", chartVars2, "100%", "100%");

// rMate 차트 준비가 완료된 경우 이 함수가 호출됩니다.
// 사용자는 이 함수를 chartVars에 반드시 등록해야 합니다.
// 여기서 작업을 하십시오.
// 차트 콜백함수 4개 존재합니다.
// 1. setLayout - 스트링으로 작성된 레이아웃 XML을 삽입합니다.
// 2. setData - 배열로 작성된 데이터를 삽입합니다.
// 3. setLayoutURL - 레이아웃 XML 경로를 지시합니다.
// 4. setDataURL - 데이터 XML 경로를 지시합니다.
// 아래 예제는 가능한 방법을 제시한 것 입니다.
// 현재 1번 차트는 레이아웃을 스트링 형태로 받아들이고 데이터는 배열 형태로 받아들입니다.
function rMateChartOnLoad()
{
    //예제1. 레이아웃 스트링 형태,데이터 배열 형태
    chart1.setLayout(layoutStr);
    chart1.setData(chartData);

    //예제2. 레이아웃 XML URL경로, 데이터 배열 형태
    //chart1.setLayoutURL(layoutURL2);
    //chart1.setData(chartData2);

    //예제3. 레이아웃 스트링 형태, 데이터 XML URL경로
    //chart1.setLayout(layoutStr2);
    //chart1.setDataURL(dataURL2);

    //예제4. 레이아웃 URL, 데이터 URL
    //chart1.setLayoutURL(layoutURL);

```

```

    //chart1.setDataURL(dataURL);
}

// 2번 차트에서 차트가 생성되면 호출하는 함수(사용자에 의해 정의됨)
function rMateChartOnLoad2()
{
    //예제2. 레이아웃 XML URL경로, 데이터 배열 형태
    chart2.setLayoutURL(layoutURL2);
    chart2.setData(chartData2);
}

// 동적으로 할당할 레이아웃 정의.
var layoutURL2 = encodeURIComponent("chartLayout.xml");
var layoutStr = "<rMateChart cornerRadius='12' borderStyle='solid'>"
    + "<Options><Caption text='Annual Report'/></Options>"
    + "<Line2DChart showDataTips='true'>"
    + "<horizontalAxis><CategoryAxis categoryField='Month'/></horizontalAxis>"
    + "<series><Line2DSeries yField='Profit' displayName='Profit'>"
    + "<showDataEffect><SeriesInterpolate/></showDataEffect>"
    + "<lineStroke><Stroke color='0xFF0000'weight='4'/>"
    + "</lineStroke></Line2DSeries>"
    + "</series></Line2DChart></rMateChart>";

// 동적으로 할당할 데이터 정의.
var chartData = [{ "Month": "Jan", "Revenue": 10000, "Cost": 5000, "Profit": 5000 },
    { "Month": "Feb", "Revenue": 15000, "Cost": 7000, "Profit": 8000 },
    { "Month": "Mar", "Revenue": 12000, "Cost": 6000, "Profit": 6000 },
    { "Month": "Apr", "Revenue": 30200, "Cost": 4000, "Profit": 26200 },
    { "Month": "May", "Revenue": 28000, "Cost": 10000, "Profit": 18000 },
    { "Month": "Jun", "Revenue": 12000, "Cost": 5000, "Profit": 7000 },
    { "Month": "Jul", "Revenue": 22000, "Cost": 10000, "Profit": 12000 },
    { "Month": "Aug", "Revenue": 13000, "Cost": 6000, "Profit": 7000 },
    { "Month": "Sep", "Revenue": 22000, "Cost": 10000, "Profit": 12000 },
    { "Month": "Oct", "Revenue": 29000, "Cost": 8000, "Profit": 21000 },
    { "Month": "Nov", "Revenue": 18000, "Cost": 7500, "Profit": 10500 },
    { "Month": "Dec", "Revenue": 30000, "Cost": 12000, "Profit": 18000 } ];

var chartData2 = [{ "Month": "Jan", "Revenue": 1000, "Cost": 500, "Profit": 500 },
    { "Month": "Feb", "Revenue": 1500, "Cost": 700, "Profit": 800 },
    { "Month": "Mar", "Revenue": 1200, "Cost": 600, "Profit": 600 },
    { "Month": "Apr", "Revenue": 3020, "Cost": 400, "Profit": 2620 },
    { "Month": "May", "Revenue": 2800, "Cost": 1000, "Profit": 1800 },
    { "Month": "Jun", "Revenue": 1200, "Cost": 500, "Profit": 700 },

```

```

    {"Month": "Jul", "Revenue": 2200, "Cost": 1000, "Profit": 1200},
    {"Month": "Aug", "Revenue": 1300, "Cost": 600, "Profit": 700},
    {"Month": "Sep", "Revenue": 2200, "Cost": 1000, "Profit": 1200},
    {"Month": "Oct", "Revenue": 2900, "Cost": 800, "Profit": 2100},
    {"Month": "Nov", "Revenue": 1800, "Cost": 750, "Profit": 1050},
    {"Month": "Dec", "Revenue": 3000, "Cost": 1200, "Profit": 1800} ];

</script>
<!-- 사용자 정의 설정 끝 -->
</head>

<body>
<table align="center" border="0" cellpadding="0" cellspacing="0">
    <tr>
        <td align="center">
            <div class="content">
                <!-- 차트가 삽입될 DIV -->
                <div id="chartHolder" style="width:600px; height:400px;">
                </div>
            </div>
        </td>
        <td align="center">
            <div class="content">
                <!-- 차트가 삽입될 DIV -->
                <div id="chartHolder2" style="width:600px; height:400px;">
                </div>
            </div>
        </td>
    </tr>
</table>
</body>
</html>

```

<예제 82 하나의 html 문서에 2개의 차트 생성하는 예제>

## 7.16. 실시간 차트 예제 – 주식 모니터링 차트

실시간 차트의 가장 일반적인 예제인 주식 모니터링 예제입니다. 이 레이아웃은 기본으로 제공되는 레이아웃입니다.

```

<rMateChart backgroundColor="0xFFFFEE" cornerRadius="12" borderStyle="solid">
    <Options>
        <Caption text="Stock Monitoring" />

```

```

        <SubCaption text="매 3초 단위로 업데이트합니다.(데이터는 랜덤)" fontSize="11"
textAlign="right"/>
        <Legend/>
    </Options>
    <DateFormatter id="dateFmt" formatString="HH시 NN분 SS초" /> //포맷터 정의
    //실시간 차트는 type은 time, 60초동안 3초단위의 데이터 표현
    <RealTimeChart id="chart" dataDisplayType="time" timePeriod="60" interval="3" showDataTips="true">
        //뒷 배경에 격자 모양을 그림.
        <backgroundElements>
            <GridLines direction="both"/>
        </backgroundElements>
        //수평축으로 DateTime축 사용, data와 축라벨은 모두 초(second)단위이고
        //데이터는 3초단위로 들어오며, 축라벨 시간 간격은 9초. GMT시간이 아닌 로컬시간표시
        <horizontalAxis>
            <DateTimeAxis id="hAxis" dataUnits="seconds" labelUnits="seconds" dataInterval="3"
interval="9" formatter="{dateFmt}" displayLocalTime="true"/>
        </horizontalAxis>
        <series>
            <Column2DSeries yField="Volume" xField="Time"
                displayName="Trading Volume"
                itemRenderer="GradientColumnItemRenderer">
                <fill> //칼럼 시리즈의 채우기 색 지정
                    <SolidColor color="0xB0C759"/>
                </fill>
                <verticalAxis> 칼럼 시리즈가 참고하는 축 정의
                    <LinearAxis id="vAxis1" title="Volume" minimum="0"
maximum="10000"/>
                </verticalAxis>
            </Column2DSeries>
            <Line2DSeries xField="Time" yField="Price" displayName="Stock Price"
                itemRenderer="CircleItemRenderer" radius="5" fill="0xFFFFF0">
                <stroke> //라인시리즈에서 데이터가 있는 곳의 표시를 나타내는 도형의 선
                    모양 정의
                    <Stroke color="0xE48701" weight="2"/>
                </stroke>
                <lineStroke> //라인시리즈의 선 모양 정의
                    <Stroke color="0xE48701" weight="4"/>
                </lineStroke>
                <verticalAxis> //라인시리즈가 참고하는 축 정의
                    <LinearAxis id="vAxis2" title="Price" minimum="0"
maximum="1000"/>
                </verticalAxis>
            </Line2DSeries>
        </series>
    </RealTimeChart>

```

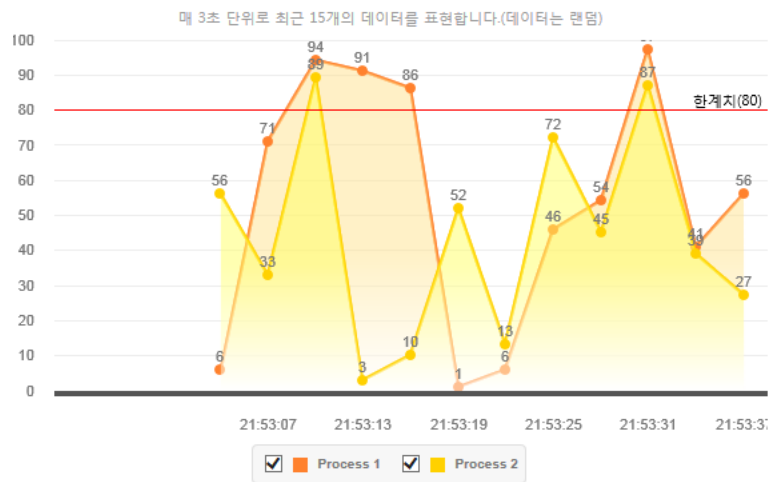


```

<verticalAxisRenderers> //두개의 축의 위치와 참고 축 정의.
    <Axis2DRenderer placement="left" axis="{vAxis1}">
    </Axis2DRenderer>
    <Axis2DRenderer placement="right" axis="{vAxis2}">
    </Axis2DRenderer>
</verticalAxisRenderers>
</RealTimeChart>
// 실제로 데이터를 폴링하는 RPC 정의.
<HttpServiceRepeater url="http://demo.riamore.net/chartTest/data4.jsp" target="{chart}" interval="3"
method="get"/>
</rMateChart>

```

<예제 83 실시간 차트 주식 모니터링 레이아웃>



<그림 58 실시간차트 - 주식 모니터링>

## 7.17. 차트의 이미지 데이터 가져오기

화면에 출력된 차트화면을 base64 인코딩 된 데이터로 가져올 수 있는 기능입니다.

HTML5 차트에서의 이미지 데이터는 HTML5 의 Canvas 에서 제공하는 toDataURL()을 사용합니다. 따라서 **HTML5 를 지원하지 않는 IE7,8 과 문서모드가 쿼크모드일 경우는 이미지 저장기능은 제공되지 않습니다.** 차트에 대한 이미지 데이터를 가져오는 것은 다음과 같이 사용합니다.

```
function snapShot(){
    var base64src = document.getElementById("chart1").saveAsImage();
    document.getElementById("img").src = base64src;
}
```

위와 같이 이미지 데이터를 가져올 차트의 id 에 접근하여 해당 차트의 saveAsImage()함수를 실행 합니다. 위 함수를 실행하면 변수 base64src 에는 차트화면을 base64 로 인코딩된 데이터가 저장 되어집니다.

**이미지 데이터를 가져올 경우 차트가 출력되어있는 도메인과 이미지를 가져오는 도메인이 다를 경우 Canvas 자체적으로 보안에러를 발생하여 올바른 이미지 데이터를 가져오지 못할 수 있습니다.**

### 7.18. 차트의 이미지 서버로 전송

차트의 이미지 서버로 전송은 차트의 이미지데이터를 가져와 해당 이미지를 서버로 전송하고 서버페이지에서는 받은 이미지 데이터를 토대로 이미지 파일을 만들어 서버에 저장 하는 방식입니다.

서버로 이미지 전송도 차트에 대한 이미지 데이터를 가져와야하기 때문에 7.17 과 같이 **HTML5 가 지원 되지 않는 브라우저와 문서모드가 쿼크모드일 경우에는 지원하지 않습니다.**

```
// 차트의 스냅샷을 base64인코딩 형태로 얻습니다.
function getSanpshot(){
    // base64로 인코딩 된 차트 이미지를 리턴합니다.
    var base64src = document.getElementById("chart1").getSnapshot();
    var data = document.getElementById("data");
    var extension = document.getElementById("extension");
    // 확장자
    extension.value = "png";
    // 이미지 소스
    data.value = base64src
    // 서버 사이드에서는 base64인코딩을 decoding하여 파일 쓰기 액션을 취합니다.
    document.getElementById("sumForm").submit();
}
```

위 소스는 javascript 로 차트의 이미지 데이터를 받고 서버로 보내는 일련의 과정입니다.

위 예제에는 나와있지 않는 폼요소 들이 존재합니다. 해당 예제는

저희 데모 페이지 -> Chart Properties -> 이미지 전송 -> 서버로 이미지 전송 페이지를 살펴보면 되겠습니다.

### 7.19. 모바일에서의 이미지 데이터 가져오기

HTML5 가 지원되는 모든 브라우저는 차트의 이미지 데이터를 가져올 수 있습니다. 이것은 모바일도 예외가 아닙니다. **다만 모바일 기기의 브라우저 버전에 따라 차이가 발생 할 수 있습니다. 안드로이드**

진저브레드 이하 버전의 기본 브라우저는 이미지 데이터를 가져올 수 없습니다. 그러나 이미지 데이터를 가져오는 기능을 지원하는 다른 브라우저로 실행한다면 해당 기능은 올바르게 작동 됩니다. 기본적으로 아이스크림 이상 버전에서는 기본 브라우저에서도 올바르게 작동됩니다.

모든 모바일 기기에서도 일반 PC 용 웹처럼 차트 이미지를 이미지 데이터로 변환하여 서버에 이미지를 전송 할 수 있습니다.

저희 모바일 샘플 <http://demo.riamore.net/m/chart/Samples/GetSnapshot.html> 통해서 확인 하실 수 있습니다.

## 8. 장애인을 위한 기능

### 8.1. 시각 장애인을 위한 대체 텍스트

시각장애인으로, 차트를 보지 못하는 분을 위하여 대체 텍스트를 제공합니다.

차트에 데이터를 삽입하였을 경우 차트 내에서 삽입된 데이터를 알아내어 대체 텍스트를 저장합니다.

대체 텍스트를 읽어 줄 수 있는 프로그램이나 기기들을 사용하여 대체 텍스트를 읽어주게 됩니다.

**프로그램이나 기기와 관련하여서는 (주)리아모어소프트에서 취급하거나 제공하는 부분은 없습니다.**

이 기능은 사용자가 특별히 조작하거나 수정해야할 부분은 없습니다.

```
chartVars += "&accessibility=true";
```

차트 설정시 chartVars 로 위와 같이 설정하여 주십시오.

### 8.2. 색맹이나 색약이신 분을 위한 패턴지원

색맹이나 색약인 분들은 차트의 색상을 구분하기 어렵습니다.

이 때문에 차트 출력시 색상이 아닌 틀정 모양을 출력하여 차트를 구분할 수 있게 도와주는 기능입니다.

지원하고 있는 패턴 모양은 20 가지이며 이는 사용자가 변경 하실 수 있습니다.

패턴을 사용하시려면 아래와 같이 설정하십시오

// 웹 접근성 패턴을 사용하시려면 usePattern을 true로 설정하여야 합니다.

```
chartVars += "&usePattern=true";
```

위 내용을 설정하시면 차트내부에서 usePattern 값을 확인 후 패턴 이미지를 불러올 경로에서 이미지를 로딩하게 됩니다. 로딩되는 패턴 이미지의 경로는 아래와 같이 수정 할 수 있습니다.

```
// 패턴 이미지를 불러올 폴더 url
rMateChartH5.patternImageUrl = "../Images/pattern/";

// 폴더 url을 기준으로 폴더 안의 패턴 이미지 파일 url
rMateChartH5.patternImageUrl = [
    "diagonal_ltr.png",
    "diagonal_rtl.png",
    "diagonal.png"
    ....
];
```

위 내용은 기본값입니다.

패턴 적용 이후에 사용자 조작에 의해 패턴을 적용 / 해제 시킬 수 있습니다.

```
document.getElementById("chart1").accessibilityPattern(true / false);
```

차트 객체에 접근하여 accessibilityPattern 이라는 함수를 실행합니다.

해당 함수로 적용되는 값은 Boolean 값이며 true 는 패턴의 적용 false 는 패턴의 해제입니다.

## 9. 테마 ( Theme )

테마는 차트의 외형에 대한 데이터를 저장해두었다가 실행시 저장해 두었던 차트 외형 데이터 내용을 가져와 차트에 적용하여 차트의 모양을 변경하는 기능입니다.

제공되는 테마 종류는 simple, cyber, modern, lovely, pastel, old 가 있습니다.

Old 는 rMateChartH5 2.5 이하 버전을 위한 테마입니다.

### 9.1. rMateChartH5 에서 제공하는 테마 등록하기

rMateChartH5에서 제공하는 theme.js를 사용하는 html에 include 합니다.

```
// rMateChartH5 테마 js
// theme.js는 Samples디렉토리 안에 있습니다.
```

```
// theme.js는 rMateChartH5 차트 라이브러리 include보다 뒤에 선언 되어야 합니다.  
<script type="text/javascript" src="./theme.js"></script>  
  
// theme.js에서 미리 만들어 둔 테마 변수 rMateChartH5.themes를 rMateChartH5에 등록 합니다.  
rMateChartH5.registerTheme(rMateChartH5.themes);
```

## 9.2. 제공되는 테마 적용하기

위에서 설명한 제공되는 테마 종류 simple, cyber, modern, lovely, pastel, old를 적용합니다.

```
// "chart1" 객체에 접근하여 simple이란 파라미터를 가지고 setTheme함수를 실행합니다.  
document.getElementById("chart1").setTheme("simple");  
// document.getElementById("chart1").setTheme("pastel");
```

## 9.3. 기본 테마로 돌아가기

특정 테마 적용 후 기본 테마로 돌아가고 싶을 경우에는 파라미터로 "default"를 설정합니다.

```
// 파라미터로 default를 설정하고 setTheme함수를 실행하면 기본 테마로 돌아가게 됩니다.  
document.getElementById("chart1").setTheme("default");
```

## 9.4. 테마 삭제하기

등록해둔 테마를 해당 페이지에서 삭제 하실 수 있습니다.

"simple"이라는 테마를 삭제 할 경우 해당 페이지 내에서 "simple"이라는 테마를 적용하고 있는 모든 차트들이 기본 테마로 돌아오게 됩니다.

"simple"테마가 적용되어져 있는 차트가 없다면 등록된 테마에서 "simple"만 삭제 되어집니다.

```
// 파라미터 "simple"을 가지고 removeTheme를 실행합니다.  
// simple 테마가 적용된 모든 차트들이 기본 테마로 돌아가게 됩니다.  
rMateChartH5.removeTheme("simple");
```

## 10. rMateChart for Flash 와 rMateChart for HTML5 의 차이점 설명

### 10.1. 지원하지 않는 기능

rMateChart for Flash(이하 플래시차트) 와 rMateChart for HTML5(이하 HTML5 차트) 는 기본적으로 레이아웃 형식 및 데이터 형식이 같습니다. 따라서 플래시 차트에서 사용한 레이아웃 및 데이터를 그대로 HTML5 차트에서 사용하여 차트를 출력 시킬 수 있습니다.

단, 플래시 차트의 몇 가지 속성 및 기능이 HTML5 차트에서 제외되었습니다. 다음은 HTML5 차트에서 제외된 기능 및 속성에 대한 리스트입니다.

플래시 차트 사용자가 HTML 5 차트로 변환하고자 하는 경우 참고하여 주십시오.

플래시 차트와 비교하여 rMateChart for HTML5 에서 지원되지 않는 기능은 다음과 같습니다.

- SideBar 기능
- 시리즈에 그림자 필터와 같이 필터 넣는 기능

아래 표는 레이아웃에서 정의하는 구체적인 태그의 속성 중 지원되지 않는 기능에 대한 대체 기능을 설명한 표입니다.

태그 명	속성 명	설명	대체 기능 설명
Axis2DRenderer, Axis3DRenderer	targetValue, targetSecondValue, targetValueName, targetSecondValueName targetValueStyleName targetSecondValueStyleName fillTargetArea targetAreaFill targetLineStroke, targetSecondLineStroke	축에 상하한선 긋기 기능이 Axis2D,3DRender 에서 제외됨.	AxisLine 태그를 활용하여 같은 기능 사용 가능합니다.
ScrollableAxis Renderer	scrollSensitivity	HTML5 퍼포먼스 관계로 제외됨.	없음,
모든 차트	seriesFilters	시리즈 그림자 삽입 기능으로 제외됨	없음
	fill	차트 영역 색상 넣기 제외됨	rMateChart 태그의

			backgroundColor 속성 사용하여 동일한 효과 가능
	selectionMode	4.0 버전 이후로 "single"만 가능 (IE7, 8미지원)	없음
	Width	항상 100% 적용시킴	없음
	Height	항상 100% 적용시킴	없음
모든 태그	textIndent	텍스트 들여쓰기 기능 제외됨	없음
모든 Series	buttonMode	데이터 포인트에 마우스-오버 시 커서 손모양 변환 속성	Chart 의 속성으로 변경됨, 즉, Chart 의 속성으로 변함
	adjustedRadius	데이터 포인트 선택 시 강조 효과 제외됨	없음
HistoryRange Selector	selectorInitialize	데이터 변경 시 히스토리 셀렉터 초기화 시킴	항상 true

<표 30 rMateChart for Flash 와 rMateChart for HTML5 의 차이점 >

**이 외에 IE7, 8 에서 인쇄를 할 경우 올바르게 인쇄가 되지 않을 수 있습니다.**

IE7,8 브라우저에서 출력되는 것은 완벽히 출력이 되지만 인쇄시에는 브라우저에서 보여지는것과는 다르게 출력이 될 수 있습니다. 예를 들어 히스토리 차트의 경우 인쇄시 차트의 아래부분의 Div 들이 투명처리가 되지 않은채 인쇄가 되어집니다. 이는 IE7, 8 의 인쇄시의 버그로 올바르게 출력되지 않고 있습니다.

## 10.2. 플래쉬 차트의 리사이즈와 HTML5 차트의 리사이즈 차이점

플래쉬 차트에서 width 와 height 크기를 "100%"로 설정 하였을 경우 브라우저 창을 늘리고 줄이면 자동적으로 차트가 커지고 줄어드게 됩니다. 하지만 HTML5 차트에서는 차트의 크기가 고정되어 있습니다. 처음 그려지는 차트 부모의 Div 크기와 그 부모의 Div 크기 안에서 차트의 크기 만큼 차트를 출력하게 됩니다. 이것을 플래쉬 차트와 마찬가지로 브라우저 창을 늘리고 줄였을 경우 차트의 크기도 줄어들어야 한다면 우선 CSS 에서 html 과 body 의 height 크기를 100%로 잡아주셔야 합니다. 그리고 차트의 부모 Div 즉 rMateChartH5.create( ... , "부모 Div Id", ..... ); 에서의 부모 Div 의 크기 width, height 를 퍼센티지 값으로 설정하여 주시고 차트의 크기또한 퍼센티지 값으로 설정하여 주십시오.

```
html{
    height:100%;
}

body{
    height:100%;
}

// rMateChart 를 생성합니다.
// 파라미터 (순서대로)
// 1. 차트의 id ( 임의로 지정하십시오. )
// 2. 차트가 위치할 div 의 id (즉, 차트의 부모 div 의 id 입니다.)
// 3. 차트 생성 시 필요한 환경 변수들의 묶음인 chartVars
// 4. 차트의 가로 사이즈 (생략 가능, 생략 시 100%)
// 5. 차트의 세로 사이즈 (생략 가능, 생략 시 100%)

rMateChartH5.create("chart1", "chartHolder", chartVars, "100%", "100%");

<!-- 차트가 삽입될 DIV -->
<div id="chartHolder" style="width:100%; height:100%;">
</div>
```

위와 같이 사용하여 주시기 바랍니다. 위의 경우에는 chartHolder 라는 Div 하나밖에 없지만 이 Div 를 감싸고 있는 부모 Div 가 있다면 감싸고 있는 Div 의 크기도 퍼센티지 값으로 적용하셔야 합니다. 그리고 위와 같이 브라우저 창을 늘이고 줄일 경우가 아닌 임의적으로 버튼클릭 등의 이벤트를 받아 차트의 크기를 변경하고 싶으실 경우에는 해당 차트의 부모 Div 의 크기를 변경 후 리사이징 함수를 실행 하시어 차트 크기를 변경하여 사용하시기 바랍니다.

아래는 예제 입니다.

```
function changeChart(w,h){
    var ch = document.getElementById("chartHolder") // 차트의 부모 Div를 가져온다.
    ch.style.width = w+"px"; // width를 600px로 변경
    ch.style.height = h+"px"; // height를 400px로 변경
    document.getElementById("chart1").resize(); //부모Div 크기변경 후 id가 chart1 인 차트를 리사이징 시킨다.
}
```

위 처럼 "chart1"에 접근하여 resize 함수를 실행하면 되겠습니다.

```
document.getElementById("chart1").resize();
```

브라우저 창을 늘이고 줄이는 것이 아닌 이상 해당 차트를 리사이즈 시키려면 위 처럼 리사이즈를 하려하는 차트 id 에 접근하여 resize()함수를 실행 하십시오.



## 11. 제품 사용시 발생 할 수 있는 문제 해결 정보

내 용	해 결 방 안
Invalid License 경고창	올바른 라이선스키를 삽입
This host is not authorized 경고창	라이선스키에 삽입한 도메인과 차트가 출력되고 있는 도메인이 같아야함
Period has expired 경고창	라이선스 기한이 만료 되었으니 새로운 라이선스로 교체
Other version soft license 경고창	올바른 제품 라이선스를 삽입
Invalid License Type 경고창	올바른 제품의 버전 라이선스를 삽입
Evaluation License 경고창	정품 라이선스를 삽입
Trial license is required 경고창	평가판 라이선스를 삽입
콘솔 창에 Invalid XML 출력	차트에 삽입되는 레이아웃 XML의 형태를 수정

<감사합니다.>