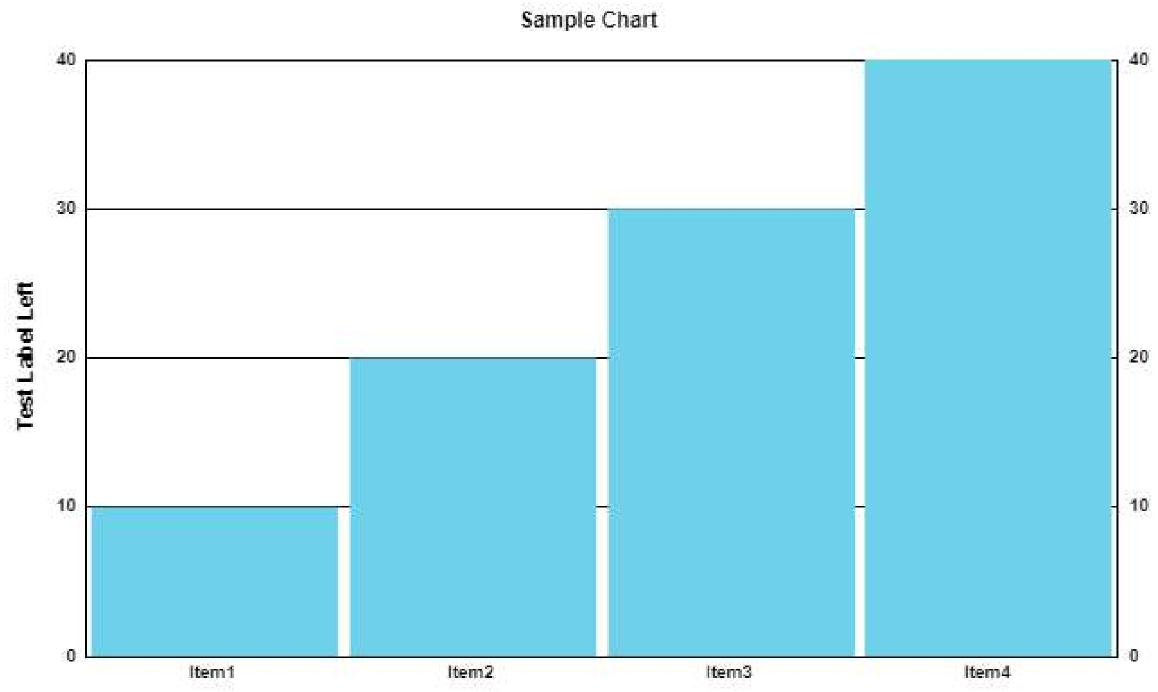
PROMPTUS 8

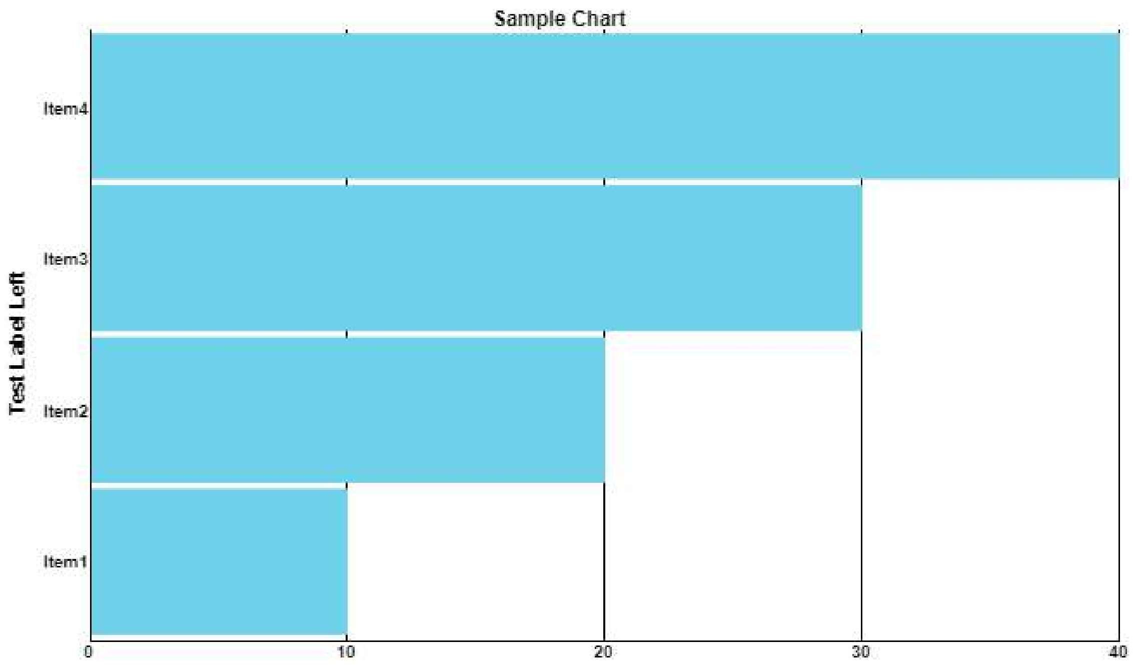
P8 Charts Usage

Basic Components in P8 Charts:

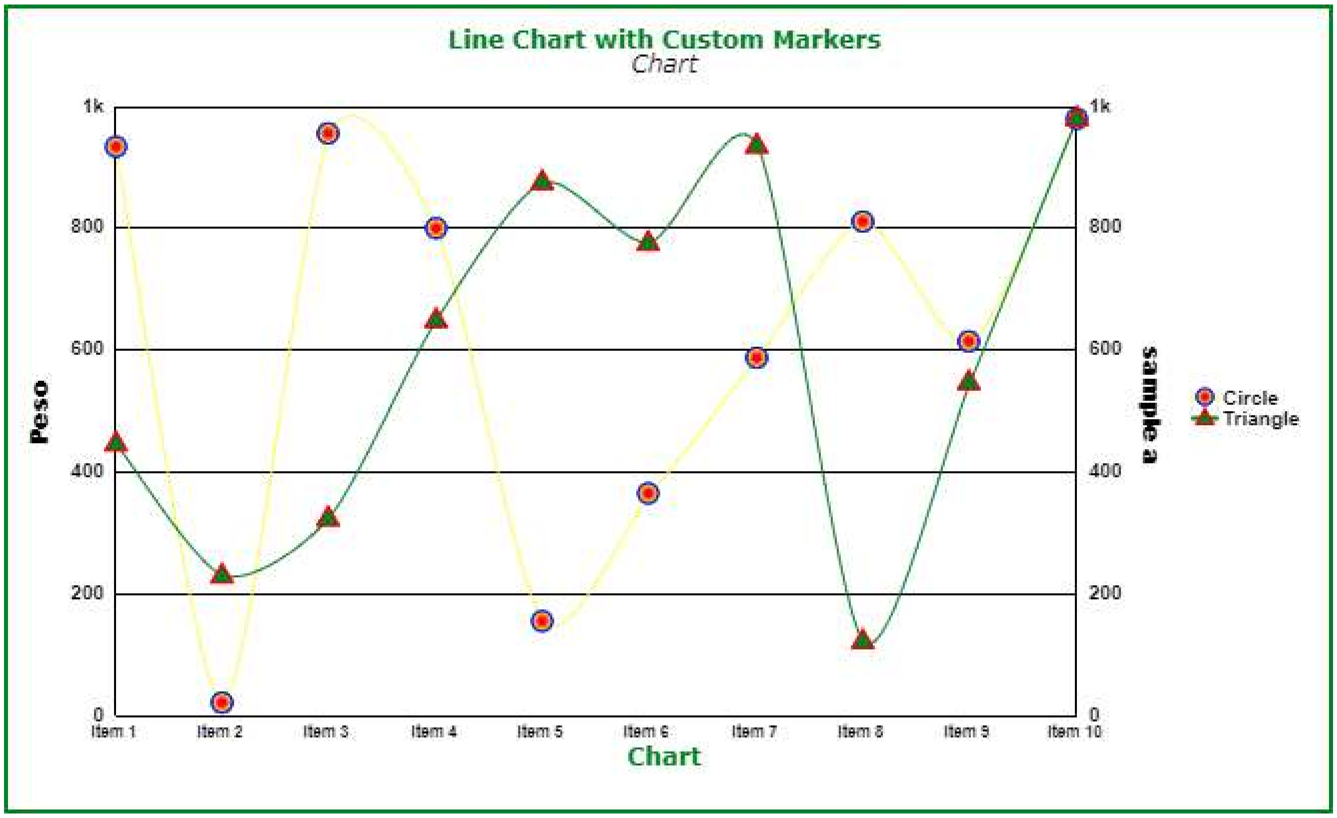
1. Bar Chart



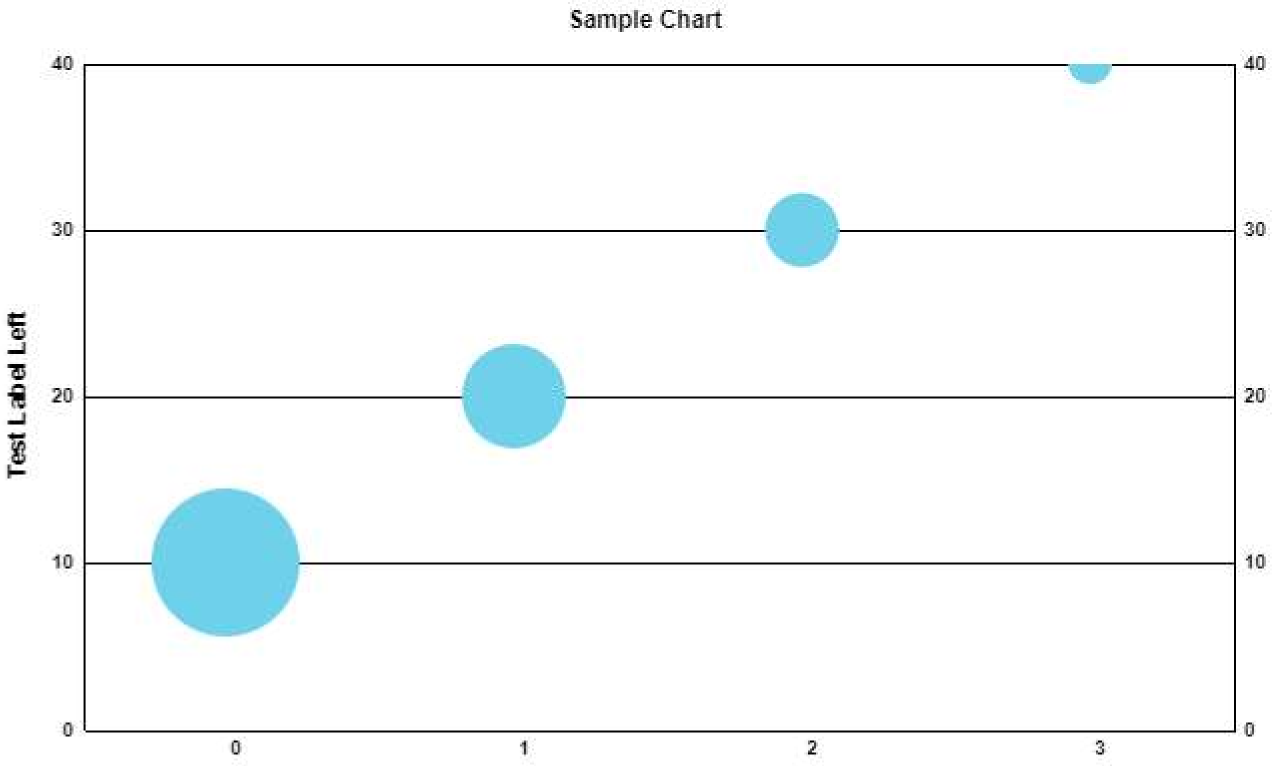
2. Horizontal Bar Chart



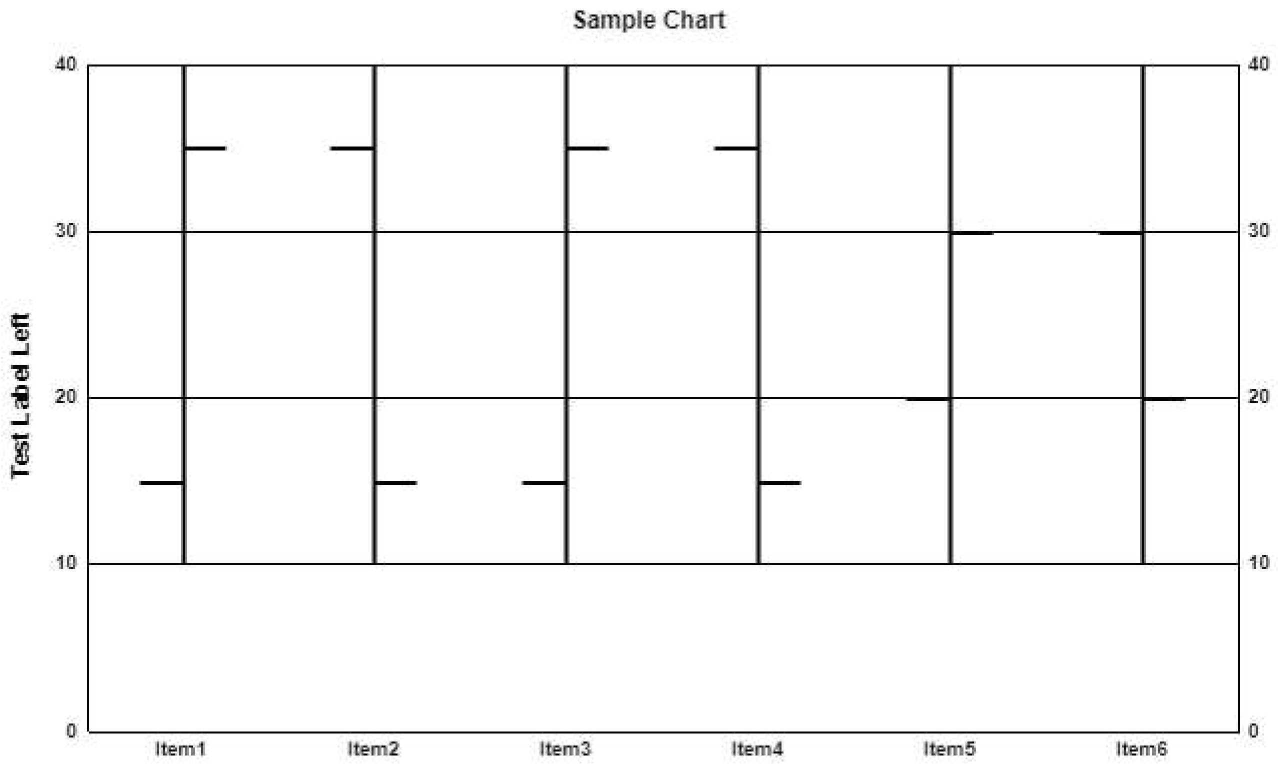
3. Line Chart



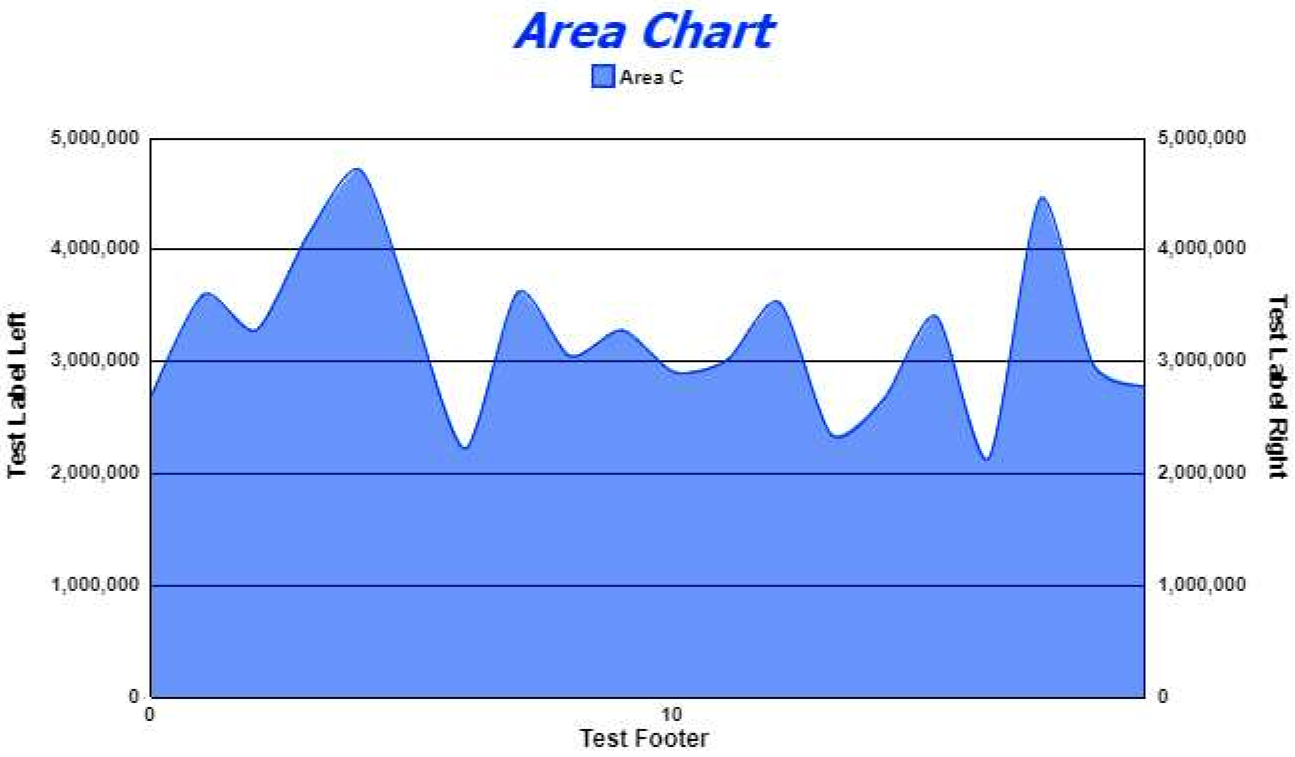
4. Bubble Chart



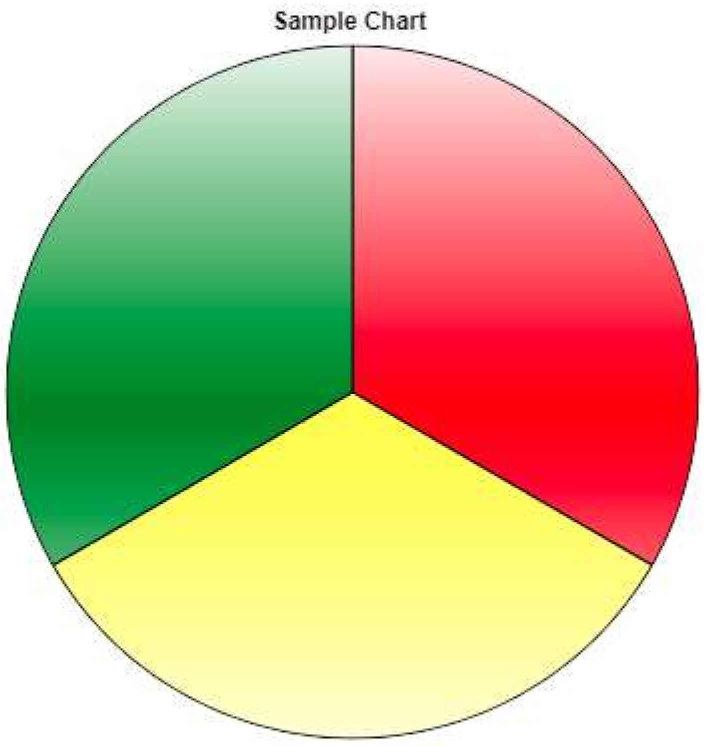
5. OHLC Chart (Open High Low Close)



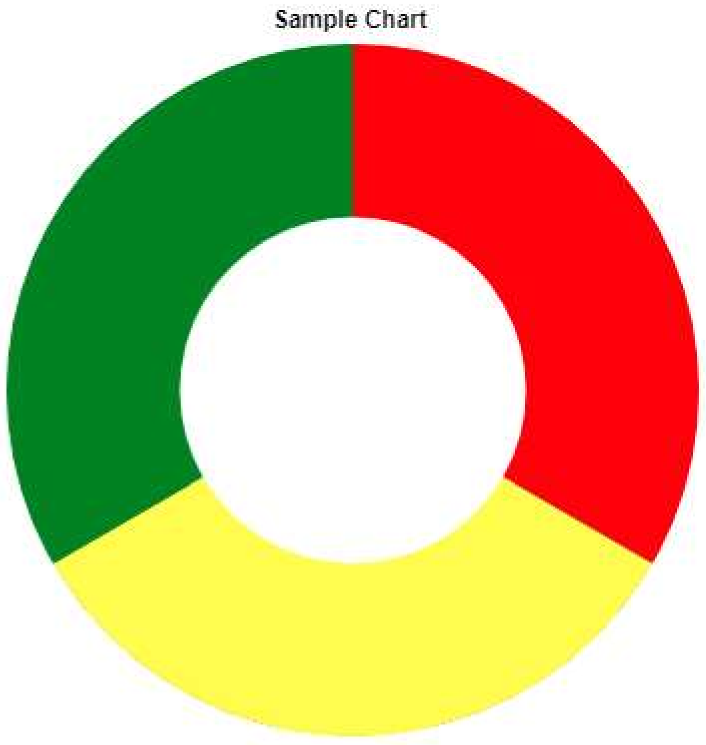
6. Area Chart



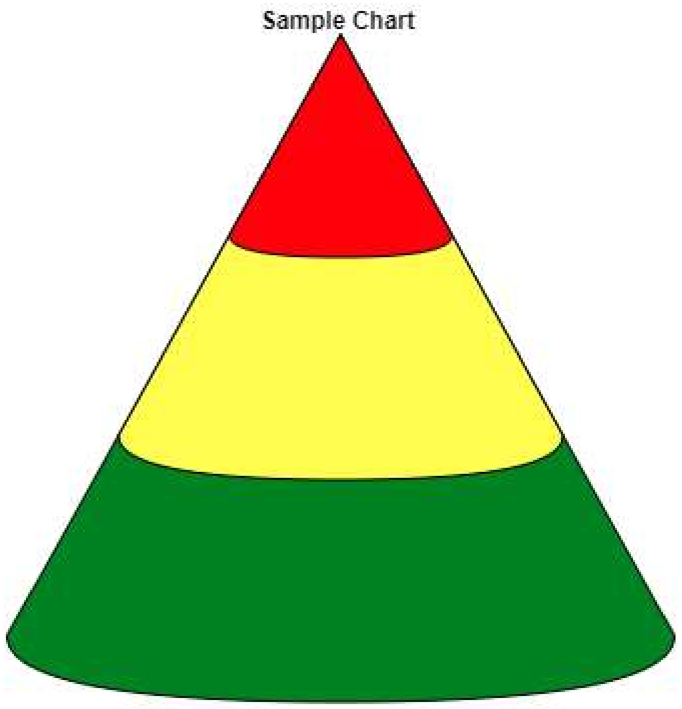
7. Pie Chart



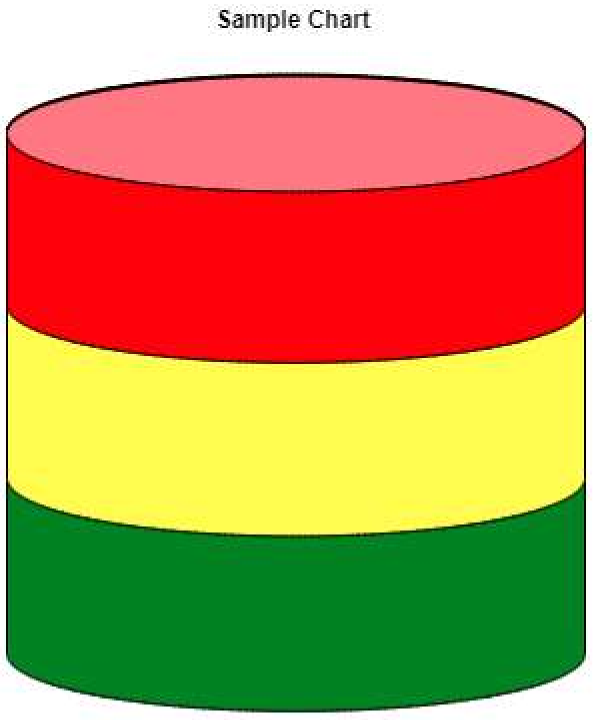
8. Doughnut Chart



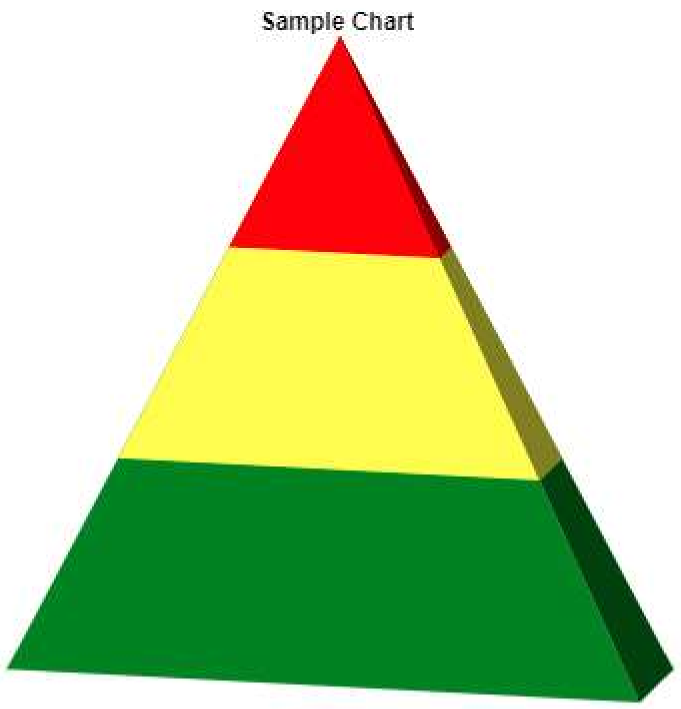
9. Cone Chart



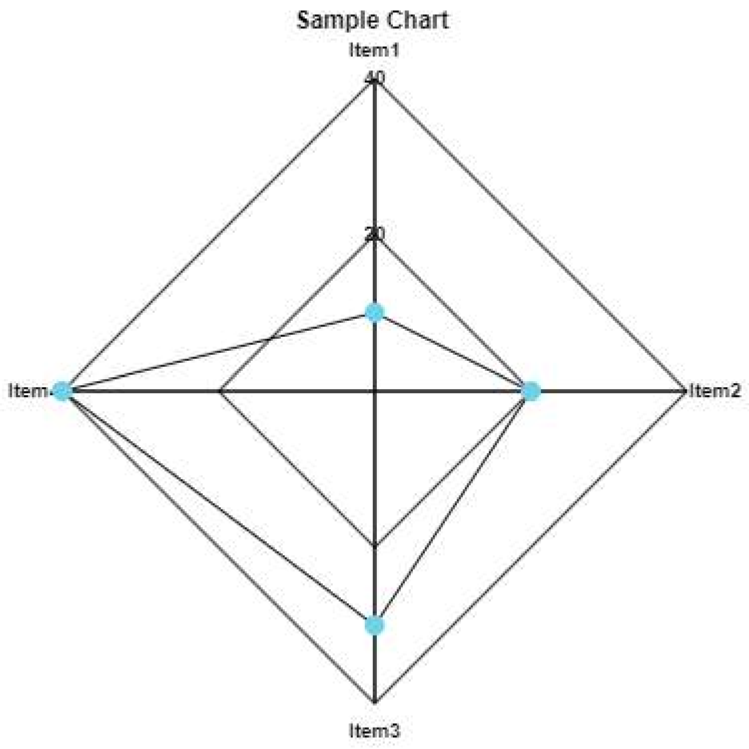
10. Cylinder Chart



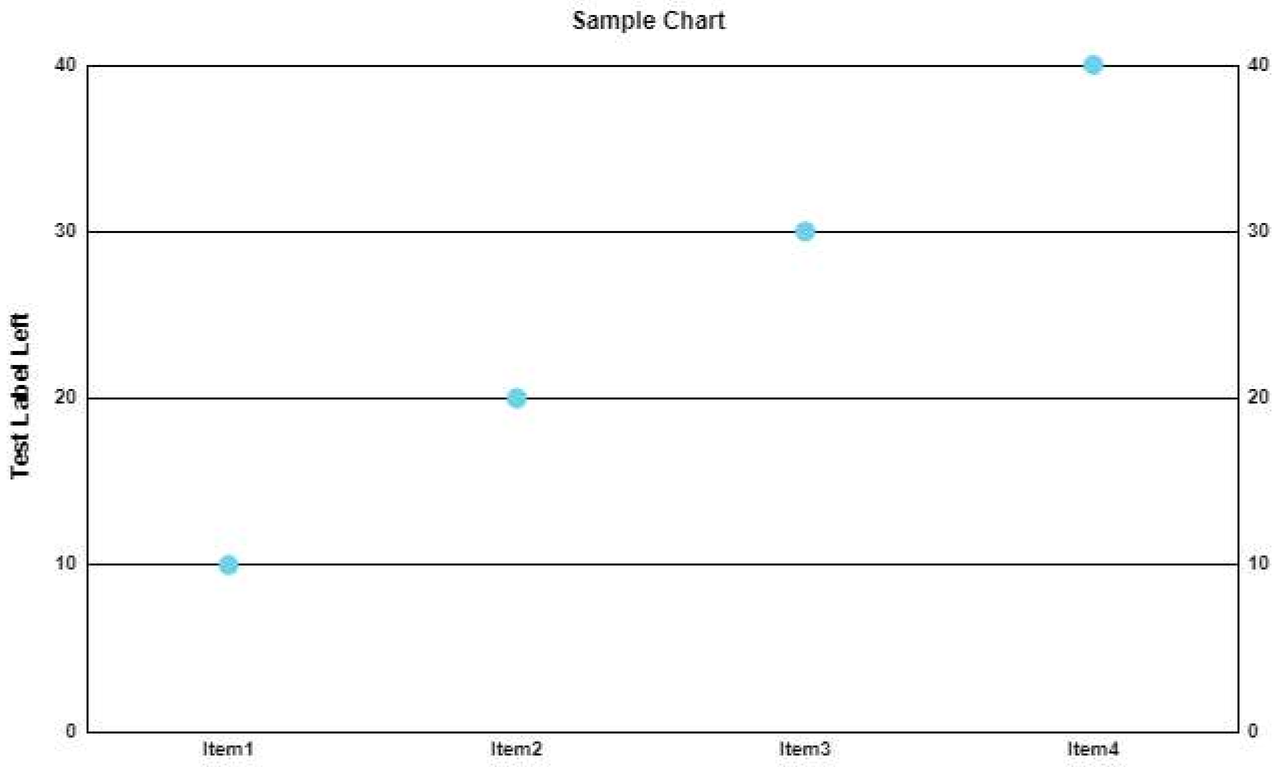
11. Pyramid Chart



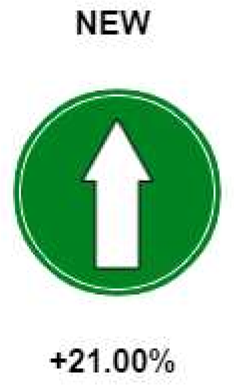
12. Radar Chart

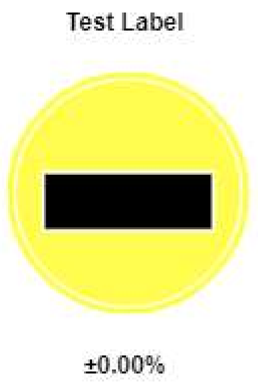


13. Scatter Chart

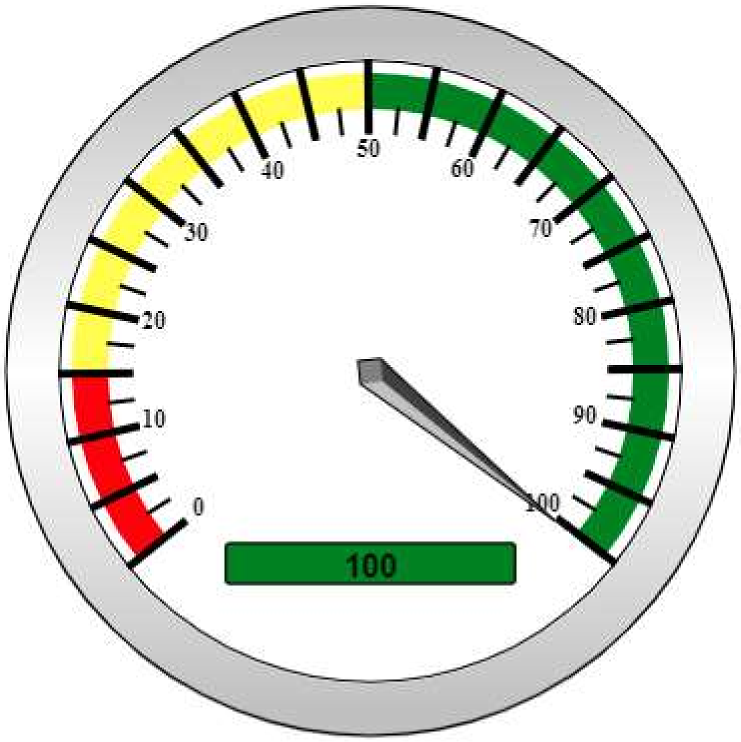


14. Trend Indicator

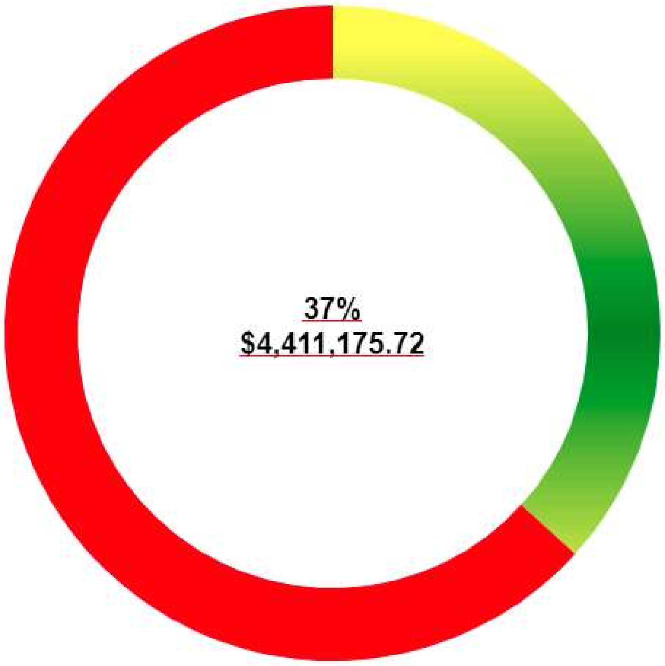




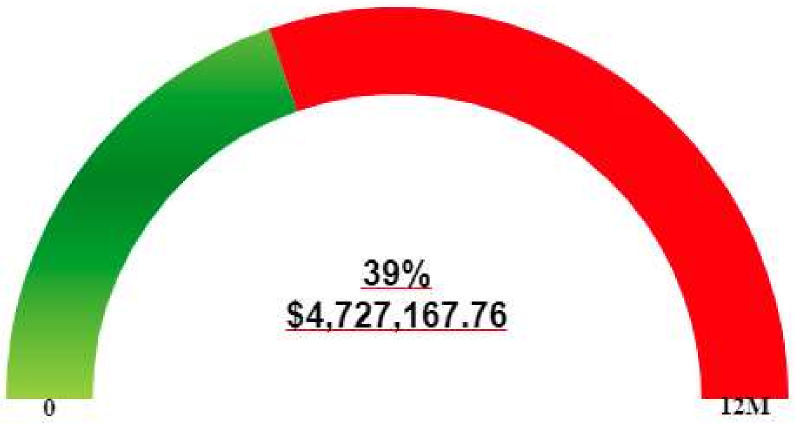
15. Gauge



16. Circular Gauge



17. Arc Gauge



CREATING CHART AND OTHER COMPONENTS

Use this code in Javascript:
format:
P8.Chart(ID, Idchart, iscanvas)

e.g.
var chart = new P8.Chart(ID, Idchart, iscanvas);

where ID = <div ID>, idchart = <canvas ID>, iscanvas = boolean

Note: if iscanvas is true, then idchart will be used, else ID will be used.
ID and idchart must be in string format.

These are the components:
P8.Chart – Bar and Line Chart
P8.SChart – Scatter Chart
P8.HChart – Horizontal Bar Chart
P8.BChart – Bubble Chart
P8.OHLCChart – OHLC and Candlestick Chart
P8.RChart – Radar Chart
P8.PieChart – Pie Chart
P8.DoughnutChart – Doughnut Chart
P8.ConeChart – Radar Chart
P8.PyramidChart – Radar Chart
P8.CylinderChart – Radar Chart
P8.TrendChart – Trend Indicator

To execute component:
chart.Render();

<u>DATA</u>

used for inputting the data value(s) of a component
format
var data = [];

Bar, Line, Scatter

```
data.push({
        index0: <string>, //index 0
        index1: 10, //index 1
        index2: 20 //index 2
});
```

Bubble

(Note: This code also works on other other charts, but you must use value only in JSON)

```
data.push({
        index0: <string>, //index 0
        index1: {value: <number>, area: <number>}, //index 1
        index2: {value: <number>, area: <number>} //index 2
});
```

Pie, Doughnut, Cone, Pyramid, and Cylinder

```
data.push({
        name: <string>
        , value: <number>
        , fill: "red"
        , filltype: undefined (color | gradient) in string format
});
```

OHLC

```
data.push({
        label: <string>
        , index: { open: <number>, high: <number>, low: <number>, close: <number> }
});
```

```
chart.SetData(data);
```

Note: index 0 will always be the label string, while index 1, index 2..., index n will be number digits in float format or in JSON object. You can also include fill, marker, and filltype inside JSON object.

OBJECT DATA

An object data is use for customizing inputs, must be in array first
format:

```
var ObjectData = [];
ObjectData.push({
	name: <string>
	, fillcolor: "<string> //hex and rgba format is also applied
	, filltype: <string> (color | gradient)
	, strokecolor: <string> //hex and rgba format is also applied
	, linecolor: <string> //hex and rgba format is also applied
	, charttype: <string> (bar | line)
	, dash: [0] //use number format in array e.g. [1, 1]
	, cap: <string> (butt | round | square)
	, join: <string> (bevel | round | miter)
	, linewidth: <number> //line width stroke for Line Chart
	, strokewidth: <number> //marker stroke
	, marker: <string> //Marker only in Line, Bubble, and Scatter chart, see Marker
	, areasize: <number>
	, group: <number>
	, prefix: <string>
	, suffix: <string>
});

chart.SetObject(ObjectData);
```

FONT

For using text formats:
format:
```
var Font = {
        color: <string>
        , fontFamily: <string>
        , fontSize: <number>
        , fontWeight: <string>
        , fontStyle: <string>
        , textdirection: <string> (off | left | right) only in measureleft and measureright
        , display: bool
        , text: <string>
}; //Must be in JSON Format
```

For fonts you can use these following codes:
```
chart.SetLabelFont(Font); //For Data Labels
chart.SetMeasureLeft(Font); //For Measure Left Font
chart.SetMeasureRight(Font); //For Measure Right Font
chart.SetMeasureFont(Font); //For Measure Font in Horizontal Bar and Radar Chart
chart.SetLegendFont(Font); //For Legend Font
```

These codes only works with text included in JSON
```
chart.SetHeader(Font);
chart.SetSubHeader(Font);
chart.SetFooter(Font);
chart.SetLabelLeft(Font);
chart.SetLabelRight(Font);
```

For Pie Chart:
```
chart.SetDataLabelFont(Font);
```

Note: text only works on Header, Sub-Header, Footer, Label Left, and Label Right

MARKER

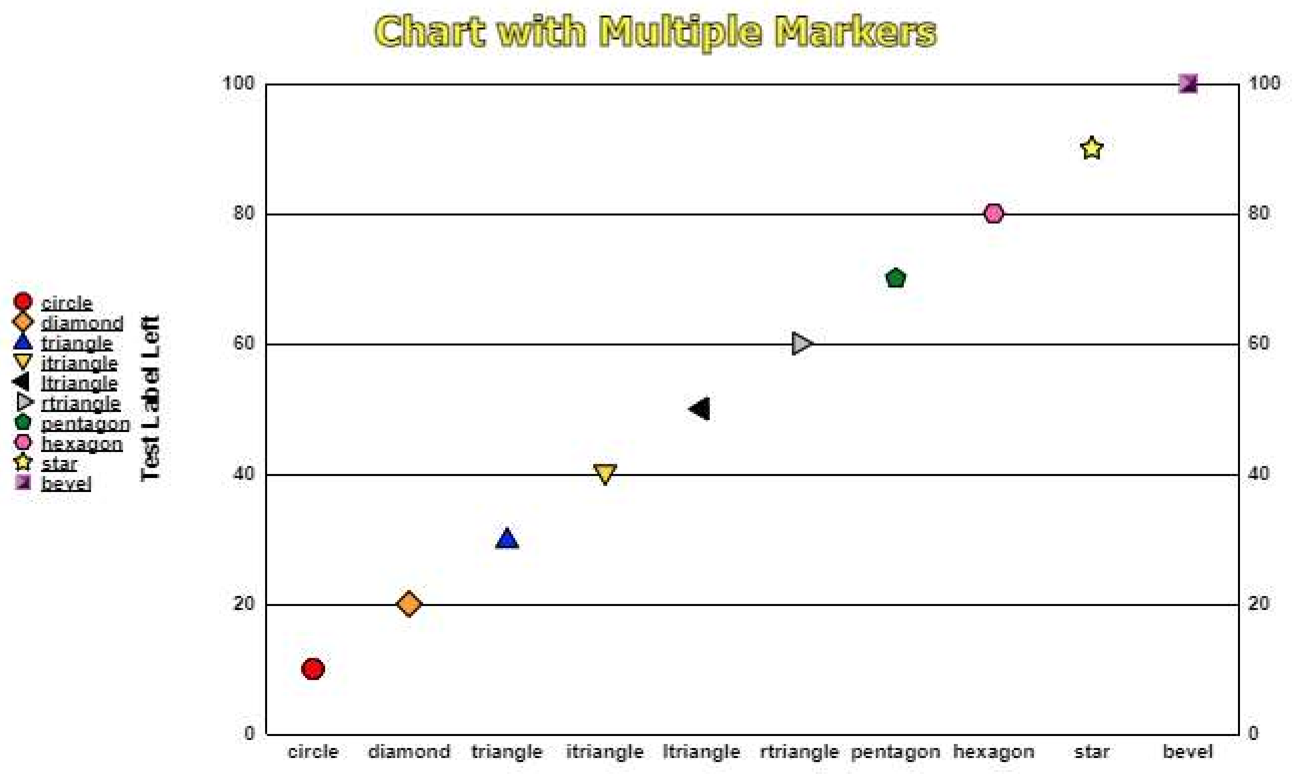
These codes only applied to Data and/or Object Data for Line, Scatter, and Bubble Charts.
All codes must be in string format

MARKER	CODE(S)
Circle	o, circle
Square	[], square
Diamond	<>, diamond
Triangle	^, triangle
Inverted Triangle	v, itriangle
Left Triangle	<, lefttriangle
Right Triangle	>, righttriangle
Rounded Square	roundsquare
Cross	X, x, cross
Plus	+, plus
MARKER	**CODE(S)**
Star	star

Pentagon	5, pentagon
Hexagon	6, hexagon
Octagon	8, octagon
Asterisk	*
Up	up
Down	down
Left	left
Right	right

Note: To remove marker, simply put null or undefined for Line Chart.

Example

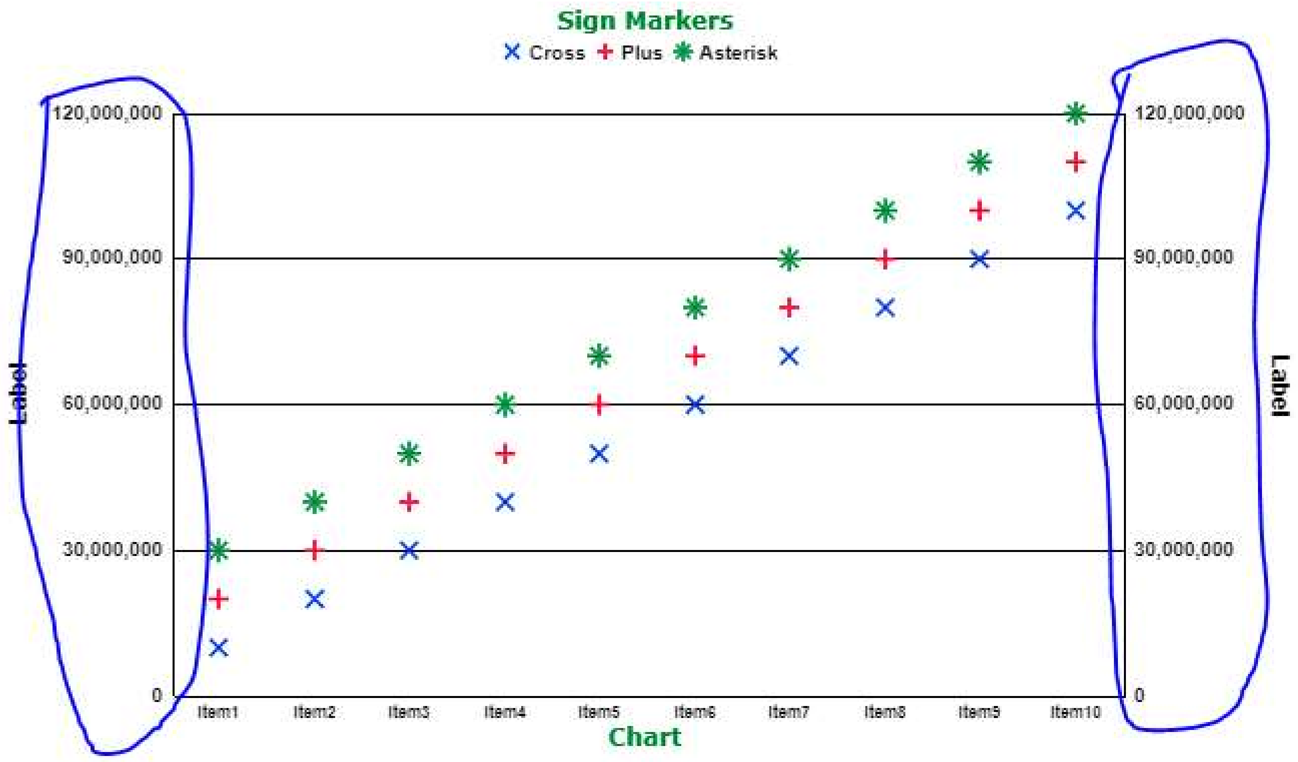


KM FLAG

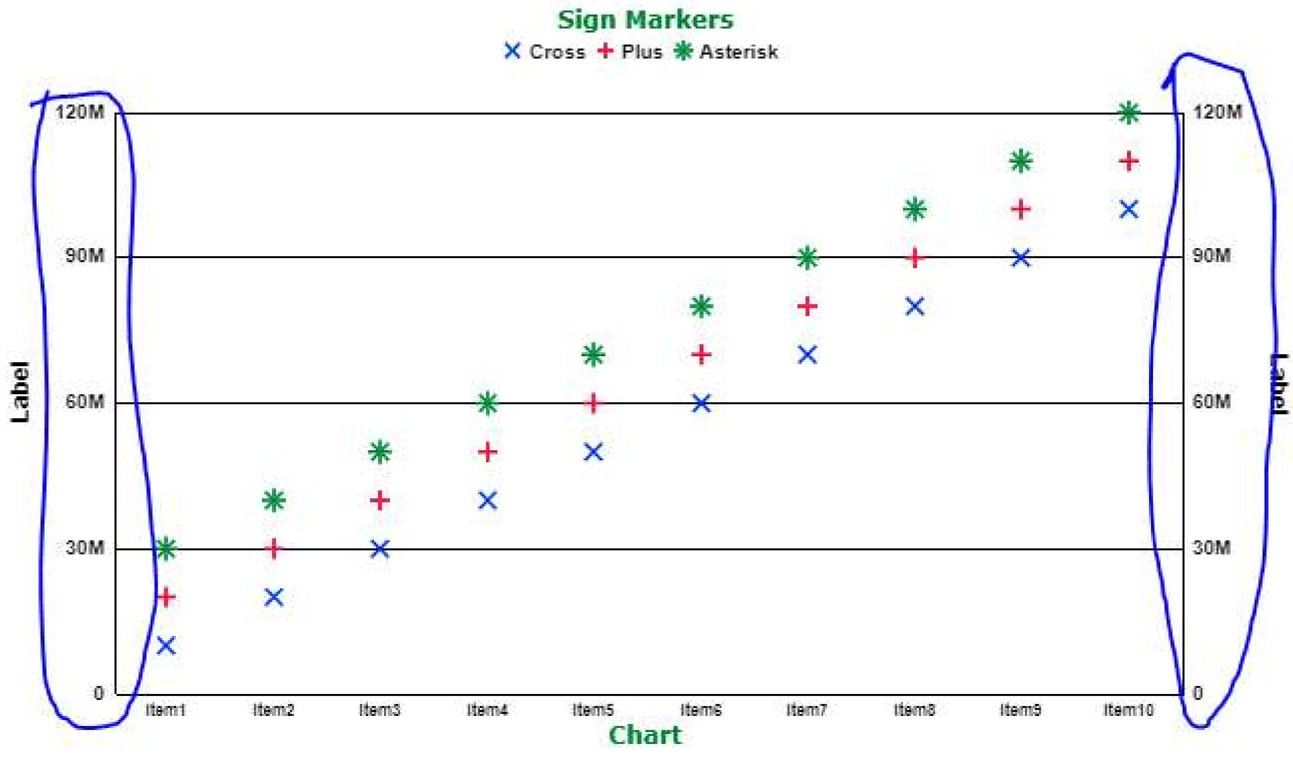
These codes use to convert digits in measure labels

format:
chart.SetKMFlag(kmflag);
where kmflag = <bool>

Examples:
If KM Flag is true



If KM Flag is false



LEGEND

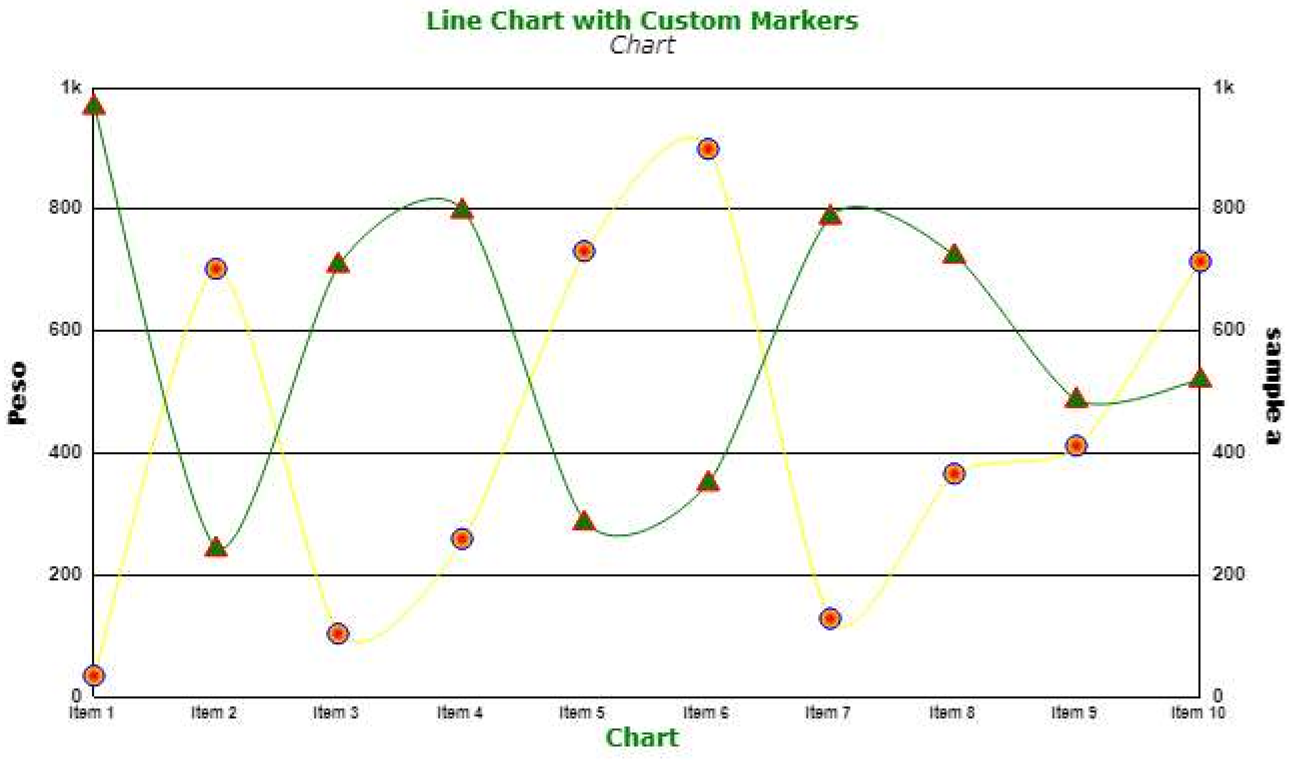
Legend Position:

chart.SetLegendPosition(position)

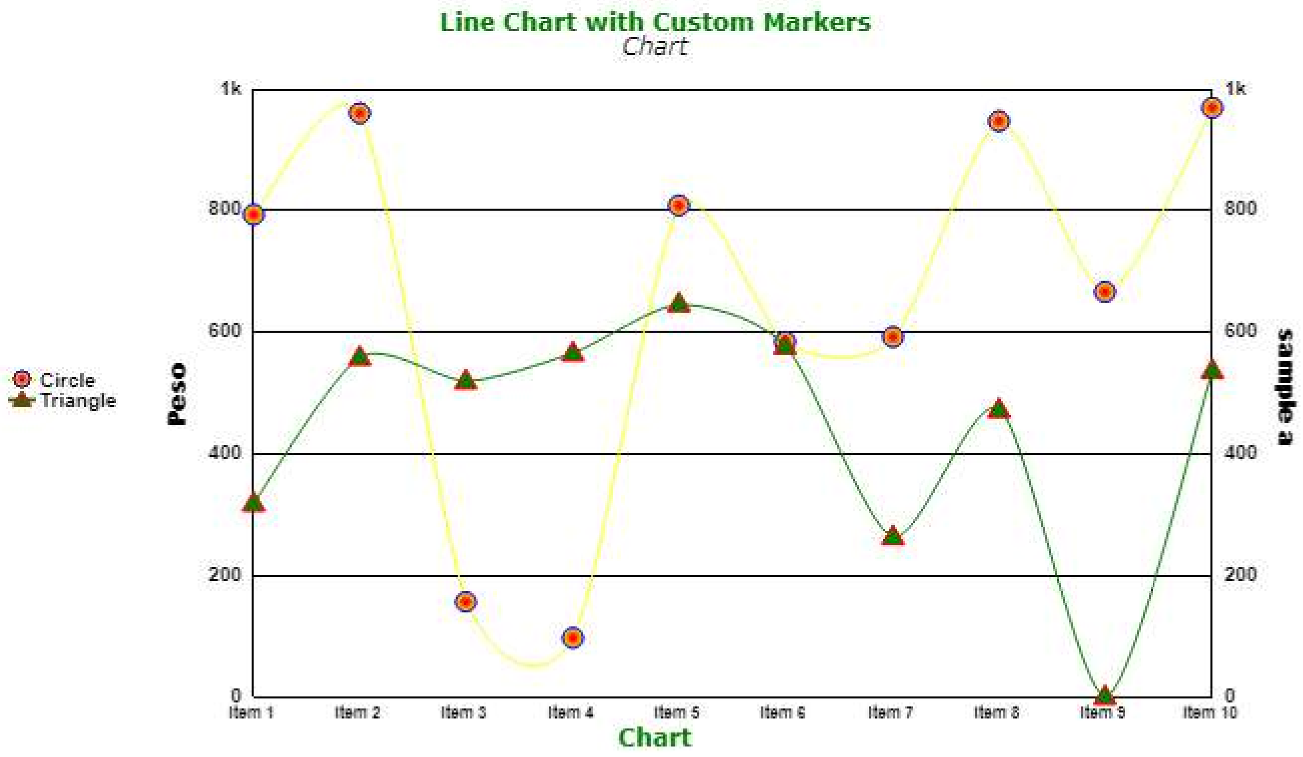
where position = <string> (left | right | top | bottom | none)
in string format
“none” is the default output

Legend Position Outputs:

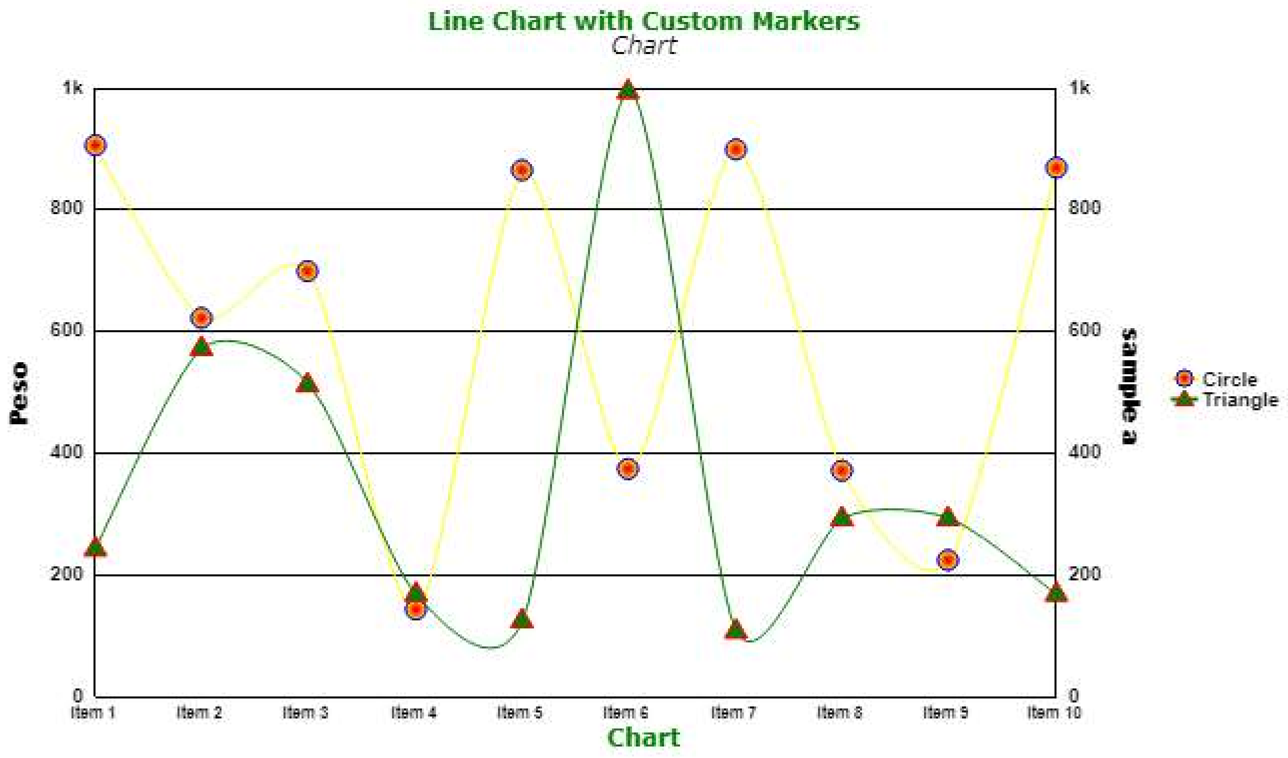
1. None



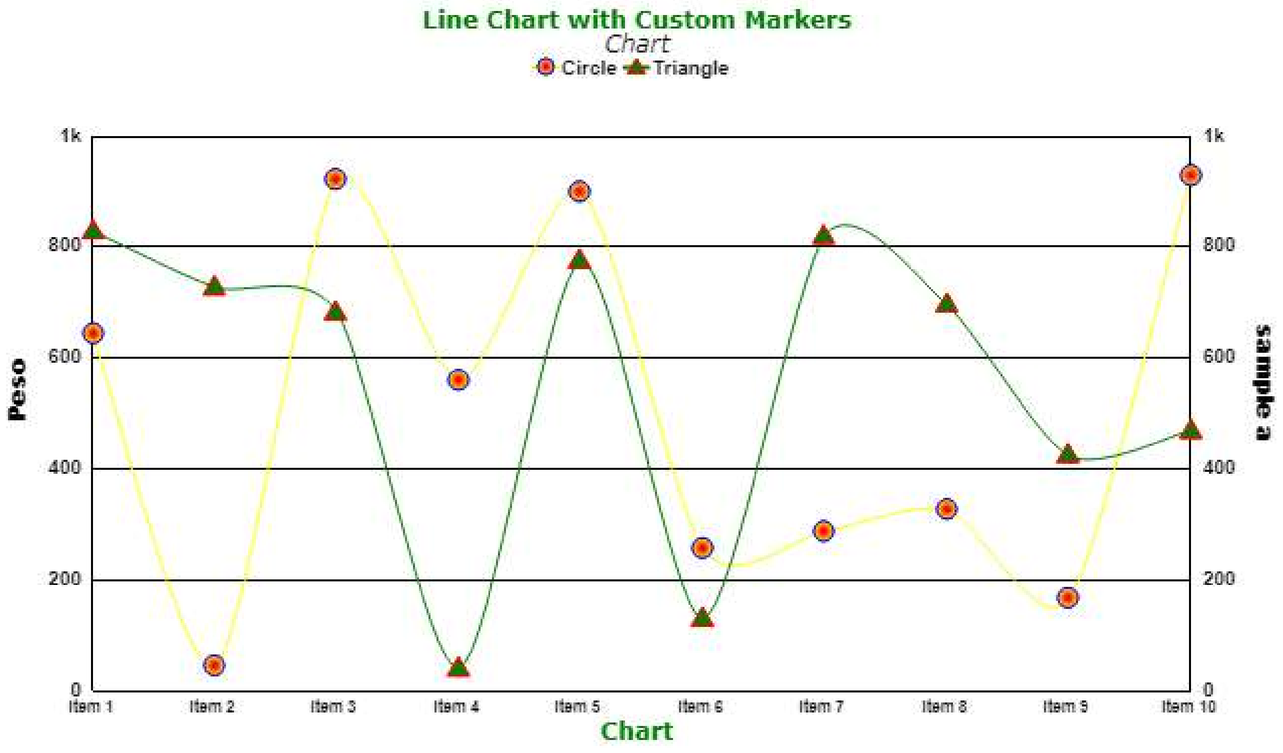
2. Left



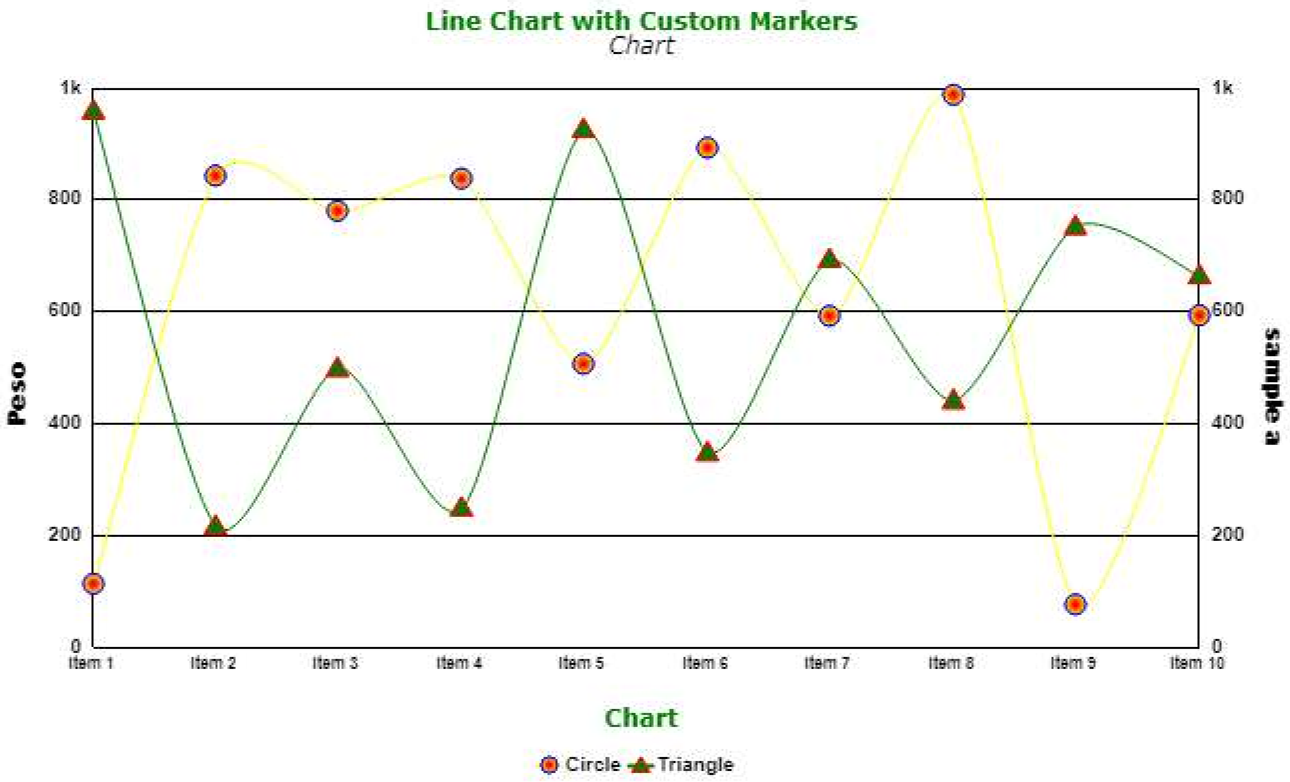
3. Right



4. Top



5. Bottom



GRID LINE
used to draw grid lines

format:
chart.SetGridLine(gridline);

```
where gridline = {
        color: <string> //fill
        , horizontalcolor: <string> //fill
        , verticalcolor: <string> //fill
        , width: <number>//grid stroke
        , internalwidth: <number> //internal grid stroke
        , internal: bool
    };
```

MAX DIGIT
- used to set max total digit of all data values

format:
chart.SetMax(max);

where max = <int value>

BORDER
used for border of canvas

format:
chart.SetBorder(border);

```
where border = {
        style: <string> //border style
        , fill: <string> //border fill
        , width: <number>//border stroke
    }
```

SIZE
- used for size of canvas

format:
chart.SetSize(canvassize);

```
where canvassize = {
            width: <number>
            , height: <number>
        }
```

BACKGROUND
- used to set background color or gradient

format:
chart.SetBackground(background);

where background = <solid color or gradient background in HTML code format>

ANIMATION
- used to execute chart animation

format:
chart.SetAnimation(animation);

where animation = <bool>

COLOR INPUTS
- these codes used for color inputs, and must use the specific type.

The default filltype or stroketype is “color”.

If filltype or stroketype is “color”, you must use the default color inputs in string format. RGBA, Hex are also applicable as well but in string format.

e.g.
```
filltype = “color”;
fill = “red”;
```

And then if filltype or stroketype is “gradient”, you must use JSON array format.

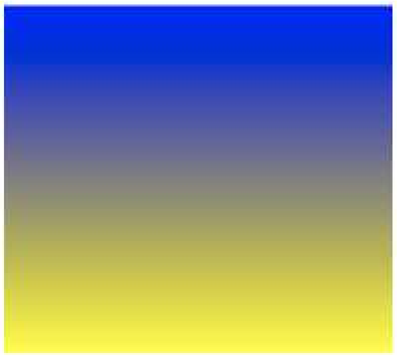
e.g.
```
filltype = “gradient”;
fill = [];
fill.push({
        color: “yellow”
        , stop: 0 //between 0-1 only
});
fill.push({
        color: “blue”
        , stop: 1 //between 0-1 only
});
```

Note: If stop is null or undefined, it will divide from what number array to it’s array length.
e.g.
```
for (i = 0; i < array.length; i++){
        colorstop = i / (array.length – 1)
}
```

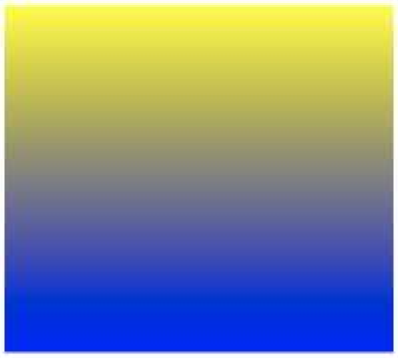
Gradient Type:

These are the gradient type outputs and must be string format:

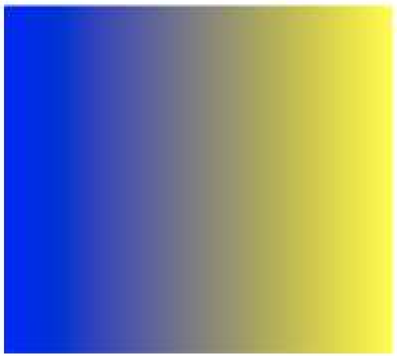
linear A



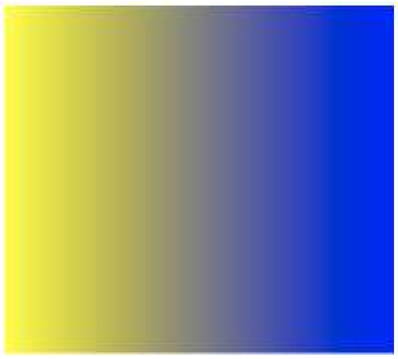
linear B



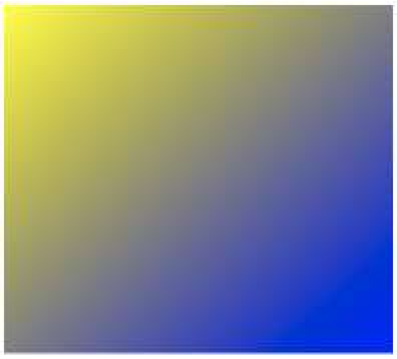
linear C



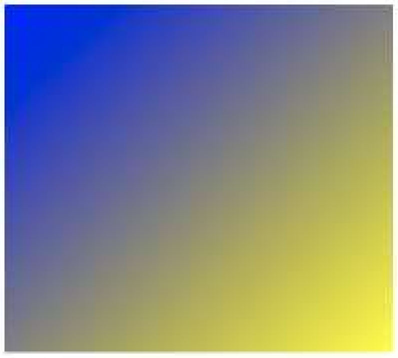
linear D



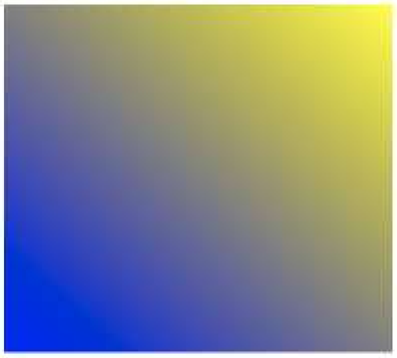
linear E



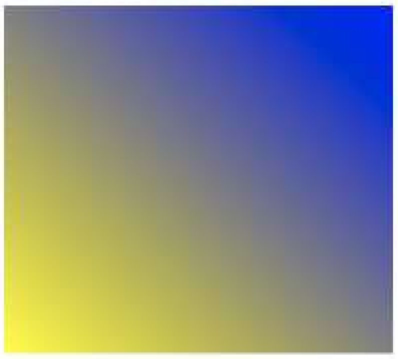
linear F



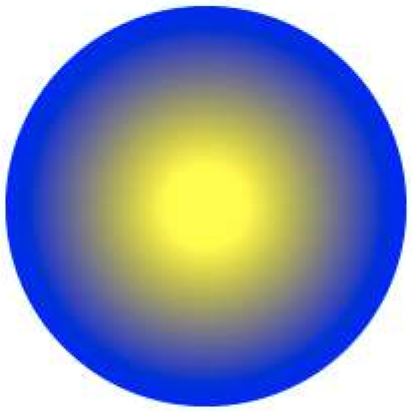
linear G



linear H



radial



Note: radial are worked for all components other than Bar Chart.

3D BAR AND OTHER SHAPES

This code only works for Bar and Horizontal Bar Chart to change in 3D view

Enable 3D

- used to enable 3D Bar Chart

format:
chart.SetEnable3D(enable3d);

where enable3d = bool (true | false)

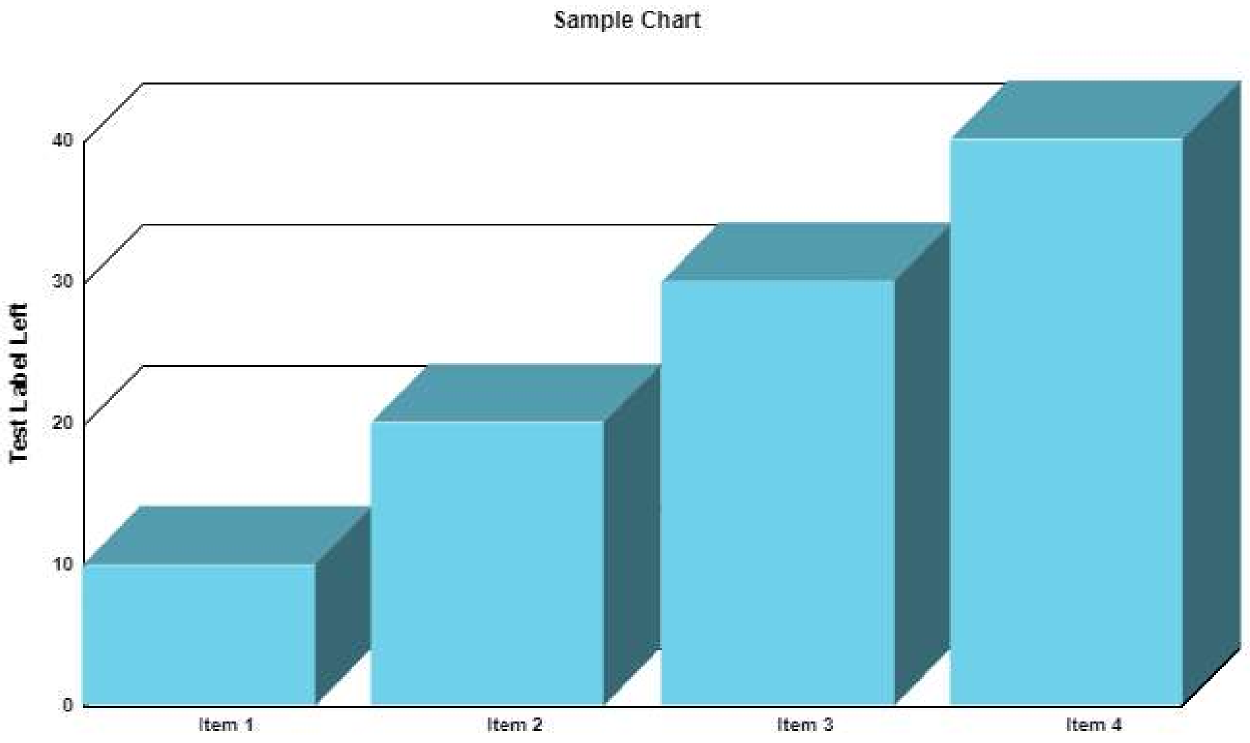
Pattern 3D

- used to change the pattern/shape of a bar

format:
chart.SetPattern3D(pattern);

where pattern = (bar | cone | cylinder | pyramid) in string format
bar is the default pattern
note: Even you did not input pattern in SetPattern3D, the default pattern is bar.

Sample:



CODES FOR GAUGE COMPONENTS:

var gauge = new P8.Gauge();

P8.Gauge – Gauge
P8.ArcGauge – Arc Gauge
P8.CircularGauge – Circular Gauge

gauge.Render(); //to execute gauge

Num Input:
Use to input the specific number

format:
gauge.SetNumInput(numinput);

where numinput = <number in float format>;

Initial Total:
Use to set the initial total of the gauge

format:
gauge.SetInitialTotal(itotal);

where itotal = <number in float format>;

Percent Low and Percent Mid:
These codes use to adjust the minimum percentage

format:
gauge.SetPercentLow(percentlow);
gauge.SetPercentMid(percentmid);

```
where percentlow = <number between 0-100>;
      percentmid = <number between 0-100>;
```

Note: if percentlow is higher than percentmid, percentlow = percentmid

Hash:
These are the codes for hash lines of gauge

format:

```
hash = {
        small: <number in float>
        , large: <number in float>
        , fill: <string> (color only)
};
```

Fonts:
Much like in other components, these code uses the same as the Font Format

format:
```
gauge.SetNumLabelFont(NumLabelFont);
gauge.SetInputFont(InputFont);
gauge.SetPercentageFont(PercentageFont);
gauge.SetRateFont(RateFont);
```

where NumLabelFont, InputFont, RateFont, and Percentage Font are:
```
Font = {
        color: <string>
        , fontFamily: <string>
        , fontSize: <number>
        , fontWeight: <string>
        , fontStyle: <string>
        , textdirection: <string> (off | left | right) only in measureleft and measureright
        , display: bool
        , text: <string>
};
```

Prefix:
Its uses the prefix of the number input in string format

format:
```
gauge.SetPrefix(prefix);
```

where prefix = <string>

Rate:
It used to display the rate, according to the percent of the input depends on low and mid percentage

format:
```
gauge.SetRate(rate);
where: rate = {
        a: <string>
        , b: <string>
        , c: <string>
};
```

Fill:

These codes uses in filling the gauge on border, needle, needle base, inner color

format:

```
gauge.SetFill(fill);

where fill = {
        border: {
                filltype: <string> (color | gradient)
                , fill: <string for color or JSON object for>
                , gradienttype: <string>
                }
        , innercolor: {
                filltype: <string> (color | gradient)
                , fill: <string for color or JSON object for>
                , gradienttype: <string>
                }
        , needle: {
                filltype: <string> (color | gradient)
                , fill: <string for color or JSON object for>
                , gradienttype: <string>
                }
        , needlebase: {
                filltype: <string> (color | gradient)
                , fill: <string for color or JSON object for>
                , stroke: <string> (color | gradient)
                , gradienttype: <string>
        }
};
```

Note: These codes are the same color input format as the Chart