

The logo features the text "Emprise JavaScript Charts" in a bold, black, sans-serif font with a white outline. The text is set against a background of a white grid on a dark green surface, with green arrows pointing in various directions.

# Emprise JavaScript Charts

## Developer's Guide

© 2008 Emprise Corporation. All Rights Reserved.

# Table of Contents

<b>Part I Getting Started</b>	<b>1</b>
1 Overview .....	1
2 Implementation Instructions .....	2
3 Creating Your First Chart .....	5
4 Customizing Your Chart .....	12
Y Axis .....	15
X Axis .....	15
Hint .....	16
Chart Body .....	16
Legend .....	17
5 Version 1.x Compatibility .....	17
Version 1.x Compatibility .....	17
Deprecated Properties, Methods and Events .....	17
<b>Part II Deployment</b>	<b>19</b>
<b>Part III Changes</b>	<b>19</b>
1 Changes in version 2.0 .....	19
2 Changes in version 1.3.1 .....	20
3 Changes in version 1.3 .....	20
4 Changes in version 1.2.1 .....	21
5 Changes in version 1.2 .....	21
6 Changes in version 1.1 .....	23
7 Changes in version 1.0.1 .....	24
8 Changes in version 1.0 .....	25
<b>Part IV Data Formats</b>	<b>26</b>
1 XML - Full .....	27
2 XML - Short .....	28
3 XML - Compact .....	29
<b>Part V API Reference</b>	<b>29</b>
1 EJSC .....	29
Properties .....	29
DefaultImagePath .....	29
DefaultColors .....	29
DefaultBarColors .....	30
DefaultPieColors .....	31
STRINGS .....	31
2 EJSC.Chart .....	32

<b>Properties</b> .....	<b>32</b>
allow_interactivity.....	32
allow_mouse_wheel_zoom.....	33
allow_move.....	33
allow_zoom.....	34
allow_hide_error.....	34
auto_find_point_by_x.....	34
auto_resize.....	35
auto_zoom.....	35
axis_bottom.....	36
axis_left.....	37
axis_right.....	38
axis_top.....	38
background.....	39
building_message.....	39
drawing_message.....	40
legend_title.....	40
max_zoom_message.....	40
message_timeouts.....	40
proximity_snap.....	40
show_hints.....	41
show_legend.....	41
show_messages.....	42
show_titlebar.....	42
title.....	42
<b>Methods</b> .....	<b>43</b>
acquireSeries.....	43
addSeries.....	43
clearSelectedPoint.....	44
exportSVG.....	44
exportSVGLegend.....	45
findClosestPoint.....	47
getMinMaxYInXRange.....	47
getZoomBoxPixelCoordinates.....	48
hideTitlebar.....	48
hideZoomBox.....	48
redraw.....	48
remove.....	49
removeSeries.....	49
selectPoint.....	49
setAutoResize.....	49
setLegendTitle.....	49
setTitle.....	50
showTitlebar.....	50
showZoomBox.....	50
<b>Events</b> .....	<b>51</b>
onAfterBuild.....	51
onAfterDraw.....	51
onAfterMove.....	51
onAfterSelectPoint.....	51
onAfterUnselectPoint.....	52
onAfterZoom.....	52
onBeforeBuild.....	52
onBeforeDbClick.....	52

onBeforeDraw.....	52
onBeforeSelectPoint.....	53
onBeforeUnselectPoint.....	53
onDbClickPoint.....	53
onShowHint.....	53
onShowMessage.....	54
onUserBeginZoom.....	54
onUserEndZoom.....	54
<b>Deprecated .....</b>	<b>55</b>
Properties .....	55
force_static_points.....	55
force_static_points_x.....	55
force_static_points_y.....	55
legendTitle .....	55
show_crosshairs.....	55
show_grid .....	55
show_mouse_position.....	55
show_x_axis .....	55
show_y_axis .....	55
x_axis_caption .....	55
x_axis_className.....	55
x_axis_extremes_ticks.....	55
x_axis_formatter.....	55
x_axis_max_tick_interval.....	56
x_axis_min_tick_interval.....	56
x_axis_minor_ticks.....	56
x_axis_size .....	56
x_axis_stagger_ticks.....	56
x_axis_tick_className .....	56
x_axis_tick_count.....	56
x_cursor_position_caption .....	56
x_cursor_position_formatter.....	56
x_max .....	56
x_min .....	56
x_value_hint_caption.....	56
x_zero_plane .....	56
y_axis_caption .....	56
y_axis_className.....	56
y_axis_extremes_ticks.....	56
y_axis_formatter.....	57
y_axis_max_tick_interval.....	57
y_axis_min_tick_interval.....	57
y_axis_minor_ticks.....	57
y_axis_size .....	57
y_axis_tick_className .....	57
y_axis_tick_count.....	57
y_cursor_position_caption .....	57
y_cursor_position_formatter.....	57
y_max .....	57
y_min .....	57
y_value_hint_caption.....	57
y_zero_plane .....	57
Methods .....	57
addXAxisBin .....	57

addYAxisBin .....	57
convertPixelToPoint.....	58
convertPointToPixel.....	58
findClosestPointInSeries .....	58
hideGrid .....	58
hideXAxis .....	58
hideYAxis .....	58
getXExtremes .....	58
getYExtremes .....	58
getZoom .....	58
removeXAxisBin .....	58
removeYAxisBin .....	58
selectClosestPoint.....	58
setCrosshairs .....	58
setXAxisCaption .....	58
setXExtremes .....	58
setYAxisCaption .....	58
setYExtremes .....	59
setZoom .....	59
showGrid .....	59
showXAxis .....	59
showYAxis .....	59
Events .....	59
onAfterShowCrosshairs .....	59
onBeforeBeginZoom .....	59
onBeforeEndZoom .....	59
onShowCrosshairs.....	59
onXAxisNeedsTicks.....	59
onYAxisNeedsTicks .....	59
<b>3 Axes Types .....</b>	<b>59</b>
<b>LinearAxis .....</b>	<b>59</b>
Properties .....	60
background (inherited).....	60
border (inherited).....	60
caption (inherited).....	61
caption_class (inherited) .....	61
color (inherited) .....	62
crosshair (inherited) .....	62
cursor_position (inherited).....	63
extremes_ticks (inherited) .....	64
force_static_points (inherited).....	64
formatter (inherited) .....	64
grid (inherited) .....	65
hint_caption (inherited) .....	66
label_class (inherited).....	66
major_ticks (inherited).....	67
max_extreme (inherited).....	67
min_extreme (inherited) .....	68
minor_ticks (inherited).....	68
size (inherited) .....	69
stagger_ticks (inherited).....	69
visible (inherited).....	70
zero_plane (inherited).....	70
Methods .....	71

addBin (inherited).....	71
getExtremes (inherited).....	72
getZoom (inherited).....	72
getZoomBoxCoordinates (inherited).....	72
hide (inherited).....	72
hideGrid (inherited).....	73
pixelToPoint (inherited).....	73
pointToPixel (inherited).....	73
removeBin (inherited).....	73
resetZoom (inherited).....	74
setCaption (inherited).....	74
setCrosshair (inherited).....	74
setExtremes (inherited).....	74
setZoom (inherited).....	75
show (inherited).....	75
showGrid (inherited).....	75
Events.....	75
onHideCrosshair (inherited).....	75
onHideCursorPosition (inherited).....	76
onNeedsTicks (inherited).....	76
onShowCrosshair (inherited).....	77
onShowCursorPosition (inherited).....	77
<b>LogarithmicAxis</b> .....	<b>77</b>
Properties.....	78
background (inherited).....	78
base.....	78
border (inherited).....	79
caption (inherited).....	79
caption_class (inherited).....	80
color (inherited).....	80
crosshair (inherited).....	81
cursor_position (inherited).....	81
extremes_ticks (inherited).....	82
force_static_points (inherited).....	82
formatter (inherited).....	83
grid (inherited).....	83
hint_caption.....	84
label_class (inherited).....	84
major_ticks (inherited).....	85
max_extreme (inherited).....	86
min_extreme (inherited).....	86
minor_ticks (inherited).....	87
size (inherited).....	87
stagger_ticks (inherited).....	88
visible (inherited).....	88
zero_plane (inherited).....	89
Methods.....	89
addBin (inherited).....	89
getExtremes (inherited).....	90
getZoom (inherited).....	90
getZoomBoxCoordinates (inherited).....	90
hide (inherited).....	91
hideGrid (inherited).....	91
pixelToPoint (inherited).....	91

pointToPixel (inherited).....	91
removeBin (inherited).....	92
resetZoom (inherited).....	92
setCaption (inherited).....	92
setCrosshair (inherited).....	93
setExtremes (inherited).....	93
setZoom (inherited).....	93
show (inherited).....	93
showGrid (inherited).....	94
Events .....	94
onHideCrosshair (inherited).....	94
onHideCursorPosition (inherited).....	94
onNeedsTicks (inherited).....	94
onShowCrosshair (inherited).....	95
onShowCursorPosition (inherited).....	95
<b>4 Series Types .....</b>	<b>96</b>
<b>EJSC.AnalogGaugeSeries .....</b>	<b>96</b>
Properties .....	97
anchor .....	97
axis .....	97
delayLoad (inherited).....	98
fillColor .....	98
fillOpacity .....	98
height .....	98
label .....	99
legendsVisible (inherited).....	99
lock .....	99
marker_position.....	100
max .....	100
min .....	100
minorTick .....	101
needle .....	101
position .....	102
range .....	102
range_degrees .....	103
ranges .....	103
start_degree .....	103
tick .....	103
tickCount .....	104
title (inherited) .....	104
visible (inherited).....	104
width .....	105
x_axis_formatter (inherited).....	105
Methods .....	105
getDataHandler (inherited).....	105
getVisibility (inherited).....	105
hide (inherited) .....	106
hideLegend (inherited).....	106
reload (inherited).....	106
setDataHandler (inherited).....	106
setTitle (inherited).....	106
show (inherited).....	107
showLegend (inherited).....	107
Events .....	107

onAfterDataAvailable (inherited) .....	107
onAfterVisibilityChange (inherited) .....	107
onBeforeVisibilityChange (inherited) .....	108
Data Formats .....	108
<b>EJSC.AreaSeries .....</b>	<b>109</b>
Properties .....	110
autosort (inherited) .....	110
closeLine .....	110
color (inherited) .....	110
coloredLegend (inherited) .....	111
delayLoad (inherited) .....	111
drawPoints (inherited) .....	111
hint_string (inherited) .....	111
legendsVisible (inherited) .....	112
lineOpacity (inherited) .....	112
lineWidth (inherited) .....	112
opacity (inherited) .....	112
pointBorderColor (inherited) .....	113
pointBorderSize (inherited) .....	113
pointColor (inherited) .....	113
pointSize (inherited) .....	113
title (inherited) .....	114
visible (inherited) .....	114
x_axis (inherited) .....	114
x_axis_formatter (inherited) .....	114
y_axis .....	115
y_axis_formatter (inherited) .....	115
Methods .....	115
getDataHandler (inherited) .....	115
getVisibility (inherited) .....	115
hide (inherited) .....	115
hideLegend (inherited) .....	116
reload (inherited) .....	116
setColor (inherited) .....	116
setColoredLegend (inherited) .....	116
setDataHandler (inherited) .....	117
setLineWidth (inherited) .....	117
setOpacity (inherited) .....	117
setTitle (inherited) .....	117
show (inherited) .....	117
showLegend (inherited) .....	118
Events .....	118
onAfterDataAvailable (inherited) .....	118
onAfterVisibilityChange (inherited) .....	118
onBeforeVisibilityChange (inherited) .....	118
onShowHint (inherited) .....	119
Data Formats .....	119
Text Replacement Options .....	120
<b>EJSC.BarSeries .....</b>	<b>120</b>
Properties .....	121
autosort (inherited) .....	121
color (inherited) .....	121
coloredLegend (inherited) .....	122
defaultColors .....	122



delayLoad (inherited).....	122
groupedBars .....	122
hint_string (inherited).....	123
intervalOffset .....	123
legendsVisible (inherited).....	124
lineOpacity (inherited).....	124
lineWidth (inherited).....	124
opacity (inherited).....	124
orientation .....	125
ranges .....	125
title (inherited) .....	126
treeLegend .....	126
useColorArray .....	126
visible (inherited).....	127
x_axis (inherited).....	127
x_axis_formatter (inherited).....	127
y_axis (inherited).....	127
y_axis_formatter (inherited).....	127
Methods .....	128
addRange .....	128
clearRanges .....	128
deleteRange .....	128
getBarSize .....	128
getDataHandler (inherited).....	129
getPoints .....	129
getVisibility (inherited).....	129
hide (inherited) .....	129
hideLegend (inherited).....	129
reload (inherited).....	130
setColor (inherited) .....	130
setColoredLegend (inherited).....	130
setDataHandler (inherited).....	130
setDefaultColors .....	131
setGroupedBars .....	131
setIntervalOffset.....	131
setLineWidth (inherited).....	131
setOpacity (inherited).....	132
setTitle (inherited).....	132
show (inherited).....	132
showLegend (inherited).....	132
Events .....	133
onAfterDataAvailable (inherited).....	133
onAfterVisibilityChange (inherited).....	133
onBarNeedsColor.....	133
onBeforeVisibilityChange (inherited).....	134
onShowHint (inherited).....	134
Data Formats .....	134
Text Replacement Options.....	136
<b>EJSC.CandlestickSeries .....</b>	<b>136</b>
Properties .....	137
autosort (inherited).....	137
color (inherited).....	137
coloredLegend (inherited).....	138
delayLoad (inherited).....	138

gain	138
hint_string (inherited)	139
intervalOffset	139
legendsVisible (inherited)	140
lineOpacity (inherited)	140
lineWidth (inherited)	140
loss	141
opacity (inherited)	141
title (inherited)	142
visible (inherited)	142
x_axis (inherited)	142
x_axis_formatter (inherited)	142
y_axis (inherited)	143
y_axis_formatter (inherited)	143
Methods	143
findClosestPoint (inherited)	143
getDataHandler (inherited)	143
getVisibility (inherited)	144
hide (inherited)	144
hideLegend (inherited)	144
reload (inherited)	144
setColor (inherited)	144
setColoredLegend (inherited)	145
setDataHandler (inherited)	145
setLineOpacity (inherited)	145
setLineWidth (inherited)	145
setOpacity (inherited)	145
setTitle (inherited)	146
show (inherited)	146
showLegend (inherited)	146
Events	146
onAfterDataAvailable (inherited)	146
onAfterVisibilityChange (inherited)	147
onBeforeVisibilityChange (inherited)	147
onShowHint (inherited)	147
Data Formats	147
Text Replacement Options	149
<b>EJSC.FloatingBarSeries</b>	<b>149</b>
Properties	150
autosort (inherited)	150
color (inherited)	150
coloredLegend (inherited)	150
defaultColors (inherited)	150
delayLoad (inherited)	151
groupedBars (inherited)	151
hint_string (inherited)	151
intervalOffset (inherited)	152
legendsVisible (inherited)	152
lineOpacity (inherited)	153
lineWidth (inherited)	153
opacity (inherited)	153
orientation (inherited)	153
ranges (inherited)	154
title (inherited)	154

treeLegend (inherited) .....	155
useColorArray (inherited) .....	155
visible (inherited) .....	155
x_axis (inherited) .....	155
x_axis_formatter (inherited) .....	156
y_axis (inherited) .....	156
y_axis_formatter (inherited) .....	156
Methods .....	156
addRange (inherited) .....	156
clearRanges (inherited) .....	157
deleteRange (inherited) .....	157
getBarSize (inherited) .....	157
getDataHandler (inherited) .....	157
getPoints (inherited) .....	157
getVisibility (inherited) .....	158
hide (inherited) .....	158
hideLegend (inherited) .....	158
reload (inherited) .....	158
setColor (inherited) .....	158
setColoredLegend (inherited) .....	159
setDataHandler (inherited) .....	159
setDefaultColors (inherited) .....	159
getGroupedBars (inherited) .....	160
setIntervalOffset (inherited) .....	160
setLineWidth (inherited) .....	160
setOpacity (inherited) .....	160
setTitle (inherited) .....	160
show (inherited) .....	160
showLegend (inherited) .....	161
Events .....	161
onAfterDataAvailable (inherited) .....	161
onAfterVisibilityChange (inherited) .....	161
onBarNeedsColor (inherited) .....	161
onBeforeVisibilityChange (inherited) .....	162
onShowHint (inherited) .....	163
Data Formats .....	163
Text Replacement Options .....	165
<b>EJSC.FunctionSeries .....</b>	<b>165</b>
Properties .....	166
color (inherited) .....	166
coloredLegend (inherited) .....	166
hint_string (inherited) .....	166
legendIsVisible (inherited) .....	167
lineOpacity (inherited) .....	167
linewidth (inherited) .....	167
title (inherited) .....	167
visible (inherited) .....	168
x_axis (inherited) .....	168
x_axis_formatter (inherited) .....	168
y_axis (inherited) .....	168
y_axis_formatter (inherited) .....	168
Methods .....	169
getVisibility (inherited) .....	169
hide (inherited) .....	169

hideLegend (inherited) .....	169
reload (inherited) .....	169
setColor (inherited) .....	170
setColoredLegend (inherited) .....	170
setLineWidth (inherited) .....	170
setTitle (inherited) .....	170
show (inherited) .....	171
showLegend (inherited) .....	171
Events .....	171
onAfterVisibilityChange (inherited) .....	171
onBeforeVisibilityChange (inherited) .....	171
onShowHint (inherited) .....	172
Text Replacement Options .....	172
<b>EJSC.LineSeries .....</b>	<b>172</b>
Properties .....	173
autosort (inherited) .....	173
color (inherited) .....	173
coloredLegend (inherited) .....	173
drawPoints .....	174
hint_string (inherited) .....	174
legendIsVisible (inherited) .....	174
lineOpacity (inherited) .....	174
linewidth (inherited) .....	175
pointBorderColor .....	175
pointBorderSize .....	175
pointColor .....	175
pointSize .....	176
title (inherited) .....	176
visible (inherited) .....	176
x_axis (inherited) .....	176
x_axis_formatter (inherited) .....	177
y_axis (inherited) .....	177
y_axis_formatter (inherited) .....	177
Methods .....	177
getDataHandler (inherited) .....	177
getVisibility (inherited) .....	178
hide (inherited) .....	178
hideLegend (inherited) .....	178
reload (inherited) .....	178
setColor (inherited) .....	179
setColoredLegend (inherited) .....	179
setDataHandler (inherited) .....	179
setLineWidth (inherited) .....	179
setTitle (inherited) .....	179
show (inherited) .....	180
showLegend (inherited) .....	180
Events .....	180
onAfterDataAvailable (inherited) .....	180
onAfterVisibilityChange (inherited) .....	180
onBeforeVisibilityChange (inherited) .....	181
onShowHint (inherited) .....	181
Data Formats .....	181
Text Replacement Options .....	182
<b>EJSC.OpenHighLowCloseSeries .....</b>	<b>183</b>

Properties .....	184
autosort (inherited) .....	184
color (inherited).....	184
coloredLegend (inherited).....	184
delayLoad (inherited).....	185
gain .....	185
hint_string (inherited).....	186
intervalOffset .....	186
legendsVisible (inherited).....	187
lineOpacity (inherited).....	187
lineWidth (inherited).....	187
loss .....	188
opacity (inherited).....	188
title (inherited) .....	189
visible (inherited).....	189
x_axis (inherited).....	189
x_axis_formatter (inherited).....	189
y_axis (inherited).....	190
y_axis_formatter (inherited).....	190
Methods .....	190
findClosestPoint (inherited).....	190
getDataHandler (inherited).....	190
getVisibility (inherited).....	190
hide (inherited) .....	191
hideLegend (inherited).....	191
reload (inherited).....	191
setColor (inherited) .....	191
setColoredLegend (inherited).....	192
setDataHandler (inherited).....	192
setLineOpacity (inherited).....	192
setLineWidth (inherited).....	192
setOpacity (inherited).....	192
setTitle (inherited).....	193
show (inherited).....	193
showLegend (inherited).....	193
Events .....	193
onAfterDataAvailable (inherited).....	193
onAfterVisibilityChange (inherited).....	194
onBeforeVisibilityChange (inherited).....	194
onShowHint (inherited).....	194
Data Formats .....	194
Text Replacement Options (COPY).....	196
<b>EJSC.PieSeries .....</b>	<b>196</b>
Properties .....	197
defaultColors .....	197
delayLoad (inherited).....	197
height .....	197
hint_string (inherited).....	197
legendsVisible (inherited).....	198
lineOpacity (inherited).....	198
lineWidth (inherited).....	198
opacity (inherited).....	198
position .....	199
title (inherited) .....	199

total_value .....	199
treeLegend .....	200
visible (inherited).....	200
width .....	200
x_axis_formatter (inherited).....	200
<b>Methods</b> .....	<b>200</b>
findCenter .....	200
findCenterOfCurve.....	201
getDataHandler (inherited) .....	201
getPoints .....	201
getTotalValue .....	202
getVisibility (inherited).....	202
hide (inherited) .....	202
hideLegend (inherited).....	202
reload (inherited).....	202
resetTotalValue.....	203
setDataHandler (inherited) .....	203
setDefaultColors .....	203
setLineWidth (inherited) .....	204
setOpacity (inherited) .....	204
setTitle (inherited) .....	204
setTotalValue .....	204
show (inherited).....	205
showLegend (inherited) .....	205
<b>Events</b> .....	<b>205</b>
onAfterDataAvailable (inherited).....	205
onAfterVisibilityChange (inherited).....	205
onBeforeVisibilityChange (inherited).....	206
onPieceNeedsColor.....	206
onShowHint (inherited).....	206
Data Formats .....	206
Text Replacement Options.....	208
<b>EJSC.ScatterSeries</b> .....	<b>208</b>
<b>Properties</b> .....	<b>209</b>
autosort (inherited).....	209
color (inherited).....	209
coloredLegend (inherited).....	209
delayLoad (inherited).....	209
hint_string (inherited).....	210
legendIsVisible (inherited).....	210
lineOpacity (inherited).....	210
linewidth (inherited) .....	210
opacity (inherited).....	211
pointSize .....	211
pointStyle .....	211
title (inherited) .....	211
visible (inherited).....	212
x_axis (inherited).....	212
x_axis_formatter (inherited).....	212
y_axis (inherited).....	212
y_axis_formatter (inherited).....	213
<b>Methods</b> .....	<b>213</b>
getDataHandler (inherited) .....	213
getVisibility (inherited).....	213

hide (inherited) .....	213
hideLegend (inherited) .....	214
reload (inherited) .....	214
setColor (inherited) .....	214
setColoredLegend (inherited) .....	214
setDataHandler (inherited) .....	214
setLineWidth (inherited) .....	215
setOpacity (inherited) .....	215
setPointStyle .....	215
setTitle (inherited) .....	215
show (inherited) .....	216
showLegend (inherited) .....	216
Events .....	216
onAfterDataAvailable (inherited) .....	216
onAfterVisibilityChange (inherited) .....	216
onBeforeVisibilityChange (inherited) .....	217
onShowHint (inherited) .....	217
Data Formats .....	217
Text Replacement Options .....	219
<b>EJSC.StackedBarSeries .....</b>	<b>219</b>
Properties .....	219
groupedBars (inherited) .....	219
hint_string (inherited) .....	219
intervalOffset (inherited) .....	220
legendsVisible (inherited) .....	220
title (inherited) .....	220
orientation (inherited) .....	221
visible (inherited) .....	221
x_axis (inherited) .....	221
y_axis (inherited) .....	221
Methods .....	222
getBarSize (inherited) .....	222
addSeries .....	222
getVisibility (inherited) .....	222
hide (inherited) .....	223
hideLegend (inherited) .....	223
reload (inherited) .....	223
removeSeries .....	223
setGroupedBars (inherited) .....	224
setIntervalOffset (inherited) .....	224
setTitle (inherited) .....	224
show (inherited) .....	224
showLegend (inherited) .....	225
Events .....	225
onAfterVisibilityChange (inherited) .....	225
onBeforeVisibilityChange (inherited) .....	225
onShowHint (inherited) .....	225
Text Replacement Options (COPY) .....	226
<b>EJSC.TrendSeries .....</b>	<b>226</b>
Properties .....	227
color (inherited) .....	227
coloredLegend (inherited) .....	227
hint_string (inherited) .....	227
legendsVisible (inherited) .....	228

lineOpacity (inherited) .....	228
lineWidth (inherited) .....	228
title (inherited) .....	228
visible (inherited).....	228
x_axis (inherited) .....	229
x_axis_formatter (inherited).....	229
y_axis (inherited) .....	229
y_axis_formatter (inherited).....	229
Methods .....	230
getVisibility (inherited).....	230
hide (inherited) .....	230
hideLegend (inherited).....	230
reload (inherited).....	230
setColor (inherited) .....	231
setColoredLegend (inherited).....	231
setLineWidth (inherited) .....	231
setTitle (inherited) .....	231
show (inherited).....	231
showLegend (inherited) .....	232
Events .....	232
onAfterVisibilityChange (inherited).....	232
onBeforeVisibilityChange (inherited).....	232
onShowHint (inherited).....	232
Text Replacement Options.....	233
<b>5 Data Handlers .....</b>	<b>233</b>
<b>EJSC.XMLDataHandler .....</b>	<b>233</b>
Properties .....	233
requestType (inherited) .....	233
url (inherited) .....	234
urlData (inherited) .....	234
Methods .....	234
getUrl (inherited) .....	234
loadData (inherited).....	235
setRequestType (inherited) .....	235
setUrl (inherited) .....	235
setUrlData (inherited) .....	235
setXMLData (inherited) .....	236
Events .....	236
onDataAvailable (inherited).....	236
onDataReady (inherited) .....	236
onNeedsData (inherited) .....	236
<b>EJSC.XMLStringDataHandler .....</b>	<b>237</b>
Methods .....	238
getXML .....	238
loadData (inherited).....	238
setXML .....	238
Events .....	238
onDataAvailable (inherited).....	238
<b>EJSC.ArrayDataHandler .....</b>	<b>239</b>
Methods .....	239
getArray .....	239
loadData (inherited).....	239
setArray .....	239
Events .....	240



onDataAvailable (inherited) .....	240
<b>EJSC.CSVFileDataHandler .....</b>	<b>240</b>
Properties .....	240
requestType (inherited) .....	240
url (inherited) .....	240
urlData (inherited) .....	241
Methods .....	241
getUrl (inherited) .....	241
loadData (inherited).....	241
setRequestType (inherited) .....	242
setUrl (inherited) .....	242
setUrlData (inherited) .....	242
setXMLData (inherited) .....	242
Events .....	243
onDataAvailable (inherited) .....	243
onDataReady (inherited) .....	243
onNeedsData (inherited) .....	243
<b>EJSC.CSVStringDataHandler .....</b>	<b>244</b>
Methods .....	244
getCSV .....	244
loadData (inherited).....	244
setCSV .....	245
Events .....	245
onDataAvailable (inherited) .....	245
<b>EJSC.JSONFileDataHandler .....</b>	<b>245</b>
Properties .....	245
requestType (inherited) .....	245
url (inherited) .....	246
urlData (inherited) .....	246
Methods .....	246
getUrl (inherited) .....	246
loadData (inherited).....	247
setRequestType (inherited) .....	247
setUrl (inherited) .....	247
setUrlData (inherited) .....	247
setXMLData (inherited) .....	248
Events .....	248
onDataAvailable (inherited) .....	248
onDataReady (inherited) .....	248
onNeedsData (inherited) .....	249
<b>EJSC.JSONStringDataHandler .....</b>	<b>249</b>
Methods .....	249
getJSON .....	249
loadData (inherited).....	250
setJSON .....	250
Events .....	250
onDataAvailable (inherited) .....	250
<b>6 Label Formatters .....</b>	<b>250</b>
<b>EJSC.NumberFormatter .....</b>	<b>250</b>
Properties .....	251
currency_align .....	251
currency_position.....	251
currency_symbol.....	251
decimal_separator .....	251

forced_decimals .....	252
negative_symbol .....	252
thousand_separator .....	252
variable_decimals .....	252
Methods .....	253
format (inherited) .....	253
<b>EJSC.DateFormatter .....</b>	<b>253</b>
Properties .....	253
format_string .....	253
timezoneOffset .....	254
useUTC .....	255
Methods .....	255
format (inherited) .....	255
<b>EJSC.StringFormatter .....</b>	<b>255</b>
Properties .....	255
append .....	255
prefix .....	256
Methods .....	256
format (inherited) .....	256
Events .....	256
onNeedsFormat .....	256
<b>7 Base Classes .....</b>	<b>257</b>
<b>EJSC.Inheritable .....</b>	<b>257</b>
<b>EJSC.AjaxDataHandler .....</b>	<b>257</b>
Properties .....	257
requestType .....	257
url .....	257
urlData .....	257
Methods .....	258
getUrl .....	258
loadData (inherited) .....	258
setRequestType .....	258
setUrl .....	259
setUrlData .....	259
setXMLData .....	259
Events .....	260
onDataAvailable (inherited) .....	260
onDataReady .....	260
onNeedsData .....	260
<b>EJSC.Axis .....</b>	<b>261</b>
Properties .....	261
background .....	261
border .....	261
caption .....	262
caption_class .....	262
color .....	263
crosshair .....	263
cursor_position .....	264
extremes_ticks .....	264
force_static_points .....	265
formatter .....	265
grid .....	266
hint_caption .....	266
label_class .....	267

major_ticks .....	267
max_extreme .....	268
minor_ticks .....	268
min_extreme .....	269
size .....	270
stagger_ticks .....	270
visible .....	271
zero_plane .....	271
Methods .....	272
addBin .....	272
getExtremes .....	272
getZoom .....	273
getZoomBoxCoordinates .....	273
hide .....	273
hideGrid .....	273
resetZoom .....	274
pixelToPoint .....	274
pointToPixel .....	274
removeBin .....	274
setCaption .....	275
setCrosshair .....	275
setExtremes .....	275
setZoom .....	275
show .....	276
showGrid .....	276
Events .....	276
onHideCrosshair .....	276
onHideCursorPosition .....	276
onNeedsTicks .....	277
onShowCrosshair .....	278
onShowCursorPosition .....	278
<b>EJSC.DataHandler .....</b>	<b>278</b>
Methods .....	278
loadData .....	278
Events .....	278
onDataAvailable .....	278
<b>EJSC.GaugeSeries .....</b>	<b>279</b>
Properties .....	279
color (inherited) .....	279
coloredLegend (inherited) .....	279
delayLoad (inherited) .....	279
legendIsVisible (inherited) .....	280
lineOpacity (inherited) .....	280
lineWidth (inherited) .....	280
opacity (inherited) .....	280
title (inherited) .....	281
visible (inherited) .....	281
x_axis_formatter (inherited) .....	281
Methods .....	281
getDataHandler (inherited) .....	281
getVisibility (inherited) .....	282
hide (inherited) .....	282
hideLegend (inherited) .....	282
reload (inherited) .....	282

setColor (inherited) .....	282
setColoredLegend (inherited) .....	283
setDataHandler (inherited) .....	283
setLineOpacity (inherited) .....	283
setLineWidth (inherited) .....	283
setOpacity (inherited) .....	283
setTitle (inherited) .....	284
show (inherited) .....	284
showLegend (inherited) .....	284
Events .....	284
onAfterDataAvailable (inherited) .....	284
onAfterVisibilityChange (inherited) .....	285
onBeforeVisibilityChange (inherited) .....	285
<b>EJSC.Formatter .....</b>	<b>285</b>
Properties .....	285
format_string .....	285
Methods .....	286
format .....	286
<b>EJSC.Point .....</b>	<b>286</b>
Properties .....	286
label .....	286
userdata .....	286
x .....	287
y .....	287
<b>EJSC.Series .....</b>	<b>287</b>
Properties .....	287
autosort .....	287
color .....	287
coloredLegend .....	288
delayLoad .....	288
hint_string .....	288
legendsVisible .....	289
lineOpacity .....	289
lineWidth .....	289
opacity .....	289
title .....	290
visible .....	290
x_axis .....	290
x_axis_formatter .....	290
y_axis .....	291
y_axis_formatter .....	291
Methods .....	291
findClosestPoint .....	291
getDataHandler .....	291
getVisibility .....	291
hide .....	292
hideLegend .....	292
reload .....	292
setColor .....	292
setColoredLegend .....	293
setDataHandler .....	293
setLineOpacity .....	293
setLineWidth .....	293
setOpacity .....	293

setTitle .....	294
show .....	294
showLegend .....	294
Events .....	294
onAfterDataAvailable.....	294
onAfterVisibilityChange .....	295
onBeforeVisibilityChange.....	295
onShowHint .....	295
<b>8 Other Classes .....</b>	<b>295</b>
<b>EJSC.BarPoint .....</b>	<b>295</b>
Properties .....	296
label (inherited) .....	296
userdata (inherited) .....	296
x (inherited) .....	296
y (inherited) .....	296
<b>EJSC.FloatingBarPoint .....</b>	<b>296</b>
Properties .....	297
label (inherited) .....	297
max .....	297
min .....	297
userdata (inherited) .....	297
x (inherited) .....	298
y (inherited) .....	298
<b>EJSC.GaugePoint .....</b>	<b>298</b>
Properties .....	298
label (inherited) .....	298
userdata (inherited) .....	299
x (inherited) .....	299
<b>EJSC.PiePoint .....</b>	<b>299</b>
Properties .....	299
label (inherited) .....	299
userdata (inherited) .....	300
x (inherited) .....	300
<b>EJSC.StockPoint .....</b>	<b>300</b>
Properties .....	300
close .....	300
high .....	300
label (inherited) .....	300
low .....	301
open .....	301
userdata (inherited) .....	301
x (inherited) .....	301
<b>EJSC.XYPoint .....</b>	<b>301</b>
Properties .....	302
label (inherited) .....	302
userdata (inherited) .....	302
x (inherited) .....	302
y (inherited) .....	302
<b>9 Using Colors .....</b>	<b>302</b>
<b>10 Text Replacement Options .....</b>	<b>303</b>
<b>11 Exporting To SVG .....</b>	<b>303</b>

**Part VI Getting Support****304**

# 1 Getting Started

## 1.1 Overview

Welcome to Emprise JavaScript Charts. Constructed entirely in JavaScript the days of annoying plugin downloads and browser security warnings are gone. With genuine ease of use and complete customization Emprise JavaScript Charts provides you with the tools you need to publish your data quickly and in a variety of formats. With its wide range of interactive features, simple and straightforward implementation, and unparalleled functionality, Emprise JavaScript Charts is the clear first choice for all your charting needs.

Here's a quick sampling of just some of the features included:

- **Interactive:** Features such as Hints, Mouse Tracking, Mouse Events, Key Tracking and Events, Zooming, Scrolling, and Crosshairs raise interactivity and user experience in web charting to a new level.
- **Axis Scaling:** There's no need to determine your data range before hand. EJSChart will calculate and scale automatically to fit whatever data it is presented with.
- **Auto Zooming, Scrolling:** Too much data and not enough screen real estate? Show it all. Let your end users zoom in on the pieces they're most interested in. Axis locking for single axis zoom, scrolling and automatic axis scaling are all included.
- **Stackable Series:** Multiple chart series can be stacked and combined to fit many charting needs.
- **Multiple Series Types:** Line, Area, Scatter, Pie, Bar and Function series are just the beginning. New series are just a few lines of JavaScript code away.
- **Ajax-Driven Data:** EJSChart supports XML-formatted data and loads data on the fly. New

series can be added and data updated in real time without page reloads.

- **Compatible:** Built with compatibility in mind and tested on all major browsers, you can be assured your charts will function consistently for the broadest range of end users. See the full list of compatible browsers on our System Requirements page.
- **Plugin Free:** 100% Pure JavaScript Charting Solution. No more worries of incompatible plugin versions or confusing security warnings. EJSChart is pure JavaScript and requires no client installation.
- **Customizable:** Every aspect of the charting display can be configured and customized through well-documented properties and methods. Want to do more than just change the color of the background? Need a series type which doesn't already exist? EJSChart is fully customizable and extendable to provide the greatest flexibility and integration for existing site designs and needs.

## 1.2 Implementation Instructions

Ease of implementation was an important factor considered in the development of EJSChart. The process from purchase to user “wow” takes just a matter of minutes and can be completed following the few easy to follow steps detailed below. Designed for all users, no previous JavaScript knowledge is required to implement EJSChart or take advantage of its many features. The steps below will guide you through the installation and testing of the EJSChart library.

1. Create a new directory in the home directory of your webpage and name it EJSChart.
2. Copy the contents (all files and directories) of the `/dist/` folder provided in your EJSChart download package into your newly created EJSChart folder on your web server.
3. Open the webpage that you wish to place an example chart on. If you do not currently have one or wish to use a test page for this purpose one has been provided for you in the How To directory included with your download. This file is named `mydemochart.html` and can be edited in your preferred html editor or even notepad.
4. Copy `<script type="text/javascript"`



`src="/EJSChart/EJSChart.js"></script>` into the head section of your webpage.

\*This path can be modified as necessary to correctly point to EJSChart.js.

```
<!doctype html public "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
  <head>
    <title>My Demo Chart</title>
    <meta name="Author" content="Emprise Corporation">
    <meta name="Description" content="EJSChart Demo Chart HTML Page">
    <script type="text/javascript" src="/EJSChart/EJSChart.js"></script>
  </head>
  <body>
    <p>This is a sample page that is used to copy and paste a demo chart
into.</p>
  </body>
</html>
```

5. Create a div on your page where you would like the chart to be displayed. The size of the chart can be specified within the div properties if desired. The div is created by pasting `<div id="myChart" style="width:450px; height:330px;"></div>` where desired in the body of your webpage. The style width and height properties can be adjusted as desired.

```
<!doctype html public "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
  <head>
    <title>My Demo Chart</title>
    <meta name="Author" content="Emprise Corporation">
    <meta name="Description" content="EJSChart Demo Chart HTML Page">
    <script type="text/javascript" src="/EJSChart/EJSChart.js"> </script>
  </head>
  <body>
    <p>This is a sample page that is used to copy and paste a demo chart
into.</p>
    <div id="myChart" style="width:450px; height:330px;"></div>
  </body>
</html>
```

6. To turn the div into a chart paste the following sample code at the end of your webpage.

```
<script type="text/javascript">
  var chart = new EJSC.Chart("myChart");
  chart.addSeries(new EJSC.AreaSeries(
    new EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,2],[5,3]]))
  );
</script>
```

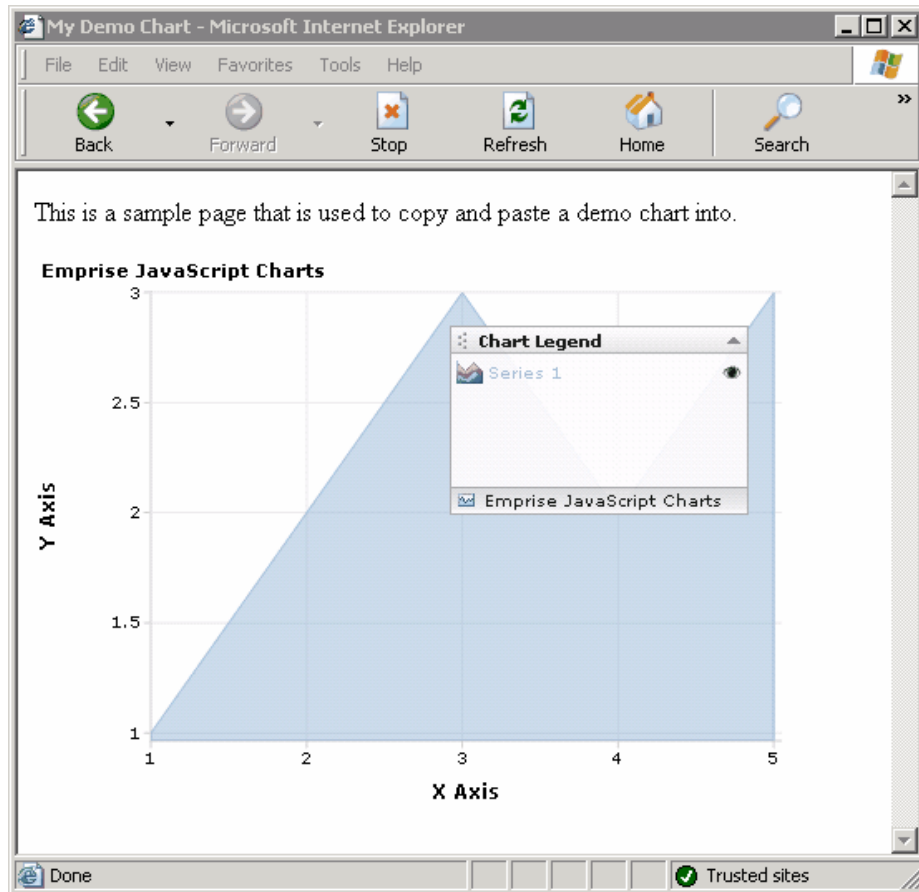
```
<!doctype html public "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
  <head>
    <title>My Demo Chart</title>
    <meta name="Author" content="Emprise Corporation">
    <meta name="Description" content="EJSChart Demo Chart HTML Page">
    <script type="text/javascript" src="/EJSChart/EJSChart.js"> </script>
  </head>
  <body>
    <p>This is a sample page that is used to copy and paste a demo chart
      into.</p>

    <div id="myChart" style="width:450px; height:330px;"></div>

    <script type="text/javascript">
      var chart = new EJSC.Chart("myChart");
      chart.addSeries(new EJSC.AreaSeries(
        new EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,2],[5,3]]))
      );
    </script>
  </body>
</html>
```

7. Upload your edited webpage to your web server. Visit your webpage and view the demo

chart located on the page where the div was placed.



### 1.3 Creating Your First Chart

Now that you have created and tested your first demonstration chart on your web server it is time to create your own chart. This demonstration will guide you through creating your first chart and introduce you to some of the many available features and methods of customization available to you in Emprise JavaScript Charts. If you have not yet completed the [Implementation Instructions](#) it is recommended that you do so before continuing with this section as they guide you through uploading the necessary files to your web server.

1. Open the webpage that you wish to place your first custom chart on. If you do not currently have one or wish to use a test page for this purpose one has been provided for you in the

HowTo directory included with your download. This file is named myfirstchart.html and can be edited in your preferred html editor or even notepad.

2. Copy `<script type="text/javascript"`

`src="/EJSChart/EJSChart.js"></script>` into the head section of your webpage.

\*This path can be modified as necessary to correctly point to EJSChart.js.

```
<!doctype html public "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
  <head>
    <title>My Demo Chart</title>
    <meta name="Author" content="Emprise Corporation">
    <meta name="Description" content="Emprise JavaScript Charts ::
Customization Example">
    <script type="text/javascript" src="/EJSChart/EJSChart.js"> </script>
  </head>
  <body>
    <p>This is a sample page that is used to copy and paste a demo chart
into.</p>
  </body>
</html>
```

3. Create a div on your page where you would like the chart to be displayed by pasting

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

where desired in the body of your webpage.

```
<!doctype html public "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
  <head>
    <title>My Demo Chart</title>
    <meta name="Author" content="Emprise Corporation">
    <meta name="Description" content="Emprise JavaScript Charts ::
Customization Example">
    <script type="text/javascript" src="/EJSChart/EJSChart.js"> </script>
  </head>
  <body>
    <p>This is a sample page that is used to copy and paste a demo chart
into.</p>
    <div id="myFirstChart" style="width:600px; height:400px;"></div>
  </body>
</html>
```

4. The next step is to create the chart object. This is done at the end of your html file, directly

before the closing body tag; `</body>`. To begin define the text as JavaScript with `<script type="text/javascript">` and create a chart object on the next line using `var chart =`

`new EJSC.Chart("myFirstChart");` Be sure that the div id as identified within the

body of your webpage matches the title you place within the parenthesis when creating your chart. In this example it is 'myFirstChart'. Failure to create the appropriate div's for the chart to be placed in will result in errors and prevent your chart from being displayed.

The properties of your chart including its visual appearance and interactivity are highly customizable by adjusting its properties. Instructions on how to accomplish this can be found on the [Customizing Your Chart](#) page of this guide.

In this example code the chart has been customized to remove the legend and set a custom title by adding:

```
<script type="text/javascript">
  var chart = new EJSC.Chart(
    "myFirstChart",
    {
      show_legend: false,
      title: "My First Custom Chart"
    }
  );
```

```
<!doctype html public "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
  <head>
    <title>My Demo Chart</title>
    <meta name="Author" content="Emprise Corporation">
    <meta name="Description" content="Emprise JavaScript Charts ::
Customization Example">
    <script type="text/javascript" src="/EJSCart/EJSCart.js"> </script>
  </head>
  <body>
    <p>This is a sample page that is used to copy and paste a demo chart
into.</p>

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

    <script type="text/javascript">
      var chart = new EJSC.Chart(
        "myFirstChart",
        {
          show_legend: false,
          title: "My First Custom Chart"
        }
      );
    </script>
  </body>
</html>
```

5. Now that the chart has been created the series must be created. The series defines how

your data will be displayed in the chart you are creating. The types of series that are available may vary by license; more details on this can be found on the LICENSE.txt file.

When creating the chart you can also choose which format you will be providing the data in. This is accomplished with the use of different Data Handlers. The formats currently supported by EJSCart are XML file, JavaScript Array, CSV file, and CSV string data. Details on the implementation of each of these data handlers as well as sample code can be found on their respective pages in the Developer Guide help file.

The properties of your series, including its visual appearance and interactivity, are customized by adjusting its properties. Instructions on how to accomplish this can be found on the [Customizing Your Chart](#) page.

In this example code a bar chart was created using the Bar Series, the color was set to green, and the bar border width was set to five pixels. The data to be graphed was specified using the EJSC.ArrayDataHandler. To accomplish this the following code was added:

```
var myChartSeries = new EJSC.BarSeries(  
    new EJSC.ArrayDataHandler(  
        [  
            ["Month 1",1],  
            ["Month 2",2],  
            ["Month 3",3],  
            ["Month 4",4],  
            ["Month 5",5]  
        ]  
    )  
);  
myChartSeries.color = 'rgb(50,210,50)';  
myChartSeries.lineWidth = 5;
```

```
<!doctype html public "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
  <head>
    <title>My Demo Chart</title>
    <meta name="Author" content="Emprise Corporation">
    <meta name="Description" content="Emprise JavaScript Charts ::
Customization Example">
    <script type="text/javascript" src="/EJSChart/EJSChart.js"> </script>
  </head>
  <body>
    <p>This is a sample page that is used to copy and paste a demo chart
into.</p>

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

    <script type="text/javascript">

      var chart = new EJSC.Chart(
        "myFirstChart",
        {
          show_legend: false,
          title: "My First Custom Chart"
        }
      );

      var myChartSeries = new EJSC.BarSeries(
        new EJSC.ArrayDataHandler(
          [
            ["Month 1",1],
            ["Month 2",2],
            ["Month 3",3],
            ["Month 4",4],
            ["Month 5",5]
          ]
        )
      );
      myChartSeries.color = 'rgb(50,210,50)';
      myChartSeries.lineWidth = 5;

    </body>
  </html>
```

6. Once created the series must then be added to the chart using the [addSeries](#) method in the chart class.

```
<!doctype html public "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
  <head>
    <title>My Demo Chart</title>
    <meta name="Author" content="Emprise Corporation">
    <meta name="Description" content="Emprise JavaScript Charts ::
Customization Example">
    <script type="text/javascript" src="/EJSChart/EJSChart.js"> </script>
  </head>
  <body>
    <p>This is a sample page that is used to copy and paste a demo chart
into.</p>

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

    <script type="text/javascript">

      var chart = new EJSC.Chart(
        "myFirstChart",
        {
          show_legend: false,
          title: "My First Custom Chart"
        }
      );

      var myChartSeries = new EJSC.BarSeries(
        new EJSC.ArrayDataHandler(
          [
            ["Month 1",1],
            ["Month 2",2],
            ["Month 3",3],
            ["Month 4",4],
            ["Month 5",5]
          ]
        )
      );
      myChartSeries.color = 'rgb(50,210,50)';
      myChartSeries.lineWidth = 5;

      chart.addSeries(myChartSeries);

    </body>
  </html>
```



## 7. Close the JavaScript by inserting `</script>` at the end.

```
<!doctype html public "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
  <head>
    <title>My Demo Chart</title>
    <meta name="Author" content="Emprise Corporation">
    <meta name="Description" content="Emprise JavaScript Charts ::
Customization Example">
    <script type="text/javascript" src="/EJSCart/EJSCart.js"> </script>
  </head>
  <body>
    <p>This is a sample page that is used to copy and paste a demo chart
into.</p>

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

    <script type="text/javascript">

      var chart = new EJSC.Chart(
        "myFirstChart",
        {
          show_legend: false,
          title: "My First Custom Chart"
        }
      );

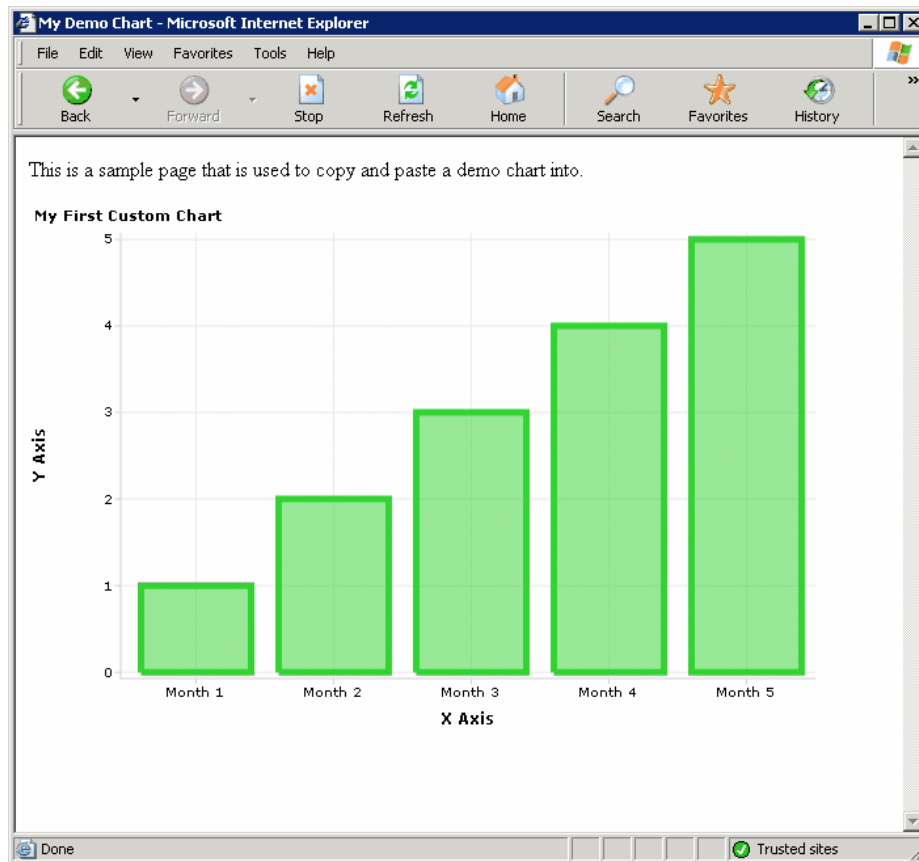
      var myChartSeries = new EJSC.BarSeries(
        new EJSC.ArrayDataHandler(
          [
            ["Month 1",1],
            ["Month 2",2],
            ["Month 3",3],
            ["Month 4",4],
            ["Month 5",5]
          ]
        )
      );
      myChartSeries.color = 'rgb(50,210,50)';
      myChartSeries.lineWidth = 5;

      chart.addSeries(myChartSeries);

    </script>

  </body>
</html>
```

8. That's it! Upload the webpage to your web server and you're done.



## 1.4 Customizing Your Chart

With Emprise JavaScript Charts, the customization options of your chart are endless. This includes visual appearance, which can be modified to integrate fully with any theme or design, as well as chart interactivity, which can range from including user capabilities such as auto zooming and custom hint captions to hidden-axis view only chart displays. Your chart can be customized on the chart and series level via the modification of the chart and series properties.

The first customization options available to you are at the chart level. They can be specified when creating the chart or alternatively after the chart object has been established. The properties available for editing and their syntax can be found in the [Chart Properties](#) section of the help documentation. In addition, examples of their implementation with sample code and the resulting effect on chart display or interactivity are available on the in the /examples/ directory of the distribution package. There are also additional examples available online at

<http://www.ejschart.com/examples/>

The following are a few quick examples of modifying commonly used properties both at and after chart creation:

- Modification of the bottom x and left y axis labels during chart creation:

```
var chart = new EJSC.Chart("myChart", {
    axis_bottom: {
        caption: "Month"
    },
    axis_left: {
        caption: "Temperature"
    }
});
```

- Modification of the bottom x and left y axis labels after chart creation:

```
var chart = new EJSC.Chart("myChart");
chart.axis_bottom.setCaption("Month");
chart.axis_left.setCaption("Temperature");
```

The properties of each series on a chart may also be customized. This is accomplished with the editing of properties in the same way as was done to customize chart properties; either at the time of series creation or following creation. The properties available for editing and their syntax can be found on the [Properties](#) page of each of the different series help sections. In addition, examples of their implementation with sample code and the resulting effect on chart display or interactivity are available on the in the /examples/ directory of the distribution package. There are also additional examples available online at <http://www.ejschart.com/examples/>

The following are a few quick examples of modifying commonly used properties both at and after chart creation:

- Modification of the series lineWidth property during creation:

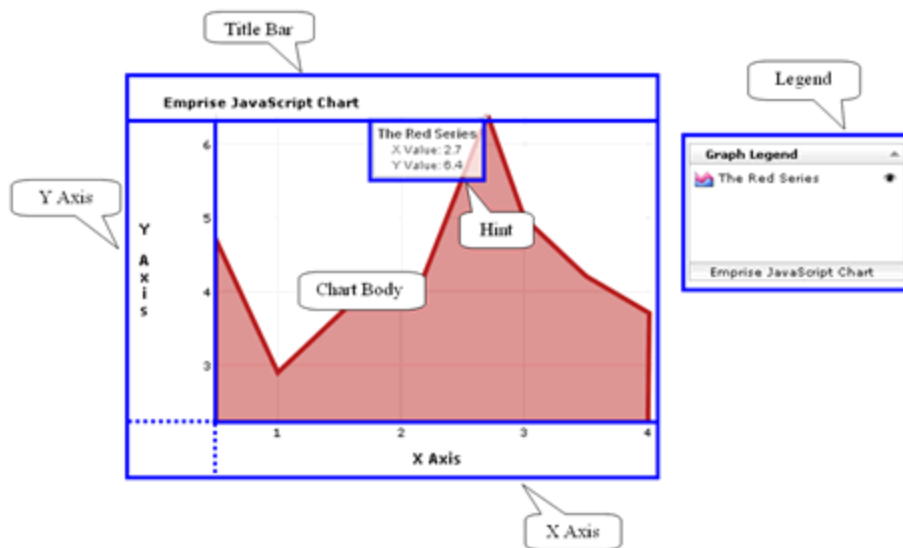
```
var myChartSeries = new EJSC.FunctionSeries(Math.sin, {lineWidth:  
4});
```

- Modification of the lineWidth property after creation:

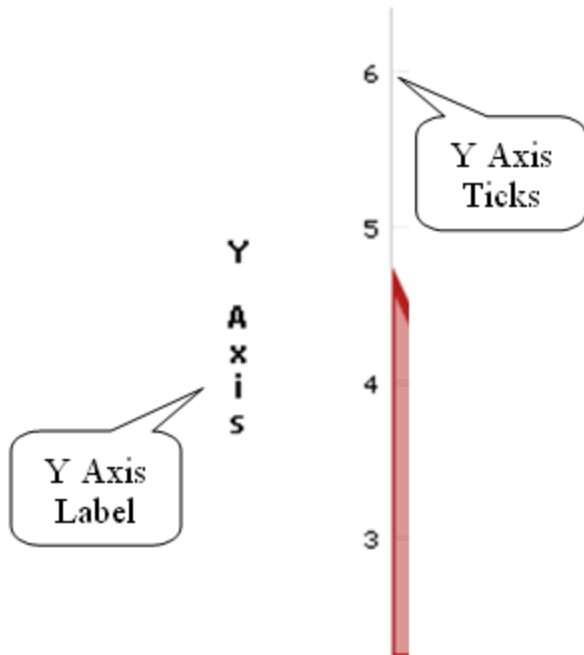
```
var myChartSeries = new EJSC.FunctionSeries(Math.sin);  
myChartSeries.setLineWidth(4);
```

In order to fully understand the power of the properties available for customizing it is important to become familiar with the various components of the chart. The following diagram identify many of these components:

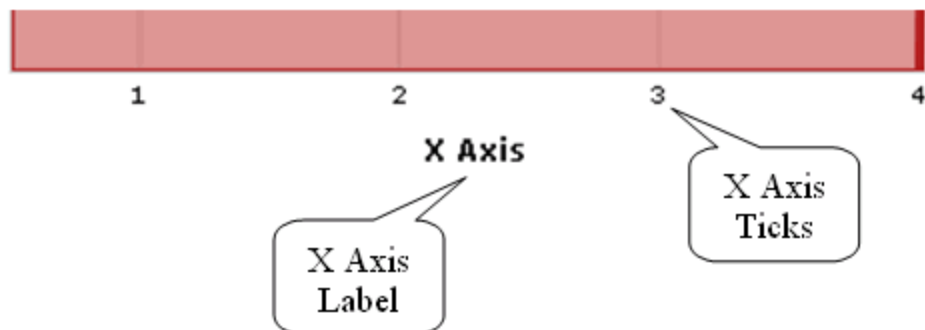
### Anatomy of a Chart



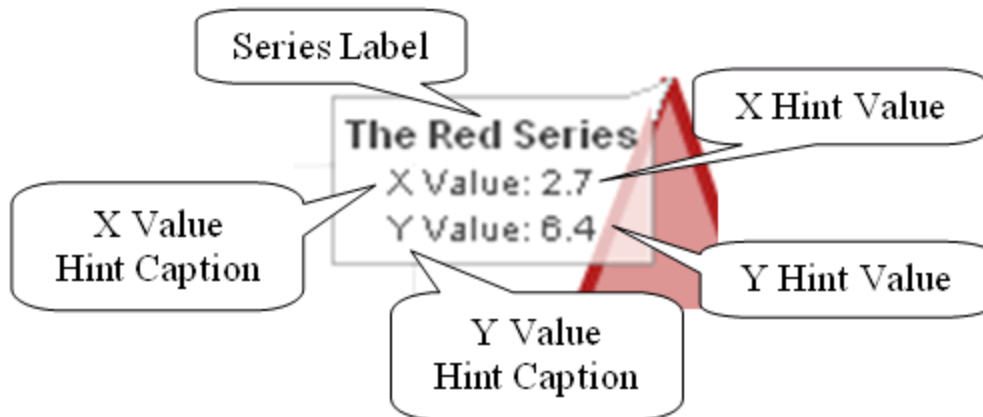
### 1.4.1 Y Axis



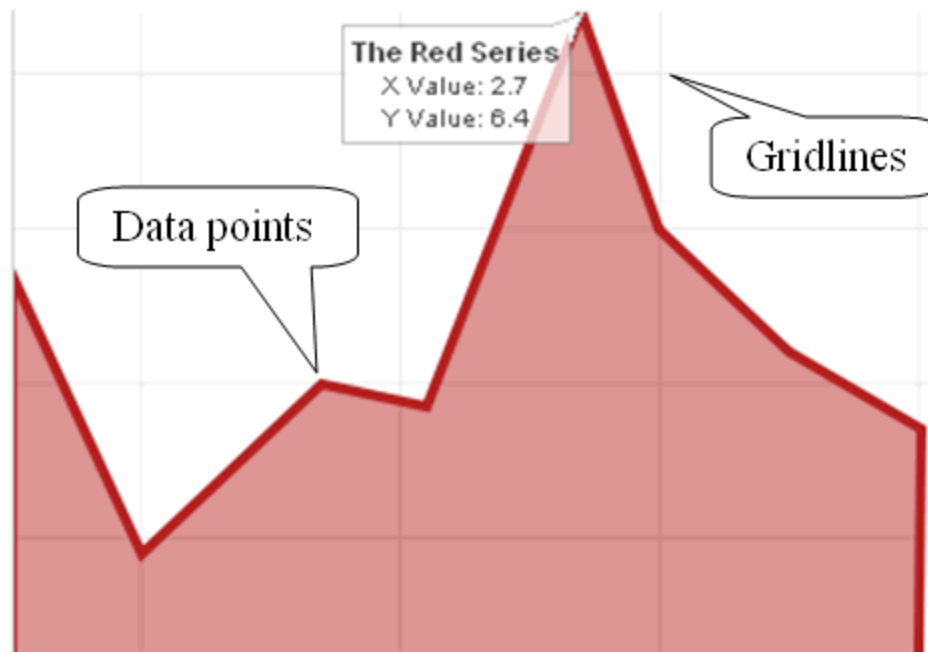
### 1.4.2 X Axis



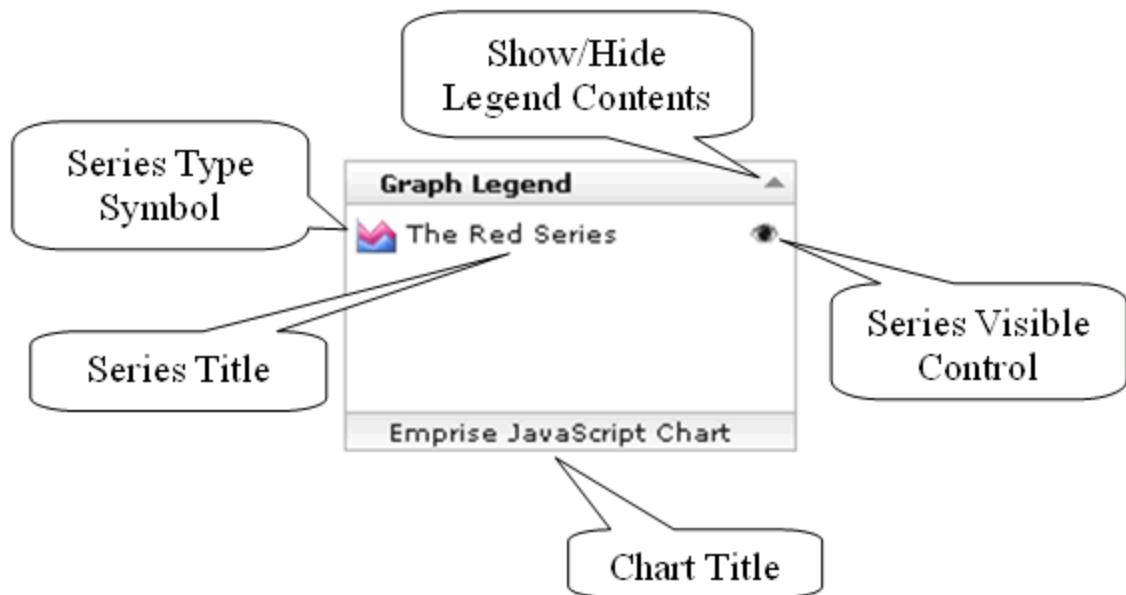
### 1.4.3 Hint



### 1.4.4 Chart Body



## 1.4.5 Legend



## 1.5 Version 1.x Compatibility

### 1.5.1 Version 1.x Compatibility

We have moved a significant number of properties and methods into separate classes to support multiple axes and new series types. You may find that properties used in charts you have already developed no longer have the effect expected on your charts. To help with the transition to the new code base we have included a version 1 compatibility JavaScript file.

To have this code included automatically into your page, simply put the following tag in the page header BEFORE you include EJSChart.js:

```
<head>
  <meta name="ejsc-v1-compatibility" content="true"/>
  <script type="text/javascript" src="/EJSChart/EJSChart.js"></script>
</head>
```

This utility script will automatically convert the deprecated properties to their version 2 counterparts.

In addition, if you are using a browser, browser extension, or other JavaScript package which implements window.console, the script will log the deprecated properties being converted in order to aid you in upgrading your code.

### 1.5.2 Deprecated Properties, Methods and Events

The following lists all properties, methods and events which have been deprecated in version 2.0 and are no longer available unless the [v1 Compatibility Script](#) is included.

**Properties**

[force\\_static\\_points](#)  
[force\\_static\\_points\\_x](#)  
[force\\_static\\_points\\_y](#)  
[legendTitle](#)  
[show\\_crosshairs](#)  
[show\\_grid](#)  
[show\\_mouse\\_position](#)  
[show\\_x\\_axis](#)  
[show\\_y\\_axis](#)  
[x\\_axis\\_caption](#)  
[x\\_axis\\_className](#)  
[x\\_axis\\_extremes\\_ticks](#)  
[x\\_axis\\_formatter](#)  
[x\\_axis\\_max\\_tick\\_interval](#)  
[x\\_axis\\_min\\_tick\\_interval](#)  
[x\\_axis\\_minor\\_ticks](#)  
[x\\_axis\\_size](#)  
[x\\_axis\\_stagger\\_ticks](#)  
[x\\_axis\\_tick\\_className](#)  
[x\\_axis\\_tick\\_count](#)  
[x\\_cursor\\_position\\_caption](#)  
[x\\_cursor\\_position\\_formatter](#)  
[x\\_max](#)  
[x\\_min](#)  
[x\\_value\\_hint\\_caption](#)  
[x\\_zero\\_plane](#)  
[y\\_axis\\_caption](#)  
[y\\_axis\\_className](#)  
[y\\_axis\\_extremes\\_ticks](#)  
[y\\_axis\\_formatter](#)  
[y\\_axis\\_max\\_tick\\_interval](#)  
[y\\_axis\\_min\\_tick\\_interval](#)  
[y\\_axis\\_minor\\_ticks](#)  
[y\\_axis\\_size](#)  
[y\\_axis\\_tick\\_className](#)  
[y\\_axis\\_tick\\_count](#)  
[y\\_cursor\\_position\\_caption](#)  
[y\\_cursor\\_position\\_formatter](#)  
[y\\_max](#)  
[y\\_min](#)  
[y\\_value\\_hint\\_caption](#)  
[y\\_zero\\_plane](#)

**Methods**

[addXAxisBin](#)  
[addYAxisBin](#)  
[convertPixelToPoint](#)  
[convertPointToPixel](#)  
[findClosestPointInSeries](#)  
[hideGrid](#)



[hideXAxis](#)  
[hideYAxis](#)  
[getXExtremes](#)  
[getYExtremes](#)  
[getZoom](#)  
[removeXAxisBin](#)  
[removeYAxisBin](#)  
[selectClosestPoint](#)  
[setCrosshairs](#)  
[setXAxisCaption](#)  
[setXExtremes](#)  
[setYAxisCaption](#)  
[setYExtremes](#)  
[setZoom](#)  
[showGrid](#)  
[showXAxis](#)  
[showYAxis](#)

#### Events

[onAfterShowCrosshairs](#)  
[onBeforeBeginZoom](#)  
[onBeforeEndZoom](#)  
[onShowCrosshairs](#)  
[onXAxisNeedsTicks](#)  
[onYAxisNeedsTicks](#)

## 2 Deployment

The Emprise JavaScript Charts distribution package includes a compressed and obfuscated version of the source code for easy deployment. The entire /dist/ directory is meant to be placed on a web site, as is, with no modification necessary.

If you have access to the source code (Developer and Enterprise editions) and have made modifications, there is a utility web page located in the /packer/ directory which hosts a compression script powered by Dean Edward's Packer (original source: <http://dean.edwards.name/packer/>)

This utility must be used to compress and obfuscate the JavaScript source code before making it live or distributing it within your own application.

## 3 Changes

### 3.1 Changes in version 2.0

#### Additions

- Multiple X and Y Axes
- Logarithmic Axis
- JSON String and File Data Handlers
- Stacked Bar Series
- Floating Bar Series
- String Formatter for prefixing and appending to labels

- OpenHighLowClose Series
- Candlestick Series
- Filled Scatter Series Points in IE
- Much optimized rendering of axis labels and ticks
- Cleaner and smaller HTML generation of chart objects
- Faster chart rendering and much better support for many charts in a single page
- Removed the need for EJSChartIE.css and EJSChartIE6.css
- Better control over styling of major tick marks
- More styling options for grid lines and axis borders
- Easier customization of cursor position elements
- Background coloring with opacity of the chart area and axes

## 3.2 Changes in version 1.3.1

### Additions

- Added Chart.y\_axis\_min\_tick\_interval, y\_axis\_max\_tick\_interval, x\_axis\_min\_tick\_interval and x\_axis\_max\_tick\_interval for finer tick control
- Added tick enhancements when using date values so that the tick interval cannot be less than one millisecond
- Added support for point specific labels in Line/Area/Scatter/Bar series

### Bug Fixes

- Fixes issues with onXAxisNeedsTicks and onYAxisNeedsTicks not positioning correctly
- Added check to ensure the selected point is valid before firing onDbClickPoint event
- Updated CSV data handlers to account for pie/gauge series and correctly map labels
- Fixed issues with text bins for Y when using XML data handlers
- Updated array data handler to account for pie/gauge series and correctly map labels
- Fixed issue with gauge border default value
- Fixed issues with gauge drawing when added to a chart but not immediately drawn

## 3.3 Changes in version 1.3

### Additions

- Added AreaSeries.closeLine property which determines if the line should return to the zero plane and draw as a closed shape.
- Added PieSeries.findCenterOfCurve(point) method
- Added support for SVG exporting

### Modifications

- Modified axis to write out ticks and labels less often
- Line and scatter series are now split into smaller draws in order to support larger data sets in IE
- BarSeries point selection now accounts for proximity\_snap setting and the bar's line width
- Added support for text labels (bins) to TrendSeries

### Bug Fixes

- Updated color conversion routine to fix issues in IE
- Added additional logic to ajax data handlers to prevent double loading
- Corrected resize issue in IE which could cause an 'Invalid Argument' error
- Updated extremes calculations to correct issues with negative numbers
- getYExtremes() now correctly returns the y axis extremes
- Added additional offset checks in point location routines to account for quirks mode in various browsers
- Fixed issues with xml data handler and processing string values
- PieSeries data points now correctly save and display their labels if defined
- Corrected drawing issues in BarSeries with negative numbers
- Fixed BarSeries drawing in IE with a large number of points
- Corrected BarSeries draw in IE to close the bar line
- Updated PieSeries to reset its available colors correctly when reloading

## 3.4 Changes in version 1.2.1

### Additions

- Added selectPoint and clearSelectedPoint methods to the chart.

### Modifications

- Enhanced support for chart with large numbers of series.
- Enabled initial implementation of Series.delayLoad (default true) which controls when a series triggers its data handler to begin data retrieval
- Updated PiePoint and PieSeries to correctly render float values for pie pieces
- Updated PieSeries to handle more 100% piece drawing cases

### Bug Fixes

- Fixed issues with series reloading where auto calculated extremes were not reset with new data (affected all series)
- Fixed print rendering issues in all browsers except Opera. The chart now renders as show on screen when printing.
- Fixed XYPoint class to properly recognize the Y value as a number as opposed to a string.
- Fixed inheritance issue with ArrayDataHandler
- Fixed issues with clearing the chart when all loaded series are set invisible
- Fixed PieSeries reload method to reset available colors
- Fixed BarSeries treeLegend property to function properly when useColorArray is false
- Fixed BarSeries reload method to reset available colors
- Fixed range drawing in AnalogGaugeSeries to correctly connect arcs before filling
- Fixed issue with x axis labels not aligning correctly when allow\_interactivity was false

## 3.5 Changes in version 1.2

### Additions

- Added classes to better control and track axis bins (text labels)
- Added support for y axis bins
- Added support for shared bins between series (unused bins)
- Added coordinate property to [Chart.y\\_zero\\_plane](#) and [Chart.x\\_zero\\_plane](#) to allow zero plane to

- fall somewhere other than 0
- Added [Chart.force\\_static\\_points\\_x](#) and [Chart.force\\_static\\_points\\_y](#) ([force\\_static\\_points](#) works as before, for x only) to force bins when data is numeric
  - Implemented [Chart.y\\_axis\\_extremes\\_ticks](#)
  - Added support for drawing minor tick marks on the [Y axis](#) and [Y axis](#)
  - Added [Chart.onXAxisNeedsTicks](#) and [onYAxisNeedsTicks](#) to allow for custom tick configurations
  - Added support for [auto sort](#) in all applicable series / data handlers. This is now set by default and may be turned off if data is correctly sorted prior to being handed to the data handler.
  - Added support for [color specifications](#) as rgb(0,0,0), rgba(0,0,0,100) and hex (#000000) everywhere colors may be specified
  - Added support for mouse wheel to zoom (controlled by [allow\\_interactivity](#), [allow\\_zoom](#), [allow\\_mouse\\_wheel\\_zoom](#))
  - Added [Chart.addYAxisBin](#), [Chart.addXAxisBin](#), [Chart.removeYAxisBin](#), [Chart.removeXAxisBin](#) methods to support manual definition (and order) of axis bins
  - Added [EJSC.XMLStringDataHandler](#) class
  - Added [drawPoints](#), [pointSize](#), [pointColor](#), [pointBorderSize](#), [pointBorderColor](#) properties to [EJSC.LineSeries](#) and [EJSC.AreaSeries](#)
  - Added support for new zero plane coordinate property to [AreaSeries](#) and [BarSeries](#)
  - Added [tree-style legend](#) for [PieSeries](#)
  - Added [tree-style legend](#) option for [BarSeries](#) (default true when [useColorArray](#) is true and [treeLegend](#) is left undefined)
  - Added [position](#) and [height/width](#) properties to [PieSeries](#) to support multiple pie series per chart
  - Added [findCenter](#) method to [PieSeries](#) to find the center point of a given piece
  - Added [getPoints](#) method to [PieSeries](#) (to allow [findCenter](#) calls to provide a valid piece object)
  - Added [orientation](#) property to [BarSeries](#) and support for horizontal bar charts
  - Added [total\\_value](#) property and [getTotalValue](#), [setTotalValue](#), [resetTotalValue](#) methods to [PieSeries](#)
  - Added [\[total\]](#) and [\[percent\]](#) as hint text replacement options for [PieSeries](#)
  - Added [x\\_cursor\\_position\\_formatter](#) and [y\\_cursor\\_position\\_formatter](#) properties to chart to allow for a different formatter for mouse position display

## Modifications

- Consolidated and cleaned up xml parsing routines
- Consolidated and cleaned up csv parsing routines
- Cached many often used math methods for performance
- Modified extremes to account for manually added bins with no associated data in series
- Rewrote data handlers to better support text labels (bins)
- Modified pie series to correctly use [lineOpacity](#)
- Updated pie piece drawing to one path creation per piece
- Added check in [AnalogGaugeSeries](#) to avoid resetting innerHTML unless the label has actually changed.
- Added check of [show\\_hints](#) before triggering on [ShowHint](#) event

## Bug Fixes

- Added media attribute to link tag to fix printing issues in firefox/netscape
- Added correct headers when sending an xml request as POST
- Fixed case when last tick is missing while using [x\\_axis\\_tick\\_count](#)
- Fixed spelling - [getZoomBoxCoordinates](#)
- Updated [lineOpacity](#) to 100 (from 1) in [PieSeries](#) so lines draw correctly
- Added check for single 100% piece to remove line when drawn (fixed full circle drawing)

- Reversed `__intercept/``__slope` properties in `TrendSeries` so they represent the correct values
- Updated number formatter to check for `-0` as final output and remove `-` sign
- `onDbClickPoint` and `onAfterSelectPoint` both now function correctly when `show_hints` is false
- Fixed unterminated string error in `TrendSeries`
- Updated style to remove border when axes are not visible (ie, pie or gauge only charts)
- Fixed constant resize issue with pie charts (and anything with x axis hidden) in IE6/opera
- Fixed issue with series drawing multiple times when added and `redraw` is false

## 3.6 Changes in version 1.1

### Additions

- Added new series type `AnalogGaugeSeries`
- Added `onBeforeBeginZoom`, `onBeforeEndZoom` and `onAfterShowCrosshairs` events to `Chart`
- Added support for local loading of chart code in IE 7 and IE6 with certain versions of the `XMLHttpRequest` ActiveX control.
- Added new axis styling properties to `Chart`: `x_axis_size`, `x_axis_className`, `x_axis_tick_className`, `x_axis_stagger_ticks`, `x_axis_tick_count`, `y_axis_size`, `y_axis_className`, `y_axis_tick_className`, `y_axis_tick_count`.
- Added `Chart.setZoom` and `getZoom` methods to in order to get the current zoom and set zoom to a particular range
- Added `Chart.getMinMaxYInXRange` method to programatically find the min and max Y values in a given X range
- Added `Chart.findClosestPointInSeries` method to programatically find the closest point to the coordinates (x,y) in a particular series
- Added `Chart.getZoomBoxCoordinates` method to get the current zoom box coordinates
- Added `Chart.displayZoomBox` method to programatically display the zoom box
- Added `BarPoint` class to support additional `BarSeries` functionality
- Added the ability to draw bars with different colors based on ranges
- Added the ability to adjust the spacing between bars using the new `intervalOffset` property.
- Added the ability to switch between grouped and overlaid bars with `BarSeries`
- Added a color pool to `BarSeries` for individually colored bars (similar to `PieSeries`)
- Added `onBarNeedsColor` to `BarSeries` for more advanced bar coloring options
- Added `setDefaultColors`, `addRange`, `deleteRange`, `clearRanges`, `setIntervalOffset` and `setGroupedBars` methods to `BarSeries`
- Added support for 'exponential' and 'logarithmic' types in `TrendSeries`
- Added `showGrid/hideGrid` methods with an optional `redraw` parameter to prevent immediate redrawing
- Optimized trend series calculation routines
- Blank X or Y axis captions now trigger the axis to be resized giving the chart additional space
- Added the ability to specify only changed members of child objects via the `options` property
- Added `Chart.convertPointToPixel` method to calculate document pixel top/left of a point in chart units
- Added `Chart.convertPixelToPoint` method to calculate chart point from a document pixels
- Added base `AjaxDataHandler` class, `XMLDataHandler` and `CSVFileData` handler now descend from this class.
- Added `requestType` property and `setRequestType` method to `AjaxDataHandler` for changing between GET and POST.
- Added `urlData` property and `setUrlData` method to `AjaxDataHandler` to allow assignment of POST data and GET url parameters
- Added `onNeedsData` event and `setXMLData` method to `AjaxDataHandler` to support 3rd party Ajax libraries for data retrieval
- Added `reload` flag to `setUrl` method (default = false) in `AjaxDataHandler` to allow for immediate

data retrieval / series reload.

- Added Chart.allow\_hide\_error property (default = false) which allows error messages to be hidden and overwritten with non-error messages
- Added additional error checking and reporting into xmlrequestpool, errors occurring during series data retrieval via XML or CSVFile data handlers will be displayed on the chart.
- Added EJSC.utility.XMLRequestPool.fatalErrors property to define which HTTP codes cause a XMLHttpRequest to fail.
- Added redraw flag to Chart.addSeries in order to postpone full redraw until all series have been added.
- Added Chart.remove() method which completely deletes a chart from the page.

## Modifications

- Moved a number of private properties into the EJSC namespace for easier patching.
- Modified load/unload events to use utility.attachEvent method so that the methods are detached automatically
- Changed non-IE browser load of stylesheet to insert link tag instead of force load stylesheet via ajax
- Reduced minimum height to 60 px
- Tick container now positioned absolute in order to support additional styling options.
- Added chart parameter to onShowCrosshairs event
- Removed case sensitivity requirement for locating support files.

## Bug Fixes

- Consolidate references to html and head tags, clear references on unload to fix IE leak
- Fixed issue with attachEvent in Opera causing errors
- Added cleanup routine to series in order to remove references to legend items and fix IE leaks
- Fixed canvas cover positioning
- x\_axis container left coordinate is now set to fix issues with table/float layouts causing odd shifts in IE
- doRecalcExtremes now called in resize method in order to re-adjust padding appropriately to the new chart size
- Fixed issues with drawing first series added when extreme values are static
- Fixed issues in recalculation of extreme values when no series data is available
- Fixed offset issues with hints and point selection
- Fixed issues with point selection routine when no point is selected
- Fixed onBeforeDbClick event to send correct reference to chart
- Fixed issue with array data handler loading when array is empty
- Fixed scatter series to not draw line through center of circle
- Fixed and optimized several issues with drawing and point selection within BarSeries with many bars
- Fixed issues with calculating spacing and widths with multiple bar series in a single chart
- Fixed issues with drawing and point selection in bar series when bar is only partially in view
- Fixed style issues with legend header and legend captions in IE

## 3.7 Changes in version 1.0.1

### Additions

- Added series.opacity property
- Added series.setOpacity method

- Added series.lineOpacity property
- Added series.setLineOpacity property
- Added useUTC property to DateFormatter (default value == true)
- Added timezoneOffset property to DateFormatter (default value == undefined) Only applicable when useUTC is true
- Added series.legendIsVisible property (default value == true)
- Added series.showLegend and series.hideLegend methods
- Added chart.legendTitle property and chart.setLegendTitle method

## Modifications

- Modified to remove all code inside target div
- Attach events to chart container to prevent selection
- Moved lineWidth property and setLineWidth method into base Series class
- Modified LineSeries to use lineOpacity properties
- Modified AreaSeries to use opacity and lineOpacity properties
- Modified ScatterSeries to use lineWidth, opacity and lineOpacity properties
- Modified PieSeries to use lineWidth, opacity and lineOpacity properties
- Modified BarSeries to use opacity and lineOpacity properties
- Modified FunctionSeries and TrendSeries to use lineOpacity property
- Text selection is now cancelled when drag/selection originates in the chart (except Opera)

## Bug Fixes

- Fixed hint and point selection issues when the chart is inside a scrollable container
- Added check and alert if attachEvent fails
- Setting y\_min/max, x\_min/max during chart creation now correctly assign extremes
- Fixed scaling errors when only a single Y or X value has been specified
- Added call to onBeforeZoom when double clicking to allow canceling of zoom out
- Added call to onAfterZoom when double clicking to zoom out
- Fixed BarSeries line drawing on Internet Explorer
- Fixed automatic resizing of y axis caption in IE when using setYAxisCaption()
- Fixed style sheets to completely hide axis when turned off
- Fixed style sheets to hide key grabber element and remove border and background from cursor position elements.
- Fixed issues with long series titles wrapping legend items
- Added additional exception handling to unload methods and storage and clearing of auto-resize interval
- Fixed errors when attempting to load empty data sets (applicable to all data handlers)
- Fixed error when loading BarSeries with a single point

## 3.8 Changes in version 1.0

### Additions

- Added an optional redraw flag into Chart.removeSeries in order to allow for multiple removes between chart redraws.
- Added support for user-defined data associated with a point
- Added support for reading user-defined data from full and short xml formats
- Added the setTitle method to the base Series class.
- Added getDataHandler method to base Series class.
- Added getUrl and setUrl methods to XMLDataHandler and CSVFileDataHandler classes.

- Added `getArray` and `setArray` methods to `ArrayDataHandler` class.
- Added `getCSV` and `setCSV` methods to `CSVStringDataHandler` class.
- Added `coloredLegend` property to `Series` to specify whether the series legend text should inherit the series color
- Added `setColoredLegend` method to `Series` in order to update the `coloredLegend` property after series creation

## Modifications

- Updated chart resize methods to monitor the chart container and update whenever necessary, this adds greater compatibility with other 3rd party packages such as Dojo when the chart is placed inside a window class.
- `Chart.removeSeries()` now deletes the series object which was removed.
- Modified series legend items to have their text color match the series color.
- Modified series legend captions to not wrap and display full text with a hint.

## Bug Fixes

- Fixed event attachments to global objects such as `document.onmouseup` to use `attachEvent` / `addEventListener`
- `Chart.removeSeries()` now correctly removes the legend objects without error.
- Fixed an issue where tall charts could lose interactivity.

## 4 Data Formats

The following is a brief summary of the data formats supported by Emprise JavaScript Charts. For additional information please see the individual class help files and example code available at <http://www.ejschart.com/help/>

### XML

The [EJSC.XMLDataHandler](#) class will load an XML file containing chart data using AJAX. The following XML formats are supported:

#### Full:

```
<graph>
  <plot>
    <point x="" y="" />
  </plot>
</graph>
```

#### Short:

```
<G>
  <L>
    <P x="" y="" />
  </L>
</G>
```

#### Compact:



```
<G>
  <L values="X|Y,X|Y,X|Y" />
</G>
```

## Array

The [EJSC.ArrayDataHandler](#) class will load chart data from a JavaScript array. The basic format of the array is as follows:

```
[
  [X Value, Y Value],
  [X Value, Y Value]
]
```

## CSV (Comma Separated Values)

The [EJSC.CSVFileDataHandler](#) and [EJSC.CSVStringDataHandler](#) classes support a comma separated list of data points. [EJSC.CSVFileDataHandler](#) will load the data points list from a file on the server using AJAX. [EJSC.CSVStringDataHandler](#) takes a string in its constructor specifying the CSV text.

```
x|y,x|y,x|y
```

## JSON (JavaScript Object Notation)

The [EJSC.JSONFileDataHandler](#) and [EJSC.JSONStringDataHandler](#) classes support JSON formatted data points. [EJSC.JSONFileDataHandler](#) will load the data points list from a file on the server using AJAX. [EJSC.JSONStringDataHandler](#) takes a string in its constructor specifying the JSON text.

```
{
  {"x":"x value", "y":"y value"},
  {"x":"x value", "y":"y value"}
}
```

## 4.1 XML - Full

The full xml format is the most verbose and the most human readable of the supported formats. This format is ideal for experimenting with chart configuration and charts with smaller datasets.

### General Usage:

```
<graph>
  <plot>
    <point x="1" y="1" />
    <point x="2" y="2" />
    <point x="3" y="3" />
    <point x="4" y="4" />
  </plot>
</graph>
```

### Series Data Formats:

The above example is used for the majority of currently supported data series types. There are exceptions such as the [EJSC.PieSeries](#), [EJSC.AnalogGaugeSeries](#) and others. For a full description of the data format for a given series, see the [Data Formats](#) topic below each series section.

### Text Labels:

At times, numeric labels and the auto scaling options in the chart are not necessary, not available or not desired for display data. A [EJSC.BarSeries](#) for instance, which is used to compare number of widgets sold by color. The chart supports providing text as the X value, rather than a number:

```
<graph>
  <plot>
    <point x="Blue" y="100" />
    <point x="Red" y="200" />
    <point x="Green" y="50" />
  </plot>
</graph>
```

### User-Defined Data:

When displaying custom hints and implementing drill down style reports it is often necessary to have an additional piece of data associated with each data point such as a database id value. The full and short xml formats support the **userdata** property which is read in and assigned to the associated [EJSC.Point](#) descendant. The value specified is entirely up to the developer. It could be a set of integers representing a database primary key value, a url which points to drill-down data or even a javascript function. The **userdata** property of a point is available during point-related events in the chart such as [EJSC.Chart.onDbClickPoint](#), [EJSC.Chart.onAfterSelectPoint](#), etc.

```
<graph>
  <plot>
    <point x="1" y="100" userdata="WHERE PointId=10432" />
    <point x="2" y="200" userdata="./drillDown2.xml" />
    <point x="3" y="50" userdata="alert('this is the userdata');" />
  </plot>
</graph>
```

## 4.2 XML - Short

The short XML format provides the same functionality as the [full format](#) but with less transfer overhead (there is no reduction in the processing overhead of extracting the point data, see the Array and CSV formats for more concise options).

The general format is as follows:

```
<G>
  <L>
    <P x="" y="" userdata="" />
  </L>
</G>
```

The <G> tag relates to the <graph> tag in the full format, the <L> tag relates to the <plot> tag and <P>

is equivalent to <point>.

Support for pie charts is provided in the same manner as with the full format, i.e.

```
<P x="" label="" />
```

## 4.3 XML - Compact

The compact xml format is geared towards larger data sets which need to reduce the overhead associated with transfer and extraction of chart data via a more verbose xml format.

The data is provided in csv-like format and stored in the "values" attribute of the <L> tag:

```
<G>  
  <L values="1|1,2|2,3|3" />  
</G>
```

For the majority of series, the above example would be used to provide X,Y coordinates.

# 5 API Reference

## 5.1 EJSC

Top level namespace for all classes and variables used by the Emprise JavaScript Charts package. Use of this namespace prevents variable name collisions with other available JavaScript packages.

### 5.1.1 Properties

#### 5.1.1.1 DefaultImagePath

##### Definition

```
string DefaultImagePath = "images/"
```

##### Description

Relative path defining the path to all images used by EJSC classes.

#### 5.1.1.2 DefaultColors

##### Definition

```
array defaultColors = [  
  "rgb(120,90,59)",  
  "rgb(53,115,53)",  
  "rgb(178,87,56)",  
  "rgb(203,143,71)",  
  "rgb(55,106,155)",  
  "rgb(205,197,51)",  
  "rgb(209,130,139)",  
  "rgb(159,153,57)",  
  "rgb(206,173,136)",
```

```
    "rgb(191,132,72)",  
    "rgb(151,135,169)",  
    "rgb(140,48,51)",  
    "rgb(59,144,187)",  
    "rgb(197,190,104)",  
    "rgb(109,136,79)",  
    "rgb(144,100,144)",  
    "rgb(181,94,94)",  
    "rgb(59,144,144)",  
    "rgb(204,136,92)",  
    "rgb(139,167,55)",  
    "rgb(205,171,66)",  
    "rgb(150,184,211)"  
]
```

### Description

This array is used in each chart to specify available series colors. Add to or modify this array in order to define a single set of default colors used by all charts on a page.

#### 5.1.1.3 DefaultBarColors

### Definition

```
array defaultBarColors = [  
    "rgb(120,90,59)",  
    "rgb(53,115,53)",  
    "rgb(178,87,56)",  
    "rgb(203,143,71)",  
    "rgb(55,106,155)",  
    "rgb(205,197,51)",  
    "rgb(209,130,139)",  
    "rgb(159,153,57)",  
    "rgb(206,173,136)",  
    "rgb(191,132,72)",  
    "rgb(151,135,169)",  
    "rgb(140,48,51)",  
    "rgb(59,144,187)",  
    "rgb(197,190,104)",  
    "rgb(109,136,79)",  
    "rgb(144,100,144)",  
    "rgb(181,94,94)",  
    "rgb(59,144,144)",  
    "rgb(204,136,92)",  
    "rgb(139,167,55)",  
    "rgb(205,171,66)",  
    "rgb(150,184,211)"  
]
```

### Description

Defines the pool of default colors available for bar series bars.

#### 5.1.1.4 DefaultPieColors

##### Definition

```
array defaultPieColors = [  
    "rgb(120,90,59) ",  
    "rgb(53,115,53) ",  
    "rgb(178,87,56) ",  
    "rgb(203,143,71) ",  
    "rgb(55,106,155) ",  
    "rgb(205,197,51) ",  
    "rgb(209,130,139) ",  
    "rgb(159,153,57) ",  
    "rgb(206,173,136) ",  
    "rgb(191,132,72) ",  
    "rgb(151,135,169) ",  
    "rgb(140,48,51) ",  
    "rgb(59,144,187) ",  
    "rgb(197,190,104) ",  
    "rgb(109,136,79) ",  
    "rgb(144,100,144) ",  
    "rgb(181,94,94) ",  
    "rgb(59,144,144) ",  
    "rgb(204,136,92) ",  
    "rgb(139,167,55) ",  
    "rgb(205,171,66) ",  
    "rgb(150,184,211) "  
]
```

##### Description

Defines the pool of default colors available for pie pieces.

#### 5.1.1.5 STRINGS

##### Definition

```
object STRINGS: {  
    building_message:    "Building Chart...",  
    max_zoom_message:   "Max Zoom Reached",  
    drawing_message:    "Drawing...",  
    chart_legend_title: "Chart Legend",  
    y_axis_caption:     "Y Axis",  
    x_axis_caption:     "X Axis"  
},
```

##### Description

Defines the default strings used by the chart.

## 5.2 EJSC.Chart

Chart class that holds all the generic information for each chart that is being created.

The constructor expects the id or reference for a DOM object (preferably a div) and an optional set of object properties. **NOTE:** The contents of the DOM object will be cleared before rendering the chart.

### Constructor

```
EJSC.Chart( string id [, object options] )
EJSC.Chart( DOMObject object [, object options] )
```

### Example

```
var myChart = new EJSC.Chart("chart", {title:"My Chart"});
```

or

```
var myChartDiv = document.getElementById("myChart");
var myChart = new EJSC.Chart(myChartDiv, {title:"My Chart"});
```

### Defining Chart Properties and Events

Chart properties may be set individually after the chart has been created or in batch using the options parameter during instantiation.

#### Setting properties individually

```
var myChart = new EJSC.Chart("chartDIV");
myChart.allow_zoom = false;
myChart.show_legend = false;
myChart.onDbClickPoint = function(point) {
    alert("Point Clicked: " + point.x + "," + point.y);
}
```

#### Setting properties in batch

```
var myChart = new EJSC.Chart(
    "chartDIV",
    {
        allow_zoom: false,
        show_legend: false,
        onDbClickPoint: function(point) {
            alert("Point Clicked: " + point.x + "," + point.y);
        }
    }
);
```

### 5.2.1 Properties

#### 5.2.1.1 allow\_interactivity

##### Definition

boolean [allow\\_interactivity](#) = true

### Description

Defines if the chart can be interacted with (ie zooming, moving, point selection, etc.).

### Example

>> *Turn off all interactivity for the chart*

```
var chart = new EJSC.Chart(  
    "chart",  
    {allow_interactivity: false}  
);
```

#### 5.2.1.2 allow\_mouse\_wheel\_zoom

### Definition

boolean [allow\\_mouse\\_wheel\\_zoom](#) = true

### Description

Defines if the chart can be zoomed via the scroll wheel on the mouse. This is automatically disabled if [allow\\_interactivity](#) or [allow\\_zoom](#) is set to false.

### Example

>> *Turn off wheel zooming for the chart.*

```
var chart = new EJSC.Chart(  
    "chart",  
    {allow_mouse_wheel_zoom: false}  
);
```

#### 5.2.1.3 allow\_move

### Definition

boolean [allow\\_move](#) = true

### Description

Defines if the chart can be moved once it has been zoomed in. This is automatically disabled if [allow\\_interactivity](#) is set to false.

### Example

>> *Turn off moving for the chart.*

```
var chart = new EJSC.Chart(  
    "chart",
```

```
        {allow_move: false}
    );
```

#### 5.2.1.4 allow\_zoom

##### Definition

boolean **allow\_zoom** = true

##### Description

Defines if the chart can be zoomed. This is automatically disabled if [allow\\_interactivity](#) is set to false.

##### Example

*>> Turn off zooming for the chart.*

```
var chart = new EJSC.Chart(
    "chart",
    {allow_zoom: false}
);
```

#### 5.2.1.5 allow\_hide\_error

##### Definition

boolean **allow\_hide\_error** = false

##### Description

Defines whether the chart can overwrite an error message with a non-error message or hide the error message after a brief period of time.

##### Example

*>> Allow all messages, including errors, to be hidden after a brief period of time*

```
var chart = new EJSC.Chart(
    "chart",
    { allow_hide_error: true }
);
```

#### 5.2.1.6 auto\_find\_point\_by\_x

##### Definition

boolean **auto\_find\_point\_by\_x** = false

##### Description

Defines if the select point routine should select points based only on the X position of the cursor.

##### Example



>> Allow point selection to occur on click anywhere within the Y axis, find the closest point based solely on X.

```
var chart = new EJSC.Chart(  
    "chart",  
    {auto_find_point_by_x: true}  
);
```

### 5.2.1.7 auto\_resize

#### Definition

boolean **auto\_resize** = true

#### Description

Defines if the chart will start a timer to check for size changes in order to automatically redraw. If you are controlling the display, size and position of the chart div container manually, it may be beneficial to turn this option off in order to eliminate the timer overhead.

#### Example

>> Turn off the chart's auto resize functionality

```
var chart = new EJSC.Chart(  
    "chart",  
    {auto_resize: false}  
);
```

### 5.2.1.8 auto\_zoom

#### Definition

string **auto\_zoom** = undefined

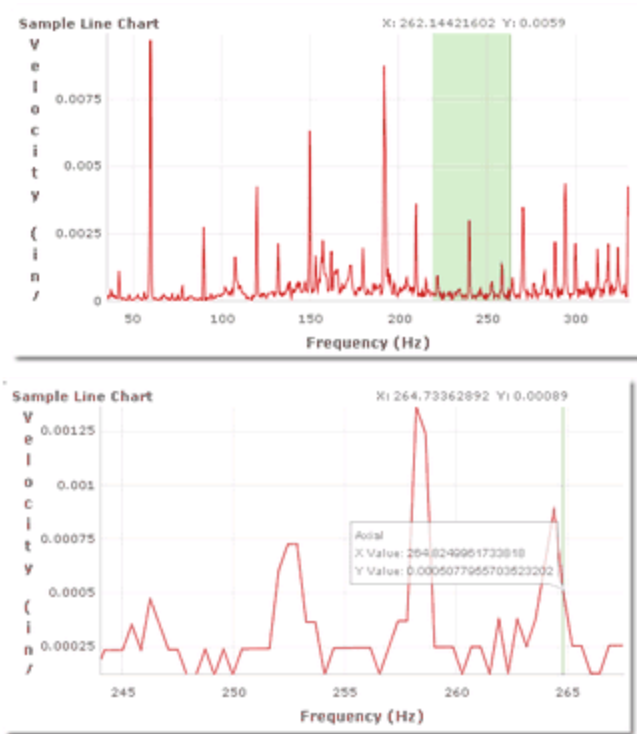
Valid property values:

'x'  
'y'

#### Description

Defines if the chart should auto-select the zoom area based on the X or Y coordinates selected by the user. The other coordinates are automatically selected based on best fit for the data defined in the range selected.

#### Example



>> Allow user to select beginning and ending X values to zoom, then auto-scale the Y axis according to the data.

```
var chart = new EJSChart(
    "chart",
    { auto_zoom: "y" }
);
```

### 5.2.1.9 axis\_bottom

#### Definition

[EJSChart.Axis axis\\_bottom](#) = EJSChart.LinearAxis

#### Description

Defines the axis that will be at the bottom of the chart. All four axes are defined as EJSChart.LinearAxis by default when the chart is created and do not need to be manually created unless a different type of axis is needed. These properties automatically allow the setting of axis properties during chart creation as shown in the example below.

Types:

[EJSChart.LinearAxis](#)

[EJSChart.LogarithmicAxis](#)

#### Example

>> *Hide the bottom axis*

```
var chart = new EJSChart(
    "chart",
    {
        axis_bottom: { visible: false }
    }
);
```

>> *Display a base 10 logarithmic axis at the bottom of the chart.*

```
var chart = new EJSChart(
    "chart",
    {
        axis_bottom: new EJSLogarithmicAxis({ base: 10 })
    }
);
```

### 5.2.1.10 axis\_left

#### Definition

[EJSChart.Axis axis\\_left](#) = EJSChart.LinearAxis

#### Description

Defines the axis that will be at the left side of the chart. All four axes are defined as EJSChart.LinearAxis by default when the chart is created and do not need to be manually created unless a different type of axis is needed. These properties automatically allow the setting of axis properties during chart creation as shown in the example below.

Types:

[EJSChart.LinearAxis](#)

[EJSChart.LogarithmicAxis](#)

#### Example

>> *Hide the left axis*

```
var chart = new EJSChart(
    "chart",
    {
        axis_left: { visible: false }
    }
);
```

>> *Display a base 10 logarithmic axis at the left side of the chart.*

```
var chart = new EJSChart(
    "chart",
    {
        axis_left: new EJSLogarithmicAxis({ base: 10 })
    }
);
```

### 5.2.1.11 axis\_right

#### Definition

[EJSC.Axis axis\\_right](#) = EJSC.LinearAxis

#### Description

Defines the axis that will at the right side of the chart. All four axes are defined as EJSC.LinearAxis by default when the chart is created and do not need to be manually created unless a different type of axis is needed. These properties automatically allow the setting of axis properties during chart creation as shown in the example below.

Types:

[EJSC.LinearAxis](#)

[EJSC.LogarithmicAxis](#)

#### Example

>> *Show the right axis*

```
var chart = new EJSC.Chart(  
    "chart",  
    {  
        axis_right: { visible: true }  
    }  
);
```

>> *Display a base 10 logarithmic axis at the right side of the chart.*

```
var chart = new EJSC.Chart(  
    "chart",  
    {  
        axis_right: new EJSC.LogarithmicAxis({ base: 10 })  
    }  
);
```

### 5.2.1.12 axis\_top

#### Definition

[EJSC.Axis axis\\_top](#) = EJSC.LinearAxis

#### Description

Defines the axis that will at the top of the chart. All four axes are defined as EJSC.LinearAxis by default when the chart is created and do not need to be manually created unless a different type of axis is needed. These properties automatically allow the setting of axis properties during chart creation as shown in the example below.

Types:

[EJSC.LinearAxis](#)

[EJSC.LogarithmicAxis](#)

## Example

>> Show the top axis

```
var chart = new EJSC.Chart(  
    "chart",  
    {  
        axis_top: { visible: true }  
    }  
);
```

>> Display a base 10 logarithmic axis at the top of the chart.

```
var chart = new EJSC.Chart(  
    "chart",  
    {  
        axis_top: new EJSC.LogarithmicAxis({ base: 10 })  
    }  
);
```

### 5.2.1.13 background

#### Definition

```
object background = {  
    color: "#FFF",  
    opacity: 0,  
    includeTitle: false  
};
```

#### Description

Defines the color and opacity of the chart area background. If the includeTitle property is set to true, the title bar area will be filled as well. If opacity is 0, no fill will occur.

**NOTE:** To set the color of then entire chart area (all axes, chart, titlebar, etc.), set the background-color style for the chart DOM object.

```
<div id="chart" style="width: 400px; height: 400px; background-color: #fff00;"></div>
```

### 5.2.1.14 building\_message

#### Definition

```
string building_message = EJSC.STRINGS["building_message"]
```

#### Description

Defines if text to display while the chart is being initially assembled. The default string used for all charts in the page is drawn from the EJSC.Strings property.

#### 5.2.1.15 drawing\_message

##### Definition

string **drawing\_message** = [EJSC.STRINGS](#)["drawing\_message"]

##### Description

Defines if text to display while the chart is being drawn. The default string used for all charts in the page is drawn from the [EJSC.Strings](#) property.

#### 5.2.1.16 legend\_title

##### Definition

string **legend\_title** = [EJSC.STRINGS](#)["chart\_legend\_title"]

##### Description

Defines the caption displayed in the chart legend title bar. To modify this caption after chart creation, use [Chart.setLegendTitle\(\)](#)

#### 5.2.1.17 max\_zoom\_message

##### Definition

string **max\_zoom\_message** = [EJSC.STRINGS](#)["max\_zoom\_message"]

##### Description

Defines if text to display when the maximum zoom has been reached. The default string used for all charts in the page is drawn from the [EJSC.Strings](#) property.

#### 5.2.1.18 message\_timeouts

##### Definition

```
object message_timeouts = {  
  progress: 500,  
  no data: 500,  
  info: 500,  
  error: 2000  
};
```

##### Description

Defines the amount of time in milliseconds to display a particular message type.

#### 5.2.1.19 proximity\_snap

##### Definition

integer **proximity\_snap** = 5

### Description

Determines the maximum number of pixels away from a point the cursor can be for point selection and hints.

### Example

*>> Allow clicks to trigger point selection up to 9 pixels away from the actual point.*

```
var chart = new EJSC.Chart(  
    "chart",  
    {proximity_snap: 8}  
);
```

#### 5.2.1.20 show\_hints

### Definition

boolean **show\_hints** = true

### Description

Defines whether hints should be selected when a point is hovered over or selected. This is automatically disabled if [allow\\_interactivity](#) is set to false.

### Example

*>> Turn off hints and point selection.*

```
var chart = new EJSC.Chart(  
    "chart",  
    {show_hints: false}  
);
```

#### 5.2.1.21 show\_legend

### Definition

boolean **show\_legend** = true

### Description

Defines if the chart legend should be displayed.

### Example

*>> Turn off legend display*

```
var chart = new EJSC.Chart(  
    "chart",  
    {show_legend: false}  
);
```

### 5.2.1.22 show\_messages

#### Definition

boolean `show_messages` = true

#### Description

Defines if progress messages should be displayed above the chart or not.

#### Example

*>> Turn off progress messages.*

```
var chart = new EJSC.Chart(  
    "chart",  
    {show_messages: false}  
);
```

### 5.2.1.23 show\_titlebar

#### Definition

boolean `show_titlebar` = true

#### Description

Defines if the titlebar (containing chart title and mouse position indicators) should be displayed above the chart.

#### Example

*>> Display the chart without a titlebar*

```
var chart = new EJSC.Chart(  
    "chart",  
    {show_titlebar: false}  
);
```

### 5.2.1.24 title

#### Definition

string `title` = "Emprise JavaScript Chart"

#### Description

Defines the title of the chart. It is displayed at the top left of the chart.

#### Example

*>> Give the chart a name.*



```
var chart = new EJSC.Chart(  
    "chart",  
    {title: "2007: Total Widget Sales by Month"}  
);
```

## 5.2.2 Methods

### 5.2.2.1 acquireSeries

#### Definition

void [acquireSeries](#)( [EJSC.Series](#) series, boolean redraw )

#### Description

Moves a series from one chart to another. This method will remove the series from the chart which currently owns it and add it to the chart making the [acquireSeries](#) call.

Sends:

series - The series to be moved

redraw - Boolean flag telling the old and new charts if they should redraw immediately.

Specifying **false** here will require that both charts [redraw\(\)](#) methods are called in order to display the results of the acquisition.

### 5.2.2.2 addSeries

#### Definition

[EJSC.Series](#) [addSeries](#)( [EJSC.Series](#) series, boolean redraw )

#### Description

Adds a new series to the chart and returns the newly added series (this is useful when creating series inline as shown in the example below). The series must descend from [EJSC.Series](#).

If redraw is false, drawing will not occur immediately but will not be entirely prevented. Certain actions such as adding an additional series with redraw = true, or chart dimensions changing may still trigger the series to draw. The default, if redraw is omitted is **true**.

**Note:** If the series has not defined a title (i.e. `Series.title = ""`) at the time [addSeries](#) is called, a default title of "Series <index>" will be assigned (where <index> is the current series index in internal series storage).

Types:

[EJSC.AnalogGaugeSeries](#)

[EJSC.AreaSeries](#)

[EJSC.BarSeries](#)

[EJSC.CandlestickSeries](#)

[EJSC.FloatingBarSeries](#)

[EJSC.FunctionSeries](#)

[EJSC.LineSeries](#)

[EJSC.OpenHighLowCloseSeries](#)

[EJSC.PieSeries](#)  
[EJSC.ScatterSeries](#)  
[EJSC.StackedBarSeries](#)  
[EJSC.TrendSeries](#)

### Example

>> Add a line series to the chart.

```
var myChart = new EJSC.Chart("chart");  
var mySeries = myChart.addSeries(new EJSC.LineSeries(...));
```

#### 5.2.2.3 clearSelectedPoint

##### Definition

void [clearSelectedPoint](#)( )

##### Description

Clears the current point selection and hides the hint if necessary.

#### 5.2.2.4 exportSVG

##### Definition

string [exportSVG](#)( object Options )

##### Options Described (parameter {default value})

- **includeHeader {true}**

The includeHeader property defines whether the result should include the xml declaration (<?xml version="1.0"?>) and the SVG doctype tags.

- **height {chart height in pixels}**

The height property defines the height attribute of the viewBox attribute in the output svg tag (see [viewbox](#)). By default this is the pixel height of the chart but may be set to an arbitrary number or percentage such as "100%"

- **width {chart width in pixels}**

The width property defines the width attribute of the viewBox attribute in the output svg tag (see [viewbox](#)). By default this is the pixel width of the chart but may be set to an arbitrary number or percentage such as "100%"

- **namespace {none}**

The namespace property defines the prefix namespace to use for the svg tags. This is useful when used with includeHeader=false if including the resulting svg inside another document type. By default, this left undefined and the resulting svg will look something like <svg ... >...</svg>. If

defined, for example as namespace = mysvg, the resulting svg will look like <mysvg:svg ...>...</mysvg:svg>.

### Description

This method will export the chart as it is currently rendered as SVG. This string can be returned to the server for processing into many different forms such as png or pdf. The server-side processing of this data is beyond the scope of this help document.

### Example:

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

    var chart = new EJSC.Chart("myChart", {});
    var series = new EJSC.LineSeries(
        new EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,2],[5,1]]),
        {}
    );
    chart.addSeries(series);

    function buttonClick() {
        alert(chart.exportSVG({
            includeHeader: false,
            height: "50%",
            width: "50%",
            namespace: "mySVG"
        }));
    }

</script>
<button onclick="buttonClick();">Export SVG</button>
```

#### 5.2.2.5 exportSVGLegend

### Definition

string [exportSVGLegend](#)( object Options )

### Options Described (parameter {default value})

- **orientation** {"horizontal"}

The orientation property defines how the legend dimensions will be calculated. Valid options are "horizontal" and "vertical". When using horizontal, the legend's width will match the chart's width and the legend items will be displayed left to right, top to bottom. The height will expand as needed to accommodate all legend items. Specifying vertical for orientation will cause the legend to match the chart's height and display the legend items top to bottom, left to right, expanding the width of the legend as necessary.

- **includeHeader {true}**

The includeHeader property defines whether the result should include the xml declaration (<?xml version="1.0"?>) and the SVG doctype tags.

- **height {chart height in pixels}**

The height property defines the height attribute of the viewBox attribute in the output svg tag (see [viewbox](#)). By default this is the pixel height of the chart but may be set to an arbitrary number or percentage such as "100%"

- **width {chart width in pixels}**

The width property defines the width attribute of the viewBox attribute in the output svg tag (see [viewbox](#)). By default this is the pixel width of the chart but may be set to an arbitrary number or percentage such as "100%"

- **namespace {none}**

The namespace property defines the prefix namespace to use for the svg tags. This is useful when used with includeHeader=false if including the resulting svg inside another document type. By default, this left undefined and the resulting svg will look something like <svg ... >...</svg>. If defined, for example as namespace = mysvg, the resulting svg will look like <mysvg:svg ...>...</mysvg:svg>.

- **border {}**

- **show {true}**

The show property determines whether or not to draw a border around the entire legend area.

- **color {"#000"}**

The color property specifies the color of the border, only applicable if show is true.

- **size {1}**

The size specifies the size (thickness) of the border drawn, only applicable if show is true.

- **show\_title {true}**

The show\_title property determines whether to draw the title area of the legend. If false or if the chart's title property is blank, the legend title area will not be drawn.

## Description

This method will export the chart legend as it relates to the currently rendered chart. The legend is exported in a more print friendly manner and may be customized using the options described above. The exported SVG string can be returned to the server for processing into many different forms such as png or pdf. The server-side processing of this data is beyond the scope of this help document.

**Example:**

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

    var chart = new EJSC.Chart("myChart", {});
    var series = new EJSC.LineSeries(
        new EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,2],[5,1]]),
        {}
    );
    chart.addSeries(series);

    function buttonClick() {
        alert(chart.exportSVG({
            height: "50%",
            width: "50%",
            border: {
                show: true,
                color: "#F00",
                size: 4
            },
            show_title: false
        }));
    }

</script>
<button onclick="buttonClick();">Export SVG</button>
```

**5.2.2.6 findClosestPoint****Definition**

[EJSC.Point](#) **findClosestPoint**( interger x, integer y, boolean select, boolean sticky )

**Description**

Given X and Y as screen pixel coordinates (not chart coordinates), this method will find and return the closest point in any of its series. Optionally, the point may be selected by sending true for the select parameter and displayed as either as a popup hint (i.e. when a user hovers over a point) by sending false for the sticky parameter or a sticky hint (i.e. a user clicks on a point) by sending true.

**5.2.2.7 getMinMaxYInXRange**

**DEPRECATED - Will be replaced in a future version**

**Definition**

object **getMinMaxYInXRange**( float x\_min, float x\_max )

RETURNS:

{ y\_min: float, y\_max: float }

**Description**

Returns the min and max Y values for the provided X range.

### 5.2.2.8 **getZoomBoxPixelCoordinates**

#### **Definition**

object **getZoomBoxPixelCoordinates**( string ScreenOrChart )

RETURNS:

{ left: integer, top: integer, right: integer, bottom: integer, width: integer, height: integer }

ScreenOrChart Values:

"screen"

"chart"

#### **Description**

Returns the current pixel coordinates of the zoom box (whether visible or not). The ScreenOrChart parameter may be used to specify if the coordinates returned are based off the top left of the chart ("chart"), or the top left of the page ("screen").

### 5.2.2.9 **hideTitlebar**

#### **Definition**

void **hideTitlebar**( )

#### **Description**

Hides the titlebar, resizes and chart area and redraws all visible series.

No effect if the titlebar is already hidden.

### 5.2.2.10 **hideZoomBox**

#### **Definition**

void **hideZoomBox**( )

#### **Description**

Hides the zoom box. No effect if the zoom box is already hidden.

### 5.2.2.11 **redraw**

#### **Definition**

void **redraw**( boolean reselectPoint )

#### **Description**

Resizes chart elements (if necessary) and redraws the entire chart and all series contained within. the **reselectPoint** parameter determines whether to retain point selection (if any) between redraws.

### 5.2.2.12 remove

#### Definition

void **remove**( )

#### Description

Removes the chart from the page, cleans up all attached events, references, timeouts and intervals and clears the contents of the parent element (specified during creation).

### 5.2.2.13 removeSeries

#### Definition

void **removeSeries**( [EJSC.Series](#) series, boolean redraw )

#### Description

Removes the specified series from the chart and triggers an immediate rescaling and redraw. Send in redraw = false if performing multiple removes and want to delay redrawing until complete.

### 5.2.2.14 selectPoint

#### Definition

void **selectPoint**( [EJSC.Point](#) point, boolean sticky )

#### Description

Selects the point provided. The sticky property specifies whether the hint window will close when the mouse moves over another point in the chart.

The point object can be retrieved using the [Chart.findClosestPoint\(\)](#) or [Series.findClosestPoint\(\)](#) methods.

### 5.2.2.15 setAutoResize

#### Definition

void **setAutoResize**( boolean auto\_resize )

#### Description

Enables or disables the [auto\\_resize](#) functionality once the chart has been created.

### 5.2.2.16 setLegendTitle

#### Definition

void **setLegendTitle**( string title )

**Description**

Updates the caption in the legend window title bar.

**5.2.2.17 setTitle****Definition**

void [setTitle](#)( string title )

**Description**

Updates the chart title shown in the title area of the chart. This method must be used to change the title once the chart has been rendered.

To set a default chart title on creation, see [EJSC.Chart.title](#).

**5.2.2.18 showTitlebar****Definition**

void [showTitlebar](#)( )

**Description**

Shows the titlebar, resizes and chart area and redraws all visible series.

No effect if the titlebar is already visible.

**5.2.2.19 showZoomBox****Definition**

void [showZoomBox](#)( coordinate x\_min, coordinate x\_max, string x\_axis, coordinate y\_min, coordinate y\_max, string y\_axis )

**Description**

Shows the zoom box at the specified chart coordinates using the axes specified.

**Example**

*>> Show the zoom box at x=10,y=10, x=100, y=100*

```
var chart = new EJSC.Chart("chart",
    {
        axis_bottom: {
            min_extreme: 0,
            max_extreme: 200
        },
        axis_left: {
```



```
        min_extreme: 0,  
        max_extreme: 2000  
    }  
};  
chart.showZoomBox(10, 200, "bottom", 100, 100, "left");
```

## 5.2.3 Events

### 5.2.3.1 onAfterBuild

#### Definition

void [onAfterBuild](#)( [EJSC.Chart](#) chart )

#### Description

Called immediately after the chart has finished writing itself to the page. Sends the chart object which triggered the event.

### 5.2.3.2 onAfterDraw

#### Definition

void [onAfterDraw](#)( [EJSC.Chart](#) chart )

#### Description

Called after a drawing option has completed. Sends the chart object which triggered the event.

### 5.2.3.3 onAfterMove

#### Definition

void [onAfterMove](#)( [EJSC.Chart](#) chart )

#### Description

Called after the chart has been dragged/moved while zoomed in. Sends the chart object which triggered the event.

### 5.2.3.4 onAfterSelectPoint

#### Definition

void [onAfterSelectPoint](#)( [EJSC.Point](#) point, [EJSC.Series](#) series, [EJSC.Chart](#) chart, DOMObject hint\_element, string HoverOrSelect )

#### Description

Called after a point has been selected due to mouse over (hover) or physical selection (click, keyboard).

Sends:

point - The point object selected.

series - The series object in which the point exists.

chart - The chart object which triggered the event (owns the series/point).

hint\_element - The DOM object which contains the hint text/markup

HoverOrSelect - String "hover" or "select" to indicate what triggered the event.

#### 5.2.3.5 onAfterUnselectPoint

##### Definition

void [onAfterUnselectPoint](#)( )

##### Description

Called after a point has lost selection.

#### 5.2.3.6 onAfterZoom

##### Definition

void [onAfterZoom](#)( [EJSC.Chart](#) chart )

##### Description

Called after the chart has been zoomed. Sends the chart object which triggered the event.

#### 5.2.3.7 onBeforeBuild

##### Definition

void [onBeforeBuild](#)( [EJSC.Chart](#) chart )

##### Description

Called immediately before the chart begins writing itself to the page. Sends the chart object which triggered the event.

#### 5.2.3.8 onBeforeDbIClick

##### Definition

boolean [onBeforeDbIClick](#)( [EJSC.Chart](#) chart )

##### Description

Called before the chart does any processing related to a double click. Return **false** may be used to cancel all additional processing.

#### 5.2.3.9 onBeforeDraw

##### Definition

boolean **onBeforeDraw**( )

#### Description

Called before the chart is drawn. If return is false, draw is canceled.

#### 5.2.3.10 onBeforeSelectPoint

##### Definition

boolean **onBeforeSelectPoint**( [EJSC.Point](#) point, [EJSC.Series](#) series, [EJSC.Chart](#) chart, DOMObject hint\_element, string HoverOrSelect )

##### Description

Called after a point has been selected due to mouse over (hover) or physical selection (click, keyboard).

Sends:

point - The point object about to be selected.

series - The series object in which the point exists.

chart - The chart object which triggered the event (owns the series/point).

hint\_element - The DOM object which contains the hint text/markup

HoverOrSelect - String "hover" or "select" to indicate what triggered the event.

#### 5.2.3.11 onBeforeUnselectPoint

##### Definition

void **onBeforeUnselectPoint**( [EJSC.Point](#) point, [EJSC.Series](#) series, [EJSC.Chart](#) chart )

##### Description

Called before a point has been unselected. If return is false, current point selection is not lost.

Sends:

point - The point object about to lose selected.

#### 5.2.3.12 onDbClickPoint

##### Definition

boolean **onDbClickPoint**( [EJSC.Point](#) point, [EJSC.Series](#) series, [EJSC.Chart](#) chart )

##### Description

Called when a point is double clicked or the ENTER key is pressed and a point is selected. If return is false, cancels zoom-out (if axis are shown).

#### 5.2.3.13 onShowHint

##### Definition

string **onShowHint**( [EJSC.Point](#) point, [EJSC.Series](#) series, [EJSC.Chart](#) chart, DOMObject hint\_element, string HoverOrSelect )

#### **Description**

Called after a point has been selected due to mouse over (hover) or physical selection (click, keyboard).

A custom string may be returned in order to specify the contents of the hint window. See [Text Replacement Options](#) for more details.

Sends:

- point - The point object selected.
- series - The series object in which the point exists.
- chart - The chart object which triggered the event (owns the series/point).
- hint\_element - The DOM object which contains the hint text/markup
- HoverOrSelect - String "hover" or "select" to indicate what triggered the event.

#### **5.2.3.14 onShowMessage**

##### **Definition**

void **onShowMessage**( string message, string type )

##### **Description**

Called when the status message is displayed (in the top right corner of the chart).

Sends:

- message - The text to be displayed
- type - A string identifying the type of message window. This may be "error" or "info"

#### **5.2.3.15 onUserBeginZoom**

##### **Definition**

boolean **onUserBeginZoom**( [EJSC.Chart](#) chart )

##### **Description**

Called when the user begins to draw the zoom box by clicking in the chart area and dragging down and to the right.

Returning false will cancel the zoom operation and hide the zoom box.

#### **5.2.3.16 onUserEndZoom**

##### **Definition**

boolean **onUserEndZoom**( [EJSC.Chart](#) chart )

##### **Description**

Called when the user finishes drawing the zoom box but before the zoom takes place.

Returning false will cancel the zoom operation and hide the zoom box.

## 5.2.4 Deprecated

### 5.2.4.1 Properties

#### 5.2.4.1.1 force\_static\_points

**DEPRECATED** See [EJSC.Axis.force\\_static\\_points](#)

#### 5.2.4.1.2 force\_static\_points\_x

**DEPRECATED** See [EJSC.Axis.force\\_static\\_points](#)

#### 5.2.4.1.3 force\_static\_points\_y

**DEPRECATED** See [EJSC.Axis.force\\_static\\_points](#)

#### 5.2.4.1.4 legendTitle

**DEPRECATED** See [EJSC.Chart.legend\\_title](#)

#### 5.2.4.1.5 show\_crosshairs

**DEPRECATED** See [EJSC.Axis.crosshair](#)

#### 5.2.4.1.6 show\_grid

**DEPRECATED** See [EJSC.Axis.grid](#)

#### 5.2.4.1.7 show\_mouse\_position

**DEPRECATED** See [EJSC.Axis.cursor\\_position](#)

#### 5.2.4.1.8 show\_x\_axis

**DEPRECATED** See [EJSC.Axis.visible](#)

#### 5.2.4.1.9 show\_y\_axis

**DEPRECATED** See [EJSC.Axis.visible](#)

#### 5.2.4.1.10 x\_axis\_caption

**DEPRECATED** See [EJSC.Axis.caption](#)

#### 5.2.4.1.11 x\_axis\_className

**DEPRECATED** See [EJSC.Axis.caption\\_class](#)

#### 5.2.4.1.12 x\_axis\_extremes\_ticks

**DEPRECATED** See [EJSC.Axis.extremes\\_ticks](#)

#### 5.2.4.1.13 x\_axis\_formatter

**DEPRECATED** See [EJSC.Axis.formatter](#)

5.2.4.1.14 x\_axis\_max\_tick\_interval

**DEPRECATED** See [EJSC.Axis.major\\_ticks](#)

5.2.4.1.15 x\_axis\_min\_tick\_interval

**DEPRECATED** See [EJSC.Axis.major\\_ticks](#)

5.2.4.1.16 x\_axis\_minor\_ticks

**DEPRECATED** See [EJSC.Axis.minor\\_ticks](#)

5.2.4.1.17 x\_axis\_size

**DEPRECATED** See [EJSC.Axis.size](#)

5.2.4.1.18 x\_axis\_stagger\_ticks

**DEPRECATED** See [EJSC.Axis.stagger\\_ticks](#)

5.2.4.1.19 x\_axis\_tick\_className

**DEPRECATED** See [EJSC.Axis.label\\_class](#)

5.2.4.1.20 x\_axis\_tick\_count

**DEPRECATED** See [EJSC.Axis.major\\_ticks](#)

5.2.4.1.21 x\_cursor\_position\_caption

**DEPRECATED** See [EJSC.Axis.cursor\\_position](#)

5.2.4.1.22 x\_cursor\_position\_formatter

**DEPRECATED** See [EJSC.Axis.cursor\\_position](#)

5.2.4.1.23 x\_max

**DEPRECATED** See [EJSC.Axis.max\\_extreme](#)

5.2.4.1.24 x\_min

**DEPRECATED** See [EJSC.Axis.min\\_extreme](#)

5.2.4.1.25 x\_value\_hint\_caption

**DEPRECATED** See [EJSC.Axis.hint\\_caption](#)

5.2.4.1.26 x\_zero\_plane

**DEPRECATED** See [EJSC.Axis.zero\\_plane](#)

5.2.4.1.27 y\_axis\_caption

**DEPRECATED** See [EJSC.Axis.caption](#)

5.2.4.1.28 y\_axis\_className

**DEPRECATED** See [EJSC.Axis.caption\\_class](#)

5.2.4.1.29 y\_axis\_extremes\_ticks

**DEPRECATED** See [EJSC.Axis.extremes\\_ticks](#)

5.2.4.1.30 y\_axis\_formatter

**DEPRECATED** See [EJSC.Axis.formatter](#)

5.2.4.1.31 y\_axis\_max\_tick\_interval

**DEPRECATED** See [EJSC.Axis.major\\_ticks](#)

5.2.4.1.32 y\_axis\_min\_tick\_interval

**DEPRECATED** See [EJSC.Axis.major\\_ticks](#)

5.2.4.1.33 y\_axis\_minor\_ticks

**DEPRECATED** See [EJSC.Axis.minor\\_ticks](#)

5.2.4.1.34 y\_axis\_size

**DEPRECATED** See [EJSC.Axis.size](#)

5.2.4.1.35 y\_axis\_tick\_className

**DEPRECATED** See [EJSC.Axis.label\\_class](#)

5.2.4.1.36 y\_axis\_tick\_count

**DEPRECATED** See [EJSC.Axis.major\\_ticks](#)

5.2.4.1.37 y\_cursor\_position\_caption

**DEPRECATED** See [EJSC.Axis.cursor\\_position](#)

5.2.4.1.38 y\_cursor\_position\_formatter

**DEPRECATED** See [EJSC.Axis.cursor\\_position](#)

5.2.4.1.39 y\_max

**DEPRECATED** See [EJSC.Axis.max\\_extreme](#)

5.2.4.1.40 y\_min

**DEPRECATED** See [EJSC.Axis.min\\_extreme](#)

5.2.4.1.41 y\_value\_hint\_caption

**DEPRECATED** See [EJSC.Axis.hint\\_caption](#)

5.2.4.1.42 y\_zero\_plane

**DEPRECATED** See [EJSC.Axis.zero\\_plane](#)

#### 5.2.4.2 Methods

5.2.4.2.1 addXAxisBin

**DEPRECATED** See [EJSC.Axis.addBin](#)

5.2.4.2.2 addYAxisBin

**DEPRECATED** See [EJSC.Axis.addBin](#)

5.2.4.2.3 convertPixelToPoint

**DEPRECATED** See [EJSC.Axis.pixelToPoint](#)

5.2.4.2.4 convertPointToPixel

**DEPRECATED** See [EJSC.Axis.pointToPixel](#)

5.2.4.2.5 findClosestPointInSeries

**DEPRECATED** See [EJSC.Series.findClosestPoint](#)

5.2.4.2.6 hideGrid

**DEPRECATED** See [EJSC.Axis.hideGrid](#)

5.2.4.2.7 hideXAxis

**DEPRECATED** See [EJSC.Axis.hide](#)

5.2.4.2.8 hideYAxis

**DEPRECATED** See [EJSC.Axis.hide](#)

5.2.4.2.9 getXExtremes

**DEPRECATED** See [EJSC.Axis.getExtremes](#)

5.2.4.2.10 getYExtremes

**DEPRECATED** See [EJSC.Axis.getExtremes](#)

5.2.4.2.11 getZoom

**DEPRECATED** See [EJSC.Axis.getZoom](#)

5.2.4.2.12 removeXAxisBin

**DEPRECATED** See [EJSC.Axis.removeBin](#)

5.2.4.2.13 removeYAxisBin

**DEPRECATED** See [EJSC.Axis.removeBin](#)

5.2.4.2.14 selectClosestPoint

**DEPRECATED** See [EJSC.Chart.findClosestPoint](#)

5.2.4.2.15 setCrosshairs

**DEPRECATED** See [EJSC.Axis.setCrosshair](#)

5.2.4.2.16 setXAxisCaption

**DEPRECATED** See [EJSC.Axis.setCaption](#)

5.2.4.2.17 setXExtremes

**DEPRECATED** See [EJSC.Axis.setExtremes](#)

5.2.4.2.18 setYAxisCaption

**DEPRECATED** See [EJSC.Axis.setCaption](#)



#### 5.2.4.2.19 setYExtremes

**DEPRECATED** See [EJSC.Axis.setExtremes](#)

#### 5.2.4.2.20 setZoom

**DEPRECATED** See [EJSC.Axis.setZoom](#)

#### 5.2.4.2.21 showGrid

**DEPRECATED** See [EJSC.Axis.showGrid](#)

#### 5.2.4.2.22 showXAxis

**DEPRECATED** See [EJSC.Axis.show](#)

#### 5.2.4.2.23 showYAxis

**DEPRECATED** See [EJSC.Axis.show](#)

### 5.2.4.3 Events

#### 5.2.4.3.1 onAfterShowCrosshairs

**DEPRECATED**

There is no direct replacement for this event. See [EJSC.Axis.onShowCrosshair](#)

#### 5.2.4.3.2 onBeforeBeginZoom

**DEPRECATED** See [EJSC.Chart.onUserBeginZoom](#)

#### 5.2.4.3.3 onBeforeEndZoom

**DEPRECATED** See [EJSC.Chart.onUserEndZoom](#)

#### 5.2.4.3.4 onShowCrosshairs

**DEPRECATED**

There is no direct replacement for this event. See [EJSC.Axis.onShowCrosshair](#)

#### 5.2.4.3.5 onXAxisNeedsTicks

**DEPRECATED** See [EJSC.Axis.onNeedsTicks](#)

#### 5.2.4.3.6 onYAxisNeedsTicks

**DEPRECATED** See [EJSC.Axis.onNeedsTicks](#)

## 5.3 Axes Types

### 5.3.1 LinearAxis

LinearAxis is the default axis type for all four chart axes.

The constructor takes an optional set of object properties used to perform initial configuration.

#### Constructor

```
EJSC.LinearAxis( [ object options] )
```

### Example

```
var chart = new EJSC.Chart("myChart", {  
    axis_bottom: new EJSC.LinearAxis({  
        caption: "Test"  
    })  
});
```

### Defining Properties and Events

LinearAxis properties may be set individually after the chart has been initialized and/or the axis has been created or in batch using the options parameter during instantiation.

#### Setting properties individually

```
var chart = new EJSC.Chart("myChart", {  
    axis_bottom: new EJSC.LinearAxis()  
});  
chart.axis_bottom.caption = "Test";  
chart.axis_bottom.color = "#F00";
```

#### Setting properties in batch

```
var chart = new EJSC.Chart("myChart", {  
    axis_bottom: new EJSC.LinearAxis({  
        caption: "Test",  
        color: "#F00"  
    })  
});
```

### 5.3.1.1 Properties

#### 5.3.1.1.1 background (inherited)

##### Definition

```
object background = {  
    color: "#fff",  
    opacity: 0,  
    includeTitle: false  
};
```

##### Description

Defines the color and opacity of the axis area background. Setting the includeTitle property to true will fill the axis caption area as well as the ticks. If opacity is set to 0, no fill will occur.

#### 5.3.1.1.2 border (inherited)

##### Definition

```
object border = {
    thickness: 1,
    color: undefined,
    opacity: 100,
    show: true
}
```

### Description

Defines the appearance of the axis side bordering the chart area. By default the border color inherits the axis color property ([EJSC.Axis.color](#))

### Example

>> Provide a 2 pixel tall green border on the bottom axis.

```
var chart = new EJSC.Chart(
    "chart",
    {
        axis_bottom:
            { border: { thickness: 2, color: "#0F0" }
            }
    }
);
```

#### 5.3.1.1.3 caption (inherited)

### Definition

```
string caption = "Axis"
```

### Description

Defines the text to be displayed below or beside the axis.

For styling the caption, see [EJSC.Axis.caption\\_class](#)

### Example

>> Customize the bottom axis caption.

```
var chart = new EJSC.Chart(
    "chart",
    {
        axis_bottom: { caption: "Year" }
    }
);
```

#### 5.3.1.1.4 caption\_class (inherited)

### Definition

```
string caption_class = ""
```

**Description**

Defines the CSS className to assign to the axis caption.

For styling the tick labels, see [EJSC.Axis.label\\_class](#)

**Example**

>> *Style the axis caption bold.*

```
<style> .AxisCaption { font-style: bold; } </style>

var chart = new EJSC.Chart(
    "chart",
    {
        axis_bottom: { caption_class: "AxisCaption" }
    }
);
```

## 5.3.1.1.5 color (inherited)

**Definition**

string **color** = "#FFF"

**Description**

Defines the default color of the axis border and tick marks. If the sub properties such as `minor_ticks.color`, `major_ticks.color`, `border.color` are left undefined, they inherit the value set here.

**Example**

>> *Color the axis border and tick marks red*

```
var chart = new EJSC.Chart(
    "chart",
    {
        axis_bottom: { color: "#F00" }
    }
);
```

## 5.3.1.1.6 crosshair (inherited)

**Definition**

```
object crosshair = {
    show: false,
    color: "#F00"
}
```

**Description**

Defines if crosshair should be shown on the chart at the current mouse coordinates as they relate to the given axis. May be enabled and disabled for each axis independently. This is automatically disabled for all axes if [EJSC.Chart.allow\\_interactivity](#) is set to false.

### Example

>> *Show crosshair based on the bottom axis mouse position.*

```
var chart = new EJSC.Chart(
    "chart",
    {
        axis_bottom: {
            crosshair: { show: true }
        }
    }
);
```

#### 5.3.1.1.7 cursor\_position (inherited)

### Definition

```
object cursor_position = {
    show: false,
    color: "#F00",
    textColor: "#FFF",
    formatter: undefined,
    caption: undefined,
    className: undefined
}
```

### Description

Defines the display properties of the cursor position feature for an axis. These properties may be used to turn the cursor position indicator on and off as well as configure the styling and color.

**show:** determines whether or not to display the cursor position while the mouse is within the chart area

**color:** sets the background and line color

**textColor:** sets the text color

**formatter:** defines a formatter to use when displaying coordinates, if left undefined it will use the axis formatter

**caption:** defines text to use as a prefix to the current coordinate

**className:** a CSS class name to be used for additional styling

### Example

>> *Turn on cursor position for the bottom axis and configure to display a related label*

```
var chart = new EJSC.Chart(
    "chart",
    {
        axis_bottom: {
            cursor_position: {
                show: true,
                caption: "Sales"
            }
        }
    }
);
```

```

        }
    }
};

```

#### 5.3.1.1.8 extremes\_ticks (inherited)

##### Definition

boolean **extremes\_ticks** = false

##### Description

Defines if the min and max values should be forced to land on the next tick mark.

##### Example

*>> Force tick marks at the min and max coordinates of the bottom axis (left and right sides of the chart)*

```

var chart = new EJSC.Chart(
    "chart",
    {
        axis_bottom: { extremes_ticks: true }
    }
);

```

#### 5.3.1.1.9 force\_static\_points (inherited)

##### Definition

boolean **force\_static\_points** = false

##### Description

Defines if the chart should force ticks to match up to every point by converting the data to strings.

##### Example

*>> Display every bottom axis data point, essentially disable auto axis scaling.*

```

var chart = new EJSC.Chart(
    "chart",
    {
        axis_bottom: { force_static_points: true }
    }
);

```

#### 5.3.1.1.10 formatter (inherited)

##### Definition

[EJSC.Formatter](#) **formatter** = EJSC.Formatter

## Description

Defines the formatter that will be used to format the tick marks on the axis before displaying them.

Types:

[EJSC.DateFormatter](#)

[EJSC.NumberFormatter](#)

[EJSC.StringFormatter](#)

## Example

*>> Display bottom axis tick labels as \$0.00*

```
var chart = new EJSC.Chart(
    "chart",
    {
        axis_bottom: {
            formatter: new EJSC.NumberFormatter( { currency_symbol: "$",
forced_decimals: 2, variable_decimals: 2 } )
        }
    }
);
```

### 5.3.1.1.11 grid (inherited)

## Definition

```
object grid = {
    thickness: 1,
    color: "rgb(230,230,230)",
    opacity: 100,
    show: true
}
```

## Description

Defines the appearance and visibility of the grid drawn in the background.

## Example

*>> Do not draw the background grid for the top and right axes.*

```
var chart = new EJSC.Chart(
    "chart",
    {
        axis_top: {
            grid: { show: false }
        },
        axis_right: {
            grid: { show: false }
        }
    }
);
```

## 5.3.1.1.12 hint\_caption (inherited)

**Definition**

string `hint_caption` = "Value:"

**Description**

Defines the text to display in front of the axis-related value in the hint (leave blank to hide the value when displaying the hint).

**Example**

>> *Customize the value label in the hint window*

```
var chart = new EJSC.Chart(  
    "chart",  
    {  
        axis_bottom: {  
            hint_caption: "Month:"  
        }  
    }  
);
```

## 5.3.1.1.13 label\_class (inherited)

**Definition**

string `label_class` = ""

**Description**

Defines the CSS className to assign to the axis tick labels. Used in conjunction with [EJSC.Axis.stagger\\_ticks](#) and [EJSC.Axis.size](#), this property allows for greater control over the size and staggering of tick labels on the top and bottom axes.

*For styling the caption, see [EJSC.Axis.caption\\_class](#)*

**Example**

>> *Style the axis tick labels grey, make them 24 pixels high to enable text wrapping and enable two levels of tick staggering.*

```
<style> .AxisTickLabels { color: #999; height: 24px; } </style>  
  
var chart = new EJSC.Chart(  
    "chart",  
    {  
        axis_bottom: {  
            label_class: "AxisTickLabels",  
            size: 48  
        }  
    }  
);
```



## 5.3.1.1.14 major\_ticks (inherited)

**Definition**

```
object major_ticks = {  
  thickness: 1,  
  size: 4,  
  color: undefined,  
  opacity: 100,  
  show: true,  
  count: undefined,  
  offset: 0,  
  min_interval: undefined,  
  max_interval: undefined  
}
```

**Description**

This set of properties defines the characteristics of the major ticks for a given axis.

**thickness:** the height or width (depending on axis orientation) of the tick mark

**size:** the amount the tick mark extends from the axis border, may be specified as a number or a string containing % for a percentage

**color:** the color of the tick marks, if left undefined this property inherits its value from

[EJSC.Axis.color](#)

**opacity:** the opacity of the tick marks

**show:** specifies whether to draw the tick marks

**count:** the number of tick marks to draw, if left undefined the axis will determine the proper amount of ticks to draw automatically based on the range of data available to the axis

**Note:** This property is not compatible with text labels (i.e. x axis values are "Gizmos", "Widgets", instead of numbers)

**offset:** distance in pixels or percent from the axis border to begin drawing the tick marks,

**min\_interval:** Defines the minimum interval between major tick marks / labels. This will override the auto generation of ticks to cap the interval to the value when defined.

**max\_interval:** Defines the maximum interval between major tick marks / labels. This will override the auto generation of ticks to cap the interval to the value when defined.

## 5.3.1.1.15 max\_extreme (inherited)

**Definition**

```
float max_extreme = undefined
```

**Description**

Defines the maximum value of the the axis. This will only affect the axis when set during chart creation and may be used to extend or truncate the value range displayed. To retrieve and set this value after the chart has been created, see [EJSC.Axis.getExtremes](#) and [EJSC.Axis.setExtremes](#)

**Example**

>> Force the axis range to extend beyond the data it contains

```
var chart = new EJSC.Chart(  
    "chart",  
    {  
        axis_bottom: {  
            max_extreme: 500.00  
        }  
    }  
);
```

#### 5.3.1.1.16 min\_extreme (inherited)

##### Definition

float **min** = undefined

##### Description

Defines the minimum value of the the axis. This will only affect the axis when set during chart creation and may be used to extend or truncate the value range displayed. To retrieve and set this value after the chart has been created, see [EJSC.Axis.getExtremes](#) and [EJSC.Axis.setExtremes](#)

##### Example

>> Force the axis range to extend beyond the data it contains

```
var chart = new EJSC.Chart(  
    "chart",  
    {  
        axis_bottom: {  
            min_extreme: -500.00  
        }  
    }  
);
```

#### 5.3.1.1.17 minor\_ticks (inherited)

##### Definition

```
object minor_ticks = {  
    show: false,  
    color: "rgb(0,0,0)",  
    opacity: 20  
    thickness: 1,  
    count: 7,  
    size: 4,  
    offset: 0  
}
```

##### Description

Defines the properties of the minor tick marks to be drawn on the axis. The color, opacity and thickness (width of ticks in pixels) properties define the style of the tick marks. The count property defines the number of tick marks to be drawn between each major tick mark. The size property defines the height of the ticks (in pixels or percent). The offset property defines the distance away from the axis the minor tick marks begin drawing.

### Example

*>> Display red minor tick marks on the bottom axis*

```
var chart = new EJSC.Chart(  
    "chart",  
    {  
        axis_bottom: {  
            minor_ticks: { show: true, color: "#FF0000" }  
        }  
    }  
);
```

#### 5.3.1.1.18 size (inherited)

### Definition

integer **size** = 20

### Description

Defines the height of horizontal axes or width of vertical axes (in pixels) of the tick area. To fully enable staggered ticks on horizontal axes, set this property to a multiple of 20 (or axis tick height), i.e. two levels = 40, three levels = 60.

For additional control over the format of the labels, see the [EJSC.Axis.label\\_class](#) property.

### Example

*>> Make axis tick area twice as tall, enabling staggered ticks (default tick label height is 20 pixels)*

```
var chart = new EJSC.Chart(  
    "chart",  
    {  
        axis_bottom: { size: 40 }  
    }  
);
```

#### 5.3.1.1.19 stagger\_ticks (inherited)

### Definition

boolean **stagger\_ticks** = true

### Description

Determines whether the axis tick labels are staggered.

**NOTE:** This property is only applicable to horizontal axes (i.e. [EJSC.Chart.axis\\_top](#) and [EJSC.Chart.axis\\_bottom](#))

### Example

>> *Disable tick staggering for the bottom axis*

```
var chart = new EJSC.Chart(  
    "chart",  
    {  
        axis_bottom: { stagger_ticks: false }  
    }  
);
```

#### 5.3.1.1.20 visible (inherited)

### Definition

boolean **visible** = true

### Description

Defines if the axis (containing axis caption and tick labels) should be displayed.

### Example

>> *Remove the bottom axis from the chart to allow additional room for data*

```
var chart = new EJSC.Chart(  
    "chart",  
    {  
        axis_bottom: { visible: false }  
    }  
);
```

#### 5.3.1.1.21 zero\_plane (inherited)

### Definition

```
object zero_plane = {  
    color: "rgb(0,0,0)",  
    show: false,  
    opacity: 100,  
    thickness: 1,  
    coordinate: 0  
}
```

### Description

Defines the properties of the zero plane line to be drawn at 0 (or whatever the coordinate property is set to). It is used to specify if the line should be shown as well as its color, thickness and opacity. The coordinate property allows the base of [EJSC.BarSeries](#), [EJSC.StackedBarSeries](#) and [EJSC.AreaSeries](#)

changed.

### Example

>> *Display a 2 pixel thick dark green line on the zero plane of the left axis*

```
var chart = new EJSC.Chart(  
    "chart",  
    {  
        axis_left: {  
            zero_plane: { show: true, color: "rgb(7,89,5)", thickness:  
2 }  
        }  
    }  
);
```

### 5.3.1.2 Methods

#### 5.3.1.2.1 addBin (inherited)

#### Definition

void **addBin**( String label, boolean redraw )

#### Description

Adds a new static label to the axis and sets the appropriate flags if necessary to force the chart to draw using static labels (as opposed to a dynamic range based on series data).

This method is useful if there is a need to include bins which may not be part of any series added (i.e. chart labels should be "Apples", "Oranges", "Pears" but the series data only contains data pertaining to Apples and Pears).

If redraw is false, drawing will not occur immediately but will not be entirely prevented. Certain actions such as adding an additional series with redraw = true, or chart dimensions changing may still trigger the chart to draw. The default, if redraw is omitted is **true**.

#### Note:

Manually added bins may only be removed if there are no series currently utilizing them.

#### Example

>> *Add bins in a specific order to ensure they appear the same each time the chart is loaded, regardless of the order in which they appear in the data.*

```
var myChart = new EJSC.Chart("chart");  
myChart.axis_bottom.addBin("Salesman 1");  
myChart.axis_bottom.addBin("Salesman 2");  
myChart.axis_bottom.addBin("Salesman 3");
```

```
var mySeries = myChart.addSeries(  
    new EJSC.BarSeries(...)
```

);

#### 5.3.1.2.2 getExtremes (inherited)

##### Definition

object [getExtremes\(\)](#)

RETURNS:  
{ min: float, max: float }

##### Description

Returns the current axis extreme values.

To set the extreme values, see [EJSC.Axis.setExtremes](#)

#### 5.3.1.2.3 getZoom (inherited)

##### Definition

object [getZoom\(\)](#)

RETURNS:  
{ min: float, max: float }

##### Description

Returns the current zoom coordinates for a given axis in chart units.

To set the zoom coordinates, see [EJSC.Axis.setZoom](#)

#### 5.3.1.2.4 getZoomBoxCoordinates (inherited)

##### Definition

void [getZoomBoxCoordinates\(\)](#)

RETURNS:  
{ min: float, max: float }

##### Description

Returns the min and max values for a given axis of the current zoom box. These values are in chart coordinates, not pixels.

#### 5.3.1.2.5 hide (inherited)

##### Definition

void [hide\(\)](#)

**Description**

Hides the axis, resizes the chart area and redraws all visible series.

No effect if the axis is already hidden.

5.3.1.2.6 hideGrid (inherited)

**Definition**

void **hideGrid**( boolean redraw )

**Description**

Hides the background grid for a given axis and if redraw is omitted or true, redraws the chart

No effect if the grid is already hidden.

5.3.1.2.7 pixelToPoint (inherited)

**Definition**

float **pixelToPoint**( integer Pixel )

**Description**

Converts the pixel coordinate into chart units based on an axis. The result will be undefined if the pixel location is outside of the chart area.

5.3.1.2.8 pointToPixel (inherited)

**Definition**

integer **pointToPixel**( number coordinate )

**Description**

Converts the chart coordinates based on axis data into the appropriate pixel position for that point (x or y depending on the orientation of the axis)

The result will be undefined if the coordinate provided is outside the currently displayed range.

5.3.1.2.9 removeBin (inherited)

**Definition**

void **removeBin**( String label, boolean redraw )

**Description**

Removes a bin (static text label) from the axis.

If redraw is false, drawing will not occur immediately but will not be entirely prevented. Certain actions such as adding an additional series with redraw = true, or chart dimensions changing may still trigger the chart to draw. The default, if redraw is omitted is **true**.

**Note:**

Bins may only be removed using this method if they were added using [EJSC.Axis.addBin](#) and no series are currently utilizing them.

5.3.1.2.10 resetZoom (inherited)

**Definition**

void [resetZoom](#)()

**Description**

Resets the zoom for a given axis to use its extreme values for min and max (as if a user had double-clicked the chart or drawn the zoom rectangle anywhere but down and right)

5.3.1.2.11 setCaption (inherited)

**Definition**

void [setCaption](#)( string caption )

**Description**

Updates the axis caption. This method must be used to change the caption once the chart has been rendered. To set a default caption on creation, see [EJSC.Axis.caption](#).

5.3.1.2.12 setCrosshair (inherited)

**Definition**

void [setCrosshair](#)( boolean visible, float coordinate, boolean fireEvent )

**Description**

This method may be used to hide, show and position the cursor position indicator. If the fireEvent parameter is set to true or left undefined the [EJSC.Axis.onShowCrosshair](#) or [EJSC.Axis.onHideCrosshair](#) events will fire.

5.3.1.2.13 setExtremes (inherited)

**Definition**

void [setExtremes](#)( float min, float max )



### Description

Updates the manual extremes for the axis. Use this method if the series data does not span the range to be displayed on the chart or to limit 100% zoom to a range smaller than the series data.

### Example

*>> Series data only spans 18 hours on the bottom axis but the chart needs to display an entire day*

```
var myChart = new EJSC.Chart( "chart" );  
myChart.axis_bottom.setExtremes( 0, 86400000 );
```

#### 5.3.1.2.14 setZoom (inherited)

##### Definition:

void **setZoom**( float min, float max )

##### Description

Sets the current zoom of the axis to the specified coordinates.

#### 5.3.1.2.15 show (inherited)

##### Definition

void **show**( )

##### Description

Shows the axis, resizes the chart area and redraws all visible series.

No effect if the axis is already visible.

#### 5.3.1.2.16 showGrid (inherited)

##### Definition

void **showGrid**( boolean redraw )

##### Description

Shows the background grid for the axis and if redraw is omitted or true, redraws the chart.

No effect if the grid is already visible.

### 5.3.1.3 Events

#### 5.3.1.3.1 onHideCrosshair (inherited)

##### Definition

void **onHideCrosshair**( [EJSC.Axis](#) axis, [EJSC.Chart](#) chart )

### Description

Called when the axis crosshair is hidden by the chart. This can occur when the user moves their mouse outside of the chart area or when [EJSC.Axis.setCrosshair](#) is called, sending false for the visible parameter and true for the fireEvent parameter.

5.3.1.3.2 onHideCursorPosition (inherited)

### Definition

void **onHideCursorPosition**( [EJSC.Axis](#) axis, [EJSC.Chart](#) chart )

### Description

This event is fired when the cursor position indicator for an axis is hidden. This will occur when the user's mouse leaves the chart area.

5.3.1.3.3 onNeedsTicks (inherited)

### Definition

array **onNeedsTicks**( float min, float max, [EJSC.Axis](#) axis, [EJSC.Chart](#) chart)

### Description

This event is triggered whenever the axis ticks need to be redrawn / recalculated. It expects an array of [ float y, string label ] to be returned which defines exactly where to put the tick marks and labels. In addition, null may be returned in order to skip custom ticks for the current draw and use the chart's build in tick controls.

**min:** The current minimum value visible on the chart for the axis.

**max:** The current maximum y value visible on the chart for the axis.

**axis:** The axis which needs ticks.

**chart:** The chart that triggered the event.

### Notes

- To use the label formatter already assigned to the axis, set label to null (i.e. [min, null])
- To use on an axis with bins (text instead of numbers), simply send in the bin (i.e. ["First Bin", null])

### Example

A typical event handler may look like the following:

```
function doBottomAxisNeedsTicks(min, max, axis, chart) {  
  
    // Display 3 tick marks, one at min, one at max and one directly in between  
    var result = new Array();
```

```
result.push( [min, null] );
result.push( [min + ((max - min) / 2), null] );
result.push( [max, null] );

// Given a chart with a min and max of 0 and 100, the resulting array looks like:
// [
//   [0, null],
//   [50, null],
//   [100, null]
// ]
return result;
}
```

#### 5.3.1.3.4 onShowCrosshair (inherited)

##### Definition

void **onShowCrosshair**( float coordinate, [EJSC.Axis](#) axis, [EJSC.Chart](#) chart )

##### Description

Called when the axis crosshair is shown by the chart. This can occur when the user moves their mouse into the chart area or when [EJSC.Axis.setCrosshair](#) is called, sending true for the visible parameter and true for the fireEvent parameter.

#### 5.3.1.3.5 onShowCursorPosition (inherited)

##### Definition

void **onShowCursorPosition**( float coordinate, [EJSC.Axis](#) axis, [EJSC.Chart](#) chart )

##### Description

Called when the cursor position indicator becomes visible or changes position on a given axis. This will occur when the user enters the chart area or moves their mouse within the chart area.

## 5.3.2 LogarithmicAxis

LogarithmicAxis may be created and assigned to any of the four chart axes in order to enable a logarithmic scale.

The constructor takes an optional set of object properties used to perform initial configuration.

##### Constructor

```
EJSC.LogarithmicAxis( [ object options] )
```

##### Example

```
var chart = new EJSC.Chart("myChart", {
  axis_bottom: new EJSC.LogarithmicAxis({
```

```
        caption: "Test",
        base: 8
    })
});
```

## Defining Properties and Events

LogarithmicAxis properties may be set individually after the chart has been initialized and/or the axis has been created or in batch using the options parameter during instantiation.

### Setting properties individually

```
var chart = new EJSC.Chart("myChart", {
    axis_bottom: new EJSC.LogarithmicAxis()
});
chart.axis_bottom.base = 8;
chart.axis_bottom.caption = "Test";
chart.axis_bottom.color = "#F00";
```

### Setting properties in batch

```
var chart = new EJSC.Chart("myChart", {
    axis_bottom: new EJSC.LinearAxis({
        base: 8,
        caption: "Test",
        color: "#F00"
    })
});
```

## 5.3.2.1 Properties

### 5.3.2.1.1 background (inherited)

#### Definition

```
object background = {
    color: "#fff",
    opacity: 0,
    includeTitle: false
};
```

#### Description

Defines the color and opacity of the axis area background. Setting the includeTitle property to true will fill the axis caption area as well as the ticks. If opacity is set to 0, no fill will occur.

### 5.3.2.1.2 base

#### Definition

```
number base = 10
```

#### Description

The term "base" refers to the number which is set to incremental powers to determine the ticks on the axis.

The tick values, if base = b, would be  $b^x$  where x is the linear incremental power.

When base = 10, and your range is 10 - 10,000, the tick marks would be:

```
10^1 = 10
10^2 = 100
10^3 = 1,000
10^4 = 10,000
```

#### 5.3.2.1.3 border (inherited)

##### Definition

```
object border = {
    thickness: 1,
    color: undefined,
    opacity: 100,
    show: true
}
```

##### Description

Defines the appearance of the axis side bordering the chart area. By default the border color inherits the axis color property ([EJSC.Axis.color](#))

##### Example

*>> Provide a 2 pixel tall green border on the bottom axis.*

```
var chart = new EJSC.Chart(
    "chart",
    {
        axis_bottom:
            { border: { thickness: 2, color: "#0F0" } }
    }
);
```

#### 5.3.2.1.4 caption (inherited)

##### Definition

```
string caption = "Axis"
```

##### Description

Defines the text to be displayed below or beside the axis.

*For styling the caption, see [EJSC.Axis.caption\\_class](#)*

### Example

>> *Customize the bottom axis caption.*

```
var chart = new EJSC.Chart(  
    "chart",  
    {  
        axis_bottom: { caption: "Year" }  
    }  
);
```

#### 5.3.2.1.5 caption\_class (inherited)

### Definition

string **caption\_class** = ""

### Description

Defines the CSS className to assign to the axis caption.

*For styling the tick labels, see [EJSC.Axis.label\\_class](#)*

### Example

>> *Style the axis caption bold.*

```
<style> .AxisCaption { font-style: bold; } </style>  
  
var chart = new EJSC.Chart(  
    "chart",  
    {  
        axis_bottom: { caption_class: "AxisCaption" }  
    }  
);
```

#### 5.3.2.1.6 color (inherited)

### Definition

string **color** = "#FFF"

### Description

Defines the default color of the axis border and tick marks. If the sub properties such as `minor_ticks.color`, `major_ticks.color`, `border.color` are left undefined, they inherit the value set here.

### Example

>> *Color the axis border and tick marks red*

```
var chart = new EJSC.Chart(  
    "chart",  
    {
```

```
        axis_bottom: { color: "#F00" }
    };
```

#### 5.3.2.1.7 crosshair (inherited)

##### Definition

```
object crosshair = {
    show: false,
    color: "#F00"
}
```

##### Description

Defines if crosshair should be shown on the chart at the current mouse coordinates as they relate to the given axis. May be enabled and disabled for each axis independently. This is automatically disabled for all axes if [EJSC.Chart.allow\\_interactivity](#) is set to false.

##### Example

*>> Show crosshair based on the bottom axis mouse position.*

```
var chart = new EJSC.Chart(
    "chart",
    {
        axis_bottom: {
            crosshair: { show: true }
        }
    }
);
```

#### 5.3.2.1.8 cursor\_position (inherited)

##### Definition

```
object cursor_position = {
    show: false,
    color: "#F00",
    textColor: "#FFF",
    formatter: undefined,
    caption: undefined,
    className: undefined
}
```

##### Description

Defines the display properties of the cursor position feature for an axis. These properties may be used to turn the cursor position indicator on and off as well as configure the styling and color.

**show:** determines whether or not to display the cursor position while the mouse is within the chart area

**color:** sets the background and line color

**textColor:** sets the text color

**formatter:** defines a formatter to use when displaying coordinates, if left undefined it will use the axis formatter

**caption:** defines text to use as a prefix to the current coordinate

**className:** a CSS class name to be used for additional styling

### Example

*>> Turn on cursor position for the bottom axis and configure to display a related label*

```
var chart = new EJSC.Chart(
    "chart",
    {
        axis_bottom: {
            cursor_position: {
                show: true,
                caption: "Sales"
            }
        }
    }
);
```

#### 5.3.2.1.9 extremes\_ticks (inherited)

### Definition

boolean **extremes\_ticks** = false

### Description

Defines if the min and max values should be forced to land on the next tick mark.

### Example

*>> Force tick marks at the min and max coordinates of the bottom axis (left and right sides of the chart)*

```
var chart = new EJSC.Chart(
    "chart",
    {
        axis_bottom: { extremes_ticks: true }
    }
);
```

#### 5.3.2.1.10 force\_static\_points (inherited)

### Definition

boolean **force\_static\_points** = false

### Description

Defines if the chart should force ticks to match up to every point by converting the data to strings.

### Example



>> *Display every bottom axis data point, essentially disable auto axis scaling.*

```
var chart = new EJSC.Chart(
    "chart",
    {
        axis_bottom: { force_static_points: true }
    }
);
```

#### 5.3.2.1.11 formatter (inherited)

##### Definition

[EJSC.Formatter](#) **formatter** = EJSC.Formatter

##### Description

Defines the formatter that will be used to format the tick marks on the axis before displaying them.

Types:

[EJSC.DateFormatter](#)

[EJSC.NumberFormatter](#)

[EJSC.StringFormatter](#)

##### Example

>> *Display bottom axis tick labels as \$0.00*

```
var chart = new EJSC.Chart(
    "chart",
    {
        axis_bottom: {
            formatter: new EJSC.NumberFormatter( { currency_symbol: "$",
forced_decimals: 2, variable_decimals: 2 } )
        }
    }
);
```

#### 5.3.2.1.12 grid (inherited)

##### Definition

```
object grid = {
    thickness: 1,
    color: "rgb(230,230,230)",
    opacity: 100,
    show: true
}
```

##### Description

Defines the appearance and visibility of the grid drawn in the background.

### Example

>> Do not draw the background grid for the top and right axes.

```
var chart = new EJSC.Chart(
    "chart",
    {
        axis_top: {
            grid: { show: false }
        },
        axis_right: {
            grid: { show: false }
        }
    }
);
```

#### 5.3.2.1.13 hint\_caption

### Definition

string **hint\_caption** = "Value:"

### Description

Defines the text to display in front of the axis-related value in the hint (leave blank to hide the value when displaying the hint).

### Example

>> Customize the value label in the hint window

```
var chart = new EJSC.Chart(
    "chart",
    {
        axis_bottom: {
            hint_caption: "Month:"
        }
    }
);
```

#### 5.3.2.1.14 label\_class (inherited)

### Definition

string **label\_class** = ""

### Description

Defines the CSS className to assign to the axis tick labels. Used in conjunction with [EJSC.Axis.stagger\\_ticks](#) and [EJSC.Axis.size](#), this property allows for greater control over the size and staggering of tick labels on the top and bottom axes.

For styling the caption, see [EJSC.Axis.caption\\_class](#)

### Example

>> Style the axis tick labels grey, make them 24 pixels high to enable text wrapping and enable two levels of tick staggering.

```
<style> .AxisTickLabels { color: #999; height: 24px; } </style>

var chart = new EJSChart(
    "chart",
    {
        axis_bottom: {
            label_class: "AxisTickLabels",
            size: 48
        }
    }
);
```

#### 5.3.2.1.15 major\_ticks (inherited)

##### Definition

```
object major_ticks = {
    thickness: 1,
    size: 4,
    color: undefined,
    opacity: 100,
    show: true,
    count: undefined,
    offset: 0,
    min_interval: undefined,
    max_interval: undefined
}
```

##### Description

This set of properties defines the characteristics of the major ticks for a given axis.

**thickness:** the height or width (depending on axis orientation) of the tick mark

**size:** the amount the tick mark extends from the axis border, may be specified as a number or a string containing % for a percentage

**color:** the color of the tick marks, if left undefined this property inherits its value from

[EJSChart.Axis.color](#)

**opacity:** the opacity of the tick marks

**show:** specifies whether to draw the tick marks

**count:** the number of tick marks to draw, if left undefined the axis will determine the proper amount of ticks to draw automatically based on the range of data available to the axis

**Note:** This property is not compatible with text labels (i.e. x axis values are "Gizmos", "Widgets", instead of numbers)

**offset:** distance in pixels or percent from the axis border to begin drawing the tick marks,

**min\_interval:** Defines the minimum interval between major tick marks / labels. This will override the auto generation of ticks to cap the interval to the value when defined.

**max\_interval:** Defines the maximum interval between major tick marks / labels. This will override the auto generation of ticks to cap the interval to the value when defined.

## 5.3.2.1.16 max\_extreme (inherited)

**Definition**

float **max\_extreme** = undefined

**Description**

Defines the maximum value of the the axis. This will only affect the axis when set during chart creation and may be used to extend or truncate the value range displayed. To retrieve and set this value after the chart has been created, see [EJSC.Axis.getExtremes](#) and [EJSC.Axis.setExtremes](#)

**Example**

*>> Force the axis range to extend beyond the data it contains*

```
var chart = new EJSC.Chart(  
    "chart",  
    {  
        axis_bottom: {  
            max_extreme: 500.00  
        }  
    }  
);
```

## 5.3.2.1.17 min\_extreme (inherited)

**Definition**

float **min** = undefined

**Description**

Defines the minimum value of the the axis. This will only affect the axis when set during chart creation and may be used to extend or truncate the value range displayed. To retrieve and set this value after the chart has been created, see [EJSC.Axis.getExtremes](#) and [EJSC.Axis.setExtremes](#)

**Example**

*>> Force the axis range to extend beyond the data it contains*

```
var chart = new EJSC.Chart(  
    "chart",  
    {  
        axis_bottom: {  
            min_extreme: -500.00  
        }  
    }  
);
```

## 5.3.2.1.18 minor\_ticks (inherited)

**Definition**

```
object minor_ticks = {  
  show: false,  
  color: "rgb(0,0,0)",  
  opacity: 20  
  thickness: 1,  
  count: 7,  
  size: 4,  
  offset: 0  
}
```

**Description**

Defines the properties of the minor tick marks to be drawn on the axis. The color, opacity and thickness (width of ticks in pixels) properties define the style of the tick marks. The count property defines the number of tick marks to be drawn between each major tick mark. The size property defines the height of the ticks (in pixels or percent). The offset property defines the distance away from the axis the minor tick marks begin drawing.

**Example**

*>> Display red minor tick marks on the bottom axis*

```
var chart = new EJSC.Chart(  
  "chart",  
  {  
    axis_bottom: {  
      minor_ticks: { show: true, color: "#FF0000" }  
    }  
  }  
);
```

## 5.3.2.1.19 size (inherited)

**Definition**

integer **size** = 20

**Description**

Defines the height of horizontal axes or width of vertical axes (in pixels) of the tick area. To fully enable staggered ticks on horizontal axes, set this property to a multiple of 20 (or axis tick height), i.e. two levels = 40, three levels = 60.

For additional control over the format of the labels, see the [EJSC.Axis.label\\_class](#) property.

**Example**

*>> Make axis tick area twice as tall, enabling staggered ticks (default tick label height is 20 pixels)*

```
var chart = new EJSC.Chart(  
    "chart",  
    {  
        axis_bottom: { size: 40 }  
    }  
);
```

#### 5.3.2.1.20 stagger\_ticks (inherited)

##### Definition

boolean **stagger\_ticks** = true

##### Description

Determines whether the axis tick labels are staggered.

**NOTE:** This property is only applicable to horizontal axes (i.e. [EJSC.Chart.axis\\_top](#) and [EJSC.Chart.axis\\_bottom](#))

##### Example

*>> Disable tick staggering for the bottom axis*

```
var chart = new EJSC.Chart(  
    "chart",  
    {  
        axis_bottom: { stagger_ticks: false }  
    }  
);
```

#### 5.3.2.1.21 visible (inherited)

##### Definition

boolean **visible** = true

##### Description

Defines if the axis (containing axis caption and tick labels) should be displayed.

##### Example

*>> Remove the bottom axis from the chart to allow additional room for data*

```
var chart = new EJSC.Chart(  
    "chart",  
    {  
        axis_bottom: { visible: false }  
    }  
);
```

## 5.3.2.1.22 zero\_plane (inherited)

**Definition**

```
object zero_plane = {
    color: "rgb(0,0,0)",
    show: false,
    opacity: 100,
    thickness: 1,
    coordinate: 0
}
```

**Description**

Defines the properties of the zero plane line to be drawn at 0 (or whatever the coordinate property is set to). It is used to specify if the line should be shown as well as its color, thickness and opacity. The coordinate property allows the base of [EJSC.BarSeries](#), [EJSC.StackedBarSeries](#) and [EJSC.AreaSeries](#) changed.

**Example**

*>> Display a 2 pixel thick dark green line on the zero plane of the left axis*

```
var chart = new EJSC.Chart(
    "chart",
    {
        axis_left: {
            zero_plane: { show: true, color: "rgb(7,89,5)", thickness:
2 }
        }
    }
);
```

**5.3.2.2 Methods**

## 5.3.2.2.1 addBin (inherited)

**Definition**

```
void addBin( String label, boolean redraw )
```

**Description**

Adds a new static label to the axis and sets the appropriate flags if necessary to force the chart to draw using static labels (as opposed to a dynamic range based on series data).

This method is useful if there is a need to include bins which may not be part of any series added (i.e. chart labels should be "Apples", "Oranges", "Pears" but the series data only contains data pertaining to Apples and Pears).

If redraw is false, drawing will not occur immediately but will not be entirely prevented. Certain actions such as adding an additional series with redraw = true, or chart dimensions changing may still trigger the chart to draw. The default, if redraw is omitted is **true**.

**Note:**

Manually added bins may only be removed if there are no series currently utilizing them.

**Example**

*>> Add bins in a specific order to ensure they appear the same each time the chart is loaded, regardless of the order in which they appear in the data.*

```
var myChart = new EJSC.Chart("chart");
myChart.axis_bottom.addBin("Salesman 1");
myChart.axis_bottom.addBin("Salesman 2");
myChart.axis_bottom.addBin("Salesman 3");

var mySeries = myChart.addSeries(
    new EJSC.BarSeries(...)
);
```

## 5.3.2.2.2 getExtremes (inherited)

**Definition**

object [getExtremes\(\)](#) ( )

RETURNS:  
{ min: float, max: float }

**Description**

Returns the current axis extreme values.

To set the extreme values, see [EJSC.Axis.setExtremes](#)

## 5.3.2.2.3 getZoom (inherited)

**Definition**

object [getZoom\(\)](#) ( )

RETURNS:  
{ min: float, max: float }

**Description**

Returns the current zoom coordinates for a given axis in chart units.

To set the zoom coordinates, see [EJSC.Axis.setZoom](#)

## 5.3.2.2.4 getZoomBoxCoordinates (inherited)

**Definition**



void **getZoomBoxCoordinates**()

RETURNS:

{ min: float, max: float }

### Description

Returns the min and max values for a given axis of the current zoom box. These values are in chart coordinates, not pixels.

#### 5.3.2.2.5 hide (inherited)

### Definition

void **hide**( )

### Description

Hides the axis, resizes the chart area and redraws all visible series.

No effect if the axis is already hidden.

#### 5.3.2.2.6 hideGrid (inherited)

### Definition

void **hideGrid**( boolean redraw )

### Description

Hides the background grid for a given axis and if redraw is omitted or true, redraws the chart

No effect if the grid is already hidden.

#### 5.3.2.2.7 pixelToPoint (inherited)

### Definition

float **pixelToPoint**( integer Pixel )

### Description

Converts the pixel coordinate into chart units based on an axis. The result will be undefined if the pixel location is outside of the chart area.

#### 5.3.2.2.8 pointToPixel (inherited)

### Definition

integer **pointToPixel**( number coordinate )

**Description**

Converts the chart coordinates based on axis data into the appropriate pixel position for that point (x or y depending on the orientation of the axis)

The result will be undefined if the coordinate provided is outside the currently displayed range.

## 5.3.2.2.9 removeBin (inherited)

**Definition**

void **removeBin**( String label, boolean redraw )

**Description**

Removes a bin (static text label) from the axis.

If redraw is false, drawing will not occur immediately but will not be entirely prevented. Certain actions such as adding an additional series with redraw = true, or chart dimensions changing may still trigger the chart to draw. The default, if redraw is omitted is **true**.

**Note:**

Bins may only be removed using this method if they were added using [EJSC.Axis.addBin](#) and no series are currently utilizing them.

## 5.3.2.2.10 resetZoom (inherited)

**Definition**

void **resetZoom**()

**Description**

Resets the zoom for a given axis to use its extreme values for min and max (as if a user had double-clicked the chart or drawn the zoom rectangle anywhere but down and right)

## 5.3.2.2.11 setCaption (inherited)

**Definition**

void **setCaption**( string caption )

**Description**

Updates the axis caption. This method must be used to change the caption once the chart has been rendered. To set a default caption on creation, see [EJSC.Axis.caption](#).

## 5.3.2.2.12 setCrosshair (inherited)

**Definition**

void **setCrosshair**( boolean visible, float coordinate, boolean fireEvent )

**Description**

This method may be used to hide, show and position the cursor position indicator. If the fireEvent parameter is set to true or left undefined the [EJSC.Axis.onShowCrosshair](#) or [EJSC.Axis.onHideCrosshair](#) events will fire.

## 5.3.2.2.13 setExtremes (inherited)

**Definition**

void **setExtremes**( float min, float max )

**Description**

Updates the manual extremes for the axis. Use this method if the series data does not span the range to be displayed on the chart or to limit 100% zoom to a range smaller than the series data.

**Example**

*>> Series data only spans 18 hours on the bottom axis but the chart needs to display an entire day*

```
var myChart = new EJSC.Chart( "chart" );  
myChart.axis_bottom.setExtremes( 0, 86400000 );
```

## 5.3.2.2.14 setZoom (inherited)

**Definition:**

void **setZoom**( float min, float max )

**Description**

Sets the current zoom of the axis to the specified coordinates.

## 5.3.2.2.15 show (inherited)

**Definition**

void **show**( )

**Description**

Shows the axis, resizes the chart area and redraws all visible series.

No effect if the axis is already visible.

#### 5.3.2.2.16 showGrid (inherited)

##### Definition

void [showGrid](#)( boolean redraw )

##### Description

Shows the background grid for the axis and if redraw is omitted or true, redraws the chart.

No effect if the grid is already visible.

#### 5.3.2.3 Events

##### 5.3.2.3.1 onHideCrosshair (inherited)

##### Definition

void [onHideCrosshair](#)( [EJSC.Axis](#) axis, [EJSC.Chart](#) chart )

##### Description

Called when the axis crosshair is hidden by the chart. This can occur when the user moves their mouse outside of the chart area or when [EJSC.Axis.setCrosshair](#) is called, sending false for the visible parameter and true for the fireEvent parameter.

##### 5.3.2.3.2 onHideCursorPosition (inherited)

##### Definition

void [onHideCursorPosition](#)( [EJSC.Axis](#) axis, [EJSC.Chart](#) chart )

##### Description

This event is fired when the cursor position indicator for an axis is hidden. This will occur when the user's mouse leaves the chart area.

##### 5.3.2.3.3 onNeedsTicks (inherited)

##### Definition

array [onNeedsTicks](#)( float min, float max, [EJSC.Axis](#) axis, [EJSC.Chart](#) chart)

##### Description

This event is triggered whenever the axis ticks need to be redrawn / recalculated. It expects an array of [ float y, string label ] to be returned which defines exactly where to put the tick marks and labels. In addition, null may be returned in order to skip custom ticks for the current draw and use the chart's build in tick controls.

**min:** The current minimum value visible on the chart for the axis.  
**max:** The current maximum y value visible on the chart for the axis.  
**axis:** The axis which needs ticks.  
**chart:** The chart that triggered the event.

### Notes

- To use the label formatter already assigned to the axis, set label to null (i.e. [min, null])
- To use on an axis with bins (text instead of numbers), simply send in the bin (i.e. ["First Bin", null])

### Example

A typical event handler may look like the following:

```
function doBottomAxisNeedsTicks(min, max, axis, chart) {

    // Display 3 tick marks, one at min, one at max and one directly in between
    var result = new Array();

    result.push( [min, null] );
    result.push( [min + ((max - min) / 2), null] );
    result.push( [max, null] );

    // Given a chart with a min and max of 0 and 100, the resulting array looks like:
    // [
    //   [0, null],
    //   [50, null],
    //   [100, null]
    // ]
    return result;
}
```

#### 5.3.2.3.4 onShowCrosshair (inherited)

### Definition

void [onShowCrosshair](#)( float coordinate, [EJSC.Axis](#) axis, [EJSC.Chart](#) chart )

### Description

Called when the axis crosshair is shown by the chart. This can occur when the user moves their mouse into the chart area or when [EJSC.Axis.setCrosshair](#) is called, sending true for the visible parameter and true for the fireEvent parameter.

#### 5.3.2.3.5 onShowCursorPosition (inherited)

### Definition

void [onShowCursorPosition](#)( float coordinate, [EJSC.Axis](#) axis, [EJSC.Chart](#) chart )

## Description

Called when the cursor position indicator becomes visible or changes position on a given axis. This will occur when the user enters the chart area or moves their mouse within the chart area.

## 5.4 Series Types

### 5.4.1 EJSC.AnalogGaugeSeries

In order to use the AnalogGaugeSeries, the file EJSCart\_Gauges.js must be included in the HTML page after the inclusion of the standard EJSCart.js file.

```
<head>
  <script type="text/javascript" src="/EJSCart/EJSCart.js"></script>
  <script type="text/javascript" src="/EJSCart/EJSCart_Gauges.js"></script>
</head>
```

The AnalogGaugeSeries is rendered as a circular (or semi-circular) gauge with a needle directed towards the current value stored in its [EJSC.GaugePoint](#).

The constructor expects an instantiated [EJSC.DataHandler](#) descendant and an optional set of object properties.

#### Constructor

```
EJSC.AnalogGaugeSeries( EJSC.DataHandler dataHandler [, object options] )
```

#### Example

```
var myAnalogGaugeSeries = new EJSC.AnalogGaugeSeries(
  new EJSC.ArrayDataHandler([[50,"Gauge Value"]]),
  { title: "New Analog Gauge Series" }
);
```

#### Defining Properties and Events

AnalogGaugeSeries properties may be set individually after the series has been created or in batch using the options parameter during instantiation.

##### Setting properties individually

```
var mySeries = new EJSC.AnalogGaugeSeries(new EJSC.ArrayDataHandler([[50,"Gauge Value"]]]);
mySeries.title = "New Analog Gauge Series";
mySeries.min = 0;
mySeries.max = 100;
```

##### Setting properties in batch

```
var mySeries = new EJSC.AnalogGaugeSeries(
  new EJSC.ArrayDataHandler([[50,"Gauge Value"]]),
```

```

        { title: "New Analog Gauge Series", min: 0, max: 100 }
    );

```

### 5.4.1.1 Properties

#### 5.4.1.1.1 anchor

##### Definition

```

object anchor = {
    color:          'rgb(0,0,0)',
    opacity:        100 ,
    size:           10
}

```

##### Description

Defines the parameters for the anchor on the gauge.

##### Properties

string **color** – Defines the color of the anchor.  
integer **opacity** – Defines the opacity of the anchor.  
integer **size** – Defines the diameter (in pixels) of the anchor.

#### 5.4.1.1.2 axis

##### Definition

```

object axis = {
    color:          'rgb(255,255,255)',
    innerBorderColor: 'rgb(0,0,0)',
    innerBorderOpacity: 100 ,
    innerBorderWidth: 1 ,
    innerBorderVisible: true ,
    opacity:        0 ,
    outerBorderColor: 'rgb(0,0,0)',
    outerBorderOpacity: 100 ,
    outerBorderWidth: 1 ,
    outerBorderVisible: true ,
    thickness:      15
}

```

##### Description

Defines the parameters for the axis on the gauge.

##### Properties

string **color** – Defines the background color of the axis.  
string **innerBorderColor** – Defines the color of the inner border.  
integer **innerBorderOpacity** – Defines the opacity of the inner border.

integer	<b>innerBorderWidth</b>	– Defines the width of the inner border.
integer	<b>innerBorderVisible</b>	– Defines whether the inner border is visible or not.
integer	<b>opacity</b>	– Defines the opacity of the background color of the axis.
string	<b>outerBorderColor</b>	– Defines the color of the outer border.
integer	<b>outerBorderOpacity</b>	– Defines the opacity of the outer border.
integer	<b>outerBorderWidth</b>	– Defines the width of the outer border.
boolean	<b>outerBorderVisible</b>	– Defines whether the outer border is visible or not.
integer	<b>thickness</b>	– Defines the thickness of the axis.

#### 5.4.1.1.3 delayLoad (inherited)

### EXPERIMENTAL

#### Definition

boolean **delayLoad** = true

#### Description

This property allows a series to force its data handler to begin data retrieval as soon as it is added to a chart. When set to false, the series will initiate data retrieval as soon as it is added to a chart. When true, the data retrieval is initialized when the series first needs to draw.

#### 5.4.1.1.4 fillColor

#### Definition

string **fillColor** = undefined

#### Description

Defines the background fill color of the gauge. Leave undefined for no fill color. This should be defined as `rgb(<RED>,<GREEN>,<BLUE>)`, i.e. `red = "rgb(255,0,0)"`

#### 5.4.1.1.5 fillOpacity

#### Definition

integer **fillOpacity** = 100

#### Description

Defines the opacity for the background fill color of the gauge.

#### 5.4.1.1.6 height

#### Definition

string **height** = "100%"

#### Description

Defines the total height of the gauge. This may be specified in percent of the chart as shown above as



the default value, or in exact pixels by specifying a number (i.e. height = 150;).

#### 5.4.1.1.7 label

##### Definition

```
object label = {
  className:    "",
  textAlign:   'center',
  position:    'centerBottom',
  lines:       1
}
```

##### Description

Defines the parameters for the label on the gauge.

##### Properties

string **className** – Defines a class (to be defined in an attached stylesheet) to be applied to the label.

string **textAlign** – Defines the alignment of the text in the label ('left', 'center', or 'right').

string **position** – Defines the position on the gauge the label will be placed (see below).

integer **lines** – Defines the number of lines to display in the label. (Text will overflow unless otherwise specified in the attached style class).

Values for position:

- bottom
- centerBottom
- centerLeft
- centerRight
- centerTop
- left
- right
- top

#### 5.4.1.1.8 legendsVisible (inherited)

##### Definition

```
boolean legendsVisible = true
```

##### Description

Determines whether the legend item associated with the series appears in the legend window. Use the Series.showLegend and Series.hideLegend methods to control legend item visibility after series creation.

#### 5.4.1.1.9 lock

##### Definition

```
object lock = {
```

```
    color:      'rgb(0,0,0)',
    offset:     5 ,
    opacity:    100 ,
    size:       6 ,
    visible:    false
}
```

### Description

Defines the parameters for the "locks", or "pegs" on the gauge.

### Properties

string **color** – Defines the color of the locks.

integer **offset** – Defines the distance (in pixels) the needle is allow to extend past the min/max value.

**Set to undefined to turn locks off.**

integer **opacity** – Defines the opacity of the locks.

integer **size** – Defines the diameter (in pixels) of the locks.

boolean **visible** – Defines whether or not to visually show the locks.

#### 5.4.1.1.10 marker\_position

### Definition

```
string marker_position = "outer"
```

### Description

Defines where the tick markers will be positioned with respect to the axis.

#### Quadrants:

- outer
- inner

#### 5.4.1.1.11 max

### Definition

```
float max = 100
```

### Description

Defines the maximum value to be displayed on the gauge.

#### 5.4.1.1.12 min

### Definition

```
float min = 0
```

### Description

Defines the minimum value to be displayed on the gauge.

#### 5.4.1.1.13 minorTick

##### Definition

```
object minorTick = {  
  color: 'rgb(150,150,150)',  
  count: 4 ,  
  offset: 0 ,  
  opacity: 100 ,  
  size: 1 ,  
  thickness: 15  
}
```

##### Description

Defines the parameters for the minor ticks on the gauge.

##### Properties

string <b>color</b>	– Defines the color of the minor tick marks.
integer <b>count</b>	– Defines the number of minor tick marks to display between each major tick mark.
integer <b>offset</b>	– Defines the distance (in pixels) the minor tick marks are pushed out from the inner axis border.
integer <b>opacity</b>	– Defines the opacity of the minor tick marks.
integer <b>size</b>	– Defines the width (in pixels) of the minor tick marks.
integer <b>thickness</b>	– Defines the thickness (in pixels) of the minor tick marks.

#### 5.4.1.1.14 needle

##### Definition

```
object needle = {  
  borderColor: 'rgb(0,0,0)',  
  borderOpacity: 100 ,  
  borderWidth: 1 ,  
  color: 'rgb(0,0,0)',  
  opacity: 100 ,  
  size: 4  
}
```

##### Description

Defines the parameters for the needle on the gauge.

##### Properties

string <b>borderColor</b>	– Defines the color of the border.
integer <b>borderOpacity</b>	– Defines the opacity of the border.

integer	<b>borderWidth</b>	– Defines the width of the border.
string	<b>color</b>	– Defines the background color of the needle.
integer	<b>opacity</b>	– Defines the opacity of the needle.
integer	<b>size</b>	– Defines the width (in pixels) of the needle at its base.

## 5.4.1.1.15 position

**Definition**

string **position** = "center"

**Description**

Defines the quadrant of the chart that the gauge will appear in.

**Quadrants:**

- topLeft
- topCenter
- topRight
- centerLeft
- center
- centerRight
- bottomLeft
- bottomCenter
- bottomRight

## 5.4.1.1.16 range

**Definition**

```
object range = {
  borderColor: 'rgb(0,0,0)',
  borderOpacity: 100,
  borderWidth: 1,
  offset: 0,
  opacity: 100,
  style: 'doughnut',
  thickness: 15
}
```

**Description**

Defines the parameters for the ranges on the gauge.

**Properties**

string	<b>borderColor</b>	– Defines the color of the border.
integer	<b>borderOpacity</b>	– Defines the opacity of the border.
integer	<b>borderWidth</b>	– Defines the width of the border.
integer	<b>offset</b>	– Defines the distance (in pixels) the ranges are pushed in from the axis.
integer	<b>opacity</b>	– Defines the opacity of the ranges.
string	<b>style</b>	– Defines the style to draw the range in ('doughnut' or 'pie')

integer **thickness** – Defines the thickness (in pixels) of the ranges.

#### 5.4.1.1.17 range\_degrees

##### Definition

integer **range\_degrees** = 180

##### Description

Defines the total angle (in degrees) the gauge will take up.

#### 5.4.1.1.18 ranges

##### Definition

array **ranges** = new Array();

##### Description

Defines a series of ranges to be marked in the gauge.

##### Implementation

```
mySeries.ranges = [ [ int min, int max, string color ] , ... ]
```

##### Example

```
mySeries.ranges = [  
    [0,10,'rgb(255,0,0)'],  
    [10,20,'rgb(0,255,0)']  
];
```

#### 5.4.1.1.19 start\_degree

##### Definition

integer **start\_degree** = 270

##### Description

Defines the angle (in degrees) at which the gauges' min value is displayed.

#### 5.4.1.1.20 tick

##### Definition

```
object tick = {  
    className:    "",  
    color:       'rgb(0,0,0)',  
    offset:      0 ,  
    opacity:     100 ,
```

```
        size:          1 ,  
        thickness:    15  
    }
```

### Description

Defines the parameters for the major ticks on the gauge.

### Properties

string <b>className</b>	– Defines a class (to be defined in an attached stylesheet) to be applied to the tick markers.
string <b>color</b>	– Defines the color of the tick marks.
integer <b>offset</b>	– Defines the distance (in pixels) the tick marks are pushed out from the inner axis border.
integer <b>opacity</b>	– Defines the opacity of the tick marks.
integer <b>size</b>	– Defines the width (in pixels) of the tick marks.
integer <b>thickness</b>	– Defines the thickness (in pixels) of the tick marks.

#### 5.4.1.1.21 tickCount

### Definition

```
integer tickCount = 11
```

### Description

Defines the number of major ticks to be displayed on the gauge's axis

#### 5.4.1.1.22 title (inherited)

### Definition

```
string title = "Series <index>"
```

### Description

Defines the series title used in hints and legend display.

**Note:** The default title of Series <index> (where <index> equals the current position in the series array for its owner chart) is assigned when a series is added to a chart using [addSeries](#). This default title is only applied if the title is blank (i.e. "") at the time the addSeries method is called.

#### 5.4.1.1.23 visible (inherited)

### Definition

```
boolean visible = true
```

### Description

Defines whether the series is visible and can draw on the chart

#### 5.4.1.1.24 width

##### Definition

string **width** = "100%"

##### Description

Defines the total width of the gauge. This may be specified in percent of the chart as shown above as the default value, or in exact pixels by specifying a number (i.e. width = 150;).

#### 5.4.1.1.25 x\_axis\_formatter (inherited)

##### Definition

string **x\_axis\_formatter** = undefined

##### Description

Defines the formatter that will be used to format hints for the X values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:

[EJSC.DateFormatter](#)

[EJSC.NumberFormatter](#)

### 5.4.1.2 Methods

#### 5.4.1.2.1 getDataHandler (inherited)

##### Definition

EJSC.DataHandler **getDataHandler**( )

##### Description

Returns the EJSC.DataHandler descendant currently assigned to the series. This is not applicable to all series and will return null if a data handler has not been defined or is not used.

#### 5.4.1.2.2 getVisibility (inherited)

##### Definition

boolean **getVisibility**( )

##### Description

Returns a boolean indicating the series current visible state

## 5.4.1.2.3 hide (inherited)

**Definition**

void **hide**( )

**Description**

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already hidden.

## 5.4.1.2.4 hideLegend (inherited)

**Definition**

void **hideLegend**( )

**Description**

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

## 5.4.1.2.5 reload (inherited)

**Definition**

void **reload**( )

**Description**

Triggers a series reload if the functionality is defined in a child class. This may involve retrieving data again or simply recalculating. While the reload() method exists in the base Series class, implementation of the protected methods triggered by calling reload() is at the discretion of the child class.

## 5.4.1.2.6 setDataHandler (inherited)

**Definition**

void **setDataHandler**( [EJSC.DataHandler](#) dataHandler, boolean reload )

**Description**

Updates the series data handler and triggers a data reload if reload parameter is **true**. This method may not be implemented in all child classes.

## 5.4.1.2.7 setTitle (inherited)

**Definition**



void **setTitle**( string title )

#### **Description**

Changes the series title and update the legend caption (if applicable).

**Note:** this will not cause updates to a hint which is visible at the time it is called, though the next time a hint is rendered it will show the new property value.

5.4.1.2.8 show (inherited)

#### **Definition**

void **show**( )

#### **Description**

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already visible.

5.4.1.2.9 showLegend (inherited)

#### **Definition**

void **showLegend**( )

#### **Description**

Changes the series legend item visible state.

No effect if the series legend item is already visible.

### **5.4.1.3 Events**

5.4.1.3.1 onAfterDataAvailable (inherited)

#### **Definition**

boolean **onAfterDataAvailable**( [EJSC.Chart](#) chart, [EJSC.Series](#) series )

#### **Description**

Fired after the series has processed its data and before the chart is told to redraw. Returning **false** from this event handler will cancel redrawing the chart.

5.4.1.3.2 onAfterVisibilityChange (inherited)

#### **Definition**

boolean **onAfterVisibilityChange**( [EJSC.Series](#) series, [EJSC.Chart](#) chart, boolean visible )

**Description**

Fired when the series visibility changes. Sends series which changed, its owner chart and its new visible property value.

Returning **false** from this event handler will cancel redrawing the chart.

## 5.4.1.3.3 onBeforeVisibilityChange (inherited)

**Definition**

boolean **onBeforeVisibilityChange**( [EJSC.Series](#) series, [EJSC.Chart](#) chart )

**Description**

Fired before the series visibility is about to change. Sends series which changed, its owner chart. The current visible state of the series may be checked using the series.visible property.

Returning **false** from this event handler will cancel the change in visibility.

5.4.1.4 **Data Formats**

The x parameter is required, label and userdata are both optional.

**XML** ([EJSC.XMLDataHandler](#), [EJSC.XMLStringDataHandler](#)):

Unused parameters may be omitted

**Full:**

```
<graph>
  <plot>
    <point x="number or string" label="string" userdata="string"/>
  </plot>
</graph>
```

**Short:**

```
<G>
  <L>
    <P x="number or string" label="string" userdata="string"/>
  </L>
</G>
```

**Compact:** The values parameter is formatted as CSV

```
<G>
  <L values="CSV string"/>
</G>
```

**Array** ([EJSC.ArrayDataHandler](#)):

```
[
    ["x", "label", "userdata"]
]
```

Null should be used to skip unused parameters:

```
[
    ["x", null, "userdata"]
]
```

**CSV** ([EJSC.CSVFileDataHandler](#), [EJSC.CSVStringDataHandler](#)):

```
"x|label|userdata"
```

Null should be used to skip unused parameters

```
"x|null|userdata"
```

**JSON** ([EJSC.JSONFileDataHandler](#), [EJSC.JSONStringDataHandler](#)):

Unused parameters may be omitted.

```
[{"x":"number or string","label":"string","userdata":"string"}]
```

## 5.4.2 EJSC.AreaSeries

AreaSeries is rendered by drawing a line from point to point and then filling the area defined.

The constructor expects an instantiated [EJSC.DataHandler](#) descendant and an optional set of object properties.

### Constructor

```
EJSC.AreaSeries( EJSC.DataHandler dataHandler [, object options] )
```

### Example

```
var mySeries = new EJSC.AreaSeries(
    new EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,3],[5,2],[6,4]]),
    { title: "New Area Series" }
);
```

### Defining Properties and Events

AreaSeries properties may be set individually after the series has been created or in batch using the options parameter during instantiation.

#### Setting properties individually

```
var mySeries = new EJSC.AreaSeries(new
EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,3],[5,2],[6,4]]));
mySeries.lineWidth = 1;
mySeries.title = "New Area Series";
```

### Setting properties in batch

```
var mySeries = new EJSC.AreaSeries(
    new EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,3],[5,2],[6,4]]),
    { lineWidth: 1, title: "New Area Series" }
);
```

## 5.4.2.1 Properties

### 5.4.2.1.1 autosort (inherited)

#### Definition

boolean **autosort** = true

#### Description

Defines whether the series applies the appropriate sort to points prior to drawing. Drawing routines are generally optimized to accept data in a certain order and this property eliminates the need for the developer to worry about that order prior to sending data to the series. If set to false, the data must be properly ordered beforehand in order to ensure the series renders correctly.

**Note:** May not be applicable to all series (i.e. [EJSC.PieSeries](#), [EJSC.AnalogGaugeSeries](#))

### 5.4.2.1.2 closeLine

#### Definition

boolean **closeLine** = true

#### Description

Defines whether the line drawn around the area returns to the zero plane to create a complete shape (true) or if it drawn only through the points in the data set (false).

### 5.4.2.1.3 color (inherited)

#### Definition

string **color** = undefined

#### Description

Defines the series color. If left undefined, the color is pulled from the array of default colors stored in the chart.

**Note:** May not be applicable to all series (i.e. [EJSC.PieSeries](#))

#### 5.4.2.1.4 coloredLegend (inherited)

##### Definition

boolean **coloredLegend** = true

##### Description

Determines whether the legend text inherits the series color. Setting to false will allow the legend text to inherit its normal cascaded color. To modify this property after series creation see [EJSC.Series.setColoredLegend\(\)](#)

#### 5.4.2.1.5 delayLoad (inherited)

##### EXPERIMENTAL

##### Definition

boolean **delayLoad** = true

##### Description

This property allows a series to force its data handler to begin data retrieval as soon as it is added to a chart. When set to false, the series will initiate data retrieval as soon as it is added to a chart. When true, the data retrieval is initialized when the series first needs to draw.

#### 5.4.2.1.6 drawPoints (inherited)

##### Definition

boolean **drawPoints** = false

##### Description

Determines whether the individual points in the series are visually indicated by round dots on the chart. The size and color of the fill and border are determined by the [pointColor](#), [pointSize](#), [pointBorderColor](#) and [pointBorderSize](#) properties.

**Note:** Setting this property to true for series which have a large number of points may impact performance as additional draws are required to render the points.

#### 5.4.2.1.7 hint\_string (inherited)

##### Definition

string **hint\_string** = undefined

##### Description

This property may be used to define a series specific string to be used when displaying point hints. This string may contain replacement indicators (see [Text Replacement Options](#)). When left undefined, a default hint defined by the series type will be displayed.

**Example**

>> *Modify the default hint to include custom text.*

```
var series = new EJSC.LineSeries(new EJSC.ArrayDataHandler(),
    {
        hint_string: "My Custom Hint<br /><strong>X: </strong>[x]<br/><strong>Y:
</strong>[y]"
    }
);
```

## 5.4.2.1.8 legendsVisible (inherited)

**Definition**

boolean **legendsVisible** = true

**Description**

Determines whether the legend item associated with the series appears in the legend window. Use the `Series.showLegend` and `Series.hideLegend` methods to control legend item visibility after series creation.

## 5.4.2.1.9 lineOpacity (inherited)

**Definition**

integer **lineOpacity** = 100

**Description**

Determines the line opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see [EJSC.Series.setLineOpacity\(\)](#)

## 5.4.2.1.10 lineWidth (inherited)

**Definition**

integer **lineWidth** = 1

**Description**

Defines the width of the line connecting series points.

## 5.4.2.1.11 opacity (inherited)

**Definition**

integer **opacity** = 50

**Description**

Determines the fill opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see [EJSC.Series.setOpacity\(\)](#)

#### 5.4.2.1.12 pointBorderColor (inherited)

##### Definition

string **pointBorderColor** = "rgb(255,255,255)"

##### Description

Defines the color of the border drawn around the points. This property is only applicable when the series [drawPoints](#) property is set to true and [pointBorderSize](#) is greater than 0.

#### 5.4.2.1.13 pointBorderSize (inherited)

##### Definition

integer **pointBorderSize** = 0

##### Description

Defines the size in pixels of the border drawn around the points. This property is only applicable when the series [drawPoints](#) property is set to true.

To draw points without any border, leave pointBorderSize set to 0.

#### 5.4.2.1.14 pointColor (inherited)

##### Definition

string **pointColor** = undefined

##### Description

Defines the color of the point (circle) drawn at each data point in the series. If left undefined the point will inherit the series color.

This property is only applicable when the series [drawPoints](#) property is set to true.

#### 5.4.2.1.15 pointSize (inherited)

##### Definition

integer **pointSize** = undefined

##### Description

Defines the size of the point (circle) drawn at each data point in the series. If left undefined the point diameter will be twice the line width of the series.

This property is only applicable when the series [drawPoints](#) property is set to true.

#### 5.4.2.1.16 title (inherited)

##### Definition

string **title** = "Series <index>"

##### Description

Defines the series title used in hints and legend display.

**Note:** The default title of Series <index> (where <index> equals the current position in the series array for its owner chart) is assigned when a series is added to a chart using [addSeries](#). This default title is only applied if the title is blank (i.e. "") at the time the addSeries method is called.

#### 5.4.2.1.17 visible (inherited)

##### Definition

boolean **visible** = true

##### Description

Defines whether the series is visible and can draw on the chart

#### 5.4.2.1.18 x\_axis (inherited)

##### Definition

string **x\_axis** = "bottom"

##### Description

Defines the horizontal axis to use for X values. Valid options are "top" or "bottom".

#### 5.4.2.1.19 x\_axis\_formatter (inherited)

##### Definition

string **x\_axis\_formatter** = undefined

##### Description

Defines the formatter that will be used to format hints for the X values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:

[EJSC.DateFormatter](#)

[EJSC.NumberFormatter](#)



## 5.4.2.1.20 y\_axis

**Definition**

string **y\_axis** = "left"

**Description**

Defines the vertical axis to use for Y values. Valid options are "left" or "right".

## 5.4.2.1.21 y\_axis\_formatter (inherited)

**Definition**

string **y\_axis\_formatter** = undefined

**Description**

Defines the formatter that will be used to format hints for the Y values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:

[EJSC.DateFormatter](#)

[EJSC.NumberFormatter](#)

**5.4.2.2 Methods**

## 5.4.2.2.1 getDataHandler (inherited)

**Definition**

EJSC.DataHandler **getDataHandler**( )

**Description**

Returns the EJSC.DataHandler descendant currently assigned to the series. This is not applicable to all series and will return null if a data handler has not been defined or is not used.

## 5.4.2.2.2 getVisibility (inherited)

**Definition**

boolean **getVisibility**( )

**Description**

Returns a boolean indicating the series current visible state

## 5.4.2.2.3 hide (inherited)

**Definition**

void **hide**( )

**Description**

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already hidden.

5.4.2.2.4 **hideLegend** (inherited)

**Definition**

void **hideLegend**( )

**Description**

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

5.4.2.2.5 **reload** (inherited)

**Definition**

void **reload**( )

**Description**

Triggers a series reload if the functionality is defined in a child class. This may involve retrieving data again or simply recalculating. While the `reload()` method exists in the base `Series` class, implementation of the protected methods triggered by calling `reload()` is at the discretion of the child class.

5.4.2.2.6 **setColor** (inherited)

**Definition**

void **setColor**( string color )

**Description**

Changes the series color and causes the chart to redraw.

5.4.2.2.7 **setColoredLegend** (inherited)

**Definition**

void **setColoredLegend**( boolean coloredLegend )

**Description**

Sets the [EJSC.Series.coloredLegend](#) property and updates the legend to reflect the change.

## 5.4.2.2.8 setDataHandler (inherited)

**Definition**

void **setDataHandler**( [EJSC.DataHandler](#) dataHandler, boolean reload )

**Description**

Updates the series data handler and triggers a data reload if reload parameter is **true**. This method may not be implemented in all child classes.

## 5.4.2.2.9 setLineWidth (inherited)

**Definition**

void **setLineWidth**( integer width )

**Description**

Updates the [lineWidth](#) property and redraws the chart.

## 5.4.2.2.10 setOpacity (inherited)

**Definition**

void **setOpacity**( integer opacity )

**Description**

Sets the [EJSC.Series.opacity](#) property and redraws the series to reflect the change.

## 5.4.2.2.11 setTitle (inherited)

**Definition**

void **setTitle**( string title )

**Description**

Changes the series title and update the legend caption (if applicable).

**Note:** this will not cause updates to a hint which is visible at the time it is called, though the next time a hint is rendered it will show the new property value.

## 5.4.2.2.12 show (inherited)

**Definition**

void **show**( )

**Description**

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already visible.

#### 5.4.2.2.13 showLegend (inherited)

##### Definition

void [showLegend](#)( )

##### Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

#### 5.4.2.3 Events

##### 5.4.2.3.1 onAfterDataAvailable (inherited)

##### Definition

boolean [onAfterDataAvailable](#)( [EJSC.Chart](#) chart, [EJSC.Series](#) series )

##### Description

Fired after the series has processed its data and before the chart is told to redraw. Returning **false** from this event handler will cancel redrawing the chart.

##### 5.4.2.3.2 onAfterVisibilityChange (inherited)

##### Definition

boolean [onAfterVisibilityChange](#)( [EJSC.Series](#) series, [EJSC.Chart](#) chart, boolean visible )

##### Description

Fired when the series visibility changes. Sends series which changed, its owner chart and its new visible property value.

Returning **false** from this event handler will cancel redrawing the chart.

##### 5.4.2.3.3 onBeforeVisibilityChange (inherited)

##### Definition

boolean [onBeforeVisibilityChange](#)( [EJSC.Series](#) series, [EJSC.Chart](#) chart )

##### Description

Fired before the series visibility is about to change. Sends series which changed, its owner chart. The

current visible state of the series may be checked using the series.visible property.

Returning **false** from this event handler will cancel the change in visibility.

#### 5.4.2.3.4 onShowHint (inherited)

##### Definition

string **onShowHint**( [EJSC.Point](#) point, [EJSC.Series](#) series, [EJSC.Chart](#) chart, string hint\_string )

##### Description

Fired before the displaying the hint, this event allows the current hint string to be overridden. See [Text Replacement Options](#) for a list of the automatic substitutions available.

#### 5.4.2.4 Data Formats

The x and y parameters are required, label and userdata are both optional.

**XML** ([EJSC.XMLDataHandler](#), [EJSC.XMLStringDataHandler](#)):

Unused parameters may be omitted

##### Full:

```

    <graph>
      <plot>
        <point x="number or string" y="number or string" label="string"
userdata="string"/>
        <point x="number or string" y="number or string" label="string"
userdata="string"/>
      </plot>
    </graph>

```

##### Short:

```

    <G>
      <L>
        <P x="number or string" y="number or string" label="string"
userdata="string"/>
        <P x="number or string" y="number or string" label="string"
userdata="string"/>
      </L>
    </G>

```

**Compact:** The values parameter is formatted as CSV

```

    <G>
      <L values="CSV string"/>
    </G>

```

**Array** ([EJSC.ArrayDataHandler](#)):

```
[
  ["x", "y", "label", "userdata"],
  ["x", "y", "label", "userdata"]
]
```

Null should be used to skip unused parameters:

```
[
  ["x", "y", null, "userdata"],
  ["x", "y", null, "userdata"]
]
```

**CSV** ([EJSC.CSVFileDataHandler](#), [EJSC.CSVStringDataHandler](#)):

```
"x|y|label|userdata,x|y|label|userdata"
```

Null should be used to skip unused parameters

```
"x|y|null|userdata,x|y|null|userdata"
```

**JSON** ([EJSC.JSONFileDataHandler](#), [EJSC.JSONStringDataHandler](#)):

Unused parameters may be omitted.

```
[
  { "x": "number or string", "y": "number or
string", "label": "string", "userdata": "string" },
  { "x": "number or string", "y": "number or
string", "label": "string", "userdata": "string" }
]
```

#### 5.4.2.5 Text Replacement Options

String	Replaced With
[chart_title]	The title of the chart
[series_title]	The title of the series the selected point resides in
[xaxis]	The caption of the series x axis
[yaxis]	The caption of the series y axis
[x]	The preformatted X value of the point
[y]	The preformatted Y value of the point
[label]	The label property of the current point

#### 5.4.3 EJSC.BarSeries

BarSeries renders its points as vertical or horizontal bars which are fixed to a baseline.

The constructor expects an instantiated [EJSC.DataHandler](#) descendant and an optional set of object properties.

## Constructor

```
EJSC.BarSeries( EJSC.DataHandler dataHandler [, object options] )
```

## Example

```
var mySeries = new EJSC.BarSeries(
  new EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,3],[5,2],[6,4]]),
  { title: "New Bar Series" }
);
```

## Defining Properties and Events

BarSeries properties may be set individually after the series has been created or in batch using the options parameter during instantiation.

### Setting properties individually

```
var mySeries = new EJSC.BarSeries(new
EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,3],[5,2],[6,4]]));
mySeries.lineWidth = 1;
mySeries.title = "New Bar Series";
```

### Setting properties in batch

```
var mySeries = new EJSC.BarSeries(
  new EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,3],[5,2],[6,4]]),
  { lineWidth: 1, title: "New Bar Series" }
);
```

## 5.4.3.1 Properties

### 5.4.3.1.1 autosort (inherited)

#### Definition

boolean **autosort** = true

#### Description

Defines whether the series applies the appropriate sort to points prior to drawing. Drawing routines are generally optimized to accept data in a certain order and this property eliminates the need for the developer to worry about that order prior to sending data to the series. If set to false, the data must be properly ordered beforehand in order to ensure the series renders correctly.

**Note:** May not be applicable to all series (i.e. [EJSC.PieSeries](#), [EJSC.AnalogGaugeSeries](#))

### 5.4.3.1.2 color (inherited)

#### Definition

string **color** = undefined

#### Description

Defines the series color. If left undefined, the color is pulled from the array of default colors stored in the chart.

**Note:** May not be applicable to all series (i.e. [EJSC.PieSeries](#))

#### 5.4.3.1.3 coloredLegend (inherited)

##### Definition

boolean **coloredLegend** = true

##### Description

Determines whether the legend text inherits the series color. Setting to false will allow the legend text to inherit its normal cascaded color. To modify this property after series creation see [EJSC.Series.setColoredLegend\(\)](#)

#### 5.4.3.1.4 defaultColors

##### Definition

array **defaultColors** = [EJSC.DefaultBarColors](#)

##### Description

Defines the pool of default colors available for bar series bars.

**NOTE:** To make use of individually colored bars, the [EJSC.BarSeries.useColorArray](#) property must be set to true.

#### 5.4.3.1.5 delayLoad (inherited)

#### **EXPERIMENTAL**

##### Definition

boolean **delayLoad** = true

##### Description

This property allows a series to force its data handler to begin data retrieval as soon as it is added to a chart. When set to false, the series will initiate data retrieval as soon as it is added to a chart. When true, the data retrieval is initialized when the series first needs to draw.

#### 5.4.3.1.6 groupedBars

##### Definition

boolean **groupedBars** = true



## Description

Defines whether the bars from different bar series in a single chart are grouped or overlaid.

**NOTE:** Only the first bar series added to the chart may use this property to affect the grouped/overlay display. To change the style after the first bar series has been added, use the [EJSC.BarSeries.setGroupedBars](#) from any bar series in the chart. Currently you cannot mix both overlaid and grouped bars within the same chart.

## Example

>> *Make the bar series overlay*

```
var chart = new EJSC.Chart("chart");
var bar1 = chart.addSeries(new EJSC.BarSeries(
    data,
    {
        groupedBars: false,
    }
));
var bar2 = chart.addSeries(new EJSC.BarSeries(data));
```

### 5.4.3.1.7 hint\_string (inherited)

## Definition

string [hint\\_string](#) = undefined

## Description

This property may be used to define a series specific string to be used when displaying point hints. This string may contain replacement indicators (see [Text Replacement Options](#)). When left undefined, a default hint defined by the series type will be displayed.

## Example

>> *Modify the default hint to include custom text.*

```
var series = new EJSC.LineSeries(new EJSC.ArrayDataHandler(),
    {
        hint_string: "My Custom Hint<br /><strong>X: </strong>[x]<br/><strong>Y:
</strong>[y]"
    }
);
```

### 5.4.3.1.8 intervalOffset

## Definition

float [intervalOffset](#) = 0.8

## Description

Defines the spacing between the bars as a percentage of 1. 1 = bars take up the entire width available, with their sides touching.

### Example

>> *Make the bars take up 1/2 their available width. (i.e. more space between bars than by default)*

```
var chart = new EJSC.Chart("chart");
var bar = chart.addSeries(new EJSC.BarSeries(
    data,
    {
        intervalOffset: 0.5
    }
));
```

#### 5.4.3.1.9 legendsVisible (inherited)

##### Definition

boolean **legendsVisible** = true

##### Description

Determines whether the legend item associated with the series appears in the legend window. Use the `Series.showLegend` and `Series.hideLegend` methods to control legend item visibility after series creation.

#### 5.4.3.1.10 lineOpacity (inherited)

##### Definition

integer **lineOpacity** = 100

##### Description

Determines the line opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see [EJSC.Series.setLineOpacity\(\)](#)

#### 5.4.3.1.11 lineWidth (inherited)

##### Definition

integer **lineWidth** = 1

##### Description

Defines the width of the line connecting series points.

#### 5.4.3.1.12 opacity (inherited)

##### Definition

integer **opacity** = 50

### Description

Determines the fill opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see [EJSC.Series.setOpacity\(\)](#)

#### 5.4.3.1.13 orientation

### Definition

string **orientation** = "vertical"

### Description

This property determines the orientation of the bar series. The supported property values are "vertical" and "horizontal".

**Note:** Currently this is a creation-time only property and should not be changed after the series has been created.

#### 5.4.3.1.14 ranges

### Definition

array **ranges** = new Array()

### Description

This property is used to store the ranges for a given bar series. The range objects are used to draw bars in varying styles based on their Y value.

The range objects stored are defined as:

```
range = {
  min:          float, // >=
  max:          float, // <
  color:        string, // Defined as rgb(RED, GREEN, BLUE), ex: "rgb(255,0,0)"
  opacity:      integer, // Represents percent, i.e. 50 = 50%
  lineOpacity:  integer, // Represents percent, i.e. 50 = 50%
  lineWidth:   integer // Represents the line width in pixels
}
```

### Example

*>> Define ranges so that bars with a Y value from 0 to 10 and 90 to 100 are colored red, bars 10 - 90 are colored green*

```
var chart = new EJSC.Chart("chart");
var bar = chart.addSeries(new EJSC.BarSeries(
  data,
```

```

    {
      ranges: [
        { 0, 10, "rgb(255,0,0)", 100, 100, 1 },
        { 90, 100, "rgb(255,0,0)", 100, 100, 1 },
        { 10, 90, "rgb(0,255,0)", 100, 100, 1 }
      ]
    }
  });

```

#### 5.4.3.1.15 title (inherited)

##### Definition

string **title** = "Series <index>"

##### Description

Defines the series title used in hints and legend display.

**Note:** The default title of Series <index> (where <index> equals the current position in the series array for its owner chart) is assigned when a series is added to a chart using [addSeries](#). This default title is only applied if the title is blank (i.e. "") at the time the addSeries method is called.

#### 5.4.3.1.16 treeLegend

##### Definition

boolean **treeLegend** = false

##### Description

Defines whether to display each bar individually in the legend.

If the [useColorArray](#) property is true and treeLegend left at its default (not defined in options), the treeLegend property will automatically be set to true to enable tree-style legend display.

#### 5.4.3.1.17 useColorArray

##### Definition

boolean **useColorArray** = false

##### Description

Defines whether to make use of the [EJSC.BarSeries.defaultColors](#) property

##### Example

*>> Use the color array to make a chart of multi colored bars*

```

var chart = new EJSC.Chart("chart");
var bar = chart.addSeries(new EJSC.BarSeries(
  data,
  {

```

```
        useColorArray: true
    }
  });
```

#### 5.4.3.1.18 visible (inherited)

##### Definition

boolean **visible** = true

##### Description

Defines whether the series is visible and can draw on the chart

#### 5.4.3.1.19 x\_axis (inherited)

##### Definition

string **x\_axis** = "bottom"

##### Description

Defines the horizontal axis to use for X values. Valid options are "top" or "bottom".

#### 5.4.3.1.20 x\_axis\_formatter (inherited)

##### Definition

string **x\_axis\_formatter** = undefined

##### Description

Defines the formatter that will be used to format hints for the X values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:

[EJSC.DateFormatter](#)

[EJSC.NumberFormatter](#)

#### 5.4.3.1.21 y\_axis (inherited)

##### Definition

string **y\_axis** = "left"

##### Description

Defines the vertical axis to use for Y values. Valid options are "left" or "right".

#### 5.4.3.1.22 y\_axis\_formatter (inherited)

##### Definition

string **y\_axis\_formatter** = undefined

### Description

Defines the formatter that will be used to format hints for the Y values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:

[EJSC.DateFormatter](#)

[EJSC.NumberFormatter](#)

## 5.4.3.2 Methods

### 5.4.3.2.1 addRange

#### Definition

void **addRange**( float min, float max, string color, integer opacity, integer lineOpacity, integer lineWidth, boolean redraw )

#### Description

Adds a new range to the bar series and optionally triggers a redraw.

### 5.4.3.2.2 clearRanges

#### Definition

void **clearRanges**( boolean redraw )

#### Description

Clears all ranges defined for the series and optionally redraws the the chart.

### 5.4.3.2.3 deleteRange

#### Definition

void **deleteRange**( float min, float max, boolean redraw )

#### Description

Deletes the matching range (min and max must exactly match an existing range) and optionally redraws the chart.

### 5.4.3.2.4 getBarSize

#### Definition

integer **getBarSize**()

**Description**

This method returns the size in pixels (width or height depending on orientation) of a single bar.

5.4.3.2.5 `getDataHandler` (inherited)**Definition**

EJSC.DataHandler [getDataHandler](#)( )

**Description**

Returns the EJSC.DataHandler descendant currently assigned to the series. This is not applicable to all series and will return null if a data handler has not been defined or is not used.

5.4.3.2.6 `getPoints`**Definition**

array [getPoints](#)()

**Description**

This method returns an array of all [EJSC.BarPoint](#) objects in the series.

5.4.3.2.7 `getVisibility` (inherited)**Definition**

boolean [getVisibility](#)( )

**Description**

Returns a boolean indicating the series current visible state

5.4.3.2.8 `hide` (inherited)**Definition**

void [hide](#)( )

**Description**

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already hidden.

5.4.3.2.9 `hideLegend` (inherited)**Definition**

void [hideLegend](#)( )

**Description**

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

5.4.3.2.10 reload (inherited)

**Definition**

void **reload**( )

**Description**

Triggers a series reload if the functionality is defined in a child class. This may involve retrieving data again or simply recalculating. While the reload() method exists in the base Series class, implementation of the protected methods triggered by calling reload() is at the discretion of the child class.

5.4.3.2.11 setColor (inherited)

**Definition**

void **setColor**( string color )

**Description**

Changes the series color and causes the chart to redraw.

5.4.3.2.12 setColoredLegend (inherited)

**Definition**

void **setColoredLegend**( boolean coloredLegend )

**Description**

Sets the [EJSC.Series.coloredLegend](#) property and updates the legend to reflect the change.

5.4.3.2.13 setDataHandler (inherited)

**Definition**

void **setDataHandler**( [EJSC.DataHandler](#) dataHandler, boolean reload )

**Description**

Updates the series data handler and triggers a data reload if reload parameter is **true**. This method may not be implemented in all child classes.



## 5.4.3.2.14 setDefaultColors

**Definition**

void [setDefaultColors](#)( array colors, boolean reload )

**Description**

Set the array of default colors available for bar series bars.

If reload = **true** the bars (if loaded) will pull their colors from the given color array and the chart will be redrawn.

If [onBarNeedsColor](#) is assigned, this method will ignore that event and load the colors from the array.

**Example**

```
var colors = [  
    "rgb('0,0,0')",           // black  
    "rgb(255,0,0)",         // red  
    "rgb('0,255,0')",       // green  
    "rgb('0,0,255')",       // blue  
    "rgb('255,255,255')"    // white  
];  
  
myBarSeries.setDefaultColors(colors, true);
```

## 5.4.3.2.15 setGroupedBars

**Definition**

void [setGroupedBars](#)( boolean grouped, boolean redraw )

**Description**

Changes the chart between grouped and overlaid bars and optionally triggers a redraw.

## 5.4.3.2.16 setIntervalOffset

**Definition**

void [setIntervalOffset](#)( float offset, boolean redraw )

**Description**

Updates the interval offset property (spacing between the bars) and optionally redraws the chart. See [EJSC.BarSeries.intervalOffset](#) for additional information.

## 5.4.3.2.17 setLineWidth (inherited)

**Definition**

void **setLineWidth**( integer width )

**Description**

Updates the [lineWidth](#) property and redraws the chart.

5.4.3.2.18 setOpacity (inherited)

**Definition**

void **setOpacity**( integer opacity )

**Description**

Sets the [EJSC.Series.opacity](#) property and redraws the series to reflect the change.

5.4.3.2.19 setTitle (inherited)

**Definition**

void **setTitle**( string title )

**Description**

Changes the series title and update the legend caption (if applicable).

**Note:** this will not cause updates to a hint which is visible at the time it is called, though the next time a hint is rendered it will show the new property value.

5.4.3.2.20 show (inherited)

**Definition**

void **show**( )

**Description**

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already visible.

5.4.3.2.21 showLegend (inherited)

**Definition**

void **showLegend**( )

**Description**

Changes the series legend item visible state.

No effect if the series legend item is already visible.

### 5.4.3.3 Events

#### 5.4.3.3.1 onAfterDataAvailable (inherited)

##### Definition

boolean **onAfterDataAvailable**( [EJSC.Chart](#) chart, [EJSC.Series](#) series )

##### Description

Fired after the series has processed its data and before the chart is told to redraw. Returning **false** from this event handler will cancel redrawing the chart.

#### 5.4.3.3.2 onAfterVisibilityChange (inherited)

##### Definition

boolean **onAfterVisibilityChange**( [EJSC.Series](#) series, [EJSC.Chart](#) chart, boolean visible )

##### Description

Fired when the series visibility changes. Sends series which changed, its owner chart and its new visible property value.

Returning **false** from this event handler will cancel redrawing the chart.

#### 5.4.3.3.3 onBarNeedsColor

##### Definition

string **onBarNeedsColor**( [EJSC.BarPoint](#) point, [EJSC.Series](#) series, [EJSC.Chart](#) chart )  
 object **onBarNeedsColor**( [EJSC.BarPoint](#) point, [EJSC.Series](#) series, [EJSC.Chart](#) chart )

##### Description

Fired when a bar needs a color. This event may return either a string (i.e. "rgb(0,255,0)") or an object with additional styling properties.

The object returned is expected to be formatted as:

```
{
    color:      string,
    opacity:   integer,
    lineOpacity: integer,
    lineWidth: integer
}
```

##### Example

*>> Display bars with userdata of "bold" with thicker lines*

```
var chart = new EJSC.Chart("chart");
```

```

var bar = chart.addSeries(new EJSC.BarSeries(
    data,
    {
        lineOpacity: 80,
        lineWidth: 1,
        onBarNeedsColor: function(point, series, chart) {

            if (point.userdata == "bold") {
                return {
                    color: "rgb(0,0,0)",
                    opacity: 100,
                    lineOpacity: 100,
                    lineWidth: 100
                };
            } else {
                return "rgb(128,128,128)";
            }
        }
    }
));

```

#### 5.4.3.3.4 onBeforeVisibilityChange (inherited)

##### Definition

boolean **onBeforeVisibilityChange**( [EJSC.Series](#) series, [EJSC.Chart](#) chart )

##### Description

Fired before the series visibility is about to change. Sends series which changed, its owner chart. The current visible state of the series may be checked using the series.visible property.

Returning **false** from this event handler will cancel the change in visibility.

#### 5.4.3.3.5 onShowHint (inherited)

##### Definition

string **onShowHint**( [EJSC.Point](#) point, [EJSC.Series](#) series, [EJSC.Chart](#) chart, string hint\_string )

##### Description

Fired before the displaying the hint, this event allows the current hint string to be overridden. See [Text Replacement Options](#) for a list of the automatic substitutions available.

#### 5.4.3.4 Data Formats

The x and y parameters are required, label and userdata are both optional.

**XML** ([EJSC.XMLDataHandler](#), [EJSC.XMLStringDataHandler](#)):

Unused parameters may be omitted

**Full:**

**<graph>**

```

        <plot>
            <point x="number or string" y="number or string" label="string"
userdata="string"/>
            <point x="number or string" y="number or string" label="string"
userdata="string"/>
        </plot>
    </graph>

```

Short:

```

    <G>
        <L>
            <P x="number or string" y="number or string" label="string"
userdata="string"/>
            <P x="number or string" y="number or string" label="string"
userdata="string"/>
        </L>
    </G>

```

**Compact:** The values parameter is formatted as CSV

```

    <G>
        <L values="CSV string"/>
    </G>

```

**Array** ([EJSC.ArrayDataHandler](#)):

```

[
    ["x", "y", "label", "userdata"],
    ["x", "y", "label", "userdata"]
]

```

Null should be used to skip unused parameters:

```

[
    ["x", "y", null, "userdata"],
    ["x", "y", null, "userdata"]
]

```

**CSV** ([EJSC.CSVFileDataHandler](#), [EJSC.CSVStringDataHandler](#)):

```
"x|y|label|userdata,x|y|label|userdata"
```

Null should be used to skip unused parameters

```
"x|y|null|userdata,x|y|null|userdata"
```

**JSON** ([EJSC.JSONFileDataHandler](#), [EJSC.JSONStringDataHandler](#)):

Unused parameters may be omitted.

```
[
  { "x": "number or string", "y": "number or
string", "label": "string", "userdata": "string" },
  { "x": "number or string", "y": "number or
string", "label": "string", "userdata": "string" }
]
```

#### 5.4.3.5 Text Replacement Options

String	Replaced With
[chart_title]	The title of the chart
[series_title]	The title of the series the selected point resides in
[xaxis]	The caption of the series x axis
[yaxis]	The caption of the series y axis
[x]	The preformatted X value of the point
[y]	The preformatted Y value of the point
[label]	The label property of the current point

#### 5.4.4 EJSC.CandlestickSeries

In order to use the CandlestickSeries, the file EJSCart\_Stock.js must be included in the HTML page after the inclusion of the standard EJSCart.js file.

```
<head>
  <script type="text/javascript" src="/EJSCart/EJSCart.js"></script>
  <script type="text/javascript" src="/EJSCart/EJSCart_Stock.js"></script>
</head>
```

CandlestickSeries renders its point data as vertical boxes with optional "wicks" to express open and close values.

The constructor expects an instantiated [EJSC.DataHandler](#) descendant and an optional set of object properties.

##### Constructor

```
EJSC.CandlestickSeries( EJSC.DataHandler dataHandler [, object options] )
```

##### Example

```
var mySeries = new EJSC.CandlestickSeries(
  new EJSC.ArrayDataHandler([[1,10,1,2,9],[2,20,7,10,19]]),
  { title: "New Candlestick Series" }
);
```

##### Defining Properties and Events

CandlestickSeries properties may be set individually after the series has been created or in batch using

the options parameter during instantiation.

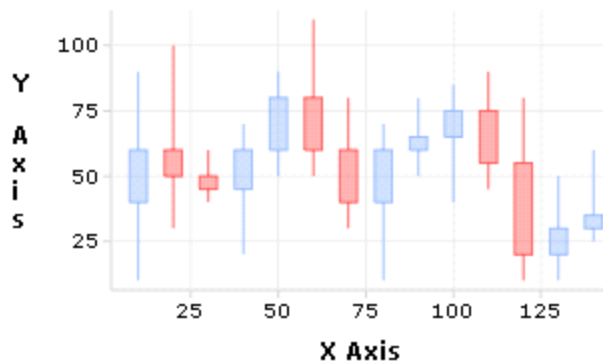
### Setting properties individually

```
var mySeries = new EJSC.CandlestickSeries(new
EJSC.ArrayDataHandler([[1,10,1,2,9],[2,20,7,10,19]]));
mySeries.lineWidth = 1;
mySeries.title = "New Candlestick Series";
```

### Setting properties in batch

```
var mySeries = new EJSC.CandlestickSeries(
    new EJSC.ArrayDataHandler([[1,10,1,2,9],[2,20,7,10,9]]),
    { lineWidth: 1, title: "New Candlestick Series" }
);
```

## Emprise JavaScript Charts



### 5.4.4.1 Properties

#### 5.4.4.1.1 autosort (inherited)

##### Definition

boolean **autosort** = true

##### Description

Defines whether the series applies the appropriate sort to points prior to drawing. Drawing routines are generally optimized to accept data in a certain order and this property eliminates the need for the developer to worry about that order prior to sending data to the series. If set to false, the data must be properly ordered beforehand in order to ensure the series renders correctly.

**Note:** May not be applicable to all series (i.e. [EJSC.PieSeries](#), [EJSC.AnalogGaugeSeries](#))

#### 5.4.4.1.2 color (inherited)

##### Definition

string **color** = undefined

**Description**

Defines the series color. If left undefined, the color is pulled from the array of default colors stored in the chart.

**Note:** May not be applicable to all series (i.e. [EJSC.PieSeries](#))

## 5.4.4.1.3 coloredLegend (inherited)

**Definition**

boolean **coloredLegend** = true

**Description**

Determines whether the legend text inherits the series color. Setting to false will allow the legend text to inherit its normal cascaded color. To modify this property after series creation see [EJSC.Series.setColoredLegend\(\)](#)

## 5.4.4.1.4 delayLoad (inherited)

**EXPERIMENTAL****Definition**

boolean **delayLoad** = true

**Description**

This property allows a series to force its data handler to begin data retrieval as soon as it is added to a chart. When set to false, the series will initiate data retrieval as soon as it is added to a chart. When true, the data retrieval is initialized when the series first needs to draw.

## 5.4.4.1.5 gain

**Definition**

```
object gain = {  
    lineColor: "rgb(151,183,247)",  
    lineOpacity: 100,  
    color: "rgb(151,183,247)",  
    opacity: 50  
}
```

**Description**

Defines the appearance of the points which represent a gain.

**Example**

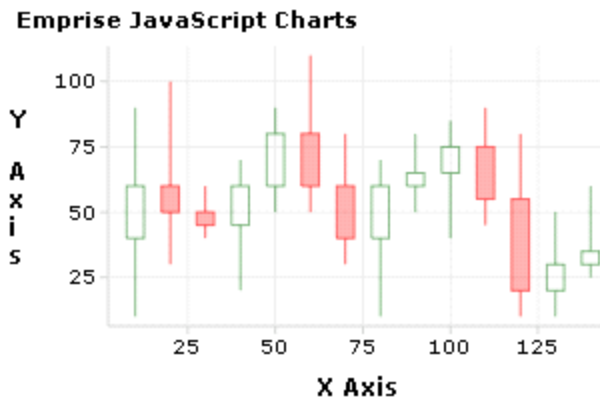
```
var series = new EJSC.CandlestickSeries(new EJSC.DataHandler(),  
    {  
        gain {
```



```

        lineColor: "#006600",
        lineOpacity: 50,
        color: "#000000",
        opacity: 0
    }
};

```



#### 5.4.4.1.6 hint\_string (inherited)

##### Definition

string **hint\_string** = undefined

##### Description

This property may be used to define a series specific string to be used when displaying point hints. This string may contain replacement indicators (see [Text Replacement Options](#)). When left undefined, a default hint defined by the series type will be displayed.

##### Example

*>> Modify the default hint to include custom text.*

```

var series = new EJSC.LineSeries(new EJSC.ArrayDataHandler(),
    {
        hint_string: "My Custom Hint<br /><strong>X: </strong>[x]<br/><strong>Y:
</strong>[y]"
    }
);

```

#### 5.4.4.1.7 intervalOffset

##### Definition

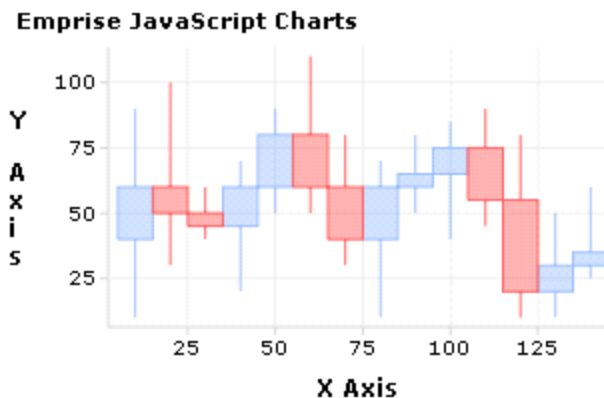
float **intervalOffset** = 0.8

##### Description

Defines the spacing between the points as a percentage of 1. 1 = points take up the entire width available, with their sides touching.

### Example

```
var chart = new EJSC.Chart("chart");
var series = chart.addSeries(new EJSC.CandlestickSeries(
    data,
    {
        intervalOffset: 1
    }
));
```



#### 5.4.4.1.8 legendsVisible (inherited)

##### Definition

boolean **legendsVisible** = true

##### Description

Determines whether the legend item associated with the series appears in the legend window. Use the `Series.showLegend` and `Series.hideLegend` methods to control legend item visibility after series creation.

#### 5.4.4.1.9 lineOpacity (inherited)

##### Definition

integer **lineOpacity** = 100

##### Description

Determines the line opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see [EJSC.Series.setLineOpacity\(\)](#)

#### 5.4.4.1.10 lineWidth (inherited)

##### Definition

integer **lineWidth** = 1

### Description

Defines the width of the line connecting series points.

#### 5.4.4.1.11 loss

### Definition

```
object loss = {
  lineColor: "rgb(249,95,95)",
  lineOpacity: 100,
  color: "rgb(249,95,95)",
  opacity: 50
}
```

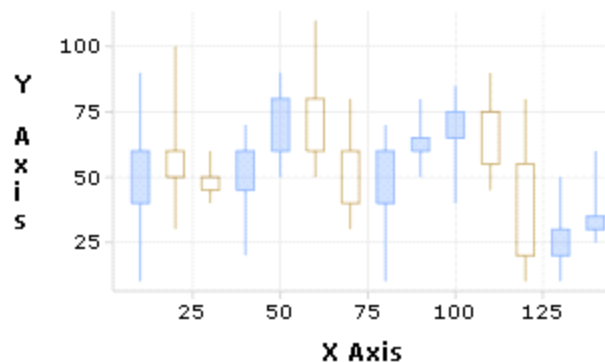
### Description

Defines the appearance of the points which represent a loss.

### Example

```
var series = new EJSC.CandlestickSeries(new EJSC.DataHandler(),
  {
    loss {
      lineColor: "#996600",
      lineOpacity: 50,
      color: "#000000",
      opacity: 0
    }
  }
);
```

#### Emprise JavaScript Charts



#### 5.4.4.1.12 opacity (inherited)

### Definition

integer **opacity** = 50

**Description**

Determines the fill opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see [EJSC.Series.setOpacity\(\)](#)

5.4.4.1.13 title (inherited)

**Definition**

string **title** = "Series <index>"

**Description**

Defines the series title used in hints and legend display.

**Note:** The default title of Series <index> (where <index> equals the current position in the series array for its owner chart) is assigned when a series is added to a chart using [addSeries](#). This default title is only applied if the title is blank (i.e. "") at the time the addSeries method is called.

5.4.4.1.14 visible (inherited)

**Definition**

boolean **visible** = true

**Description**

Defines whether the series is visible and can draw on the chart

5.4.4.1.15 x\_axis (inherited)

**Definition**

string **x\_axis** = "bottom"

**Description**

Defines the horizontal axis to use for X values. Valid options are "top" or "bottom".

5.4.4.1.16 x\_axis\_formatter (inherited)

**Definition**

string **x\_axis\_formatter** = undefined

**Description**

Defines the formatter that will be used to format hints for the X values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:

[EJSC.DateFormatter](#)

[EJSC.NumberFormatter](#)

5.4.4.1.17 `y_axis` (inherited)

#### Definition

string `y_axis` = "left"

#### Description

Defines the vertical axis to use for Y values. Valid options are "left" or "right".

5.4.4.1.18 `y_axis_formatter` (inherited)

#### Definition

string `y_axis_formatter` = undefined

#### Description

Defines the formatter that will be used to format hints for the Y values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:

[EJSC.DateFormatter](#)

[EJSC.NumberFormatter](#)

### 5.4.4.2 Methods

5.4.4.2.1 `findClosestPoint` (inherited)

#### Definition

[EJSC.Point](#) `findClosestPoint`( integer pixel )

#### Description

Returns the point object closest to the pixel coordinate given or null if no points are found.

5.4.4.2.2 `getDataHandler` (inherited)

#### Definition

[EJSC.DataHandler](#) `getDataHandler`( )

#### Description

Returns the [EJSC.DataHandler](#) descendant currently assigned to the series. This is not applicable to all series and will return null if a data handler has not been defined or is not used.

## 5.4.4.2.3 getVisibility (inherited)

**Definition**

boolean **getVisibility**( )

**Description**

Returns a boolean indicating the series current visible state

## 5.4.4.2.4 hide (inherited)

**Definition**

void **hide**( )

**Description**

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already hidden.

## 5.4.4.2.5 hideLegend (inherited)

**Definition**

void **hideLegend**( )

**Description**

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

## 5.4.4.2.6 reload (inherited)

**Definition**

void **reload**( )

**Description**

Triggers a series reload if the functionality is defined in a child class. This may involve retrieving data again or simply recalculating. While the reload() method exists in the base Series class, implementation of the protected methods triggered by calling reload() is at the discretion of the child class.

## 5.4.4.2.7 setColor (inherited)

**Definition**

void **setColor**( string color )

#### **Description**

Changes the series color and causes the chart to redraw.

#### 5.4.4.2.8 setColoredLegend (inherited)

#### **Definition**

void **setColoredLegend**( boolean coloredLegend )

#### **Description**

Sets the [EJSC.Series.coloredLegend](#) property and updates the legend to reflect the change.

#### 5.4.4.2.9 setDataHandler (inherited)

#### **Definition**

void **setDataHandler**( [EJSC.DataHandler](#) dataHandler, boolean reload )

#### **Description**

Updates the series data handler and triggers a data reload if reload parameter is **true**. This method may not be implemented in all child classes.

#### 5.4.4.2.10 setLineOpacity (inherited)

#### **Definition**

void **setLineOpacity**( integer opacity )

#### **Description**

Sets the [EJSC.Series.lineOpacity](#) property and redraws the series to reflect the change.

#### 5.4.4.2.11 setLineWidth (inherited)

#### **Definition**

void **setLineWidth**( integer width )

#### **Description**

Updates the [lineWidth](#) property and redraws the chart.

#### 5.4.4.2.12 setOpacity (inherited)

#### **Definition**

void **setOpacity**( integer opacity )

**Description**

Sets the [EJSC.Series.opacity](#) property and redraws the series to reflect the change.

5.4.4.2.13 setTitle (inherited)

**Definition**

void [setTitle](#)( string title )

**Description**

Changes the series title and update the legend caption (if applicable).

**Note:** this will not cause updates to a hint which is visible at the time it is called, though the next time a hint is rendered it will show the new property value.

5.4.4.2.14 show (inherited)

**Definition**

void [show](#)( )

**Description**

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already visible.

5.4.4.2.15 showLegend (inherited)

**Definition**

void [showLegend](#)( )

**Description**

Changes the series legend item visible state.

No effect if the series legend item is already visible.

**5.4.4.3 Events**

5.4.4.3.1 onAfterDataAvailable (inherited)

**Definition**

boolean [onAfterDataAvailable](#)( [EJSC.Chart](#) chart, [EJSC.Series](#) series )

**Description**



Fired after the series has processed its data and before the chart is told to redraw. Returning **false** from this event handler will cancel redrawing the chart.

#### 5.4.4.3.2 onAfterVisibilityChange (inherited)

##### Definition

boolean **onAfterVisibilityChange**( [EJSC.Series](#) series, [EJSC.Chart](#) chart, boolean visible )

##### Description

Fired when the series visibility changes. Sends series which changed, its owner chart and its new visible property value.

Returning **false** from this event handler will cancel redrawing the chart.

#### 5.4.4.3.3 onBeforeVisibilityChange (inherited)

##### Definition

boolean **onBeforeVisibilityChange**( [EJSC.Series](#) series, [EJSC.Chart](#) chart )

##### Description

Fired before the series visibility is about to change. Sends series which changed, its owner chart. The current visible state of the series may be checked using the series.visible property.

Returning **false** from this event handler will cancel the change in visibility.

#### 5.4.4.3.4 onShowHint (inherited)

##### Definition

string **onShowHint**( [EJSC.Point](#) point, [EJSC.Series](#) series, [EJSC.Chart](#) chart, string hint\_string )

##### Description

Fired before the displaying the hint, this event allows the current hint string to be overridden. See [Text Replacement Options](#) for a list of the automatic substitutions available.

#### 5.4.4.4 Data Formats

**XML** ([EJSC.XMLDataHandler](#), [EJSC.XMLStringDataHandler](#)):

Unused parameters may be omitted

**Full:**

```
<graph>
  <plot>
    <point x="number or string" high="number" low="number"
open="number" close="number" label="string" userdata="string"/>
```

```

        <point x="number or string" high="number" low="number"
open="number" close="number" label="string" userdata="string"/>
    </plot>
</graph>

```

Short:

```

    <G>
        <L>
            <P x="number or string" high="number" low="number"
open="number" close="number" label="string" userdata="string"/>
            <P x="number or string" high="number" low="number"
open="number" close="number" label="string" userdata="string"/>
        </L>
    </G>

```

**Compact:** The values parameter is formatted as CSV

```

    <G>
        <L values="CSV string"/>
    </G>

```

**Array** ([EJSC.ArrayDataHandler](#)):

```

[
    ["x", "high", "low", "open", "close", "label", "userdata"],
    ["x", "high", "low", "open", "close", "label", "userdata"]
]

```

Null should be used to skip unused parameters:

```

[
    ["x", null, null, "open", "close", null, "userdata"],
    ["x", null, null, "open", "close", null, "userdata"]
]

```

**CSV** ([EJSC.CSVFileDataHandler](#), [EJSC.CSVStringDataHandler](#)):

```
"x|high|low|open|close|label|userdata,x|high|low|open|close|label|userdata"
```

Null should be used to skip unused parameters

```
"x|null|null|open|close|null|userdata,x|null|null|open|close|null|userdata"
```

**JSON** ([EJSC.JSONFileDataHandler](#), [EJSC.JSONStringDataHandler](#)):

Unused parameters may be omitted.

```

[
    { "x": "number or string", "high": "number", "low": "number",

```

```
"open": "number", "close": "number", label: "string", "userdata": "string"},
  {"x": "number or string", "high": "number", "low": "number",
"open": "number", "close": "number", label: "string", "userdata": "string"}
]
```

#### 5.4.4.5 Text Replacement Options

String	Replaced With
[chart_title]	The title of the chart
[series_title]	The title of the series the selected point resides in
[xaxis]	The caption of the series x axis
[yaxis]	The caption of the series y axis
[x]	The preformatted X value of the point
[high]	The preformatted high value of the point
[low]	The preformatted low value of the point
[open]	The preformatted open value of the point
[close]	The preformatted close value of the point
[label]	The label property of the current point

### 5.4.5 EJSC.FloatingBarSeries

FloatingBarSeries renders its points as vertical or horizontal bars which are not fixed to a baseline.

The constructor expects an instantiated [EJSC.DataHandler](#) descendant and an optional set of object properties.

#### Constructor

```
EJSC.FloatingBarSeries( EJSC.DataHandler dataHandler [, object options] )
```

#### Example

```
var mySeries = new EJSC.FloatingBarSeries(
  new EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,3],[5,2],[6,4]]),
  { title: "New Floating Bar Series" }
);
```

#### Defining Properties and Events

FloatingBarSeries properties may be set individually after the series has been created or in batch using the options parameter during instantiation.

##### Setting properties individually

```
var mySeries = new EJSC.FloatingBarSeries(new
EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,3],[5,2],[6,4]]));
mySeries.lineWidth = 1;
mySeries.title = "New Floating Bar Series";
```

##### Setting properties in batch

```
var mySeries = new EJSC.FloatingBarSeries(
  new EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,3],[5,2],[6,4]]),
```

```
        { lineWidth: 1, title: "New Floating Bar Series" }  
    );
```

### 5.4.5.1 Properties

#### 5.4.5.1.1 autosort (inherited)

##### Definition

boolean **autosort** = true

##### Description

Defines whether the series applies the appropriate sort to points prior to drawing. Drawing routines are generally optimized to accept data in a certain order and this property eliminates the need for the developer to worry about that order prior to sending data to the series. If set to false, the data must be properly ordered beforehand in order to ensure the series renders correctly.

**Note:** May not be applicable to all series (i.e. [EJSC.PieSeries](#), [EJSC.AnalogGaugeSeries](#))

#### 5.4.5.1.2 color (inherited)

##### Definition

string **color** = undefined

##### Description

Defines the series color. If left undefined, the color is pulled from the array of default colors stored in the chart.

**Note:** May not be applicable to all series (i.e. [EJSC.PieSeries](#))

#### 5.4.5.1.3 coloredLegend (inherited)

##### Definition

boolean **coloredLegend** = true

##### Description

Determines whether the legend text inherits the series color. Setting to false will allow the legend text to inherit its normal cascaded color. To modify this property after series creation see [EJSC.Series.setColoredLegend\(\)](#)

#### 5.4.5.1.4 defaultColors (inherited)

##### Definition

array **defaultColors** = [EJSC.DefaultBarColors](#)

##### Description

Defines the pool of default colors available for bar series bars.

**NOTE:** To make use of individually colored bars, the [EJSC.BarSeries.useColorArray](#) property must be set to true.

#### 5.4.5.1.5 delayLoad (inherited)

### EXPERIMENTAL

#### Definition

boolean **delayLoad** = true

#### Description

This property allows a series to force its data handler to begin data retrieval as soon as it is added to a chart. When set to false, the series will initiate data retrieval as soon as it is added to a chart. When true, the data retrieval is initialized when the series first needs to draw.

#### 5.4.5.1.6 groupedBars (inherited)

#### Definition

boolean **groupedBars** = true

#### Description

Defines whether the bars from different bar series in a single chart are grouped or overlaid.

**NOTE:** Only the first bar series added to the chart may use this property to affect the grouped/overlay display. To change the style after the first bar series has been added, use the [EJSC.BarSeries.setGroupedBars](#) from any bar series in the chart. Currently you cannot mix both overlaid and grouped bars within the same chart.

#### Example

>> *Make the bar series overlay*

```
var chart = new EJSC.Chart("chart");
var bar1 = chart.addSeries(new EJSC.BarSeries(
    data,
    {
        groupedBars: false,
    }
));
var bar2 = chart.addSeries(new EJSC.BarSeries(data));
```

#### 5.4.5.1.7 hint\_string (inherited)

#### Definition

string **hint\_string** = undefined

### Description

This property may be used to define a series specific string to be used when displaying point hints. This string may contain replacement indicators (see [Text Replacement Options](#)). When left undefined, a default hint defined by the series type will be displayed.

### Example

>> *Modify the default hint to include custom text.*

```
var series = new EJSC.LineSeries(new EJSC.ArrayDataHandler(),
    {
        hint_string: "My Custom Hint<br /><strong>X: </strong>[x]<br/><strong>Y:
</strong>[y]"
    }
);
```

#### 5.4.5.1.8 intervalOffset (inherited)

### Definition

float **intervalOffset** = 0.8

### Description

Defines the spacing between the bars as a percentage of 1. 1 = bars take up the entire width available, with their sides touching.

### Example

>> *Make the bars take up 1/2 their available width. (i.e. more space between bars than by default)*

```
var chart = new EJSC.Chart("chart");
var bar = chart.addSeries(new EJSC.BarSeries(
    data,
    {
        intervalOffset: 0.5
    }
));
```

#### 5.4.5.1.9 legendsVisible (inherited)

### Definition

boolean **legendsVisible** = true

### Description

Determines whether the legend item associated with the series appears in the legend window. Use the Series.showLegend and Series.hideLegend methods to control legend item visibility after series creation.

## 5.4.5.1.10 lineOpacity (inherited)

**Definition**

integer **lineOpacity** = 100

**Description**

Determines the line opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see [EJSC.Series.setLineOpacity\(\)](#)

## 5.4.5.1.11 lineWidth (inherited)

**Definition**

integer **lineWidth** = 1

**Description**

Defines the width of the line connecting series points.

## 5.4.5.1.12 opacity (inherited)

**Definition**

integer **opacity** = 50

**Description**

Determines the fill opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see [EJSC.Series.setOpacity\(\)](#)

## 5.4.5.1.13 orientation (inherited)

**Definition**

string **orientation** = "vertical"

**Description**

This property determines the orientation of the bar series. The supported property values are "vertical" and "horizontal".

**Note:** Currently this is a creation-time only property and should not be changed after the series has been created.

## 5.4.5.1.14 ranges (inherited)

**Definition**

array **ranges** = new Array()

**Description**

This property is used to store the ranges for a given bar series. The range objects are used to draw bars in varying styles based on their Y value.

The range objects stored are defined as:

```
range = {
  min:          float, // >=
  max:          float, // <
  color:        string, // Defined as rgb(RED, GREEN, BLUE), ex: "rgb(255,0,0)"
  opacity:      integer, // Represents percent, i.e. 50 = 50%
  lineOpacity:  integer, // Represents percent, i.e. 50 = 50%
  lineWidth:   integer // Represents the line width in pixels
}
```

**Example**

>> Define ranges so that bars with a Y value from 0 to 10 and 90 to 100 are colored red, bars 10 - 90 are colored green

```
var chart = new EJSC.Chart("chart");
var bar = chart.addSeries(new EJSC.BarSeries(
  data,
  {
    ranges: [
      { 0, 10, "rgb(255,0,0)", 100, 100, 1 },
      { 90, 100, "rgb(255,0,0)", 100, 100, 1 },
      { 10, 90, "rgb(0,255,0)", 100, 100, 1 }
    ]
  }
));
```

## 5.4.5.1.15 title (inherited)

**Definition**

string **title** = "Series <index>"

**Description**

Defines the series title used in hints and legend display.

**Note:** The default title of Series <index> (where <index> equals the current position in the series array for its owner chart) is assigned when a series is added to a chart using [addSeries](#). This default title is only applied if the title is blank (i.e. "") at the time the addSeries method is called.



## 5.4.5.1.16 treeLegend (inherited)

**Definition**

boolean **treeLegend** = false

**Description**

Defines whether to display each bar individually in the legend.

If the [useColorArray](#) property is true and treeLegend left at its default (not defined in options), the treeLegend property will automatically be set to true to enable tree-style legend display.

## 5.4.5.1.17 useColorArray (inherited)

**Definition**

boolean **useColorArray** = false

**Description**

Defines whether to make use of the [EJSC.BarSeries.defaultColors](#) property

**Example**

*>> Use the color array to make a chart of multi colored bars*

```
var chart = new EJSC.Chart("chart");
var bar = chart.addSeries(new EJSC.BarSeries(
    data,
    {
        useColorArray: true
    }
));
```

## 5.4.5.1.18 visible (inherited)

**Definition**

boolean **visible** = true

**Description**

Defines whether the series is visible and can draw on the chart

## 5.4.5.1.19 x\_axis (inherited)

**Definition**

string **x\_axis** = "bottom"

**Description**

Defines the horizontal axis to use for X values. Valid options are "top" or "bottom".

#### 5.4.5.1.20 x\_axis\_formatter (inherited)

##### Definition

string **x\_axis\_formatter** = undefined

##### Description

Defines the formatter that will be used to format hints for the X values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:

[EJSC.DateFormatter](#)

[EJSC.NumberFormatter](#)

#### 5.4.5.1.21 y\_axis (inherited)

##### Definition

string **y\_axis** = "left"

##### Description

Defines the vertical axis to use for Y values. Valid options are "left" or "right".

#### 5.4.5.1.22 y\_axis\_formatter (inherited)

##### Definition

string **y\_axis\_formatter** = undefined

##### Description

Defines the formatter that will be used to format hints for the Y values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:

[EJSC.DateFormatter](#)

[EJSC.NumberFormatter](#)

### 5.4.5.2 Methods

#### 5.4.5.2.1 addRange (inherited)

##### Definition

void **addRange**( float min, float max, string color, integer opacity, integer lineOpacity, integer lineWidth, boolean redraw )

**Description**

Adds a new range to the bar series and optionally triggers a redraw.

5.4.5.2.2 clearRanges (inherited)

**Definition**

void [clearRanges](#)( boolean redraw )

**Description**

Clears all ranges defined for the series and optionally redraws the the chart.

5.4.5.2.3 deleteRange (inherited)

**Definition**

void [deleteRange](#)( float min, float max, boolean redraw )

**Description**

Deletes the matching range (min and max must exactly match an existing range) and optionally redraws the chart.

5.4.5.2.4 getBarSize (inherited)

**Definition**

integer [getBarSize](#)()

**Description**

This method returns the size in pixels (width or height depending on orientation) of a single bar.

5.4.5.2.5 getDataHandler (inherited)

**Definition**

EJSC.DataHandler [getDataHandler](#)( )

**Description**

Returns the EJSC.DataHandler descendant currently assigned to the series. This is not applicable to all series and will return null if a data handler has not been defined or is not used.

5.4.5.2.6 getPoints (inherited)

**Definition**

array [getPoints](#)()

**Description**

This method returns an array of all [EJSC.BarPoint](#) objects in the series.

## 5.4.5.2.7 getVisibility (inherited)

Enter topic text here.

## 5.4.5.2.8 hide (inherited)

**Definition**

void [hide](#)( )

**Description**

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already hidden.

## 5.4.5.2.9 hideLegend (inherited)

**Definition**

void [hideLegend](#)( )

**Description**

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

## 5.4.5.2.10 reload (inherited)

**Definition**

void [reload](#)( )

**Description**

Triggers a series reload if the functionality is defined in a child class. This may involve retrieving data again or simply recalculating. While the reload() method exists in the base Series class, implementation of the protected methods triggered by calling reload() is at the discretion of the child class.

## 5.4.5.2.11 setColor (inherited)

**Definition**

void [setColor](#)( string color )

**Description**

Changes the series color and causes the chart to redraw.

#### 5.4.5.2.12 setColoredLegend (inherited)

##### Definition

void **setColoredLegend**( boolean coloredLegend )

##### Description

Sets the [EJSC.Series.coloredLegend](#) property and updates the legend to reflect the change.

#### 5.4.5.2.13 setDataHandler (inherited)

##### Definition

void **setDataHandler**( [EJSC.DataHandler](#) dataHandler, boolean reload )

##### Description

Updates the series data handler and triggers a data reload if reload parameter is **true**. This method may not be implemented in all child classes.

#### 5.4.5.2.14 setDefaultColors (inherited)

##### Definition

void **setDefaultColors**( array colors, boolean reload )

##### Description

Set the array of default colors available for bar series bars.

If reload = **true** the bars (if loaded) will pull their colors from the given color array and the chart will be redrawn.

If [onBarNeedsColor](#) is assigned, this method will ignore that event and load the colors from the array.

##### Example

```
var colors = [  
    "rgb('0,0,0')",      // black  
    "rgb(255,0,0)",     // red  
    "rgb('0,255,0')",   // green  
    "rgb('0,0,255')",   // blue  
    "rgb('255,255,255') // white  
];  
  
myBarSeries.setDefaultColors(colors, true);
```

## 5.4.5.2.15 getGroupedBars (inherited)

Enter topic text here.

## 5.4.5.2.16 setIntervalOffset (inherited)

**Definition**

void [setIntervalOffset](#)( float offset, boolean redraw )

**Description**

Updates the interval offset property (spacing between the bars) and optionally redraws the chart. See [EJSC.BarSeries.intervalOffset](#) for additional information.

## 5.4.5.2.17 setLineWidth (inherited)

**Definition**

void [setLineWidth](#)( integer width )

**Description**

Updates the [lineWidth](#) property and redraws the chart.

## 5.4.5.2.18 setOpacity (inherited)

**Definition**

void [setOpacity](#)( integer opacity )

**Description**

Sets the [EJSC.Series.opacity](#) property and redraws the series to reflect the change.

## 5.4.5.2.19 setTitle (inherited)

**Definition**

void [setTitle](#)( string title )

**Description**

Changes the series title and update the legend caption (if applicable).

**Note:** this will not cause updates to a hint which is visible at the time it is called, though the next time a hint is rendered it will show the new property value.

## 5.4.5.2.20 show (inherited)

**Definition**

void [show](#)( )

#### Description

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already visible.

5.4.5.2.21 [showLegend](#) (inherited)

#### Definition

void [showLegend](#)( )

#### Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

### 5.4.5.3 Events

5.4.5.3.1 [onAfterDataAvailable](#) (inherited)

#### Definition

boolean [onAfterDataAvailable](#)( [EJSC.Chart](#) chart, [EJSC.Series](#) series )

#### Description

Fired after the series has processed its data and before the chart is told to redraw. Returning **false** from this event handler will cancel redrawing the chart.

5.4.5.3.2 [onAfterVisibilityChange](#) (inherited)

#### Definition

boolean [onAfterVisibilityChange](#)( [EJSC.Series](#) series, [EJSC.Chart](#) chart, boolean visible )

#### Description

Fired when the series visibility changes. Sends series which changed, its owner chart and its new visible property value.

Returning **false** from this event handler will cancel redrawing the chart.

5.4.5.3.3 [onBarNeedsColor](#) (inherited)

#### Definition

string [onBarNeedsColor](#)( [EJSC.BarPoint](#) point, [EJSC.Series](#) series, [EJSC.Chart](#) chart )  
 object [onBarNeedsColor](#)( [EJSC.BarPoint](#) point, [EJSC.Series](#) series, [EJSC.Chart](#) chart )

### Description

Fired when a bar needs a color. This event may return either a string (i.e. "rgb(0,255,0)") or an object with additional styling properties.

The object returned is expected to be formatted as:

```
{
    color:      string,
    opacity:    integer,
    lineOpacity: integer,
    lineWidth:  integer
}
```

### Example

>> *Display bars with userdata of "bold" with thicker lines*

```
var chart = new EJSC.Chart("chart");
var bar = chart.addSeries(new EJSC.BarSeries(
    data,
    {
        lineOpacity: 80,
        lineWidth: 1,
        onBarNeedsColor: function(point, series, chart) {

            if (point.userdata == "bold") {
                return {
                    color: "rgb(0,0,0)",
                    opacity: 100,
                    lineOpacity: 100,
                    lineWidth: 100
                };
            } else {
                return "rgb(128,128,128)";
            }
        }
    }
));
```

#### 5.4.5.3.4 onBeforeVisibilityChange (inherited)

### Definition

boolean **onBeforeVisibilityChange**( [EJSC.Series](#) series, [EJSC.Chart](#) chart )

### Description

Fired before the series visibility is about to change. Sends series which changed, its owner chart. The current visible state of the series may be checked using the series.visible property.

Returning **false** from this event handler will cancel the change in visibility.



## 5.4.5.3.5 onShowHint (inherited)

**Definition**

string **onShowHint**( [EJSC.Point](#) point, [EJSC.Series](#) series, [EJSC.Chart](#) chart, string hint\_string )

**Description**

Fired before the displaying the hint, this event allows the current hint string to be overridden. See [Text Replacement Options](#) for a list of the automatic substitutions available.

## 5.4.5.4 Data Formats

**XML** ([EJSC.XMLDataHandler](#), [EJSC.XMLStringDataHandler](#)):

Unused parameters may be omitted

**Full:**

**Orientation == "vertical"**

```
<graph>
  <plot>
    <point x="number or string" min="number" max="number"
label="string" userdata="string"/>
    <point x="number or string" min="number" max="number"
label="string" userdata="string"/>
  </plot>
</graph>
```

**Orientation == "horizontal"**

```
<graph>
  <plot>
    <point y="number or string" min="number" max="number"
label="string" userdata="string"/>
    <point y="number or string" min="number" max="number"
label="string" userdata="string"/>
  </plot>
</graph>
```

**Short:**

**Orientation == "vertical"**

```
<G>
  <L>
    <P x="number or string" min="number" max="number"
label="string" userdata="string"/>
    <P x="number or string" min="number" max="number"
label="string" userdata="string"/>
  </L>
```

```

    <G>
    Orientation == "horizontal"
    <G>
        <L>
            <P x="number or string" min="number" max="number"
label="string" userdata="string"/>
            <P x="number or string" min="number" max="number"
label="string" userdata="string"/>
        </L>
    </G>

```

**Compact:** The values parameter is formatted as CSV

```

    <G>
        <L values="CSV string"/>
    </G>

```

**Array** ([EJSC.ArrayDataHandler](#)):

```

[
    ["x or y", "min", "max", "label", "userdata"],
    ["x or y", "min", "max", "label", "userdata"]
]

```

Null should be used to skip unused parameters:

```

[
    ["x or y", min, max, null, "userdata"],
    ["x or y", min, max, null, "userdata"]
]

```

**CSV** ([EJSC.CSVFileDataHandler](#), [EJSC.CSVStringDataHandler](#)):

```
"x or y|min|max|label|userdata,x or y|min|max||label|userdata"
```

Null should be used to skip unused parameters

```
"x or y|min|max|null|userdata,x or y|min|max|null|userdata"
```

**JSON** ([EJSC.JSONFileDataHandler](#), [EJSC.JSONStringDataHandler](#)):

Unused parameters may be omitted.

**Orientation == "vertical"**

```

[
    { "y": "number or string", "min": "number", "max": "number",
label: "string", "userdata": "string" },

```

```

        {"y":"number or string", "min":"number", "max":"number",
label":"string","userdata":"string"}
    ]

    Orientation == "horizontal"

    [
        {"x":"number or string", "min":"number", "max":"number",
label":"string","userdata":"string"},
        {"x":"number or string", "min":"number", "max":"number",
label":"string","userdata":"string"}
    ]

```

#### 5.4.5.5 Text Replacement Options

String	Replaced With
[chart_title]	The title of the chart
[series_title]	The title of the series the selected point resides in
[xaxis]	The caption of the series x axis
[yaxis]	The caption of the series y axis
[x]	The preformatted X value of the point
[y]	The preformatted Y value of the point
[label]	The label property of the current point
[min]	The preformatted min value of the current point
[max]	The preformatted max value of the current point

#### 5.4.6 EJSC.FunctionSeries

The FunctionSeries is rendered as a line based on the results of the function specified.

The constructor expects a function which can be called as function(x), and an optional set of object properties.

##### Constructor

```
EJSC.FunctionSeries( function fn [, object options] )
```

##### Example

```
var mySeries = new EJSC.FunctionSeries(
    Math.cos,
    { title: "New Function Series" }
);
```

##### Defining Properties and Events

FunctionSeries properties may be set individually after the series has been created or in batch using the options parameter during instantiation.

##### Setting properties individually

```
var mySeries = new EJSC.FunctionSeries(Math.cos);
```

```
mySeries.lineWidth = 2;  
mySeries.title = "New Function Series";
```

### Setting properties in batch

```
var mySeries = new EJSC.FunctionSeries(  
    Math.cos,  
    { lineWidth: 2, title: "New Function Series" }  
);
```

## 5.4.6.1 Properties

### 5.4.6.1.1 color (inherited)

#### Definition

string **color** = undefined

#### Description

Defines the series color. If left undefined, the color is pulled from the array of default colors stored in the chart.

**Note:** May not be applicable to all series (i.e. [EJSC.PieSeries](#))

### 5.4.6.1.2 coloredLegend (inherited)

#### Definition

boolean **coloredLegend** = true

#### Description

Determines whether the legend text inherits the series color. Setting to false will allow the legend text to inherit its normal cascaded color. To modify this property after series creation see [EJSC.Series.setColoredLegend\(\)](#)

### 5.4.6.1.3 hint\_string (inherited)

#### Definition

string **hint\_string** = undefined

#### Description

This property may be used to define a series specific string to be used when displaying point hints. This string may contain replacement indicators (see [Text Replacement Options](#)). When left undefined, a default hint defined by the series type will be displayed.

#### Example

>> *Modify the default hint to include custom text.*

```
var series = new EJSC.LineSeries(new EJSC.ArrayDataHandler(),
    {
        hint_string: "My Custom Hint<br /><strong>X: </strong>[x]<br/><strong>Y:
</strong>[y]"
    }
);
```

#### 5.4.6.1.4 legendsVisible (inherited)

##### Definition

boolean **legendsVisible** = true

##### Description

Determines whether the legend item associated with the series appears in the legend window. Use the `Series.showLegend` and `Series.hideLegend` methods to control legend item visibility after series creation.

#### 5.4.6.1.5 lineOpacity (inherited)

##### Definition

integer **lineOpacity** = 100

##### Description

Determines the line opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see [EJSC.Series.setLineOpacity\(\)](#)

#### 5.4.6.1.6 lineWidth (inherited)

##### Definition

integer **lineWidth** = 1

##### Description

Defines the width of the line connecting series points.

#### 5.4.6.1.7 title (inherited)

##### Definition

string **title** = "Series <index>"

##### Description

Defines the series title used in hints and legend display.

**Note:** The default title of Series <index> (where <index> equals the current position in the series array for its owner chart) is assigned when a series is added to a chart using [addSeries](#). This default title is only applied if the title is blank (i.e. "") at the time the addSeries method is called.

#### 5.4.6.1.8 visible (inherited)

##### Definition

boolean **visible** = true

##### Description

Defines whether the series is visible and can draw on the chart

#### 5.4.6.1.9 x\_axis (inherited)

##### Definition

string **x\_axis** = "bottom"

##### Description

Defines the horizontal axis to use for X values. Valid options are "top" or "bottom".

#### 5.4.6.1.10 x\_axis\_formatter (inherited)

##### Definition

string **x\_axis\_formatter** = undefined

##### Description

Defines the formatter that will be used to format hints for the X values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:

[EJSC.DateFormatter](#)

[EJSC.NumberFormatter](#)

#### 5.4.6.1.11 y\_axis (inherited)

##### Definition

string **y\_axis** = "left"

##### Description

Defines the vertical axis to use for Y values. Valid options are "left" or "right".

#### 5.4.6.1.12 y\_axis\_formatter (inherited)

##### Definition

string [y\\_axis\\_formatter](#) = undefined

### Description

Defines the formatter that will be used to format hints for the Y values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:

[EJSC.DateFormatter](#)

[EJSC.NumberFormatter](#)

## 5.4.6.2 Methods

5.4.6.2.1 [getVisibility](#) (inherited)

### Definition

boolean [getVisibility](#)( )

### Description

Returns a boolean indicating the series current visible state

5.4.6.2.2 [hide](#) (inherited)

### Definition

void [hide](#)( )

### Description

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already hidden.

5.4.6.2.3 [hideLegend](#) (inherited)

### Definition

void [hideLegend](#)( )

### Description

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

5.4.6.2.4 [reload](#) (inherited)

### Definition

void **reload**( )

#### **Description**

Triggers a series reload if the functionality is defined in a child class. This may involve retrieving data again or simply recalculating. While the reload() method exists in the base Series class, implementation of the protected methods triggered by calling reload() is at the discretion of the child class.

#### 5.4.6.2.5 setColor (inherited)

#### **Definition**

void **setColor**( string color )

#### **Description**

Changes the series color and causes the chart to redraw.

#### 5.4.6.2.6 setColoredLegend (inherited)

#### **Definition**

void **setColoredLegend**( boolean coloredLegend )

#### **Description**

Sets the [EJSC.Series.coloredLegend](#) property and updates the legend to reflect the change.

#### 5.4.6.2.7 setLineWidth (inherited)

#### **Definition**

void **setLineWidth**( integer width )

#### **Description**

Updates the [lineWidth](#) property and redraws the chart.

#### 5.4.6.2.8 setTitle (inherited)

#### **Definition**

void **setTitle**( string title )

#### **Description**

Changes the series title and update the legend caption (if applicable).

**Note:** this will not cause updates to a hint which is visible at the time it is called, though the next time a hint is rendered it will show the new property value.



#### 5.4.6.2.9 show (inherited)

##### Definition

void [show](#)( )

##### Description

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already visible.

#### 5.4.6.2.10 showLegend (inherited)

##### Definition

void [showLegend](#)( )

##### Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

### 5.4.6.3 Events

#### 5.4.6.3.1 onAfterVisibilityChange (inherited)

##### Definition

boolean [onAfterVisibilityChange](#)( [EJSC.Series](#) series, [EJSC.Chart](#) chart, boolean visible )

##### Description

Fired when the series visibility changes. Sends series which changed, its owner chart and its new visible property value.

Returning **false** from this event handler will cancel redrawing the chart.

#### 5.4.6.3.2 onBeforeVisibilityChange (inherited)

##### Definition

boolean [onBeforeVisibilityChange](#)( [EJSC.Series](#) series, [EJSC.Chart](#) chart )

##### Description

Fired before the series visibility is about to change. Sends series which changed, its owner chart. The current visible state of the series may be checked using the series.visible property.

Returning **false** from this event handler will cancel the change in visibility.

## 5.4.6.3.3 onShowHint (inherited)

**Definition**

string **onShowHint**( [EJSC.Point](#) point, [EJSC.Series](#) series, [EJSC.Chart](#) chart, string hint\_string )

**Description**

Fired before the displaying the hint, this event allows the current hint string to be overridden. See [Text Replacement Options](#) for a list of the automatic substitutions available.

## 5.4.6.4 Text Replacement Options

String	Replaced With
[chart_title]	The title of the chart
[series_title]	The title of the series the selected point resides in
[xaxis]	The caption of the series x axis
[yaxis]	The caption of the series y axis
[x]	The preformatted X value of the point
[y]	The preformatted Y value of the point
[label]	The label property of the current point

## 5.4.7 EJSC.LineSeries

LineSeries is rendered by drawing a line from point to point.

The constructor expects an instantiated [EJSC.DataHandler](#) descendant and an optional set of object properties.

**Constructor**

```
EJSC.LineSeries( EJSC.DataHandler dataHandler [, object options] )
```

**Example**

```
var mySeries = new EJSC.LineSeries(
  new EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,3],[5,2],[6,4]]),
  { title: "New Line Series" }
);
```

**Defining Properties and Events**

LineSeries properties may be set individually after the series has been created or in batch using the options parameter during instantiation.

**Setting properties individually**

```
var mySeries = new EJSC.LineSeries(new
EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,3],[5,2],[6,4]]));
mySeries.lineWidth = 2;
```

```
mySeries.title = "New Line Series";
```

### Setting properties in batch

```
var mySeries = new EJSC.LineSeries(  
    new EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,3],[5,2],[6,4]]),  
    { lineWidth: 2, title: "New Line Series" }  
);
```

## 5.4.7.1 Properties

### 5.4.7.1.1 autosort (inherited)

#### Definition

boolean **autosort** = true

#### Description

Defines whether the series applies the appropriate sort to points prior to drawing. Drawing routines are generally optimized to accept data in a certain order and this property eliminates the need for the developer to worry about that order prior to sending data to the series. If set to false, the data must be properly ordered beforehand in order to ensure the series renders correctly.

**Note:** May not be applicable to all series (i.e. [EJSC.PieSeries](#), [EJSC.AnalogGaugeSeries](#))

### 5.4.7.1.2 color (inherited)

#### Definition

string **color** = undefined

#### Description

Defines the series color. If left undefined, the color is pulled from the array of default colors stored in the chart.

**Note:** May not be applicable to all series (i.e. [EJSC.PieSeries](#))

### 5.4.7.1.3 coloredLegend (inherited)

#### Definition

boolean **coloredLegend** = true

#### Description

Determines whether the legend text inherits the series color. Setting to false will allow the legend text to inherit its normal cascaded color. To modify this property after series creation see [EJSC.Series.setColoredLegend\(\)](#)

## 5.4.7.1.4 drawPoints

**Definition**

boolean **drawPoints** = false

**Description**

Determines whether the individual points in the series are visually indicated by round dots on the chart. The size and color of the fill and border are determined by the [pointColor](#), [pointSize](#), [pointBorderColor](#) and [pointBorderSize](#) properties.

**Note:** Setting this property to true for series which have a large number of points may impact performance as additional draws are required to render the points.

## 5.4.7.1.5 hint\_string (inherited)

**Definition**

string **hint\_string** = undefined

**Description**

This property may be used to define a series specific string to be used when displaying point hints. This string may contain replacement indicators (see [Text Replacement Options](#)). When left undefined, a default hint defined by the series type will be displayed.

**Example**

*>> Modify the default hint to include custom text.*

```
var series = new EJSC.LineSeries(new EJSC.ArrayDataHandler(),
    {
        hint_string: "My Custom Hint<br /><strong>X: </strong>[x]<br/><strong>Y:
</strong>[y]"
    }
);
```

## 5.4.7.1.6 legendsVisible (inherited)

**Definition**

boolean **legendsVisible** = true

**Description**

Determines whether the legend item associated with the series appears in the legend window. Use the `Series.showLegend` and `Series.hideLegend` methods to control legend item visibility after series creation.

## 5.4.7.1.7 lineOpacity (inherited)

**Definition**

integer **lineOpacity** = 100

### Description

Determines the line opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see [EJSC.Series.setLineOpacity\(\)](#)

#### 5.4.7.1.8 lineWidth (inherited)

### Definition

integer **lineWidth** = 1

### Description

Defines the width of the line connecting series points.

#### 5.4.7.1.9 pointBorderColor

### Definition

string **pointBorderColor** = "rgb(255,255,255)"

### Description

Defines the color of the border drawn around the points. This property is only applicable when the series [drawPoints](#) property is set to true and [pointBorderSize](#) is greater than 0.

#### 5.4.7.1.10 pointBorderSize

### Definition

integer **pointBorderSize** = 0

### Description

Defines the size in pixels of the border drawn around the points. This property is only applicable when the series [drawPoints](#) property is set to true.

To draw points without any border, leave pointBorderSize set to 0.

#### 5.4.7.1.11 pointColor

### Definition

string **pointColor** = undefined

### Description

Defines the color of the point (circle) drawn at each data point in the series. If left undefined the point will

inherit the series color.

This property is only applicable when the series [drawPoints](#) property is set to true.

#### 5.4.7.1.12 pointSize

##### Definition

integer **pointSize** = undefined

##### Description

Defines the size of the point (circle) drawn at each data point in the series. If left undefined the point diameter will be twice the line width of the series.

This property is only applicable when the series [drawPoints](#) property is set to true.

#### 5.4.7.1.13 title (inherited)

##### Definition

string **title** = "Series <index>"

##### Description

Defines the series title used in hints and legend display.

**Note:** The default title of Series <index> (where <index> equals the current position in the series array for its owner chart) is assigned when a series is added to a chart using [addSeries](#). This default title is only applied if the title is blank (i.e. "") at the time the addSeries method is called.

#### 5.4.7.1.14 visible (inherited)

##### Definition

boolean **visible** = true

##### Description

Defines whether the series is visible and can draw on the chart

#### 5.4.7.1.15 x\_axis (inherited)

##### Definition

string **x\_axis** = "bottom"

##### Description

Defines the horizontal axis to use for X values. Valid options are "top" or "bottom".

#### 5.4.7.1.16 x\_axis\_formatter (inherited)

##### Definition

string **x\_axis\_formatter** = undefined

##### Description

Defines the formatter that will be used to format hints for the X values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:

[EJSC.DateFormatter](#)

[EJSC.NumberFormatter](#)

#### 5.4.7.1.17 y\_axis (inherited)

##### Definition

string **y\_axis** = "left"

##### Description

Defines the vertical axis to use for Y values. Valid options are "left" or "right".

#### 5.4.7.1.18 y\_axis\_formatter (inherited)

##### Definition

string **y\_axis\_formatter** = undefined

##### Description

Defines the formatter that will be used to format hints for the Y values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:

[EJSC.DateFormatter](#)

[EJSC.NumberFormatter](#)

### 5.4.7.2 Methods

Enter topic text here.

#### 5.4.7.2.1 getDataHandler (inherited)

##### Definition

EJSC.DataHandler **getDataHandler**( )

##### Description

Returns the EJSC.DataHandler descendant currently assigned to the series. This is not applicable to all series and will return null if a data handler has not been defined or is not used.

#### 5.4.7.2.2 getVisibility (inherited)

##### **Definition**

boolean **getVisibility**( )

##### **Description**

Returns a boolean indicating the series current visible state

#### 5.4.7.2.3 hide (inherited)

##### **Definition**

void **hide**( )

##### **Description**

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already hidden.

#### 5.4.7.2.4 hideLegend (inherited)

##### **Definition**

void **hideLegend**( )

##### **Description**

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

#### 5.4.7.2.5 reload (inherited)

##### **Definition**

void **reload**( )

##### **Description**

Triggers a series reload if the functionality is defined in a child class. This may involve retrieving data again or simply recalculating. While the reload() method exists in the base Series class, implementation of the protected methods triggered by calling reload() is at the discretion of the child class.



## 5.4.7.2.6 setColor (inherited)

**Definition**

void **setColor**( string color )

**Description**

Changes the series color and causes the chart to redraw.

## 5.4.7.2.7 setColoredLegend (inherited)

**Definition**

void **setColoredLegend**( boolean coloredLegend )

**Description**

Sets the [EJSC.Series.coloredLegend](#) property and updates the legend to reflect the change.

## 5.4.7.2.8 setDataHandler (inherited)

**Definition**

void **setDataHandler**( [EJSC.DataHandler](#) dataHandler, boolean reload )

**Description**

Updates the series data handler and triggers a data reload if reload parameter is **true**. This method may not be implemented in all child classes.

## 5.4.7.2.9 setLineWidth (inherited)

**Definition**

void **setLineWidth**( integer width )

**Description**

Updates the [lineWidth](#) property and redraws the chart.

## 5.4.7.2.10 setTitle (inherited)

**Definition**

void **setTitle**( string title )

**Description**

Changes the series title and update the legend caption (if applicable).

**Note:** this will not cause updates to a hint which is visible at the time it is called, though the next time a hint is rendered it will show the new property value.

#### 5.4.7.2.11 show (inherited)

##### Definition

void [show](#)( )

##### Description

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already visible.

#### 5.4.7.2.12 showLegend (inherited)

##### Definition

void [showLegend](#)( )

##### Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

### 5.4.7.3 Events

Enter topic text here.

#### 5.4.7.3.1 onAfterDataAvailable (inherited)

##### Definition

boolean [onAfterDataAvailable](#)( [EJSC.Chart](#) chart, [EJSC.Series](#) series )

##### Description

Fired after the series has processed its data and before the chart is told to redraw. Returning **false** from this event handler will cancel redrawing the chart.

#### 5.4.7.3.2 onAfterVisibilityChange (inherited)

##### Definition

boolean [onAfterVisibilityChange](#)( [EJSC.Series](#) series, [EJSC.Chart](#) chart, boolean visible )

##### Description

Fired when the series visibility changes. Sends series which changed, its owner chart and its new visible property value.

Returning **false** from this event handler will cancel redrawing the chart.

#### 5.4.7.3.3 onBeforeVisibilityChange (inherited)

##### Definition

boolean **onBeforeVisibilityChange**( [EJSC.Series](#) series, [EJSC.Chart](#) chart )

##### Description

Fired before the series visibility is about to change. Sends series which changed, its owner chart. The current visible state of the series may be checked using the series.visible property.

Returning **false** from this event handler will cancel the change in visibility.

#### 5.4.7.3.4 onShowHint (inherited)

##### Definition

string **onShowHint**( [EJSC.Point](#) point, [EJSC.Series](#) series, [EJSC.Chart](#) chart, string hint\_string )

##### Description

Fired before the displaying the hint, this event allows the current hint string to be overridden. See [Text Replacement Options](#) for a list of the automatic substitutions available.

#### 5.4.7.4 Data Formats

The x and y parameters are required, label and userdata are both optional.

**XML** ([EJSC.XMLDataHandler](#), [EJSC.XMLStringDataHandler](#)):

Unused parameters may be omitted

##### Full:

```

    <graph>
      <plot>
        <point x="number or string" y="number or string" label="string"
userdata="string"/>
        <point x="number or string" y="number or string" label="string"
userdata="string"/>
      </plot>
    </graph>

```

##### Short:

```

    <G>
      <L>
        <P x="number or string" y="number or string" label="string"
userdata="string"/>
        <P x="number or string" y="number or string" label="string"
userdata="string"/>

```

```

        </L>
    </G>

```

**Compact:** The values parameter is formatted as CSV

```

<G>
    <L values="CSV string"/>
</G>

```

**Array** ([EJSC.ArrayDataHandler](#)):

```

[
    ["x", "y", "label", "userdata"],
    ["x", "y", "label", "userdata"]
]

```

Null should be used to skip unused parameters:

```

[
    ["x", "y", null, "userdata"],
    ["x", "y", null, "userdata"]
]

```

**CSV** ([EJSC.CSVFileDataHandler](#), [EJSC.CSVStringDataHandler](#)):

```
"x|y|label|userdata,x|y|label|userdata"
```

Null should be used to skip unused parameters

```
"x|y|null|userdata,x|y|null|userdata"
```

**JSON** ([EJSC.JSONFileDataHandler](#), [EJSC.JSONStringDataHandler](#)):

Unused parameters may be omitted.

```

[
    { "x": "number or string", "y": "number or
string", "label": "string", "userdata": "string" },
    { "x": "number or string", "y": "number or
string", "label": "string", "userdata": "string" }
]

```

#### 5.4.7.5 Text Replacement Options

String	Replaced With
[chart_title]	The title of the chart
[series_title]	The title of the series the selected point resides in
[xaxis]	The caption of the series x axis
[yaxis]	The caption of the series y axis
[x]	The preformatted X value of the point
[y]	The preformatted Y value of the point

[label]

The label property of the current point

## 5.4.8 EJSC.OpenHighLowCloseSeries

In order to use the OpenHighLowCloseSeries, the file EJSCChart\_Stock.js must be included in the HTML page after the inclusion of the standard EJSCChart.js file.

```
<head>
  <script type="text/javascript" src="/EJSCChart/EJSCChart.js"></script>
  <script type="text/javascript" src="/EJSCChart/EJSCChart_Stock.js"></script>
</head>
```

OpenHighLowCloseSeries renders its point data as vertical lines with optional horizontal "wicks" to express open and close values.

The constructor expects an instantiated [EJSC.DataHandler](#) descendant and an optional set of object properties.

### Constructor

```
EJSC.OpenHighLowCloseSeries( EJSC.DataHandler dataHandler [, object options] )
```

### Example

```
var mySeries = new EJSC.OpenHighLowCloseSeries(
  new EJSC.ArrayDataHandler([[1,10,1,2,9],[2,20,7,10,19]]),
  { title: "New OpenHighLowClose Series" }
);
```

### Defining Properties and Events

OpenHighLowCloseSeries properties may be set individually after the series has been created or in batch using the options parameter during instantiation.

#### Setting properties individually

```
var mySeries = new EJSC.OpenHighLowCloseSeries(new
EJSC.ArrayDataHandler([[1,10,1,2,9],[2,20,7,10,19]]));
mySeries.lineWidth = 1;
mySeries.title = "New OpenHighLowClose Series";
```

#### Setting properties in batch

```
var mySeries = new EJSC.OpenHighLowCloseSeries(
  new EJSC.ArrayDataHandler([[1,10,1,2,9],[2,20,7,10,9]]),
  { lineWidth: 1, title: "New OpenHighLowClose Series" }
);
```



### 5.4.8.1 Properties

#### 5.4.8.1.1 autosort (inherited)

##### Definition

boolean **autosort** = true

##### Description

Defines whether the series applies the appropriate sort to points prior to drawing. Drawing routines are generally optimized to accept data in a certain order and this property eliminates the need for the developer to worry about that order prior to sending data to the series. If set to false, the data must be properly ordered beforehand in order to ensure the series renders correctly.

**Note:** May not be applicable to all series (i.e. [EJSC.PieSeries](#), [EJSC.AnalogGaugeSeries](#))

#### 5.4.8.1.2 color (inherited)

##### Definition

string **color** = undefined

##### Description

Defines the series color. If left undefined, the color is pulled from the array of default colors stored in the chart.

**Note:** May not be applicable to all series (i.e. [EJSC.PieSeries](#))

#### 5.4.8.1.3 coloredLegend (inherited)

##### Definition

boolean **coloredLegend** = true

##### Description

Determines whether the legend text inherits the series color. Setting to false will allow the legend text to inherit its normal cascaded color. To modify this property after series creation see [EJSC.Series.setColoredLegend\(\)](#)

#### 5.4.8.1.4 delayLoad (inherited)

### EXPERIMENTAL

#### Definition

boolean **delayLoad** = true

#### Description

This property allows a series to force its data handler to begin data retrieval as soon as it is added to a chart. When set to false, the series will initiate data retrieval as soon as it is added to a chart. When true, the data retrieval is initialized when the series first needs to draw.

#### 5.4.8.1.5 gain

#### Definition

```
object gain = {  
    lineColor: "rgb(151,183,247)",  
    lineOpacity: 100  
}
```

#### Description

Defines the appearance of the points which represent a gain.

#### Example

```
var series = new EJSC.OpenHighLowCloseSeries(new EJSC.DataHandler(),  
    {  
        gain {  
            lineColor: "#006600",  
            lineOpacity: 50  
        }  
    }  
);
```



#### 5.4.8.1.6 hint\_string (inherited)

##### Definition

string `hint_string` = undefined

##### Description

This property may be used to define a series specific string to be used when displaying point hints. This string may contain replacement indicators (see [Text Replacement Options](#)). When left undefined, a default hint defined by the series type will be displayed.

##### Example

*>> Modify the default hint to include custom text.*

```
var series = new EJSC.LineSeries(new EJSC.ArrayDataHandler(),
    {
        hint_string: "My Custom Hint<br /><strong>X: </strong>[x]<br/><strong>Y:
</strong>[y]"
    }
);
```

#### 5.4.8.1.7 intervalOffset

##### Definition

float `intervalOffset` = 0.8

##### Description

Defines the spacing between the points as a percentage of 1. 1 = points take up the entire width available, with their edges touching.

##### Example

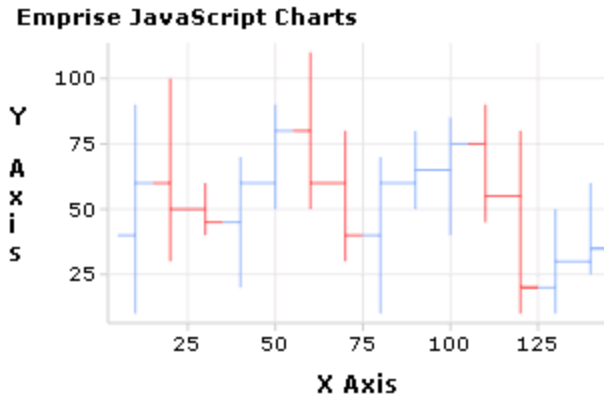
```
var chart = new EJSC.Chart("chart");
var series = chart.addSeries(new EJSC.OpenHighLowCloseSeries(
    data,
```



```

    {
      intervalOffset: 1
    }
  ));

```



#### 5.4.8.1.8 legendsVisible (inherited)

##### Definition

boolean **legendsVisible** = true

##### Description

Determines whether the legend item associated with the series appears in the legend window. Use the `Series.showLegend` and `Series.hideLegend` methods to control legend item visibility after series creation.

#### 5.4.8.1.9 lineOpacity (inherited)

##### Definition

integer **lineOpacity** = 100

##### Description

Determines the line opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see [EJSC.Series.setLineOpacity\(\)](#)

#### 5.4.8.1.10 lineWidth (inherited)

##### Definition

integer **lineWidth** = 1

##### Description

Defines the width of the line connecting series points.

## 5.4.8.1.11 loss

**Definition**

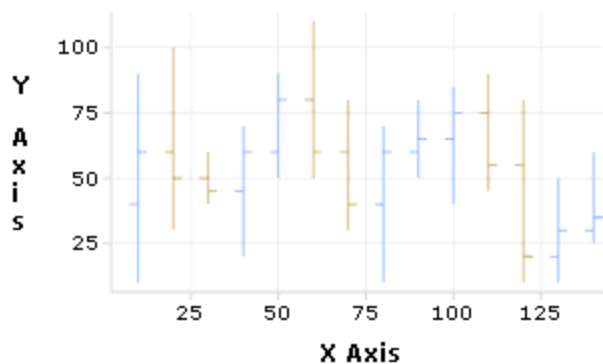
```
object loss = {  
    lineColor: "rgb(249,95,95)",  
    lineOpacity: 100  
}
```

**Description**

Defines the appearance of the points which represent a loss.

**Example**

```
var series = new EJSC.OpenHighLowCloseSeries(new EJSC.DataHandler(),  
    {  
        loss {  
            lineColor: "#996600",  
            lineOpacity: 50  
        }  
    }  
);
```

**Emprise JavaScript Charts**

## 5.4.8.1.12 opacity (inherited)

**Definition**

```
integer opacity = 50
```

**Description**

Determines the fill opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see [EJSC.Series.setOpacity\(\)](#)

## 5.4.8.1.13 title (inherited)

**Definition**

string **title** = "Series <index>"

**Description**

Defines the series title used in hints and legend display.

**Note:** The default title of Series <index> (where <index> equals the current position in the series array for its owner chart) is assigned when a series is added to a chart using [addSeries](#). This default title is only applied if the title is blank (i.e. "") at the time the addSeries method is called.

## 5.4.8.1.14 visible (inherited)

**Definition**

boolean **visible** = true

**Description**

Defines whether the series is visible and can draw on the chart

## 5.4.8.1.15 x\_axis (inherited)

**Definition**

string **x\_axis** = "bottom"

**Description**

Defines the horizontal axis to use for X values. Valid options are "top" or "bottom".

## 5.4.8.1.16 x\_axis\_formatter (inherited)

**Definition**

string **x\_axis\_formatter** = undefined

**Description**

Defines the formatter that will be used to format hints for the X values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:

[EJSC.DateFormatter](#)

[EJSC.NumberFormatter](#)

5.4.8.1.17 `y_axis` (inherited)**Definition**

string `y_axis` = "left"

**Description**

Defines the vertical axis to use for Y values. Valid options are "left" or "right".

5.4.8.1.18 `y_axis_formatter` (inherited)**Definition**

string `y_axis_formatter` = undefined

**Description**

Defines the formatter that will be used to format hints for the Y values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:

[EJSC.DateFormatter](#)

[EJSC.NumberFormatter](#)

**5.4.8.2 Methods**5.4.8.2.1 `findClosestPoint` (inherited)**Definition**

[EJSC.Point](#) `findClosestPoint`( integer pixel )

**Description**

Returns the point object closest to the pixel coordinate given or null if no points are found.

5.4.8.2.2 `getDataHandler` (inherited)**Definition**

[EJSC.DataHandler](#) `getDataHandler`( )

**Description**

Returns the [EJSC.DataHandler](#) descendant currently assigned to the series. This is not applicable to all series and will return null if a data handler has not been defined or is not used.

5.4.8.2.3 `getVisibility` (inherited)**Definition**

boolean **getVisibility**( )

#### **Description**

Returns a boolean indicating the series current visible state

#### 5.4.8.2.4 hide (inherited)

#### **Definition**

void **hide**( )

#### **Description**

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already hidden.

#### 5.4.8.2.5 hideLegend (inherited)

#### **Definition**

void **hideLegend**( )

#### **Description**

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

#### 5.4.8.2.6 reload (inherited)

#### **Definition**

void **reload**( )

#### **Description**

Triggers a series reload if the functionality is defined in a child class. This may involve retrieving data again or simply recalculating. While the reload() method exists in the base Series class, implementation of the protected methods triggered by calling reload() is at the discretion of the child class.

#### 5.4.8.2.7 setColor (inherited)

#### **Definition**

void **setColor**( string color )

#### **Description**

Changes the series color and causes the chart to redraw.

## 5.4.8.2.8 setColoredLegend (inherited)

**Definition**

void **setColoredLegend**( boolean coloredLegend )

**Description**

Sets the [EJSC.Series.coloredLegend](#) property and updates the legend to reflect the change.

## 5.4.8.2.9 setDataHandler (inherited)

**Definition**

void **setDataHandler**( [EJSC.DataHandler](#) dataHandler, boolean reload )

**Description**

Updates the series data handler and triggers a data reload if reload parameter is **true**. This method may not be implemented in all child classes.

## 5.4.8.2.10 setLineOpacity (inherited)

**Definition**

void **setLineOpacity**( integer opacity )

**Description**

Sets the [EJSC.Series.lineOpacity](#) property and redraws the series to reflect the change.

## 5.4.8.2.11 setLineWidth (inherited)

**Definition**

void **setLineWidth**( integer width )

**Description**

Updates the [lineWidth](#) property and redraws the chart.

## 5.4.8.2.12 setOpacity (inherited)

**Definition**

void **setOpacity**( integer opacity )

**Description**

Sets the [EJSC.Series.opacity](#) property and redraws the series to reflect the change.

## 5.4.8.2.13 setTitle (inherited)

**Definition**

void [setTitle](#)( string title )

**Description**

Changes the series title and update the legend caption (if applicable).

**Note:** this will not cause updates to a hint which is visible at the time it is called, though the next time a hint is rendered it will show the new property value.

## 5.4.8.2.14 show (inherited)

**Definition**

void [show](#)( )

**Description**

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already visible.

## 5.4.8.2.15 showLegend (inherited)

**Definition**

void [showLegend](#)( )

**Description**

Changes the series legend item visible state.

No effect if the series legend item is already visible.

**5.4.8.3 Events**

## 5.4.8.3.1 onAfterDataAvailable (inherited)

**Definition**

boolean [onAfterDataAvailable](#)( [EJSC.Chart](#) chart, [EJSC.Series](#) series )

**Description**

Fired after the series has processed its data and before the chart is told to redraw. Returning **false** from this event handler will cancel redrawing the chart.

## 5.4.8.3.2 onAfterVisibilityChange (inherited)

**Definition**

boolean **onAfterVisibilityChange**( [EJSC.Series](#) series, [EJSC.Chart](#) chart, boolean visible )

**Description**

Fired when the series visibility changes. Sends series which changed, its owner chart and its new visible property value.

Returning **false** from this event handler will cancel redrawing the chart.

## 5.4.8.3.3 onBeforeVisibilityChange (inherited)

**Definition**

boolean **onBeforeVisibilityChange**( [EJSC.Series](#) series, [EJSC.Chart](#) chart )

**Description**

Fired before the series visibility is about to change. Sends series which changed, its owner chart. The current visible state of the series may be checked using the series.visible property.

Returning **false** from this event handler will cancel the change in visibility.

## 5.4.8.3.4 onShowHint (inherited)

**Definition**

string **onShowHint**( [EJSC.Point](#) point, [EJSC.Series](#) series, [EJSC.Chart](#) chart, string hint\_string )

**Description**

Fired before the displaying the hint, this event allows the current hint string to be overridden. See [Text Replacement Options](#) for a list of the automatic substitutions available.

## 5.4.8.4 Data Formats

**XML** ([EJSC.XMLDataHandler](#), [EJSC.XMLStringDataHandler](#)):

Unused parameters may be omitted

**Full:**

```

<graph>
  <plot>
    <point x="number or string" high="number" low="number"
open="number" close="number" label="string" userdata="string"/>
    <point x="number or string" high="number" low="number"
open="number" close="number" label="string" userdata="string"/>

```



```

        </plot>
    </graph>

```

Short:

```

    <G>
        <L>
            <P x="number or string" high="number" low="number"
open="number" close="number" label="string" userdata="string"/>
            <P x="number or string" high="number" low="number"
open="number" close="number" label="string" userdata="string"/>
        </L>
    </G>

```

**Compact:** The values parameter is formatted as CSV

```

    <G>
        <L values="CSV string"/>
    </G>

```

**Array** ([EJSC.ArrayDataHandler](#)):

```

[
    ["x", "high", "low", "open", "close", "label", "userdata"],
    ["x", "high", "low", "open", "close", "label", "userdata"]
]

```

Null should be used to skip unused parameters:

```

[
    ["x", null, null, "open", "close", null, "userdata"],
    ["x", null, null, "open", "close", null, "userdata"]
]

```

**CSV** ([EJSC.CSVFileDataHandler](#), [EJSC.CSVStringDataHandler](#)):

```
"x|high|low|open|close|label|userdata,x|high|low|open|close|label|userdata"
```

Null should be used to skip unused parameters

```
"x|null|null|open|close|null|userdata,x|null|null|open|close|null|userdata"
```

**JSON** ([EJSC.JSONFileDataHandler](#), [EJSC.JSONStringDataHandler](#)):

Unused parameters may be omitted.

```

[
    {"x":"number or string", "high":"number", "low":"number",
"open":"number", "close":"number", label:"string","userdata":"string"},
    {"x":"number or string", "high":"number", "low":"number",

```

```
"open": "number", "close": "number", label: "string", "userdata": "string"}
]
```

#### 5.4.8.5 Text Replacement Options (COPY)

String	Replaced With
[chart_title]	The title of the chart
[series_title]	The title of the series the selected point resides in
[xaxis]	The caption of the series x axis
[yaxis]	The caption of the series y axis
[x]	The preformatted X value of the point
[high]	The preformatted high value of the point
[low]	The preformatted low value of the point
[open]	The preformatted open value of the point
[close]	The preformatted close value of the point
[label]	The label property of the current point

#### 5.4.9 EJSC.PieSeries

PieSeries is rendered by drawing slices to form an ellipse. Each slice represents a percentage of the total of the sum of all point values in the dataset.

The constructor expects an instantiated [EJSC.DataHandler](#) descendant and an optional set of object properties.

##### Constructor

```
EJSC.PieSeries( EJSC.DataHandler dataHandler [, object options] )
```

##### Example

```
var mySeries = new EJSC.PieSeries(
    new EJSC.ArrayDataHandler([[10,"Label 1"],[20,"Label 2"],[120,"Label3"]]),
    { title: "New Pie Series" }
);
```

##### Defining Properties and Events

PieSeries properties may be set individually after the series has been created or in batch using the options parameter during instantiation.

##### Setting properties individually

```
var mySeries = new EJSC.PieSeries(new EJSC.ArrayDataHandler([[10,"Label
1"],[20,"Label 2"],[120,"Label 3"]]]);
mySeries.title = "New Pie Series";
```

##### Setting properties in batch

```
var mySeries = new EJSC.PieSeries(
    ArrayDataHandler([[10,"Label 1"],[20,"Label 2"],[120,"Label 3"]]),
    { title: "New Pie Series" }
);
```

### 5.4.9.1 Properties

#### 5.4.9.1.1 defaultColors

##### Definition

array **defaultColors** = [EJSC.DefaultPieColors](#)

##### Description

Defines the pool of default colors available for pie pieces.

#### 5.4.9.1.2 delayLoad (inherited)

##### **EXPERIMENTAL**

##### Definition

boolean **delayLoad** = true

##### Description

This property allows a series to force its data handler to begin data retrieval as soon as it is added to a chart. When set to false, the series will initiate data retrieval as soon as it is added to a chart. When true, the data retrieval is initialized when the series first needs to draw.

#### 5.4.9.1.3 height

##### Definition

string **height** = "100%"

##### Description

Defines the total height of the pie series. This may be specified in percent of the chart as shown above as the default value, or in exact pixels by specifying a number (i.e. height = 150).

#### 5.4.9.1.4 hint\_string (inherited)

##### Definition

string **hint\_string** = undefined

##### Description

This property may be used to define a series specific string to be used when displaying point hints. This string may contain replacement indicators (see [Text Replacement Options](#)). When left undefined, a default hint defined by the series type will be displayed.

##### Example

>> *Modify the default hint to include custom text.*

```
var series = new EJSC.LineSeries(new EJSC.ArrayDataHandler(),
    {
        hint_string: "My Custom Hint<br /><strong>X: </strong>[x]<br/><strong>Y:
</strong>[y]"
    }
);
```

#### 5.4.9.1.5 legendsVisible (inherited)

##### Definition

boolean **legendsVisible** = true

##### Description

Determines whether the legend item associated with the series appears in the legend window. Use the `Series.showLegend` and `Series.hideLegend` methods to control legend item visibility after series creation.

#### 5.4.9.1.6 lineOpacity (inherited)

##### Definition

integer **lineOpacity** = 100

##### Description

Determines the line opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see [EJSC.Series.setLineOpacity\(\)](#)

#### 5.4.9.1.7 lineWidth (inherited)

##### Definition

integer **lineWidth** = 1

##### Description

Defines the width of the line connecting series points.

#### 5.4.9.1.8 opacity (inherited)

##### Definition

integer **opacity** = 50

##### Description

Determines the fill opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see [EJSC.Series.setOpacity\(\)](#)

## 5.4.9.1.9 position

**Definition**

string **position** = "center"

**Description**

Defines the quadrant of the chart that the pie will appear in.

**Quadrants:**

- topLeft
- topCenter
- topRight
- leftCenter
- center
- rightCenter
- bottomLeft
- bottomCenter
- bottomRight

## 5.4.9.1.10 title (inherited)

**Definition**

string **title** = "Series <index>"

**Description**

Defines the series title used in hints and legend display.

**Note:** The default title of Series <index> (where <index> equals the current position in the series array for its owner chart) is assigned when a series is added to a chart using [addSeries](#). This default title is only applied if the title is blank (i.e. "") at the time the addSeries method is called.

## 5.4.9.1.11 total\_value

**Definition**

integer **total\_value** = undefined

**Description**

Defines the total value (sum) of all pie pieces in the series. If left undefined, this value will be calculated automatically by adding all the piece values when the series data is loaded.

This property is only applicable during series creation. To determine or modify the total value once the series has been created use the [getTotalValue](#) and [setTotalValue](#) methods.

To force the chart to recalculate the total value based on actual series data, use the [resetTotalValue](#) method.

## 5.4.9.1.12 treeLegend

**Definition**

boolean **treeLegend** = true

**Description**

Defines whether to display each pie piece individually in the legend.

## 5.4.9.1.13 visible (inherited)

**Definition**

boolean **visible** = true

**Description**

Defines whether the series is visible and can draw on the chart

## 5.4.9.1.14 width

**Definition**

string **width** = "100%"

**Description**

Defines the total width of the pie series. This may be specified in percent of the chart as shown above as the default value, or in exact pixels by specifying a number (i.e. width = 150).

## 5.4.9.1.15 x\_axis\_formatter (inherited)

**Definition**

string **x\_axis\_formatter** = undefined

**Description**

Defines the formatter that will be used to format hints for the X values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:

[EJSC.DateFormatter](#)

[EJSC.NumberFormatter](#)

**5.4.9.2 Methods**

## 5.4.9.2.1 findCenter

**Definition**

object **findCenter**( [EJSC.PiePoint](#) point )

The object returned is formatted as:

```
{
  x: integer,
  y: integer
}
```

### Description

This method takes a pie point (see [getPoints](#)) and returns an object containing the x and y screen coordinates of the piece's center.

#### 5.4.9.2.2 findCenterOfCurve

### Definition

object **findCenterOfCurve**( [EJSC.PiePoint](#) point )

The object returned is formatted as:

```
{
  x: integer,
  y: integer
}
```

### Description

This method takes a pie point (see [getPoints](#)) and returns an object containing the x and y screen coordinates of the center of the piece's arc.

#### 5.4.9.2.3 getDataHandler (inherited)

### Definition

[EJSC.DataHandler](#) **getDataHandler**( )

### Description

Returns the [EJSC.DataHandler](#) descendant currently assigned to the series. This is not applicable to all series and will return null if a data handler has not been defined or is not used.

#### 5.4.9.2.4 getPoints

### Definition

array **getPoints**()

### Description

This method returns an array of all [EJSC.PiePoint](#) objects in the series.

## 5.4.9.2.5 getTotalValue

**Definition**

integer [getTotalValue](#)( )

**Description**

Returns the current total value of the pie series. If the [total\\_value](#) property was set during creation or the [setTotalValue](#) has been called, the static value will be returned. If the series is set to calculate the total value based on the series data, the dynamic value will be returned.

## 5.4.9.2.6 getVisibility (inherited)

**Definition**

boolean [getVisibility](#)( )

**Description**

Returns a boolean indicating the series current visible state

## 5.4.9.2.7 hide (inherited)

**Definition**

void [hide](#)( )

**Description**

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already hidden.

## 5.4.9.2.8 hideLegend (inherited)

**Definition**

void [hideLegend](#)( )

**Description**

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

## 5.4.9.2.9 reload (inherited)

**Definition**

void [reload](#)( )



### Description

Triggers a series reload if the functionality is defined in a child class. This may involve retrieving data again or simply recalculating. While the reload() method exists in the base Series class, implementation of the protected methods triggered by calling reload() is at the discretion of the child class.

#### 5.4.9.2.10 resetTotalValue

### Definition

```
void resetTotalValue( boolean redraw )
```

### Description

This method causes the series to recalculate its total value based on the actual series data and optionally redraws the series to reflect the change.

To retrieve the total value after this method is called, use [getTotalValue](#).

#### 5.4.9.2.11 setDataHandler (inherited)

### Definition

```
void setDataHandler( EJSC.DataHandler dataHandler, boolean reload )
```

### Description

Updates the series data handler and triggers a data reload if reload parameter is **true**. This method may not be implemented in all child classes.

#### 5.4.9.2.12 setDefaultColors

### Definition

```
void setDefaultColors( array colors, boolean reload )
```

### Description

Set the array of default colors available for pie pieces.

If reload = **true** the pie pieces (if loaded) will pull their colors from the given color array and the chart will be redrawn.

If [onPieceNeedsColor](#) is assigned, this method will ignore that event and load the colors from the array.

### Example

```
var colors = [  
    "rgb(0,0,0)",           // black  
    "rgb(255,0,0)",        // red
```

```
        "rgb("0,255,0)",      // green
        "rgb("0,0,255)",     // blue
        "rgb("255,255,255)"  // white
    ];

    myPieSeries.setDefaultColors(colors, true);
```

#### 5.4.9.2.13 setLineWidth (inherited)

##### Definition

void [setLineWidth](#)( integer width )

##### Description

Updates the [lineWidth](#) property and redraws the chart.

#### 5.4.9.2.14 setOpacity (inherited)

##### Definition

void [setOpacity](#)( integer opacity )

##### Description

Sets the [EJSC.Series.opacity](#) property and redraws the series to reflect the change.

#### 5.4.9.2.15 setTitle (inherited)

##### Definition

void [setTitle](#)( string title )

##### Description

Changes the series title and update the legend caption (if applicable).

**Note:** this will not cause updates to a hint which is visible at the time it is called, though the next time a hint is rendered it will show the new property value.

#### 5.4.9.2.16 setTotalValue

##### Definition

void [setTotalValue](#)( integer value, boolean redraw )

##### Description

Sets the total value of the pie series and optionally redraws the series to reflect the change.

To force the chart to calculate its total value based on the actual series data, call [resetTotalValue](#).

#### 5.4.9.2.17 show (inherited)

##### Definition

void [show](#)( )

##### Description

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already visible.

#### 5.4.9.2.18 showLegend (inherited)

##### Definition

void [showLegend](#)( )

##### Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

### 5.4.9.3 Events

#### 5.4.9.3.1 onAfterDataAvailable (inherited)

##### Definition

boolean [onAfterDataAvailable](#)( [EJSC.Chart](#) chart, [EJSC.Series](#) series )

##### Description

Fired after the series has processed its data and before the chart is told to redraw. Returning **false** from this event handler will cancel redrawing the chart.

#### 5.4.9.3.2 onAfterVisibilityChange (inherited)

##### Definition

boolean [onAfterVisibilityChange](#)( [EJSC.Series](#) series, [EJSC.Chart](#) chart, boolean visible )

##### Description

Fired when the series visibility changes. Sends series which changed, its owner chart and its new visible property value.

Returning **false** from this event handler will cancel redrawing the chart.

## 5.4.9.3.3 onBeforeVisibilityChange (inherited)

**Definition**

boolean **onBeforeVisibilityChange**( [EJSC.Series](#) series, [EJSC.Chart](#) chart )

**Description**

Fired before the series visibility is about to change. Sends series which changed, its owner chart. The current visible state of the series may be checked using the series.visible property.

Returning **false** from this event handler will cancel the change in visibility.

## 5.4.9.3.4 onPieceNeedsColor

**Definition**

boolean **onPieceNeedsColor**( [EJSC.PiePoint](#) point, [EJSC.PieSeries](#) series, [EJSC.Chart](#) chart )

**Description**

If assigned, this event is triggered for each piece in a pie series immediately before it pulls a color from the default colors array. The event provides the [EJSC.PiePoint](#) object which represents the piece which needs a color, the [EJSC.PieSeries](#) which owns the piece and the [EJSC.Chart](#) which owns the series. Return a color string formatted as "rgb(<red value>, <green value>, <blue value>)", i.e. Orange would be "rgb(237,173,0)"

## 5.4.9.3.5 onShowHint (inherited)

**Definition**

string **onShowHint**( [EJSC.Point](#) point, [EJSC.Series](#) series, [EJSC.Chart](#) chart, string hint\_string )

**Description**

Fired before the displaying the hint, this event allows the current hint string to be overridden. See [Text Replacement Options](#) for a list of the automatic substitutions available.

**5.4.9.4 Data Formats**

The x parameter is required, label and userdata are both optional.

**XML** ([EJSC.XMLDataHandler](#), [EJSC.XMLStringDataHandler](#)):

Unused parameters may be omitted

**Full:**

```
<graph>
  <plot>
    <point x="number or string" label="string" userdata="string"/>
    <point x="number or string" label="string" userdata="string"/>
```

```

    </plot>
  </graph>

```

Short:

```

  <G>
    <L>
      <P x="number or string" label="string" userdata="string"/>
      <P x="number or string" label="string" userdata="string"/>
    </L>
  </G>

```

**Compact:** The values parameter is formatted as CSV

```

  <G>
    <L values="CSV string"/>
  </G>

```

**Array** ([EJSC.ArrayDataHandler](#)):

```

[
  ["x", "label", "userdata"],
  ["x", "label", "userdata"]
]

```

Null should be used to skip unused parameters:

```

[
  ["x", null, "userdata"],
  ["x", null, "userdata"]
]

```

**CSV** ([EJSC.CSVFileDataHandler](#), [EJSC.CSVStringDataHandler](#)):

```
"x|label|userdata,x|label|userdata"
```

Null should be used to skip unused parameters

```
"x|null|userdata,x|null|userdata"
```

**JSON** ([EJSC.JSONFileDataHandler](#), [EJSC.JSONStringDataHandler](#)):

Unused parameters may be omitted.

```

[
  { "x": "number or string", "label": "string", "userdata": "string" },
  { "x": "number or string", "label": "string", "userdata": "string" }
]

```

### 5.4.9.5 Text Replacement Options

String	Replaced With
[chart_title]	The title of the chart
[series_title]	The title of the series the selected point resides in
[x]	The preformatted value of the point
[total]	The total value represented by the sum of all piece values
[percent]	The percent of the total value the current point represents
[label]	The label property of the current point

### 5.4.10 EJSC.ScatterSeries

ScatterSeries is rendered by drawing a styled point for each x,y coordinate in the dataset.

The constructor expects an instantiated [EJSC.DataHandler](#) descendant and an optional set of object properties.

#### Constructor

```
EJSC.ScatterSeries( EJSC.DataHandler dataHandler [, object options] )
```

#### Example

```
var mySeries = new EJSC.ScatterSeries(
    new EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,3],[5,2],[6,4]]),
    { title: "New Scatter Series" }
);
```

#### Defining Properties and Events

ScatterSeries properties may be set individually after the series has been created or in batch using the options parameter during instantiation.

##### Setting properties individually

```
var mySeries = new EJSC.ScatterSeries(new
EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,3],[5,2],[6,4]]));
mySeries.pointSize = 3;
mySeries.pointStyle = "diamond";
mySeries.title = "New Scatter Series";
```

##### Setting properties in batch

```
var mySeries = new EJSC.ScatterSeries(
    new EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,3],[5,2],[6,4]]),
    { pointSize: 3, pointStyle: "diamond", title: "New Scatter Series" }
);
```

### 5.4.10.1 Properties

#### 5.4.10.1.1 autosort (inherited)

##### Definition

boolean **autosort** = true

##### Description

Defines whether the series applies the appropriate sort to points prior to drawing. Drawing routines are generally optimized to accept data in a certain order and this property eliminates the need for the developer to worry about that order prior to sending data to the series. If set to false, the data must be properly ordered beforehand in order to ensure the series renders correctly.

**Note:** May not be applicable to all series (i.e. [EJSC.PieSeries](#), [EJSC.AnalogGaugeSeries](#))

#### 5.4.10.1.2 color (inherited)

##### Definition

string **color** = undefined

##### Description

Defines the series color. If left undefined, the color is pulled from the array of default colors stored in the chart.

**Note:** May not be applicable to all series (i.e. [EJSC.PieSeries](#))

#### 5.4.10.1.3 coloredLegend (inherited)

##### Definition

boolean **coloredLegend** = true

##### Description

Determines whether the legend text inherits the series color. Setting to false will allow the legend text to inherit its normal cascaded color. To modify this property after series creation see [EJSC.Series.setColoredLegend\(\)](#)

#### 5.4.10.1.4 delayLoad (inherited)

##### **EXPERIMENTAL**

##### Definition

boolean **delayLoad** = true

##### Description

This property allows a series to force its data handler to begin data retrieval as soon as it is added to a

chart. When set to false, the series will initiate data retrieval as soon as it is added to a chart. When true, the data retrieval is initialized when the series first needs to draw.

#### 5.4.10.1.5 hint\_string (inherited)

##### Definition

string **hint\_string** = undefined

##### Description

This property may be used to define a series specific string to be used when displaying point hints. This string may contain replacement indicators (see [Text Replacement Options](#)). When left undefined, a default hint defined by the series type will be displayed.

##### Example

*>> Modify the default hint to include custom text.*

```
var series = new EJSC.LineSeries(new EJSC.ArrayDataHandler(),
    {
        hint_string: "My Custom Hint<br /><strong>X: </strong>[x]<br/><strong>Y:
</strong>[y]"
    }
);
```

#### 5.4.10.1.6 legendIsVisible (inherited)

##### Definition

boolean **legendIsVisible** = true

##### Description

Determines whether the legend item associated with the series appears in the legend window. Use the `Series.showLegend` and `Series.hideLegend` methods to control legend item visibility after series creation.

#### 5.4.10.1.7 lineOpacity (inherited)

##### Definition

integer **lineOpacity** = 100

##### Description

Determines the line opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see [EJSC.Series.setLineOpacity\(\)](#)

#### 5.4.10.1.8 lineWidth (inherited)

##### Definition



integer **lineWidth** = 1

#### **Description**

Defines the width of the line connecting series points.

5.4.10.1.9 opacity (inherited)

#### **Definition**

integer **opacity** = 50

#### **Description**

Determines the fill opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see [EJSC.Series.setOpacity\(\)](#)

5.4.10.1.10 pointSize

#### **Definition**

integer **pointSize** = 4

#### **Description**

Defines the size of the point to be drawn. Point size has no effect on point selection. In order to increase the distance from the actual coordinate that a click will select or hover a point, see [EJSC.Chart.proximity\\_snap](#)

5.4.10.1.11 pointStyle

#### **Definition**

string **pointStyle** = "triangle"

Styles:

- triangle
- box
- circle
- diamond

#### **Description**

Defines the shape of the point to be drawn. Point shape has no effect on point selection. In order to increase the distance from the actual coordinate that a click will select or hover a point, see [EJSC.Chart.proximity\\_snap](#)

5.4.10.1.12 title (inherited)

#### **Definition**

string **title** = "Series <index>"

### Description

Defines the series title used in hints and legend display.

**Note:** The default title of Series <index> (where <index> equals the current position in the series array for its owner chart) is assigned when a series is added to a chart using [addSeries](#). This default title is only applied if the title is blank (i.e. "") at the time the addSeries method is called.

5.4.10.1.13 visible (inherited)

### Definition

boolean **visible** = true

### Description

Defines whether the series is visible and can draw on the chart

5.4.10.1.14 x\_axis (inherited)

### Definition

string **x\_axis** = "bottom"

### Description

Defines the horizontal axis to use for X values. Valid options are "top" or "bottom".

5.4.10.1.15 x\_axis\_formatter (inherited)

### Definition

string **x\_axis\_formatter** = undefined

### Description

Defines the formatter that will be used to format hints for the X values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:

[EJSC.DateFormatter](#)

[EJSC.NumberFormatter](#)

5.4.10.1.16 y\_axis (inherited)

### Definition

string **y\_axis** = "left"

**Description**

Defines the vertical axis to use for Y values. Valid options are "left" or "right".

5.4.10.1.17 `y_axis_formatter` (inherited)

**Definition**

string `y_axis_formatter` = undefined

**Description**

Defines the formatter that will be used to format hints for the Y values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:

[EJSC.DateFormatter](#)

[EJSC.NumberFormatter](#)

**5.4.10.2 Methods**

5.4.10.2.1 `getDataHandler` (inherited)

**Definition**

EJSC.DataHandler `getDataHandler`( )

**Description**

Returns the EJSC.DataHandler descendant currently assigned to the series. This is not applicable to all series and will return null if a data handler has not been defined or is not used.

5.4.10.2.2 `getVisibility` (inherited)

**Definition**

boolean `getVisibility`( )

**Description**

Returns a boolean indicating the series current visible state

5.4.10.2.3 `hide` (inherited)

**Definition**

void `hide`( )

**Description**

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already hidden.

#### 5.4.10.2.4 hideLegend (inherited)

##### Definition

void [hideLegend](#)( )

##### Description

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

#### 5.4.10.2.5 reload (inherited)

##### Definition

void [reload](#)( )

##### Description

Triggers a series reload if the functionality is defined in a child class. This may involve retrieving data again or simply recalculating. While the reload() method exists in the base Series class, implementation of the protected methods triggered by calling reload() is at the discretion of the child class.

#### 5.4.10.2.6 setColor (inherited)

##### Definition

void [setColor](#)( string color )

##### Description

Changes the series color and causes the chart to redraw.

#### 5.4.10.2.7 setColoredLegend (inherited)

##### Definition

void [setColoredLegend](#)( boolean coloredLegend )

##### Description

Sets the [EJSC.Series.coloredLegend](#) property and updates the legend to reflect the change.

#### 5.4.10.2.8 setDataHandler (inherited)

##### Definition

void [setDataHandler](#)( [EJSC.DataHandler](#) dataHandler, boolean reload )

**Description**

Updates the series data handler and triggers a data reload if reload parameter is **true**. This method may not be implemented in all child classes.

## 5.4.10.2.9 setLineWidth (inherited)

**Definition**

void **setLineWidth**( integer width )

**Description**

Updates the [lineWidth](#) property and redraws the chart.

## 5.4.10.2.10 setOpacity (inherited)

**Definition**

void **setOpacity**( integer opacity )

**Description**

Sets the [EJSC.Series.opacity](#) property and redraws the series to reflect the change.

## 5.4.10.2.11 setPointStyle

**Definition**

void **setPointStyle**( integer size, string style )

**Description**

Updates the [pointSize](#) and/or [pointStyle](#) and redraws the chart. Size or style update may be avoided by sending undefined.

See [EJSC.ScatterSeries.pointStyle](#) for a list of available styles.

**Example**

*>> Update the size and style of scatter series points*

```
mySeries.setPointStyle( 5, "diamond" );
```

*>> Update only the style of scatter series points*

```
mySeries.setPointStyle( undefined, "box" );
```

## 5.4.10.2.12 setTitle (inherited)

**Definition**

void **setTitle**( string title )

#### **Description**

Changes the series title and update the legend caption (if applicable).

**Note:** this will not cause updates to a hint which is visible at the time it is called, though the next time a hint is rendered it will show the new property value.

#### 5.4.10.2.13 show (inherited)

#### **Definition**

void **show**( )

#### **Description**

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already visible.

#### 5.4.10.2.14 showLegend (inherited)

#### **Definition**

void **showLegend**( )

#### **Description**

Changes the series legend item visible state.

No effect if the series legend item is already visible.

### 5.4.10.3 Events

#### 5.4.10.3.1 onAfterDataAvailable (inherited)

#### **Definition**

boolean **onAfterDataAvailable**( [EJSC.Chart](#) chart, [EJSC.Series](#) series )

#### **Description**

Fired after the series has processed its data and before the chart is told to redraw. Returning **false** from this event handler will cancel redrawing the chart.

#### 5.4.10.3.2 onAfterVisibilityChange (inherited)

#### **Definition**

boolean **onAfterVisibilityChange**( [EJSC.Series](#) series, [EJSC.Chart](#) chart, boolean visible )

**Description**

Fired when the series visibility changes. Sends series which changed, its owner chart and its new visible property value.

Returning **false** from this event handler will cancel redrawing the chart.

## 5.4.10.3.3 onBeforeVisibilityChange (inherited)

**Definition**

boolean **onBeforeVisibilityChange**( [EJSC.Series](#) series, [EJSC.Chart](#) chart )

**Description**

Fired before the series visibility is about to change. Sends series which changed, its owner chart. The current visible state of the series may be checked using the series.visible property.

Returning **false** from this event handler will cancel the change in visibility.

## 5.4.10.3.4 onShowHint (inherited)

**Definition**

string **onShowHint**( [EJSC.Point](#) point, [EJSC.Series](#) series, [EJSC.Chart](#) chart, string hint\_string )

**Description**

Fired before the displaying the hint, this event allows the current hint string to be overridden. See [Text Replacement Options](#) for a list of the automatic substitutions available.

**5.4.10.4 Data Formats**

The x and y parameters are required, label and userdata are both optional.

**XML** ([EJSC.XMLDataHandler](#), [EJSC.XMLStringDataHandler](#)):

Unused parameters may be omitted

**Full:**

```

    <graph>
      <plot>
        <point x="number or string" y="number or string" label="string"
userdata="string"/>
        <point x="number or string" y="number or string" label="string"
userdata="string"/>
      </plot>
    </graph>

```

**Short:**

```

    <G>
        <L>
            <P x="number or string" y="number or string" label="string"
userdata="string"/>
            <P x="number or string" y="number or string" label="string"
userdata="string"/>
        </L>
    </G>

```

**Compact:** The values parameter is formatted as CSV

```

    <G>
        <L values="CSV string"/>
    </G>

```

**Array** ([EJSC.ArrayDataHandler](#)):

```

[
    ["x", "y", "label", "userdata"],
    ["x", "y", "label", "userdata"]
]

```

Null should be used to skip unused parameters:

```

[
    ["x", "y", null, "userdata"],
    ["x", "y", null, "userdata"]
]

```

**CSV** ([EJSC.CSVFileDataHandler](#), [EJSC.CSVStringDataHandler](#)):

```
"x|y|label|userdata,x|y|label|userdata"
```

Null should be used to skip unused parameters

```
"x|y|null|userdata,x|y|null|userdata"
```

**JSON** ([EJSC.JSONFileDataHandler](#), [EJSC.JSONStringDataHandler](#)):

Unused parameters may be omitted.

```

[
    {"x":"number or string","y":"number or
string","label":"string","userdata":"string"},
    {"x":"number or string","y":"number or
string","label":"string","userdata":"string"}
]

```



### 5.4.10.5 Text Replacement Options

String	Replaced With
[chart_title]	The title of the chart
[series_title]	The title of the series the selected point resides in
[xaxis]	The caption of the series x axis
[yaxis]	The caption of the series y axis
[x]	The preformatted X value of the point
[y]	The preformatted Y value of the point
[label]	The label property of the current point

## 5.4.11 EJSC.StackedBarSeries

Enter topic text here.

### 5.4.11.1 Properties

#### 5.4.11.1.1 groupedBars (inherited)

#### Definition

boolean **groupedBars** = true

#### Description

Defines whether the bars from different bar series in a single chart are grouped or overlaid.

**NOTE:** Only the first bar series added to the chart may use this property to affect the grouped/overlay display. To change the style after the first bar series has been added, use the [EJSC.BarSeries.setGroupedBars](#) from any bar series in the chart. Currently you cannot mix both overlaid and grouped bars within the same chart.

#### Example

>> *Make the bar series overlay*

```
var chart = new EJSC.Chart("chart");
var bar1 = chart.addSeries(new EJSC.BarSeries(
    data,
    {
        groupedBars: false,
    }
));
var bar2 = chart.addSeries(new EJSC.BarSeries(data));
```

#### 5.4.11.1.2 hint\_string (inherited)

#### Definition

string **hint\_string** = undefined

#### Description

This property may be used to define a series specific string to be used when displaying point hints. This string may contain replacement indicators (see [Text Replacement Options](#)). When left undefined, a default hint defined by the series type will be displayed.

### Example

>> *Modify the default hint to include custom text.*

```
var series = new EJSC.LineSeries(new EJSC.ArrayDataHandler(),
    {
        hint_string: "My Custom Hint<br /><strong>X: </strong>[x]<br/><strong>Y:
</strong>[y]"
    }
);
```

#### 5.4.11.1.3 intervalOffset (inherited)

### Definition

float **intervalOffset** = 0.8

### Description

Defines the spacing between the bars as a percentage of 1. 1 = bars take up the entire width available, with their sides touching.

### Example

>> *Make the bars take up 1/2 their available width. (i.e. more space between bars than by default)*

```
var chart = new EJSC.Chart("chart");
var bar = chart.addSeries(new EJSC.BarSeries(
    data,
    {
        intervalOffset: 0.5
    }
));
```

#### 5.4.11.1.4 legendsVisible (inherited)

### Definition

boolean **legendsVisible** = true

### Description

Determines whether the legend item associated with the series appears in the legend window. Use the Series.showLegend and Series.hideLegend methods to control legend item visibility after series creation.

#### 5.4.11.1.5 title (inherited)

### Definition

```
string title = "Series <index>"
```

**Description**

Defines the series title used in hints and legend display.

**Note:** The default title of Series <index> (where <index> equals the current position in the series array for its owner chart) is assigned when a series is added to a chart using [addSeries](#). This default title is only applied if the title is blank (i.e. "") at the time the addSeries method is called.

## 5.4.11.1.6 orientation (inherited)

**Definition**

```
string orientation = "vertical"
```

**Description**

This property determines the orientation of the bar series. The supported property values are "vertical" and "horizontal".

**Note:** Currently this is a creation-time only property and should not be changed after the series has been created.

## 5.4.11.1.7 visible (inherited)

**Definition**

```
boolean visible = true
```

**Description**

Defines whether the series is visible and can draw on the chart

## 5.4.11.1.8 x\_axis (inherited)

**Definition**

```
string x_axis = "bottom"
```

**Description**

Defines the horizontal axis to use for X values. Valid options are "top" or "bottom".

## 5.4.11.1.9 y\_axis (inherited)

**Definition**

```
string y_axis = "left"
```

**Description**

Defines the vertical axis to use for Y values. Valid options are "left" or "right".

### 5.4.11.2 Methods

#### 5.4.11.2.1 getBarSize (inherited)

##### Definition

integer [getBarSize\(\)](#)

##### Description

This method returns the size in pixels (width or height depending on orientation) of a single bar.

#### 5.4.11.2.2 addSeries

##### Definition

[EJSC.BarSeries](#) [addSeries](#)( [EJSC.BarSeries](#) series, boolean redraw )

##### Description

Adds a new series to the stacked bar series and returns the newly added series (this is useful when creating series inline as shown in the example below). The series must be a [EJSC.BarSeries](#) and have the same orientation as the stacked bar series being added to..

If redraw is false, drawing will not occur immediately but will not be entirely prevented. Certain actions such as adding an additional series with redraw = true, or chart dimensions changing may still trigger the series to draw. The default, if redraw is omitted is **true**.

**Note:** If the series has not defined a title (i.e. Series.title = "") at the time addSeries is called, a default title of "Series <index>" will be assigned (where <index> is the current series index in internal series storage).

##### Example

*>> Add new bar series to an existing stacked bar series.*

```
var myChart = new EJSC.Chart("chart");  
var myStackedSeries = myChart.addSeries(new EJSC.StackedBarSeries(data, { }));
```

```
var myBarSeries1 = myStackedSeries.addSeries(new EJSC.BarSeries(...));  
var myBarSeries2 = myStackedSeries.addSeries(new EJSC.BarSeries(...));  
var myBarSeries3 = myStackedSeries.addSeries(new EJSC.BarSeries(...));  
var myBarSeries4 = myStackedSeries.addSeries(new EJSC.BarSeries(...));  
var myBarSeries5 = myStackedSeries.addSeries(new EJSC.BarSeries(...));
```

#### 5.4.11.2.3 getVisibility (inherited)

##### Definition

boolean **getVisibility**( )

#### **Description**

Returns a boolean indicating the series current visible state

#### 5.4.11.2.4 hide (inherited)

#### **Definition**

void **hide**( )

#### **Description**

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already hidden.

#### 5.4.11.2.5 hideLegend (inherited)

#### **Definition**

void **hideLegend**( )

#### **Description**

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

#### 5.4.11.2.6 reload (inherited)

#### **Definition**

void **reload**( )

#### **Description**

Triggers a series reload if the functionality is defined in a child class. This may involve retrieving data again or simply recalculating. While the reload() method exists in the base Series class, implementation of the protected methods triggered by calling reload() is at the discretion of the child class.

#### 5.4.11.2.7 removeSeries

#### **Definition**

void **removeSeries**( [EJSC.BarSeries](#) series, boolean redraw )

#### **Description**

Removes the specified series from the stacked bar series and triggers an immediate rescaling and redraw. Send in `redraw = false` if performing multiple removes and want to delay redrawing until complete.

#### 5.4.11.2.8 `setGroupedBars` (inherited)

##### Definition

void `setGroupedBars`( boolean grouped, boolean redraw )

##### Description

Changes the chart between grouped and overlaid bars and optionally triggers a redraw.

#### 5.4.11.2.9 `setIntervalOffset` (inherited)

##### Definition

void `setIntervalOffset`( float offset, boolean redraw )

##### Description

Updates the interval offset property (spacing between the bars) and optionally redraws the chart. See [EJSC.BarSeries.intervalOffset](#) for additional information.

#### 5.4.11.2.10 `setTitle` (inherited)

##### Definition

void `setTitle`( string title )

##### Description

Changes the series title and update the legend caption (if applicable).

**Note:** this will not cause updates to a hint which is visible at the time it is called, though the next time a hint is rendered it will show the new property value.

#### 5.4.11.2.11 `show` (inherited)

##### Definition

void `show`( )

##### Description

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already visible.

#### 5.4.11.2.12 showLegend (inherited)

##### Definition

void [showLegend](#)( )

##### Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

### 5.4.11.3 Events

#### 5.4.11.3.1 onAfterVisibilityChange (inherited)

##### Definition

boolean [onAfterVisibilityChange](#)( [EJSC.Series](#) series, [EJSC.Chart](#) chart, boolean visible )

##### Description

Fired when the series visibility changes. Sends series which changed, its owner chart and its new visible property value.

Returning **false** from this event handler will cancel redrawing the chart.

#### 5.4.11.3.2 onBeforeVisibilityChange (inherited)

##### Definition

boolean [onBeforeVisibilityChange](#)( [EJSC.Series](#) series, [EJSC.Chart](#) chart )

##### Description

Fired before the series visibility is about to change. Sends series which changed, its owner chart. The current visible state of the series may be checked using the series.visible property.

Returning **false** from this event handler will cancel the change in visibility.

#### 5.4.11.3.3 onShowHint (inherited)

##### Definition

string [onShowHint](#)( [EJSC.Point](#) point, [EJSC.Series](#) series, [EJSC.Chart](#) chart, string hint\_string )

##### Description

Fired before the displaying the hint, this event allows the current hint string to be overridden. See [Text Replacement Options](#) for a list of the automatic substitutions available.

#### 5.4.11.4 Text Replacement Options (COPY)

Enter topic text here.

### 5.4.12 EJSC.TrendSeries

The TrendSeries is rendered as a line based on the results of a function applied to the related series' data points.

The constructor expects a reference series, a `trendType` and an optional set of object properties.

The trend series then "trends" the data in that referenceSeries according to the trendType into a function, and plots the function in the visible area of the chart.

#### Constructor

```
EJSC.TrendSeries( EJSC.Series referenceSeries, string trendType [, object options] )
```

Valid options for trendType:

```
"linear"  
"power"  
"exponential"  
"logarithmic"
```

#### Example

```
var mySeries1 = new EJSC.ScatterSeries(...);  
var mySeries2 = new EJSC.TrendSeries(  
    mySeries1,  
    "linear",  
    { title: "New Trend Series" }  
);
```

#### Defining Properties and Events

TrendSeries properties may be set individually after the series has been created or in batch using the options parameter during instantiation.

#### Setting properties individually

```
var mySeries = new EJSC.TrendSeries(myDataSeries, "linear");  
mySeries.lineWidth = 2;  
mySeries.title = "New Trend Series";
```

#### Setting properties in batch

```
var mySeries = new EJSC.TrendSeries(  
    myDataSeries,  
    "linear",  
    { lineWidth: 2, title: "New Trend Series" }
```



```
);
```

### 5.4.12.1 Properties

#### 5.4.12.1.1 color (inherited)

##### Definition

string **color** = undefined

##### Description

Defines the series color. If left undefined, the color is pulled from the array of default colors stored in the chart.

**Note:** May not be applicable to all series (i.e. [EJSC.PieSeries](#))

#### 5.4.12.1.2 coloredLegend (inherited)

##### Definition

boolean **coloredLegend** = true

##### Description

Determines whether the legend text inherits the series color. Setting to false will allow the legend text to inherit its normal cascaded color. To modify this property after series creation see [EJSC.Series.setColoredLegend\(\)](#)

#### 5.4.12.1.3 hint\_string (inherited)

##### Definition

string **hint\_string** = undefined

##### Description

This property may be used to define a series specific string to be used when displaying point hints. This string may contain replacement indicators (see [Text Replacement Options](#)). When left undefined, a default hint defined by the series type will be displayed.

##### Example

*>> Modify the default hint to include custom text.*

```
var series = new EJSC.LineSeries(new EJSC.ArrayDataHandler(),
    {
        hint_string: "My Custom Hint<br /><strong>X: </strong>[x]<br/><strong>Y:
</strong>[y]"
    }
);
```

## 5.4.12.1.4 legendIsVisible (inherited)

**Definition**

boolean **legendIsVisible** = true

**Description**

Determines whether the legend item associated with the series appears in the legend window. Use the `Series.showLegend` and `Series.hideLegend` methods to control legend item visibility after series creation.

## 5.4.12.1.5 lineOpacity (inherited)

**Definition**

integer **lineOpacity** = 100

**Description**

Determines the line opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see [EJSC.Series.setLineOpacity\(\)](#)

## 5.4.12.1.6 lineWidth (inherited)

**Definition**

integer **lineWidth** = 1

**Description**

Defines the width of the line connecting series points.

## 5.4.12.1.7 title (inherited)

**Definition**

string **title** = "Series <index>"

**Description**

Defines the series title used in hints and legend display.

**Note:** The default title of Series <index> (where <index> equals the current position in the series array for its owner chart) is assigned when a series is added to a chart using [addSeries](#). This default title is only applied if the title is blank (i.e. "") at the time the `addSeries` method is called.

## 5.4.12.1.8 visible (inherited)

**Definition**

boolean **visible** = true

### Description

Defines whether the series is visible and can draw on the chart

#### 5.4.12.1.9 x\_axis (inherited)

### Definition

string **x\_axis** = "bottom"

### Description

Defines the horizontal axis to use for X values. Valid options are "top" or "bottom".

#### 5.4.12.1.10 x\_axis\_formatter (inherited)

### Definition

string **x\_axis\_formatter** = undefined

### Description

Defines the formatter that will be used to format hints for the X values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:

[EJSC.DateFormatter](#)

[EJSC.NumberFormatter](#)

#### 5.4.12.1.11 y\_axis (inherited)

### Definition

string **y\_axis** = "left"

### Description

Defines the vertical axis to use for Y values. Valid options are "left" or "right".

#### 5.4.12.1.12 y\_axis\_formatter (inherited)

### Definition

string **y\_axis\_formatter** = undefined

### Description

Defines the formatter that will be used to format hints for the Y values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:

[EJSC.DateFormatter](#)

[EJSC.NumberFormatter](#)

### 5.4.12.2 Methods

#### 5.4.12.2.1 getVisibility (inherited)

##### Definition

boolean **getVisibility**( )

##### Description

Returns a boolean indicating the series current visible state

#### 5.4.12.2.2 hide (inherited)

##### Definition

void **hide**( )

##### Description

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already hidden.

#### 5.4.12.2.3 hideLegend (inherited)

##### Definition

void **hideLegend**( )

##### Description

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

#### 5.4.12.2.4 reload (inherited)

##### Definition

void **reload**( )

##### Description

Triggers a series reload if the functionality is defined in a child class. This may involve retrieving data again or simply recalculating. While the reload() method exists in the base Series class, implementation of the protected methods triggered by calling reload() is at the discretion of the child class.

## 5.4.12.2.5 setColor (inherited)

**Definition**

void [setColor](#)( string color )

**Description**

Changes the series color and causes the chart to redraw.

## 5.4.12.2.6 setColoredLegend (inherited)

**Definition**

void [setColoredLegend](#)( boolean coloredLegend )

**Description**

Sets the [EJSC.Series.coloredLegend](#) property and updates the legend to reflect the change.

## 5.4.12.2.7 setLineWidth (inherited)

**Definition**

void [setLineWidth](#)( integer width )

**Description**

Updates the [lineWidth](#) property and redraws the chart.

## 5.4.12.2.8 setTitle (inherited)

**Definition**

void [setTitle](#)( string title )

**Description**

Changes the series title and update the legend caption (if applicable).

**Note:** this will not cause updates to a hint which is visible at the time it is called, though the next time a hint is rendered it will show the new property value.

## 5.4.12.2.9 show (inherited)

**Definition**

void [show](#)( )

**Description**

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already visible.

#### 5.4.12.2.10 showLegend (inherited)

##### Definition

void [showLegend](#)( )

##### Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

#### 5.4.12.3 Events

##### 5.4.12.3.1 onAfterVisibilityChange (inherited)

##### Definition

boolean [onAfterVisibilityChange](#)( [EJSC.Series](#) series, [EJSC.Chart](#) chart, boolean visible )

##### Description

Fired when the series visibility changes. Sends series which changed, its owner chart and its new visible property value.

Returning **false** from this event handler will cancel redrawing the chart.

##### 5.4.12.3.2 onBeforeVisibilityChange (inherited)

##### Definition

boolean [onBeforeVisibilityChange](#)( [EJSC.Series](#) series, [EJSC.Chart](#) chart )

##### Description

Fired before the series visibility is about to change. Sends series which changed, its owner chart. The current visible state of the series may be checked using the series.visible property.

Returning **false** from this event handler will cancel the change in visibility.

##### 5.4.12.3.3 onShowHint (inherited)

##### Definition

string [onShowHint](#)( [EJSC.Point](#) point, [EJSC.Series](#) series, [EJSC.Chart](#) chart, string hint\_string )

##### Description

Fired before the displaying the hint, this event allows the current hint string to be overridden. See [Text Replacement Options](#) for a list of the automatic substitutions available.

#### 5.4.12.4 Text Replacement Options

String	Replaced With
[chart_title]	The title of the chart
[series_title]	The title of the series the selected point resides in
[xaxis]	The caption of the series x axis
[yaxis]	The caption of the series y axis
[x]	The preformatted X value of the point
[y]	The preformatted Y value of the point
[label]	The label property of the current point

## 5.5 Data Handlers

### 5.5.1 EJSC.XMLDataHandler

XMLDataHandler loads the requested XML file, determines the format (Full, Short or Compact) and extracts the data points into a format that the Series classes can manipulate.

The constructor expects the URL to the XML data file and an optional set of object properties.

#### Constructor

```
EJSC.XMLDataHandler( string url [, object options] )
```

#### Example

```
var myDataHandler = new EJSC.XMLDataHandler(
    "http://www.ejschart.com/data.xml"
);
```

#### Formats:

The XMLDataHandler supports three formats:

- [Full](#)
- [Short](#)
- [Compact](#)

#### 5.5.1.1 Properties

5.5.1.1.1 requestType (inherited)

#### Definition

```
string requestType = "GET"
```

#### Description

Defines the type of HTTP request to be made. Valid options are "GET" and "POST"

#### 5.5.1.1.2 url (inherited)

##### Definition

string `url` = null

##### Description

Defines the url currently being used to retrieve data. Descendants of `AjaxDataHandler` generally allow this to be set in the constructor method. To modify this value after creation, call the [setUrl](#) method which allows the data to be reloaded immediately.

#### 5.5.1.1.3 urlData (inherited)

##### Definition

string `urlData` = null

##### Description

Defines the data to be included for both POST and GET requests. When performing a GET request, this data (if defined) will be appended to the URL before the request is made. For POST requests, this data will be sent as the body of the request.

**NOTE:** For GET requests you must include the proper syntax for appending the url stored in the url property. Example:

```
url = "http://localhost/getData"  
urlData = "?param1=30"
```

or

```
url = "http://localhost/getData?param1=30"  
urlData = "&extra=1"
```

### 5.5.1.2 Methods

#### 5.5.1.2.1 getUrl (inherited)

##### Definition

string `getUrl()`

##### Description

Returns the url string currently stored in the `AjaxDataHandler` descendant.

**DEPRECATED, see [EJSC.AjaxDataHandler.url](#)**



## 5.5.1.2.2 loadData (inherited)

**Definition**

void **loadData**( )

**Description**

Placeholder method which should be used to clear (if stored) and reload data. This method must be overridden in all descendant classes.

## 5.5.1.2.3 setRequestType (inherited)

**Definition**

void **setRequestType**( string requestType, boolean reload )

**Description**

Updates the requestType stored in the AjaxDataHandler descendant and optionally reloads / reprocesses the data source and updates the series.

Valid options for requestType are "GET" and "POST"

**Note:** If the reload parameter is omitted or false, the data will not reload automatically and must be done manually via [EJSC.Series.reload\(\)](#)

## 5.5.1.2.4 setUrl (inherited)

**Definition**

void **setUrl**( string url, boolean reload )

**Description**

Updates the url string stored in the AjaxDataHandler descendant and optionally reloads the data immediately.

**Note:** If the reload parameter is omitted or false, the data will not reload automatically and must be done manually via [EJSC.Series.reload\(\)](#)

## 5.5.1.2.5 setUrlData (inherited)

**Definition**

void **setUrlData**( string urlData, boolean reload )

**Description**

Updates the urlData stored in the AjaxDataHandler descendant and optionally reloads / reprocesses the data source and updates the series.

**Note:** If the reload parameter is omitted or false, the data will not reload automatically and must be done manually via [EJSC.Series.reload\(\)](#)

#### 5.5.1.2.6 setXMLData (inherited)

##### Definition

void [setXMLData](#)( XMLHttpRequest request )

##### Description

This method allows passing of an already retrieved XMLHttpRequest object to an AjaxDataHandler descendant. When used in conjunction with the [onNeedsData](#) event, this method allows interaction with pre-existing 3rd party Ajax libraries instead of the request pool built into the chart library.

If the data handler is not currently in the loading state (i.e. [onNeedsData](#) was triggered), calling this method will have no effect.

#### 5.5.1.3 Events

##### 5.5.1.3.1 onDataAvailable (inherited)

##### Definition

void [onDataAvailable](#)()

##### Description

Placeholder for storing an event to fire after the a descendant class has loaded and processed its data.

**Important:** This event should be assigned by the EJSC.Series object.

##### 5.5.1.3.2 onDataReady (inherited)

##### Definition

boolean [onDataReady](#)( XMLHttpRequest response, EJSC.AjaxDataHandler data\_handler, EJSC.Series series, EJSC.Chart chart)

##### Description

This event is fired when data has arrived completely and is ready for processing by the data handler. Returning true from this event will cause the data handler to process its data normally, returning false will cause processing to be canceled.

##### 5.5.1.3.3 onNeedsData (inherited)

##### Definition

boolean [onNeedsData](#)( [EJSC.AjaxDataHandler](#) dataHandler, [EJSC.Series](#) series, [EJSC.Chart](#) chart )  
XMLHttpRequest [onNeedsData](#)( [EJSC.AjaxDataHandler](#) dataHandler, [EJSC.Series](#) series, [EJSC.Chart](#)

chart )

### Description

This event allows for additional control over when and how data is retrieved by the data handler. If assigned, the event will be triggered whenever the its owner series has requested the data handler retrieve its data. The event passes a reference to the current data handler, its owner series and the owner chart.

The result of this event is expected to be either a boolean value or an XMLHttpRequest object which has already been retrieved (responseXML is a valid, well-formed chart xml document).

### RESULT:

- **true:** The data handler will ignore other properties such as url and requestType and hold itself in a loading state and wait for its setXMLData method to be called and provided a valid XMLHttpRequest object. In this manner, a separate Ajax library may be used to retrieve data.
- **false:** The data handler will ignore other properties such as url and requestType and cancel its loading process.
- **XMLHttpRequest:** The data handler will ignore other properties such as url and requestType and immediately begin processing the data stored in responseXML.

## 5.5.2 EJSC.XMLStringDataHandler

XMLStringDataHandler processes the provided xml-formatted string, determines the format (Full, Short or Compact) and extracts the data points into a format that the Series classes can manipulate.

The constructor expects a properly formatted string containing XML data and an optional set of object properties.

### Constructor

```
EJSC.XMLStringDataHandler( string xml [, object options] )
```

### Example

```
var myDataHandler = new EJSC.XMLStringDataHandler(
    '<?xml version="1.0"?><graph><plot><point x="1" y="1" /><point x="2" y="2" /><point x="3" y="3" /></plot></graph>'
);
```

### Formats:

The XMLStringDataHandler supports three formats:

- [Full](#)
- [Short](#)
- [Compact](#)

### 5.5.2.1 Methods

#### 5.5.2.1.1 getXML

**Definition**

string [getXML\(\)](#)

**Description**

Returns the xml string currently stored in the XMLStringDataHandler.

#### 5.5.2.1.2 loadData (inherited)

**Definition**

void [loadData\(\)](#)

**Description**

Placeholder method which should be used to clear (if stored) and reload data. This method must be overridden in all descendant classes.

#### 5.5.2.1.3 setXML

**Definition**

void [setXML\(\)](#) ( string xml )

**Description**

Updates the xml string stored in the XMLStringDataHandler.

**Note:** This will not cause the data to be refreshed, see [EJSC.Series.reload\(\)](#) for more information.

### 5.5.2.2 Events

#### 5.5.2.2.1 onDataAvailable (inherited)

**Definition**

void [onDataAvailable\(\)](#)

**Description**

Placeholder for storing an event to fire after the a descendant class has loaded and processed its data.

**Important:** This event should be assigned by the EJSC.Series object.

### 5.5.3 EJSC.ArrayDataHandler

ArrayDataHandler converts the JavaScript array of points into a format that the Series classes can manipulate.

The constructor expects a multi-dimensional array of data and an optional set of object properties.

#### Constructor

```
EJSC.ArrayDataHandler( array data [, object options] )
```

#### Example

```
var myDataHandler = new EJSC.ArrayDataHandler(
    [ [1,1], [2,2], [3,3], [4,4] ]
);
```

#### 5.5.3.1 Methods

##### 5.5.3.1.1 `getArray`

#### Definition

array `getArray()`

#### Description

Returns the data array currently stored in the ArrayDataHandler.

##### 5.5.3.1.2 `loadData` (inherited)

#### Definition

void `loadData()`

#### Description

Placeholder method which should be used to clear (if stored) and reload data. This method must be overridden in all descendant classes.

##### 5.5.3.1.3 `setArray`

#### Definition

void `setArray`( array data )

#### Description

Updates the data array stored in the ArrayDataHandler.

**Note:** This will not cause the data to be refreshed, see [EJSC.Series.reload\(\)](#) for more

information.

### 5.5.3.2 Events

#### 5.5.3.2.1 onDataAvailable (inherited)

##### Definition

void [onDataAvailable\(\)](#)

##### Description

Placeholder for storing an event to fire after the a descendant class has loaded and processed its data.

**Important:** This event should be assigned by the EJSC.Series object.

## 5.5.4 EJSC.CSVFileDataHandler

CSVFileDataHandler loads the requested csv file and extracts the data points into a format that the Series classes can manipulate.

The constructor expects the URL to the csv data file and an optional set of object properties.

##### Constructor

```
EJSC.CSVFileDataHandler( string url [, object options] )
```

##### Example

```
var myDataHandler = new EJSC.CSVFileDataHandler(  
    "http://www.ejschart.com/data.csv"  
);
```

### 5.5.4.1 Properties

#### 5.5.4.1.1 requestType (inherited)

##### Definition

string [requestType](#) = "GET"

##### Description

Defines the type of HTTP request to be made. Valid options are "GET" and "POST"

#### 5.5.4.1.2 url (inherited)

##### Definition

string [url](#) = null

##### Description

Defines the url currently being used to retrieve data. Descendants of AjaxDataHandler generally allow this to be set in the constructor method. To modify this value after creation, call the [setUrl](#) method which allows the data to be reloaded immediately.

#### 5.5.4.1.3 urlData (inherited)

##### Definition

string [urlData](#) = null

##### Description

Defines the data to be included for both POST and GET requests. When performing a GET request, this data (if defined) will be appended to the URL before the request is made. For POST requests, this data will be sent as the body of the request.

**NOTE:** For GET requests you must include the proper syntax for appending the url stored in the url property. Example:

```
url = "http://localhost/getData"  
urlData = "?param1=30"
```

or

```
url = "http://localhost/getData?param1=30"  
urlData = "&extra=1"
```

#### 5.5.4.2 Methods

##### 5.5.4.2.1 getUrl (inherited)

##### Definition

string [getUrl](#)( )

##### Description

Returns the url string currently stored in the AjaxDataHandler descendant.

**DEPRECATED, see [EJSC.AjaxDataHandler.url](#)**

##### 5.5.4.2.2 loadData (inherited)

##### Definition

void [loadData](#)( )

##### Description

Placeholder method which should be used to clear (if stored) and reload data. This method must be overridden in all descendant classes.

## 5.5.4.2.3 setRequestType (inherited)

**Definition**

void **setRequestType**( string requestType, boolean reload )

**Description**

Updates the requestType stored in the AjaxDataHandler descendant and optionally reloads / reprocesses the data source and updates the series.

Valid options for requestType are "GET" and "POST"

**Note:** If the reload parameter is omitted or false, the data will not reload automatically and must be done manually via [EJSC.Series.reload\(\)](#)

## 5.5.4.2.4 setUrl (inherited)

**Definition**

void **setUrl**( string url, boolean reload )

**Description**

Updates the url string stored in the AjaxDataHandler descendant and optionally reloads the data immediately.

**Note:** If the reload parameter is omitted or false, the data will not reload automatically and must be done manually via [EJSC.Series.reload\(\)](#)

## 5.5.4.2.5 setUrlData (inherited)

**Definition**

void **setUrlData**( string urlData, boolean reload )

**Description**

Updates the urlData stored in the AjaxDataHandler descendant and optionally reloads / reprocesses the data source and updates the series.

**Note:** If the reload parameter is omitted or false, the data will not reload automatically and must be done manually via [EJSC.Series.reload\(\)](#)

## 5.5.4.2.6 setXMLData (inherited)

**Definition**

void **setXMLData**( XMLHttpRequest request )

**Description**



This method allows passing of an already retrieved XMLHttpRequest object to an AjaxDataHandler descendant. When used in conjunction with the [onNeedsData](#) event, this method allows interaction with pre-existing 3rd party Ajax libraries instead of the request pool built into the chart library.

If the data handler is not currently in the loading state (i.e. [onNeedsData](#) was triggered), calling this method will have no effect.

### 5.5.4.3 Events

#### 5.5.4.3.1 onDataAvailable (inherited)

##### Definition

void [onDataAvailable](#)()

##### Description

Placeholder for storing an event to fire after the a descendant class has loaded and processed its data.

**Important:** This event should be assigned by the EJSK.Series object.

#### 5.5.4.3.2 onDataReady (inherited)

##### Definition

boolean [onDataReady](#)( XMLHttpRequest response, EJSK.AjaxDataHandler data\_handler, EJSK.Series series, EJSK.Chart chart)

##### Description

This event is fired when data has arrived completely and is ready for processing by the data handler. Returning true from this event will cause the data handler to process its data normally, returning false will cause processing to be canceled.

#### 5.5.4.3.3 onNeedsData (inherited)

##### Definition

boolean [onNeedsData](#)( [EJSK.AjaxDataHandler](#) dataHandler, [EJSK.Series](#) series, [EJSK.Chart](#) chart )  
XMLHttpRequest [onNeedsData](#)( [EJSK.AjaxDataHandler](#) dataHandler, [EJSK.Series](#) series, [EJSK.Chart](#) chart )

##### Description

This event allows for additional control over when and how data is retrieved by the data handler. If assigned, the event will be triggered whenever the its owner series has requested the data handler retrieve its data. The event passes a reference to the current data handler, its owner series and the owner chart.

The result of this event is expected to be either a boolean value or an XMLHttpRequest object which has already been retrieved (responseXML is a valid, well-formed chart xml document).

**RESULT:**

- **true:** The data handler will ignore other properties such as url and requestType and hold itself in a loading state and wait for its setXMLData method to be called and provided a valid XMLHttpRequest object. In this manner, a separate Ajax library may be used to retrieve data.
- **false:** The data handler will ignore other properties such as url and requestType and cancel its loading process.
- **XMLHttpRequest:** The data handler will ignore other properties such as url and requestType and immediately begin processing the data stored in responseXML.

### 5.5.5 EJSC.CSVStringDataHandler

CSVStringDataHandler converts the csv string provided into a format that the Series classes can manipulate.

The constructor expects a properly formatted string and an optional set of object properties.

**Constructor**

```
EJSC.CSVStringDataHandler( string data [, object options] )
```

**Example**

```
var myDataHandler = new EJSC.CSVStringDataHandler(  
    "1|1,2|2,3|3,4|4"  
);
```

#### 5.5.5.1 Methods

##### 5.5.5.1.1 getCSV

**Definition**

string [getCSV](#)( )

**Description**

Returns the csv string currently stored in the CSVStringDataHandler.

##### 5.5.5.1.2 loadData (inherited)

**Definition**

void [loadData](#)( )

**Description**

Placeholder method which should be used to clear (if stored) and reload data. This method must be overridden in all descendant classes.

### 5.5.5.1.3 setCSV

#### Definition

```
void setCSV( string csv )
```

#### Description

Updates the csv string stored in the CSVStringDataHandler.

**Note:** This will not cause the data to be refreshed, see [EJSC.Series.reload\(\)](#) for more information.

### 5.5.5.2 Events

#### 5.5.5.2.1 onDataAvailable (inherited)

#### Definition

```
void onDataAvailable()
```

#### Description

Placeholder for storing an event to fire after the a descendant class has loaded and processed its data.

**Important:** This event should be assigned by the EJSC.Series object.

### 5.5.6 EJSC.JSONFileDataHandler

JSONFileDataHandler loads the requested JSON file and extracts the data points into a format that the Series classes can manipulate.

The constructor expects the URL to the JSON data file and an optional set of object properties.

#### Constructor

```
EJSC.JSONFileDataHandler( string url [, object options] )
```

#### Example

```
var myDataHandler = new EJSC.JSONDataHandler(  
    "http://www.ejschart.com/data.json"  
);
```

See [Data Formats](#) for additional information.

#### 5.5.6.1 Properties

##### 5.5.6.1.1 requestType (inherited)

#### Definition

```
string requestType = "GET"
```

**Description**

Defines the type of HTTP request to be made. Valid options are "GET" and "POST"

## 5.5.6.1.2 url (inherited)

**Definition**

```
string url = null
```

**Description**

Defines the url currently being used to retrieve data. Descendants of AjaxDataHandler generally allow this to be set in the constructor method. To modify this value after creation, call the [setUrl](#) method which allows the data to be reloaded immediately.

## 5.5.6.1.3 urlData (inherited)

**Definition**

```
string urlData = null
```

**Description**

Defines the data to be included for both POST and GET requests. When performing a GET request, this data (if defined) will be appended to the URL before the request is made. For POST requests, this data will be sent as the body of the request.

**NOTE:** For GET requests you must include the proper syntax for appending the url stored in the url property. Example:

```
url = "http://localhost/getData"  
urlData = "?param1=30"
```

or

```
url = "http://localhost/getData?param1=30"  
urlData = "&extra=1"
```

**5.5.6.2 Methods**

## 5.5.6.2.1 getUrl (inherited)

**Definition**

```
string getUrl( )
```

**Description**

Returns the url string currently stored in the AjaxDataHandler descendant.

**DEPRECATED**, see [EJSC.AjaxDataHandler.url](#)

#### 5.5.6.2.2 loadData (inherited)

##### Definition

void [loadData](#)( )

##### Description

Placeholder method which should be used to clear (if stored) and reload data. This method must be overridden in all descendant classes.

#### 5.5.6.2.3 setRequestType (inherited)

##### Definition

void [setRequestType](#)( string requestType, boolean reload )

##### Description

Updates the requestType stored in the AjaxDataHandler descendant and optionally reloads / reprocesses the data source and updates the series.

Valid options for requestType are "GET" and "POST"

**Note:** If the reload parameter is omitted or false, the data will not reload automatically and must be done manually via [EJSC.Series.reload\(\)](#)

#### 5.5.6.2.4 setUrl (inherited)

##### Definition

void [setUrl](#)( string url, boolean reload )

##### Description

Updates the url string stored in the AjaxDataHandler descendant and optionally reloads the data immediately.

**Note:** If the reload parameter is omitted or false, the data will not reload automatically and must be done manually via [EJSC.Series.reload\(\)](#)

#### 5.5.6.2.5 setUrlData (inherited)

##### Definition

void [setUrlData](#)( string urlData, boolean reload )

**Description**

Updates the urlData stored in the AjaxDataHandler descendant and optionally reloads / reprocesses the data source and updates the series.

**Note:** If the reload parameter is omitted or false, the data will not reload automatically and must be done manually via [EJSC.Series.reload\(\)](#)

## 5.5.6.2.6 setXMLData (inherited)

**Definition**

void [setXMLData](#)( XMLHttpRequest request )

**Description**

This method allows passing of an already retrieved XMLHttpRequest object to an AjaxDataHandler descendant. When used in conjunction with the [onNeedsData](#) event, this method allows interaction with pre-existing 3rd party Ajax libraries instead of the request pool built into the chart library.

If the data handler is not currently in the loading state (i.e. [onNeedsData](#) was triggered), calling this method will have no effect.

**5.5.6.3 Events**

## 5.5.6.3.1 onDataAvailable (inherited)

**Definition**

void [onDataAvailable](#)()

**Description**

Placeholder for storing an event to fire after the a descendant class has loaded and processed its data.

**Important:** This event should be assigned by the EJSC.Series object.

## 5.5.6.3.2 onDataReady (inherited)

**Definition**

boolean [onDataReady](#)( XMLHttpRequest response, EJSC.AjaxDataHandler data\_handler, EJSC.Series series, EJSC.Chart chart)

**Description**

This event is fired when data has arrived completely and is ready for processing by the data handler. Returning true from this event will cause the data handler to process its data normally, returning false will cause processing to be canceled.

## 5.5.6.3.3 onNeedsData (inherited)

**Definition**

boolean **onNeedsData**( [EJSC.AjaxDataHandler](#) dataHandler, [EJSC.Series](#) series, [EJSC.Chart](#) chart )  
 XMLHttpRequest **onNeedsData**( [EJSC.AjaxDataHandler](#) dataHandler, [EJSC.Series](#) series, [EJSC.Chart](#) chart )

**Description**

This event allows for additional control over when and how data is retrieved by the data handler. If assigned, the event will be triggered whenever the its owner series has requested the data handler retrieve its data. The event passes a reference to the current data handler, its owner series and the owner chart.

The result of this event is expected to be either a boolean value or an XMLHttpRequest object which has already been retrieved (responseXML is a valid, well-formed chart xml document).

**RESULT:**

- **true:** The data handler will ignore other properties such as url and requestType and hold itself in a loading state and wait for its setXMLData method to be called and provided a valid XMLHttpRequest object. In this manner, a separate Ajax library may be used to retrieve data.
- **false:** The data handler will ignore other properties such as url and requestType and cancel its loading process.
- **XMLHttpRequest:** The data handler will ignore other properties such as url and requestType and immediately begin processing the data stored in responseXML.

**5.5.7 EJSC.JSONStringDataHandler**

JSONStringDataHandler converts the JSON string provided into a format that the Series classes can manipulate.

The constructor expects a properly formatted string and an optional set of object properties.

**Constructor**

```
EJSC.JSONStringDataHandler( string data [, object options] )
```

**Example**

```
var myDataHandler = new EJSC.JSONStringDataHandler(
    '{"x": "1", "y": "2"}, {"x": "2", "y": "4"}'
);
```

**5.5.7.1 Methods**

## 5.5.7.1.1 getJSON

**Definition**

string [getJSON](#)( )

#### Description

Returns the JSON string currently stored in the JSONStringDataHandler.

#### 5.5.7.1.2 loadData (inherited)

#### Definition

void [loadData](#)( )

#### Description

Placeholder method which should be used to clear (if stored) and reload data. This method must be overridden in all descendant classes.

#### 5.5.7.1.3 setJSON

#### Definition

void [setJSON](#)( string JSON )

#### Description

Updates the JSON string stored in the JSONStringDataHandler.

**Note:** This will not cause the data to be refreshed, see [EJSC.Series.reload\(\)](#) for more information.

### 5.5.7.2 Events

#### 5.5.7.2.1 onDataAvailable (inherited)

#### Definition

void [onDataAvailable](#)()

#### Description

Placeholder for storing an event to fire after the a descendant class has loaded and processed its data.

**Important:** This event should be assigned by the EJSC.Series object.

## 5.6 Label Formatters

### 5.6.1 EJSC.NumberFormatter

Use this formatter when you want more control over the display of numeric values in your charts. It can be applied to the chart axis as well as series hints.

#### Constructor



EJSC.NumberFormatter([options])

### Example

```
var nf = new EJSC.NumberFormatter({
    currency_symbol: "$",
    currency_position: "inner",
    negative_symbol: "("
});
```

#### 5.6.1.1 Properties

##### 5.6.1.1.1 currency\_align

### Definition

string **currency\_align** = "left"

### Description

Valid options are "left" and "right".

##### 5.6.1.1.2 currency\_position

### Definition

string **currency\_position** = "inner"

### Description

Valid options are "inner" and "outer".

##### 5.6.1.1.3 currency\_symbol

### Definition

string **currency\_symbol** = ""

### Description

Specifies the symbol used to indicate currency.

**Note:** If blank "" then no currency marker is used.

##### 5.6.1.1.4 decimal\_separator

### Definition

string **decimal\_separator** = "."

### Description

Specifies the symbol used to separate the whole number part from the fractional part.

Common values are "." or ","

#### 5.6.1.1.5 forced\_decimals

##### **Definition**

integer **forced\_decimals** = undefined

##### **Description**

Specifies the number of decimal places which MUST be displayed (0's are appended if necessary).

#### 5.6.1.1.6 negative\_symbol

##### **Definition**

string **negative\_symbol** = "-"

##### **Description**

Specifies the formatting of negative numbers.

Valid options are "-" and "()"

#### 5.6.1.1.7 thousand\_separator

##### **Definition**

string **thousand\_separator** = ","

##### **Description**

Specifies the symbol used to separate thousands groupings.

Common values are:

"," renders as 1,000,000

"." renders as 1.000.000

"" renders as 1000000

#### 5.6.1.1.8 variable\_decimals

##### **Definition**

string **variable\_decimals** = undefined

##### **Description**

Specifies the number of decimal places which MAY be displayed (values are truncated to this length)

If left undefined, values are not truncated.

### 5.6.1.2 Methods

#### 5.6.1.2.1 format (inherited)

##### Definition

string **format**( float value, integer precision )

##### Description

Format is called when a value needs to be formatted for display in the axis, hint, cursor position, etc.

The behavior of the format method is to round a numeric value to the precision specified. Descendants do not need to implement the precision parameter.

## 5.6.2 EJSC.DateFormatter

Use this formatter when you want to display date values in your charts. It can be applied to the chart axis, series hints and cursor position labels by assigning the appropriate formatter property.

Dates are supported in UTC format by specifying a JavaScript date (i.e. number of milliseconds since midnight January 01, 1970) for a point value.

See the [format\\_string](#) property for information on supported date formatting options.

##### Constructor

```
EJSC.DateFormatter([options])
```

##### Example

```
var df = new EJSC.DateFormatter({
    format_string: "YYYY-MM-DD"
});
```

### 5.6.2.1 Properties

#### 5.6.2.1.1 format\_string

##### Definition

string **format\_string** = ""

##### Description

The following format options are available:

Format	Description
YYYY	Four (4) digit year (i.e. 2007)
YY	Two (2) digit year (i.e. 07)

MMMM	Full month name (i.e. January)
MMM	Short month name (i.e. Jan)
MM	Two (2) digit month of the year with leading zeros (i.e. 01)
M	Month of the year without leading zeros
DDDD	Full day of the week (i.e. Monday)
DDD	Short day of the week (i.e. Mon)
DD	Two (2) digit day of the month with leading zeros (i.e. 04)
D	Day of the month without leading zeros
HH	Hour of the day in 24-hour format with leading zeros
H	Hour of the day in 24-hour format without leading zeros
hh	Hour of the day in 12-hour format with leading zeros
h	Hour of the day in 12-hour format without leading zeros
NN	Minutes with leading zeros
N	Minutes without leading zeros
SS	Seconds with leading zeros
S	Seconds without leading zeros
ZZZ	Milliseconds (3 places) with leading zeros
ZZ	Milliseconds (2 places) with leading zeros
Z	Milliseconds without leading zeros
AA	AM/PM notation
A	A/P notation
aa	am/pm notation
a	a/p notation

#### 5.6.2.1.2 timezoneOffset

##### Definition

integer **timezoneOffset** = undefined

##### Description

Used in conjunction with useUTC, this property may be used to display dates in the specified time zone. The timezoneOffset property is set by specifying the number of minutes (positive or negative) to offset the given time.

##### Example

*>> Display times in Eastern Standard Time (EST)*

```
var chart = new EJSC.Chart("myChart");
    chart.axis_bottom.formatter = new EJSC.DateFormatter({
        format_string: "HH:NN:SS",
        useUTC: true,
        timezoneOffset: -300
    });
```

### 5.6.2.1.3 useUTC

#### Definition

boolean **useUTC** = true

#### Description

Set this property to false to use the workstation local time to calculate dates. This may be useful when passing in dates generated by JavaScript on the client machine to indicate current time.

## 5.6.2.2 Methods

### 5.6.2.2.1 format (inherited)

#### Definition

string **format**( float value, integer precision )

#### Description

Format is called when a value needs to be formatted for display in the axis, hint, cursor position, etc.

The behavior of the format method is to round a numeric value to the precision specified. Descendants do not need to implement the precision parameter.

## 5.6.3 EJSC.StringFormatter

Use this formatter when you want to prefix or append data to the value being displayed in your charts. It can be applied to the chart axis, series hints and cursor position labels by assigning the appropriate formatter property.

#### Constructor

```
EJSC.StringFormatter([options])
```

#### Example

```
var sf = new EJSC.StringFormatter({
    prefix: "Before (",
    append: ") After"
});
```

### 5.6.3.1 Properties

#### 5.6.3.1.1 append

#### Definition

string **append** = undefined

**Description**

The string stored in the append property will be applied to every value processed by the formatter.

## 5.6.3.1.2 prefix

**Definition**

string **prefix** = undefined

**Description**

The string stored in the prefix property will be applied to every value processed by the formatter.

**5.6.3.2 Methods**

## 5.6.3.2.1 format (inherited)

**Definition**

string **format**(float value, integer precision )

**Description**

Format is called when a value needs to be formatted for display in the axis, hint, cursor position, etc.

The behavior of the format method is to round a numeric value to the precision specified. Descendants do not need to implement the precision parameter.

**5.6.3.3 Events**

## 5.6.3.3.1 onNeedsFormat

**Definition**

string **onNeedsFormat**(number value, EJSC.StringFormatter formatter)

**Description**

This event (when assigned) will fire every time the formatter needs to process a value. The string returned from this event will be displayed in place of the value in the chart.

**Example**

*>> Adjust all values by 100*

```
var chart = new EJSC.Chart("myChart");
    chart.x_axis_formatter = new EJSC.StringFormatter({
        onNeedsFormat: function(value, formatter) {
            return value + 100;
        }
    });
```

## 5.7 Base Classes

### 5.7.1 EJSC.Inheritable

Base class from which all EJSC classes descend. There are no public properties, methods or events for this class at this time.

#### Constructor

none

### 5.7.2 EJSC.AjaxDataHandler

Base data handler class for all Ajax data handlers (i.e. [EJSC.XMLDataHandler](#), [EJSC.CSVFileDataHandler](#), [EJSC.JSONFileDataHandler](#)). This base class defines common properties, methods and events but does not implement any data processing. This class should not be used directly.

#### Constructor

none

#### 5.7.2.1 Properties

##### 5.7.2.1.1 requestType

#### Definition

string `requestType` = "GET"

#### Description

Defines the type of HTTP request to be made. Valid options are "GET" and "POST"

##### 5.7.2.1.2 url

#### Definition

string `url` = null

#### Description

Defines the url currently being used to retrieve data. Descendants of AjaxDataHandler generally allow this to be set in the constructor method. To modify this value after creation, call the [setUrl](#) method which allows the data to be reloaded immediately.

##### 5.7.2.1.3 urlData

#### Definition

```
string urlData = null
```

### Description

Defines the data to be included for both POST and GET requests. When performing a GET request, this data (if defined) will be appended to the URL before the request is made. For POST requests, this data will be sent as the body of the request.

**NOTE:** For GET requests you must include the proper syntax for appending the url stored in the url property. Example:

```
url = "http://localhost/getData"  
urlData = "?param1=30"
```

or

```
url = "http://localhost/getData?param1=30"  
urlData = "&extra=1"
```

## 5.7.2.2 Methods

### 5.7.2.2.1 getUrl

#### Definition

```
string getUrl( )
```

#### Description

Returns the url string currently stored in the AjaxDataHandler descendant.

**DEPRECATED, see [EJSC.AjaxDataHandler.url](#)**

### 5.7.2.2.2 loadData (inherited)

#### Definition

```
void loadData( )
```

#### Description

Placeholder method which should be used to clear (if stored) and reload data. This method must be overridden in all descendant classes.

### 5.7.2.2.3 setRequestType

#### Definition

```
void setRequestType( string requestType, boolean reload )
```

#### Description



Updates the requestType stored in the AjaxDataHandler descendant and optionally reloads / reprocesses the data source and updates the series.

Valid options for requestType are "GET" and "POST"

**Note:** If the reload parameter is omitted or false, the data will not reload automatically and must be done manually via [EJSC.Series.reload\(\)](#)

#### 5.7.2.2.4 setUrl

##### Definition

void [setUrl](#)( string url, boolean reload )

##### Description

Updates the url string stored in the AjaxDataHandler descendant and optionally reloads the data immediately.

**Note:** If the reload parameter is omitted or false, the data will not reload automatically and must be done manually via [EJSC.Series.reload\(\)](#)

#### 5.7.2.2.5 setUrlData

##### Definition

void [setUrlData](#)( string urlData, boolean reload )

##### Description

Updates the urlData stored in the AjaxDataHandler descendant and optionally reloads / reprocesses the data source and updates the series.

**Note:** If the reload parameter is omitted or false, the data will not reload automatically and must be done manually via [EJSC.Series.reload\(\)](#)

#### 5.7.2.2.6 setXMLData

##### Definition

void [setXMLData](#)( XMLHttpRequest request )

##### Description

This method allows passing of an already retrieved XMLHttpRequest object to an AjaxDataHandler descendant. When used in conjunction with the [onNeedsData](#) event, this method allows interaction with pre-existing 3rd party Ajax libraries instead of the request pool built into the chart library.

If the data handler is not currently in the loading state (i.e. [onNeedsData](#) was triggered), calling this method will have no effect.

### 5.7.2.3 Events

#### 5.7.2.3.1 onDataAvailable (inherited)

##### Definition

void [onDataAvailable\(\)](#)

##### Description

Placeholder for storing an event to fire after the a descendant class has loaded and processed its data.

**Important:** This event should be assigned by the EJSK.Series object.

#### 5.7.2.3.2 onDataReady

##### Definition

boolean [onDataReady](#)( XMLHttpRequest response, EJSK.AjaxDataHandler data\_handler, EJSK.Series series, EJSK.Chart chart)

##### Description

This event is fired when data has arrived completely and is ready for processing by the data handler. Returning true from this event will cause the data handler to process its data normally, returning false will cause processing to be canceled.

#### 5.7.2.3.3 onNeedsData

##### Definition

boolean [onNeedsData](#)( [EJSK.AjaxDataHandler](#) dataHandler, [EJSK.Series](#) series, [EJSK.Chart](#) chart )  
XMLHttpRequest [onNeedsData](#)( [EJSK.AjaxDataHandler](#) dataHandler, [EJSK.Series](#) series, [EJSK.Chart](#) chart )

##### Description

This event allows for additional control over when and how data is retrieved by the data handler. If assigned, the event will be triggered whenever the its owner series has requested the data handler retrieve its data. The event passes a reference to the current data handler, its owner series and the owner chart.

The result of this event is expected to be either a boolean value or an XMLHttpRequest object which has already been retrieved (responseXML is a valid, well-formed chart xml document).

##### RESULT:

- **true:** The data handler will ignore other properties such as url and requestType and hold itself in a loading state and wait for its setXMLData method to be called and provided a valid XMLHttpRequest object. In this manner, a separate Ajax library may be used to retrieve data.
- **false:** The data handler will ignore other properties such as url and requestType and cancel its

loading process.

- **XMLHttpRequest:** The data handler will ignore other properties such as url and requestType and immediately begin processing the data stored in responseXML.

### 5.7.3 EJSC.Axis

Base class for all Axis classes (i.e. [EJSC.LinearAxis](#), [EJSC.LogarithmicAxis](#)). This base class defines common properties, methods and events but does not implement any drawing or calculation routines. This class should not be used directly.

#### Constructor

none

#### 5.7.3.1 Properties

##### 5.7.3.1.1 background

#### Definition

```
object background = {
    color: "#fff",
    opacity: 0,
    includeTitle: false
};
```

#### Description

Defines the color and opacity of the axis area background. Setting the includeTitle property to true will fill the axis caption area as well as the ticks. If opacity is set to 0, no fill will occur.

##### 5.7.3.1.2 border

#### Definition

```
object border = {
    thickness: 1,
    color: undefined,
    opacity: 100,
    show: true
}
```

#### Description

Defines the appearance of the axis side bordering the chart area. By default the border color inherits the axis color property ([EJSC.Axis.color](#))

#### Example

>> Provide a 2 pixel tall green border on the bottom axis.

```
var chart = new EJSC.Chart(
```

```
        "chart",
        {
            axis_bottom:
                { border: { thickness: 2, color: "#0F0" } }
        }
    );
```

#### 5.7.3.1.3 caption

##### Definition

string **caption** = "Axis"

##### Description

Defines the text to be displayed below or beside the axis.

*For styling the caption, see [EJSC.Axis.caption\\_class](#)*

##### Example

>> *Customize the bottom axis caption.*

```
var chart = new EJSC.Chart(
    "chart",
    {
        axis_bottom: { caption: "Year" }
    }
);
```

#### 5.7.3.1.4 caption\_class

##### Definition

string **caption\_class** = ""

##### Description

Defines the CSS className to assign to the axis caption.

*For styling the tick labels, see [EJSC.Axis.label\\_class](#)*

##### Example

>> *Style the axis caption bold.*

```
<style> .AxisCaption { font-style: bold; } </style>

var chart = new EJSC.Chart(
    "chart",
    {
        axis_bottom: { caption_class: "AxisCaption" }
    }
);
```

## 5.7.3.1.5 color

**Definition**

string **color** = "#FFF"

**Description**

Defines the default color of the axis border and tick marks. If the sub properties such as `minor_ticks.color`, `major_ticks.color`, `border.color` are left undefined, they inherit the value set here.

**Example**

*>> Color the axis border and tick marks red*

```
var chart = new EJSC.Chart(  
    "chart",  
    {  
        axis_bottom: { color: "#F00" }  
    }  
);
```

## 5.7.3.1.6 crosshair

**Definition**

```
object crosshair = {  
    show: false,  
    color: "#F00"  
}
```

**Description**

Defines if crosshair should be shown on the chart at the current mouse coordinates as they relate to the given axis. May be enabled and disabled for each axis independently. This is automatically disabled for all axes if [EJSC.Chart.allow\\_interactivity](#) is set to false.

**Example**

*>> Show crosshair based on the bottom axis mouse position.*

```
var chart = new EJSC.Chart(  
    "chart",  
    {  
        axis_bottom: {  
            crosshair: { show: true }  
        }  
    }  
);
```

## 5.7.3.1.7 cursor\_position

**Definition**

```
object cursor_position = {  
  show: false,  
  color: "#F00",  
  textColor: "#FFF",  
  formatter: undefined,  
  caption: undefined,  
  className: undefined  
}
```

**Description**

Defines the display properties of the cursor position feature for an axis. These properties may be used to turn the cursor position indicator on and off as well as configure the styling and color.

**show:** determines whether or not to display the cursor position while the mouse is within the chart area

**color:** sets the background and line color

**textColor:** sets the text color

**formatter:** defines a formatter to use when displaying coordinates, if left undefined it will use the axis formatter

**caption:** defines text to use as a prefix to the current coordinate

**className:** a CSS class name to be used for additional styling

**Example**

>> *Turn on cursor position for the bottom axis and configure to display a related label*

```
var chart = new EJSC.Chart(  
  "chart",  
  {  
    axis_bottom: {  
      cursor_position: {  
        show: true,  
        caption: "Sales"  
      }  
    }  
  }  
);
```

## 5.7.3.1.8 extremes\_ticks

**Definition**

boolean **extremes\_ticks** = false

**Description**

Defines if the min and max values should be forced to land on the next tick mark.

## Example

>> Force tick marks at the min and max coordinates of the bottom axis (left and right sides of the chart)

```
var chart = new EJSC.Chart(
    "chart",
    {
        axis_bottom: { extremes_ticks: true }
    }
);
```

### 5.7.3.1.9 force\_static\_points

#### Definition

boolean [force\\_static\\_points](#) = false

#### Description

Defines if the chart should force ticks to match up to every point by converting the data to strings.

#### Example

>> Display every bottom axis data point, essentially disable auto axis scaling.

```
var chart = new EJSC.Chart(
    "chart",
    {
        axis_bottom: { force_static_points: true }
    }
);
```

### 5.7.3.1.10 formatter

#### Definition

[EJSC.Formatter](#) [formatter](#) = EJSC.Formatter

#### Description

Defines the formatter that will be used to format the tick marks on the axis before displaying them.

Types:

[EJSC.DateFormatter](#)

[EJSC.NumberFormatter](#)

[EJSC.StringFormatter](#)

#### Example

>> Display bottom axis tick labels as \$0.00

```
var chart = new EJSC.Chart(
```

```

        "chart",
        {
            axis_bottom: {
                formatter: new EJSC.NumberFormatter( { currency_symbol: "$",
forced_decimals: 2, variable_decimals: 2 } )
            }
        }
    );

```

#### 5.7.3.1.11 grid

##### Definition

```

object grid = {
    thickness: 1,
    color: "rgb(230,230,230)",
    opacity: 100,
    show: true
}

```

##### Description

Defines the appearance and visibility of the grid drawn in the background.

##### Example

*>> Do not draw the background grid for the top and right axes.*

```

var chart = new EJSC.Chart(
    "chart",
    {
        axis_top: {
            grid: { show: false }
        },
        axis_right: {
            grid: { show: false }
        }
    }
);

```

#### 5.7.3.1.12 hint\_caption

##### Definition

```

string hint_caption = "Value:"

```

##### Description

Defines the text to display in front of the axis-related value in the hint (leave blank to hide the value when displaying the hint).

##### Example

*>> Customize the value label in the hint window*



```

var chart = new EJSC.Chart(
    "chart",
    {
        axis_bottom: {
            hint_caption: "Month:"
        }
    }
);

```

#### 5.7.3.1.13 label\_class

##### Definition

string **label\_class** = ""

##### Description

Defines the CSS className to assign to the axis tick labels. Used in conjunction with [EJSC.Axis.stagger\\_ticks](#) and [EJSC.Axis.size](#), this property allows for greater control over the size and staggering of tick labels on the top and bottom axes.

*For styling the caption, see [EJSC.Axis.caption\\_class](#)*

##### Example

*>> Style the axis tick labels grey, make them 24 pixels high to enable text wrapping and enable two levels of tick staggering.*

```

<style> .AxisTickLabels { color: #999; height: 24px; } </style>

var chart = new EJSC.Chart(
    "chart",
    {
        axis_bottom: {
            label_class: "AxisTickLabels",
            size: 48
        }
    }
);

```

#### 5.7.3.1.14 major\_ticks

##### Definition

```

object major_ticks = {
    thickness: 1,
    size: 4,
    color: undefined,
    opacity: 100,
    show: true,
    count: undefined,
    offset: 0,
    min_interval: undefined,
    max_interval: undefined
}

```

```
}

```

### Description

This set of properties defines the characteristics of the major ticks for a given axis.

**thickness:** the height or width (depending on axis orientation) of the tick mark

**size:** the amount the tick mark extends from the axis border, may be specified as a number or a string containing % for a percentage

**color:** the color of the tick marks, if left undefined this property inherits its value from

[EJSC.Axis.color](#)

**opacity:** the opacity of the tick marks

**show:** specifies whether to draw the tick marks

**count:** the number of tick marks to draw, if left undefined the axis will determine the proper amount of ticks to draw automatically based on the range of data available to the axis

**Note:** This property is not compatible with text labels (i.e. x axis values are "Gizmos", "Widgets", instead of numbers)

**offset:** distance in pixels or percent from the axis border to begin drawing the tick marks,

**min\_interval:** Defines the minimum interval between major tick marks / labels. This will override the auto generation of ticks to cap the interval to the value when defined.

**max\_interval:** Defines the maximum interval between major tick marks / labels. This will override the auto generation of ticks to cap the interval to the value when defined.

#### 5.7.3.1.15 max\_extreme

### Definition

float **max\_extreme** = undefined

### Description

Defines the maximum value of the the axis. This will only affect the axis when set during chart creation and may be used to extend or truncate the value range displayed. To retrieve and set this value after the chart has been created, see [EJSC.Axis.getExtremes](#) and [EJSC.Axis.setExtremes](#)

### Example

*>> Force the axis range to extend beyond the data it contains*

```
var chart = new EJSC.Chart(
    "chart",
    {
        axis_bottom: {
            max_extreme: 500.00
        }
    }
);
```

#### 5.7.3.1.16 minor\_ticks

### Definition

```
object minor_ticks = {
  show: false,
  color: "rgb(0,0,0)",
  opacity: 20
  thickness: 1,
  count: 7,
  size: 4,
  offset: 0
}
```

### Description

Defines the properties of the minor tick marks to be drawn on the axis. The color, opacity and thickness (width of ticks in pixels) properties define the style of the tick marks. The count property defines the number of tick marks to be drawn between each major tick mark. The size property defines the height of the ticks (in pixels or percent). The offset property defines the distance away from the axis the minor tick marks begin drawing.

### Example

*>> Display red minor tick marks on the bottom axis*

```
var chart = new EJSC.Chart(
  "chart",
  {
    axis_bottom: {
      minor_ticks: { show: true, color: "#FF0000" }
    }
  }
);
```

#### 5.7.3.1.17 min\_extreme

### Definition

float **min** = undefined

### Description

Defines the minimum value of the the axis. This will only affect the axis when set during chart creation and may be used to extend or truncate the value range displayed. To retrieve and set this value after the chart has been created, see [EJSC.Axis.getExtremes](#) and [EJSC.Axis.setExtremes](#)

### Example

*>> Force the axis range to extend beyond the data it contains*

```
var chart = new EJSC.Chart(
  "chart",
  {
    axis_bottom: {
      min_extreme: -500.00
    }
  }
);
```

```
);
```

#### 5.7.3.1.18 size

##### Definition

integer **size** = 20

##### Description

Defines the height of horizontal axes or width of vertical axes (in pixels) of the tick area. To fully enable staggered ticks on horizontal axes, set this property to a multiple of 20 (or axis tick height), i.e. two levels = 40, three levels = 60.

For additional control over the format of the labels, see the [EJSC.Axis.label\\_class](#) property.

##### Example

*>> Make axis tick area twice as tall, enabling staggered ticks (default tick label height is 20 pixels)*

```
var chart = new EJSC.Chart(  
    "chart",  
    {  
        axis_bottom: { size: 40 }  
    }  
);
```

#### 5.7.3.1.19 stagger\_ticks

##### Definition

boolean **stagger\_ticks** = true

##### Description

Determines whether the axis tick labels are staggered.

**NOTE:** This property is only applicable to horizontal axes (i.e. [EJSC.Chart.axis\\_top](#) and [EJSC.Chart.axis\\_bottom](#))

##### Example

*>> Disable tick staggering for the bottom axis*

```
var chart = new EJSC.Chart(  
    "chart",  
    {  
        axis_bottom: { stagger_ticks: false }  
    }  
);
```

## 5.7.3.1.20 visible

**Definition**

boolean **visible** = true

**Description**

Defines if the axis (containing axis caption and tick labels) should be displayed.

**Example**

>> *Remove the bottom axis from the chart to allow additional room for data*

```
var chart = new EJSC.Chart(  
    "chart",  
    {  
        axis_bottom: { visible: false }  
    }  
);
```

## 5.7.3.1.21 zero\_plane

**Definition**

```
object zero_plane = {  
    color: "rgb(0,0,0)",  
    show: false,  
    opacity: 100,  
    thickness: 1,  
    coordinate: 0  
}
```

**Description**

Defines the properties of the zero plane line to be drawn at 0 (or whatever the coordinate property is set to). It is used to specify if the line should be shown as well as its color, thickness and opacity. The coordinate property allows the base of [EJSC.BarSeries](#), [EJSC.StackedBarSeries](#) and [EJSC.AreaSeries](#) changed.

**Example**

>> *Display a 2 pixel thick dark green line on the zero plane of the left axis*

```
var chart = new EJSC.Chart(  
    "chart",  
    {  
        axis_left: {  
            zero_plane: { show: true, color: "rgb(7,89,5)", thickness:  
2 }  
        }  
    }  
);
```

### 5.7.3.2 Methods

#### 5.7.3.2.1 addBin

##### Definition

void **addBin**( String label, boolean redraw )

##### Description

Adds a new static label to the axis and sets the appropriate flags if necessary to force the chart to draw using static labels (as opposed to a dynamic range based on series data).

This method is useful if there is a need to include bins which may not be part of any series added (i.e. chart labels should be "Apples", "Oranges", "Pears" but the series data only contains data pertaining to Apples and Pears).

If redraw is false, drawing will not occur immediately but will not be entirely prevented. Certain actions such as adding an additional series with redraw = true, or chart dimensions changing may still trigger the chart to draw. The default, if redraw is omitted is **true**.

##### Note:

Manually added bins may only be removed if there are no series currently utilizing them.

##### Example

*>> Add bins in a specific order to ensure they appear the same each time the chart is loaded, regardless of the order in which they appear in the data.*

```
var myChart = new EJSC.Chart("chart");
myChart.axis_bottom.addBin("Salesman 1");
myChart.axis_bottom.addBin("Salesman 2");
myChart.axis_bottom.addBin("Salesman 3");

var mySeries = myChart.addSeries(
    new EJSC.BarSeries(...)
);
```

#### 5.7.3.2.2 getExtremes

##### Definition

object **getExtremes**()

RETURNS:

{ min: float, max: float }

##### Description

Returns the current axis extreme values.

To set the extreme values, see [EJSC.Axis.setExtremes](#)

#### 5.7.3.2.3 getZoom

##### Definition

object **getZoom**( )

RETURNS:  
{ min: float, max: float }

##### Description

Returns the current zoom coordinates for a given axis in chart units.

To set the zoom coordinates, see [EJSC.Axis.setZoom](#)

#### 5.7.3.2.4 getZoomBoxCoordinates

##### Definition

void **getZoomBoxCoordinates**()

RETURNS:  
{ min: float, max: float }

##### Description

Returns the min and max values for a given axis of the current zoom box. These values are in chart coordinates, not pixels.

#### 5.7.3.2.5 hide

##### Definition

void **hide**( )

##### Description

Hides the axis, resizes the chart area and redraws all visible series.

No effect if the axis is already hidden.

#### 5.7.3.2.6 hideGrid

##### Definition

void **hideGrid**( boolean redraw )

##### Description

Hides the background grid for a given axis and if redraw is omitted or true, redraws the chart

No effect if the grid is already hidden.

#### 5.7.3.2.7 resetZoom

##### Definition

void **resetZoom**()

##### Description

Resets the zoom for a given axis to use its extreme values for min and max (as if a user had double-clicked the chart or drawn the zoom rectangle anywhere but down and right)

#### 5.7.3.2.8 pixelToPoint

##### Definition

float **pixelToPoint**( integer Pixel )

##### Description

Converts the pixel coordinate into chart units based on an axis. The result will be undefined if the pixel location is outside of the chart area.

#### 5.7.3.2.9 pointToPixel

##### Definition

integer **pointToPixel**( number coordinate )

##### Description

Converts the chart coordinates based on axis data into the appropriate pixel position for that point (x or y depending on the orientation of the axis)

The result will be undefined if the coordinate provided is outside the currently displayed range.

#### 5.7.3.2.10 removeBin

##### Definition

void **removeBin**( String label, boolean redraw )

##### Description

Removes a bin (static text label) from the axis.

If redraw is false, drawing will not occur immediately but will not be entirely prevented. Certain actions such as adding an additional series with redraw = true, or chart dimensions changing may still trigger



the chart to draw. The default, if redraw is omitted is **true**.

**Note:**

Bins may only be removed using this method if they were added using [EJSC.Axis.addBin](#) and no series are currently utilizing them.

5.7.3.2.11 setCaption

**Definition**

void [setCaption](#)( string caption )

**Description**

Updates the axis caption. This method must be used to change the caption once the chart has been rendered. To set a default caption on creation, see [EJSC.Axis.caption](#).

5.7.3.2.12 setCrosshair

**Definition**

void [setCrosshair](#)( boolean visible, float coordinate, boolean fireEvent )

**Description**

This method may be used to hide, show and position the cursor position indicator. If the fireEvent parameter is set to true or left undefined the [EJSC.Axis.onShowCrosshair](#) or [EJSC.Axis.onHideCrosshair](#) events will fire.

5.7.3.2.13 setExtremes

**Definition**

void [setExtremes](#)( float min, float max )

**Description**

Updates the manual extremes for the axis. Use this method if the series data does not span the range to be displayed on the chart or to limit 100% zoom to a range smaller than the series data.

**Example**

*>> Series data only spans 18 hours on the bottom axis but the chart needs to display an entire day*

```
var myChart = new EJSC.Chart( "chart" );
myChart.axis_bottom.setExtremes( 0, 86400000 );
```

5.7.3.2.14 setZoom

**Definition:**

void [setZoom](#)( float min, float max )

**Description**

Sets the current zoom of the axis to the specified coordinates.

5.7.3.2.15 show

**Definition**

void [show](#)( )

**Description**

Shows the axis, resizes the chart area and redraws all visible series.

No effect if the axis is already visible.

5.7.3.2.16 showGrid

**Definition**

void [showGrid](#)( boolean redraw )

**Description**

Shows the background grid for the axis and if redraw is omitted or true, redraws the chart.

No effect if the grid is already visible.

**5.7.3.3 Events**

5.7.3.3.1 onHideCrosshair

**Definition**

void [onHideCrosshair](#)( [EJSC.Axis](#) axis, [EJSC.Chart](#) chart )

**Description**

Called when the axis crosshair is hidden by the chart. This can occur when the user moves their mouse outside of the chart area or when [EJSC.Axis.setCrosshair](#) is called, sending false for the visible parameter and true for the fireEvent parameter.

5.7.3.3.2 onHideCursorPosition

**Definition**

void [onHideCursorPosition](#)( [EJSC.Axis](#) axis, [EJSC.Chart](#) chart )

**Description**

This event is fired when the cursor position indicator for an axis is hidden. This will occur when the user's mouse leaves the chart area.

#### 5.7.3.3.3 onNeedsTicks

### Definition

array **onNeedsTicks**( float min, float max, [EJSC.Axis](#) axis, [EJSC.Chart](#) chart)

### Description

This event is triggered whenever the axis ticks need to be redrawn / recalculated. It expects an array of [ float y, string label ] to be returned which defines exactly where to put the tick marks and labels. In addition, null may be returned in order to skip custom ticks for the current draw and use the chart's build in tick controls.

**min**: The current minimum value visible on the chart for the axis.

**max**: The current maximum y value visible on the chart for the axis.

**axis**: The axis which needs ticks.

**chart**: The chart that triggered the event.

### Notes

- To use the label formatter already assigned to the axis, set label to null (i.e. [min, null])
- To use on an axis with bins (text instead of numbers), simply send in the bin (i.e. ["First Bin", null])

### Example

A typical event handler may look like the following:

```
function doBottomAxisNeedsTicks(min, max, axis, chart) {

    // Display 3 tick marks, one at min, one at max and one directly in between
    var result = new Array();

    result.push( [min, null] );
    result.push( [min + ((max - min) / 2), null] );
    result.push( [max, null] );

    // Given a chart with a min and max of 0 and 100, the resulting array looks like:
    // [
    //   [0, null],
    //   [50, null],
    //   [100, null]
    // ]
    return result;
}
```

#### 5.7.3.3.4 onShowCrosshair

##### Definition

void **onShowCrosshair**( float coordinate, [EJSC.Axis](#) axis, [EJSC.Chart](#) chart )

##### Description

Called when the axis crosshair is shown by the chart. This can occur when the user moves their mouse into the chart area or when [EJSC.Axis.setCrosshair](#) is called, sending true for the visible parameter and true for the fireEvent parameter.

#### 5.7.3.3.5 onShowCursorPosition

##### Definition

void **onShowCursorPosition**( float coordinate, [EJSC.Axis](#) axis, [EJSC.Chart](#) chart )

##### Description

Called when the cursor position indicator becomes visible or changes position on a given axis. This will occur when the user enters the chart area or moves their mouse within the chart area.

### 5.7.4 EJSC.DataHandler

Base data handler class that defines properties, methods and events common to all data handler descendants. This class does not implement any actual data loading and should not be instantiated. All data handlers should descend from DataHandler or one of its descendants in order to function properly with the [EJSC.Series](#) descendants.

##### Constructor

none

#### 5.7.4.1 Methods

##### 5.7.4.1.1 loadData

##### Definition

void **loadData**( )

##### Description

Placeholder method which should be used to clear (if stored) and reload data. This method must be overridden in all descendant classes.

#### 5.7.4.2 Events

##### 5.7.4.2.1 onDataAvailable

##### Definition

void [onDataAvailable\(\)](#)

### Description

Placeholder for storing an event to fire after the a descendant class has loaded and processed its data.

**Important:** This event should be assigned by the `EJSC.Series` object.

## 5.7.5 EJSC.GaugeSeries

Base series class that holds all the generic information for each gauge series that is being created. All gauge type series should descend from this class or one of its descendants in order to function properly with the [EJSC.Chart](#) class.

### Constructor

none

### 5.7.5.1 Properties

5.7.5.1.1 `color` (inherited)

#### Definition

string `color` = undefined

#### Description

Defines the series color. If left undefined, the color is pulled from the array of default colors stored in the chart.

**Note:** May not be applicable to all series (i.e. [EJSC.PieSeries](#))

5.7.5.1.2 `coloredLegend` (inherited)

#### Definition

boolean `coloredLegend` = true

#### Description

Determines whether the legend text inherits the series color. Setting to false will allow the legend text to inherit its normal cascaded color. To modify this property after series creation see [EJSC.Series.setColoredLegend\(\)](#)

5.7.5.1.3 `delayLoad` (inherited)

### EXPERIMENTAL

#### Definition

boolean `delayLoad` = true

**Description**

This property allows a series to force its data handler to begin data retrieval as soon as it is added to a chart. When set to false, the series will initiate data retrieval as soon as it is added to a chart. When true, the data retrieval is initialized when the series first needs to draw.

## 5.7.5.1.4 legendsVisible (inherited)

**Definition**

boolean **legendsVisible** = true

**Description**

Determines whether the legend item associated with the series appears in the legend window. Use the `Series.showLegend` and `Series.hideLegend` methods to control legend item visibility after series creation.

## 5.7.5.1.5 lineOpacity (inherited)

**Definition**

integer **lineOpacity** = 100

**Description**

Determines the line opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see [EJSC.Series.setLineOpacity\(\)](#)

## 5.7.5.1.6 lineWidth (inherited)

**Definition**

integer **lineWidth** = 1

**Description**

Defines the width of the line connecting series points.

## 5.7.5.1.7 opacity (inherited)

**Definition**

integer **opacity** = 50

**Description**

Determines the fill opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see [EJSC.Series.setOpacity\(\)](#)

## 5.7.5.1.8 title (inherited)

**Definition**

string **title** = "Series <index>"

**Description**

Defines the series title used in hints and legend display.

**Note:** The default title of Series <index> (where <index> equals the current position in the series array for its owner chart) is assigned when a series is added to a chart using [addSeries](#). This default title is only applied if the title is blank (i.e. "") at the time the addSeries method is called.

## 5.7.5.1.9 visible (inherited)

**Definition**

boolean **visible** = true

**Description**

Defines whether the series is visible and can draw on the chart

## 5.7.5.1.10 x\_axis\_formatter (inherited)

**Definition**

string **x\_axis\_formatter** = undefined

**Description**

Defines the formatter that will be used to format hints for the X values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:

[EJSC.DateFormatter](#)

[EJSC.NumberFormatter](#)

**5.7.5.2 Methods**

## 5.7.5.2.1 getDataHandler (inherited)

**Definition**

EJSC.DataHandler **getDataHandler**( )

**Description**

Returns the EJSC.DataHandler descendant currently assigned to the series. This is not applicable to all series and will return null if a data handler has not been defined or is not used.

## 5.7.5.2.2 getVisibility (inherited)

**Definition**

boolean **getVisibility**( )

**Description**

Returns a boolean indicating the series current visible state

## 5.7.5.2.3 hide (inherited)

**Definition**

void **hide**( )

**Description**

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already hidden.

## 5.7.5.2.4 hideLegend (inherited)

**Definition**

void **hideLegend**( )

**Description**

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

## 5.7.5.2.5 reload (inherited)

**Definition**

void **reload**( )

**Description**

Triggers a series reload if the functionality is defined in a child class. This may involve retrieving data again or simply recalculating. While the reload() method exists in the base Series class, implementation of the protected methods triggered by calling reload() is at the discretion of the child class.

## 5.7.5.2.6 setColor (inherited)

**Definition**



void **setColor**( string color )

#### **Description**

Changes the series color and causes the chart to redraw.

#### 5.7.5.2.7 setColoredLegend (inherited)

#### **Definition**

void **setColoredLegend**( boolean coloredLegend )

#### **Description**

Sets the [EJSC.Series.coloredLegend](#) property and updates the legend to reflect the change.

#### 5.7.5.2.8 setDataHandler (inherited)

#### **Definition**

void **setDataHandler**( [EJSC.DataHandler](#) dataHandler, boolean reload )

#### **Description**

Updates the series data handler and triggers a data reload if reload parameter is **true**. This method may not be implemented in all child classes.

#### 5.7.5.2.9 setLineOpacity (inherited)

#### **Definition**

void **setLineOpacity**( integer opacity )

#### **Description**

Sets the [EJSC.Series.lineOpacity](#) property and redraws the series to reflect the change.

#### 5.7.5.2.10 setLineWidth (inherited)

#### **Definition**

void **setLineWidth**( integer width )

#### **Description**

Updates the [lineWidth](#) property and redraws the chart.

#### 5.7.5.2.11 setOpacity (inherited)

#### **Definition**

void **setOpacity**( integer opacity )

**Description**

Sets the [EJSC.Series.opacity](#) property and redraws the series to reflect the change.

5.7.5.2.12 setTitle (inherited)

**Definition**

void [setTitle](#)( string title )

**Description**

Changes the series title and update the legend caption (if applicable).

**Note:** this will not cause updates to a hint which is visible at the time it is called, though the next time a hint is rendered it will show the new property value.

5.7.5.2.13 show (inherited)

**Definition**

void [show](#)( )

**Description**

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already visible.

5.7.5.2.14 showLegend (inherited)

**Definition**

void [showLegend](#)( )

**Description**

Changes the series legend item visible state.

No effect if the series legend item is already visible.

**5.7.5.3 Events**

5.7.5.3.1 onAfterDataAvailable (inherited)

**Definition**

boolean [onAfterDataAvailable](#)( [EJSC.Chart](#) chart, [EJSC.Series](#) series )

**Description**

Fired after the series has processed its data and before the chart is told to redraw. Returning **false** from this event handler will cancel redrawing the chart.

#### 5.7.5.3.2 onAfterVisibilityChange (inherited)

##### Definition

boolean **onAfterVisibilityChange**( [EJSC.Series](#) series, [EJSC.Chart](#) chart, boolean visible )

##### Description

Fired when the series visibility changes. Sends series which changed, its owner chart and its new visible property value.

Returning **false** from this event handler will cancel redrawing the chart.

#### 5.7.5.3.3 onBeforeVisibilityChange (inherited)

##### Definition

boolean **onBeforeVisibilityChange**( [EJSC.Series](#) series, [EJSC.Chart](#) chart )

##### Description

Fired before the series visibility is about to change. Sends series which changed, its owner chart. The current visible state of the series may be checked using the series.visible property.

Returning **false** from this event handler will cancel the change in visibility.

## 5.7.6 EJSC.Formatter

The top level Formatter class is used as a base for all label formatters. It defines a `format_string` property as a guide for descendants. The `format` method is required to be overridden in all descendant classes in order to function properly when used with the chart classes.

### Constructor

`EJSC.Formatter()`

### 5.7.6.1 Properties

#### 5.7.6.1.1 format\_string

##### Definition

string **format\_string** = undefined

##### Description

Defines the format string. Classes that descend from `EJSC.Formatter` may use this property to store their custom format strings.

### 5.7.6.2 Methods

#### 5.7.6.2.1 format

##### Definition

string **format**( float value, integer precision )

##### Description

Format is called when a value needs to be formatted for display in the axis, hint, cursor position, etc.

The behavior of the format method is to round a numeric value to the precision specified. Descendants do not need to implement the precision parameter.

### 5.7.7 EJSC.Point

Base point class that holds all the generic information for each point that is being created. All points should descend from this class or one of its descendants in order to function properly with the [EJSC.Chart](#) and [EJSC.Series](#) (and descendant) classes.

##### Constructor

none

#### 5.7.7.1 Properties

##### 5.7.7.1.1 label

##### Definition

string **label** = null

##### Description

Defines the point label. Currently this is only implemented in [EJSC.PieSeries](#) [EJSC.PiePoint](#) objects.

##### 5.7.7.1.2 userdata

##### Definition

string **userdata** = null

##### Description

Stores a user-defined string related to the point. This may be used to hold information such as database id values, drill down urls or any other related data. Currently only supported by the [EJSC.XMLDataHandler](#) (full and short formats).

## 5.7.7.1.3 x

**Definition**

float **x** = null

**Description**

Defines the point X value. This may be mapped to a text label automatically by the data handler in instances where the point data is not numeric.

## 5.7.7.1.4 y

**Definition**

float **y** = null

**Description**

Defines point Y value.

## 5.7.8 EJSC.Series

Base series class that holds all the generic information for each series that is being created. All series should descend from this class or one of its descendants in order to function properly with the [EJSC.Chart](#) class.

**Constructor**

none

### 5.7.8.1 Properties

## 5.7.8.1.1 autosort

**Definition**

boolean **autosort** = true

**Description**

Defines whether the series applies the appropriate sort to points prior to drawing. Drawing routines are generally optimized to accept data in a certain order and this property eliminates the need for the developer to worry about that order prior to sending data to the series. If set to false, the data must be properly ordered beforehand in order to ensure the series renders correctly.

**Note:** May not be applicable to all series (i.e. [EJSC.PieSeries](#), [EJSC.AnalogGaugeSeries](#))

## 5.7.8.1.2 color

**Definition**

string **color** = undefined

### Description

Defines the series color. If left undefined, the color is pulled from the array of default colors stored in the chart.

**Note:** May not be applicable to all series (i.e. [EJSC.PieSeries](#))

#### 5.7.8.1.3 coloredLegend

### Definition

boolean **coloredLegend** = true

### Description

Determines whether the legend text inherits the series color. Setting to false will allow the legend text to inherit its normal cascaded color. To modify this property after series creation see [EJSC.Series.setColoredLegend\(\)](#)

#### 5.7.8.1.4 delayLoad

### EXPERIMENTAL

### Definition

boolean **delayLoad** = true

### Description

This property allows a series to force its data handler to begin data retrieval as soon as it is added to a chart. When set to false, the series will initiate data retrieval as soon as it is added to a chart. When true, the data retrieval is initialized when the series first needs to draw.

#### 5.7.8.1.5 hint\_string

### Definition

string **hint\_string** = undefined

### Description

This property may be used to define a series specific string to be used when displaying point hints. This string may contain replacement indicators (see [Text Replacement Options](#)). When left undefined, a default hint defined by the series type will be displayed.

### Example

*>> Modify the default hint to include custom text.*

```
var series = new EJSC.LineSeries(new EJSC.ArrayDataHandler(),  
    {
```

```
        hint_string: "My Custom Hint  
</strong>[y]"  
    }  
);
```

#### 5.7.8.1.6 legendsVisible

##### Definition

boolean **legendsVisible** = true

##### Description

Determines whether the legend item associated with the series appears in the legend window. Use the `Series.showLegend` and `Series.hideLegend` methods to control legend item visibility after series creation.

#### 5.7.8.1.7 lineOpacity

##### Definition

integer **lineOpacity** = 100

##### Description

Determines the line opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see [EJSC.Series.setLineOpacity\(\)](#)

#### 5.7.8.1.8 lineWidth

##### Definition

integer **lineWidth** = 1

##### Description

Defines the width of the line connecting series points.

#### 5.7.8.1.9 opacity

##### Definition

integer **opacity** = 50

##### Description

Determines the fill opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see [EJSC.Series.setOpacity\(\)](#)

## 5.7.8.1.10 title

**Definition**

string **title** = "Series <index>"

**Description**

Defines the series title used in hints and legend display.

**Note:** The default title of Series <index> (where <index> equals the current position in the series array for its owner chart) is assigned when a series is added to a chart using [addSeries](#). This default title is only applied if the title is blank (i.e. "") at the time the addSeries method is called.

## 5.7.8.1.11 visible

**Definition**

boolean **visible** = true

**Description**

Defines whether the series is visible and can draw on the chart

## 5.7.8.1.12 x\_axis

**Definition**

string **x\_axis** = "bottom"

**Description**

Defines the horizontal axis to use for X values. Valid options are "top" or "bottom".

## 5.7.8.1.13 x\_axis\_formatter

**Definition**

string **x\_axis\_formatter** = undefined

**Description**

Defines the formatter that will be used to format hints for the X values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:

[EJSC.DateFormatter](#)

[EJSC.NumberFormatter](#)



## 5.7.8.1.14 y\_axis

**Definition**

string **y\_axis** = "left"

**Description**

Defines the vertical axis to use for Y values. Valid options are "left" or "right".

## 5.7.8.1.15 y\_axis\_formatter

**Definition**

string **y\_axis\_formatter** = undefined

**Description**

Defines the formatter that will be used to format hints for the Y values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:

[EJSC.DateFormatter](#)

[EJSC.NumberFormatter](#)

**5.7.8.2 Methods**

## 5.7.8.2.1 findClosestPoint

**Definition**

[EJSC.Point](#) **findClosestPoint**( integer pixel )

**Description**

Returns the point object closest to the pixel coordinate given or null if no points are found.

## 5.7.8.2.2 getDataHandler

**Definition**

EJSC.DataHandler **getDataHandler**( )

**Description**

Returns the EJSC.DataHandler descendant currently assigned to the series. This is not applicable to all series and will return null if a data handler has not been defined or is not used.

## 5.7.8.2.3 getVisibility

**Definition**

boolean **getVisibility**( )

**Description**

Returns a boolean indicating the series current visible state

5.7.8.2.4 hide

**Definition**

void **hide**( )

**Description**

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already hidden.

5.7.8.2.5 hideLegend

**Definition**

void **hideLegend**( )

**Description**

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

5.7.8.2.6 reload

**Definition**

void **reload**( )

**Description**

Triggers a series reload if the functionality is defined in a child class. This may involve retrieving data again or simply recalculating. While the reload() method exists in the base Series class, implementation of the protected methods triggered by calling reload() is at the discretion of the child class.

5.7.8.2.7 setColor

**Definition**

void **setColor**( string color )

**Description**

Changes the series color and causes the chart to redraw.

## 5.7.8.2.8 setColoredLegend

**Definition**

void **setColoredLegend**( boolean coloredLegend )

**Description**

Sets the [EJSC.Series.coloredLegend](#) property and updates the legend to reflect the change.

## 5.7.8.2.9 setDataHandler

**Definition**

void **setDataHandler**( [EJSC.DataHandler](#) dataHandler, boolean reload )

**Description**

Updates the series data handler and triggers a data reload if reload parameter is **true**. This method may not be implemented in all child classes.

## 5.7.8.2.10 setLineOpacity

**Definition**

void **setLineOpacity**( integer opacity )

**Description**

Sets the [EJSC.Series.lineOpacity](#) property and redraws the series to reflect the change.

## 5.7.8.2.11 setLineWidth

**Definition**

void **setLineWidth**( integer width )

**Description**

Updates the [lineWidth](#) property and redraws the chart.

## 5.7.8.2.12 setOpacity

**Definition**

void **setOpacity**( integer opacity )

**Description**

Sets the [EJSC.Series.opacity](#) property and redraws the series to reflect the change.

## 5.7.8.2.13 setTitle

**Definition**

void [setTitle](#)( string title )

**Description**

Changes the series title and update the legend caption (if applicable).

**Note:** this will not cause updates to a hint which is visible at the time it is called, though the next time a hint is rendered it will show the new property value.

## 5.7.8.2.14 show

**Definition**

void [show](#)( )

**Description**

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already visible.

## 5.7.8.2.15 showLegend

**Definition**

void [showLegend](#)( )

**Description**

Changes the series legend item visible state.

No effect if the series legend item is already visible.

**5.7.8.3 Events**

## 5.7.8.3.1 onAfterDataAvailable

**Definition**

boolean [onAfterDataAvailable](#)( [EJSC.Chart](#) chart, [EJSC.Series](#) series )

**Description**

Fired after the series has processed its data and before the chart is told to redraw. Returning **false** from this event handler will cancel redrawing the chart.

#### 5.7.8.3.2 onAfterVisibilityChange

##### Definition

boolean **onAfterVisibilityChange**( [EJSC.Series](#) series, [EJSC.Chart](#) chart, boolean visible )

##### Description

Fired when the series visibility changes. Sends series which changed, its owner chart and its new visible property value.

Returning **false** from this event handler will cancel redrawing the chart.

#### 5.7.8.3.3 onBeforeVisibilityChange

##### Definition

boolean **onBeforeVisibilityChange**( [EJSC.Series](#) series, [EJSC.Chart](#) chart )

##### Description

Fired before the series visibility is about to change. Sends series which changed, its owner chart. The current visible state of the series may be checked using the series.visible property.

Returning **false** from this event handler will cancel the change in visibility.

#### 5.7.8.3.4 onShowHint

##### Definition

string **onShowHint**( [EJSC.Point](#) point, [EJSC.Series](#) series, [EJSC.Chart](#) chart, string hint\_string )

##### Description

Fired before the displaying the hint, this event allows the current hint string to be overridden. See [Text Replacement Options](#) for a list of the automatic substitutions available.

## 5.8 Other Classes

### 5.8.1 EJSC.BarPoint

BarPoint is used to store an X,Y point value (and optionally a label and/or user defined data). This is used in the [EJSC.BarSeries](#) class to store data for each bar to be drawn on the chart. This object is created automatically once data is made available via a [EJSC.DataHandler](#) descendant. BarPoint objects should generally not be created manually.

The constructor expects the X and Y value as well as the [EJSC.BarSeries](#) which owns the point and optionally (leave null if not used) a label and a userdata string value.

##### Constructor

```
EJSC.BarPoint( float x, float y, string label, string userdata, EJSC.BarSeries  
owner )
```

### 5.8.1.1 Properties

#### 5.8.1.1.1 label (inherited)

##### Definition

string **label** = null

##### Description

Defines the point label. Currently this is only implemented in [EJSC.PieSeries](#) [EJSC.PiePoint](#) objects.

#### 5.8.1.1.2 userdata (inherited)

##### Definition

string **userdata** = null

##### Description

Stores a user-defined string related to the point. This may be used to hold information such as database id values, drill down urls or any other related data. Currently only supported by the [EJSC.XMLDataHandler](#) (full and short formats).

#### 5.8.1.1.3 x (inherited)

##### Definition

float **x** = null

##### Description

Defines the point X value. This may be mapped to a text label automatically by the data handler in instances where the point data is not numeric.

#### 5.8.1.1.4 y (inherited)

##### Definition

float **y** = null

##### Description

Defines point Y value.

## 5.8.2 EJSC.FloatingBarPoint

FloatingBarPoint is used to store X and min and max Y, or Y and min and max X values for use with a

[EJSC.FloatingBarSeries](#). This information is used by the [EJSC.FloatingBarSeries](#) class to store data for each bar to be drawn on the chart. This object is created automatically once data is made available via a [EJSC.DataHandler](#) descendant. FloatingBarPoint objects should generally not be created manually.

The constructor expects the X, Y, min and max values as well as the [EJSC.BarSeries](#) which owns the point and optionally (leave null if not used) a label and a userdata string value.

### Constructor

```
EJSC.FloatingBarPoint( float x, float y, float min, float max, string label,
string userdata, EJSC.FloatingBarSeries owner )
```

## 5.8.2.1 Properties

### 5.8.2.1.1 label (inherited)

#### Definition

string **label** = null

#### Description

Defines the point label. Currently this is only implemented in [EJSC.PieSeries](#) [EJSC.PiePoint](#) objects.

### 5.8.2.1.2 max

#### Definition

float **max** = null

#### Description

Defines point max value (x or y depending on the series orientation).

### 5.8.2.1.3 min

#### Definition

float **min** = null

#### Description

Defines point min value (x or y depending on the series orientation).

### 5.8.2.1.4 userdata (inherited)

#### Definition

string **userdata** = null

#### Description

Stores a user-defined string related to the point. This may be used to hold information such as database id values, drill down urls or any other related data. Currently only supported by the [EJSC.XMLDataHandler](#) (full and short formats).

5.8.2.1.5 x (inherited)

#### Definition

float **x** = null

#### Description

Defines the point X value. This may be mapped to a text label automatically by the data handler in instances where the point data is not numeric.

5.8.2.1.6 y (inherited)

#### Definition

float **y** = null

#### Description

Defines point Y value.

### 5.8.3 EJSC.GaugePoint

GaugePoint is used to store an X point value (and optionally a label). This is used in the [EJSC.GaugeSeries](#) class to store data the gauge indicator. This object is created automatically once data is made available via a [EJSC.DataHandler](#) descendant. GaugePoint objects should generally not be created manually.

The constructor expects the X value as well as the [EJSC.GaugeSeries](#) which owns the point and optionally (leave null if not used) a label string value.

#### Constructor

```
EJSC.GaugePoint( float x, string label, EJSC.GaugeSeries owner )
```

#### 5.8.3.1 Properties

5.8.3.1.1 label (inherited)

#### Definition

string **label** = null

#### Description

Defines the point label. Currently this is only implemented in [EJSC.PieSeries](#) [EJSC.PiePoint](#) objects.



#### 5.8.3.1.2 userdata (inherited)

##### Definition

string **userdata** = null

##### Description

Stores a user-defined string related to the point. This may be used to hold information such as database id values, drill down urls or any other related data. Currently only supported by the [EJSC.XMLDataHandler](#) (full and short formats).

#### 5.8.3.1.3 x (inherited)

##### Definition

float **x** = null

##### Description

Defines the point X value. This may be mapped to a text label automatically by the data handler in instances where the point data is not numeric.

### 5.8.4 EJSC.PiePoint

PiePoint is used to store an X point value (and optionally a label value). This is used in the [EJSC.PieSeries](#) class store data for each pie piece to be drawn on the chart. This object is created automatically by once data is made available via a [EJSC.DataHandler](#) descendant. PiePoint objects should generally not be created manually.

The constructor expects the X value as well as the [EJSC.PieSeries](#) which owns the point and optionally (leave null if not used) label and userdata string values.

##### Constructor

```
EJSC.PiePoint( number x, string label, string userdata, EJSC.PieSeries owner )
```

#### 5.8.4.1 Properties

##### 5.8.4.1.1 label (inherited)

##### Definition

string **label** = null

##### Description

Defines the point label. Currently this is only implemented in [EJSC.PieSeries](#) [EJSC.PiePoint](#) objects.

#### 5.8.4.1.2 userdata (inherited)

##### Definition

string **userdata** = null

##### Description

Stores a user-defined string related to the point. This may be used to hold information such as database id values, drill down urls or any other related data. Currently only supported by the [EJSC.XMLDataHandler](#) (full and short formats).

#### 5.8.4.1.3 x (inherited)

##### Definition

float **x** = null

##### Description

Defines the point X value. This may be mapped to a text label automatically by the data handler in instances where the point data is not numeric.

### 5.8.5 EJSC.StockPoint

StockPoint is used to store the data necessary to display stock-related series such as [EJSC.CandlestickSeries](#) and [EJSC.OpenHighLowCloseseries](#). The standard properties are an X point value which defines where along the horizontal axis the point will appear, and the open, close, high and low values used to render the series in the chart. In addition to the standard properties, each point can also store a label and user defined data which may be utilized later during user interaction with the series. StockPoint objects should generally not be created manually.

The constructor expects the X value as well as the open, close, high, and low values, a series which owns the point and optionally (leave null if not used) a label and a userdata string value.

##### Constructor

```
EJSC.StockPoint( float x, float high, float low, float open, float close,  
string label, string userdata, EJSC.Series owner )
```

#### 5.8.5.1 Properties

##### 5.8.5.1.1 close

Enter topic text here.

##### 5.8.5.1.2 high

Enter topic text here.

##### 5.8.5.1.3 label (inherited)

##### Definition

string **label** = null

#### Description

Defines the point label. Currently this is only implemented in [EJSC.PieSeries](#) [EJSC.PiePoint](#) objects.

#### 5.8.5.1.4 low

Enter topic text here.

#### 5.8.5.1.5 open

Enter topic text here.

#### 5.8.5.1.6 userdata (inherited)

#### Definition

string **userdata** = null

#### Description

Stores a user-defined string related to the point. This may be used to hold information such as database id values, drill down urls or any other related data. Currently only supported by the [EJSC.XMLDataHandler](#) (full and short formats).

#### 5.8.5.1.7 x (inherited)

#### Definition

float **x** = null

#### Description

Defines the point X value. This may be mapped to a text label automatically by the data handler in instances where the point data is not numeric.

## 5.8.6 EJSC.XYPoint

XYPoint is used to store a X,Y point value. This is used in series such as [EJSC.LineSeries](#), [EJSC.AreaSeries](#), [EJSC.ScatterSeries](#) and [EJSC.BarSeries](#) to store each point to be drawn on the chart. This object is created automatically by each series once data is made available via a [EJSC.DataHandler](#) descendant. XYPoint objects should generally not be created manually.

The constructor expects the X and Y values as well as the [EJSC.Series](#) which owns the point and an optional (leave null if not used) userdata string value.

#### Constructor

```
EJSC.XYPoint( number x, number y, string userdata, EJSC.Series owner )
```

### 5.8.6.1 Properties

#### 5.8.6.1.1 label (inherited)

**Definition**

string **label** = null

**Description**

Defines the point label. Currently this is only implemented in [EJSC.PieSeries](#) [EJSC.PiePoint](#) objects.

#### 5.8.6.1.2 userdata (inherited)

**Definition**

string **userdata** = null

**Description**

Stores a user-defined string related to the point. This may be used to hold information such as database id values, drill down urls or any other related data. Currently only supported by the [EJSC.XMLDataHandler](#) (full and short formats).

#### 5.8.6.1.3 x (inherited)

**Definition**

float **x** = null

**Description**

Defines the point X value. This may be mapped to a text label automatically by the data handler in instances where the point data is not numeric.

#### 5.8.6.1.4 y (inherited)

**Definition**

float **y** = null

**Description**

Defines point Y value.

## 5.9 Using Colors

**Using Colors**

Colors may be specified in any of the following formats:

RGB: `"rgb( <red>, <green>, <blue> )"`

*example: "rgb(255,0,0)"*

RGBA: `"rgba( <red>, <green>, <blue>, <opacity> )"`

*example: "rgba(255,0,0,50)"*

HEX: `"#<red><green><blue>"`

*example: "#FF0000"*

SHORT HEX: `"#<red><green><blue>"`

*example "#F00"*

## 5.10 Text Replacement Options

Each series has its own set of text replacement options.

## 5.11 Exporting To SVG

### Including SVG Export Functionality

To add the SVG Export functionality to your current implementation, simply include the file immediately after the EJSChart.js

```
<head>
  <script type="text/javascript" src="/EJSChart/EJSChart.js"></script>
  <script type="text/javascript" src="/EJSChart/EJSChart_SVGExport.js"></script>
</head>
```

### Exporting SVG

Exporting SVG from your chart is as simple as making a single JavaScript call. The following example shows how to call the [exportSVG](#) method and obtain the resulting SVG. The default configuration will produce complete SVG including headers and sized to match the dimensions of source chart. If additional customization is needed, the options parameter allows for settings to be changed at the time of export.

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

  var chart = new EJSC.Chart("myChart", {});
  var series = new EJSC.LineSeries(
    new EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,2],[5,1]]),
    {}
  );
  chart.addSeries(series);

  function buttonClick() {
    alert(chart.exportSVG());
  }
</script>
```

```
</script>  
<button onclick="buttonClick();">Export SVG</button>
```

## 6 Getting Support

There are a wide variety of technical support options available for Emprise software products. For users who have purchased the Developer or Enterprise editions, and customers with maintenance plans, support options may include Priority E-mail Support and our Technical Support Hotline. Technical support documents, help files and access to our support forums are available to all users of Emprise software products.

### Documentation

Emprise JavaScript Charts documentation is available in several formats:

- [Online Web Documentation \(Searchable\)](#)

Downloadable Formats:

- [PDF Manual](#)
- [Windows HTML Help \(.chm\)](#)
- [Classic Windows Help \(.hlp\)](#)

### Example Charts

Our website contains a wide variety of example charts to assist you in creating and customizing your own. All examples include full JavaScript and data descriptions as well as links to the relevant help files for each property used.

- [Line Chart](#)
- [Area Chart](#)
- [Bar Chart](#)
- [Pie Chart](#)
- [More...](#)

### Support Forums

Share questions, suggestions, and information about your Emprise software products with other

---

users and the Emprise support and development staff in our [Support Forums](#).

## Contact Us

If you cannot find an answer to your question from the resources listed above, send our support department a message and we will get back to you as soon as we can.

### General Information & Questions

[info@ejschart.com](mailto:info@ejschart.com)

### Sales Related Inquiries

[sales@ejschart.com](mailto:sales@ejschart.com)

### Technical Support

[support@ejschart.com](mailto:support@ejschart.com)

### Phone

Sales: +1 (860) 464-8555

Support: (See support contract)

### Mailing Address:

PO Box 129

Ledyard, CT 06339

USA