

JSChartsTM

Developer's Guide



© 2008 Emprise Corporation. All Rights Reserved.
For more information on this and other products visit:
www.emprisecorporation.com

Table of Contents

Foreword	0
Part I Getting Started	30
1 Overview	31
2 Implementation Instructions.....	33
3 Creating Your First Chart.....	35
4 Customizing Your Chart.....	40
5 Anatomy of a Chat.....	41
Y Axis	42
X Axis	43
Hint	44
Chart Body	45
Legend	46
6 Version 1.x Compatability.....	46
Deprecated Properties, Methods and Events	46
7 Deployment.....	48
Part II Changes	48
1 Changes in version 2.3.....	48
2 Changes in version 2.1.....	49
3 Changes in version 2.0.1.....	49
4 Changes in version 2.0.....	50
5 Changes in version 1.3.1.....	51
6 Changes in version 1.3.....	51
7 Changes in version 1.2.1.....	52
8 Changes in version 1.2.....	52
9 Changes in version 1.1.....	54
10 Changes in version 1.0.1.....	55
11 Changes in version 1.0.....	56
Part III Data Formats	57
1 XML - Full	58
2 XML - Short.....	59
3 XML - Compact.....	59
Part IV API Reference	60
1 EJSC	60
Properties	60
DefaultImagePath.....	60
DefaultColors.....	60

DefaultBarColors.....	61
DefaultPieColors.....	61
STRINGS.....	62
2 EJSC.Chart	62
Properties	63
allow_interactivity.....	63
allow_mouse_wheel_zoom.....	63
allow_move.....	64
allow_zoom.....	64
allow_hide_error.....	64
auto_find_point_by_x.....	65
auto_resize.....	65
auto_zoom.....	65
axis_bottom.....	66
axis_left	67
axis_right.....	67
axis_top	68
background.....	69
building_message.....	69
drawing_message.....	69
legend_state.....	69
legend_title.....	70
max_zoom_message.....	70
message_timeouts.....	70
proximity_snap.....	70
show_hints.....	71
show_legend.....	71
show_messages.....	71
show_titlebar.....	72
title	72
Methods	72
acquireSeries.....	72
addSeries.....	73
clearSelectedPoint.....	73
exportSVG.....	74
exportSVGLegend.....	75
findClosestPoint.....	76
getLegendState.....	77
getMinMaxYInXRange.....	77
getSeries.....	77
getZoomBoxPixelCoordinates.....	77
hideTitlebar	78
hideZoomBox.....	78
legendMinimize.....	78
legendRestore.....	78
redraw	78
remove	79
removeSeries.....	79
selectPoint.....	79
setAutoResize.....	79
setLegendTitle.....	79
setShowLegend	80
setTitle	80
showTitlebar.....	80

showZoomBox.....	80
Events	81
onAfterBuild.....	81
onAfterDraw.....	81
onAfterDrawSeries.....	81
onAfterMove.....	81
onAfterSelectPoint.....	82
onAfterUnselectPoint.....	82
onAfterZoom.....	82
onBeforeBuild.....	82
onBeforeDbClick.....	83
onBeforeDraw.....	83
onBeforeDrawSeries.....	83
onBeforeSelectPoint.....	83
onBeforeUnselectPoint.....	84
onContextMenu.....	84
onDbClickPoint.....	84
onShowHint.....	84
onShowMessage.....	85
onUserBeginZoom.....	85
onUserEndZoom.....	85
Deprecated	85
Properties.....	85
force_static_points.....	85
force_static_points_x.....	86
force_static_points_y.....	86
legendTitle	86
show_crosshairs.....	86
show_grid	86
show_mouse_position.....	86
show_x_axis	86
show_y_axis	86
x_axis_caption	86
x_axis_className.....	86
x_axis_extremes_ticks.....	86
x_axis_formatter.....	86
x_axis_max_tick_interval.....	86
x_axis_min_tick_interval.....	86
x_axis_minor_ticks.....	86
x_axis_size	86
x_axis_stagger_ticks.....	87
x_axis_tick_className.....	87
x_axis_tick_count.....	87
x_cursor_position_caption.....	87
x_cursor_position_formatter.....	87
x_max	87
x_min	87
x_value_hint_caption.....	87
x_zero_plane	87
y_axis_caption	87
y_axis_className.....	87
y_axis_extremes_ticks.....	87
y_axis_formatter.....	87
y_axis_max_tick_interval.....	87

y_axis_min_tick_interval.....	87
y_axis_minor_ticks.....	87
y_axis_size	88
y_axis_tick_className.....	88
y_axis_tick_count.....	88
y_cursor_position_caption.....	88
y_cursor_position_formatter.....	88
y_max	88
y_min	88
y_value_hint_caption.....	88
y_zero_plane	88
Methods	88
addXAxisBin	88
addYAxisBin	88
convertPixelToPoint.....	88
convertPointToPixel.....	88
findClosestPointInSeries.....	88
hideGrid	88
hideXAxis	89
hideYAxis	89
getXExtremes	89
getYExtremes	89
getZoom	89
removeXAxisBin.....	89
removeYAxisBin.....	89
selectClosestPoint.....	89
setCrosshairs	89
setXAxisCaption.....	89
setXExtremes	89
setYAxisCaption.....	89
setYExtremes	89
setZoom	89
showGrid	89
showXAxis	89
showYAxis	90
Events	90
onAfterShowCrosshairs.....	90
onBeforeBeginZoom.....	90
onBeforeEndZoom.....	90
onShowCrosshairs.....	90
onXAxisNeedsTicks.....	90
onYAxisNeedsTicks.....	90
3 Axis Types	90
LinearAxis	90
Properties.....	91
background (inherited).....	91
border (inherited).....	91
caption (inherited).....	92
caption_class (inherited).....	92
color (inherited)	92
crosshair (inherited).....	93
cursor_position (inherited).....	93
extremes_ticks (inherited).....	94
force_static_points (inherited).....	94

formatter (inherited).....	94
grid (inherited)	95
hint_caption (inherited).....	95
label_class (inherited).....	96
major_ticks (inherited).....	96
max_extreme (inherited).....	97
min_extreme (inherited).....	97
minor_ticks (inherited).....	98
size (inherited)	98
stagger_ticks (inherited).....	99
visible (inherited).....	99
zero_plane (inherited).....	99
Methods	100
addBin (inherited).....	100
getExtremes (inherited).....	100
getZoom (inherited).....	101
getZoomBoxCoordinates (inherited).....	101
hide (inherited).....	101
hideGrid (inherited).....	101
pixelToPoint (inherited).....	102
pointToPixel (inherited).....	102
removeBin (inherited).....	102
resetZoom (inherited).....	103
setCaption (inherited).....	103
setCrosshair (inherited).....	103
setExtremes (inherited).....	103
setZoom (inherited).....	104
show (inherited).....	104
showGrid (inherited).....	104
Events	104
onHideCrosshair (inherited).....	104
onHideCursorPosition (inherited).....	104
onNeedsTicks (inherited).....	105
onShowCrosshair (inherited).....	105
onShowCursorPosition (inherited).....	106
LogarithmicAxis	106
Properties.....	107
background (inherited).....	107
base	107
border (inherited).....	107
caption (inherited).....	108
caption_class (inherited).....	108
color (inherited).....	108
crosshair (inherited).....	109
cursor_position (inherited).....	109
extremes_ticks (inherited).....	110
force_static_points (inherited).....	110
formatter (inherited).....	111
grid (inherited)	111
hint_caption	111
label_class (inherited).....	112
major_ticks (inherited).....	112
max_extreme (inherited).....	113
min_extreme (inherited).....	113

minor_ticks (inherited).....	114
size (inherited).....	114
stagger_ticks (inherited).....	115
visible (inherited).....	115
zero_plane (inherited).....	115
Methods	116
addBin (inherited).....	116
getExtremes (inherited).....	117
getZoom (inherited).....	117
getZoomBoxCoordinates (inherited).....	117
hide (inherited).....	117
hideGrid (inherited).....	118
pixelToPoint (inherited).....	118
pointToPixel (inherited).....	118
removeBin (inherited).....	118
resetZoom (inherited).....	119
setCaption (inherited).....	119
setCrosshair (inherited).....	119
setExtremes (inherited).....	119
setZoom (inherited).....	120
show (inherited).....	120
showGrid (inherited).....	120
Events	120
onHideCrosshair (inherited).....	120
onHideCursorPosition (inherited).....	121
onNeedsTicks (inherited).....	121
onShowCrosshair (inherited).....	122
onShowCursorPosition (inherited).....	122
DateAxis	122
Properties.....	123
background (inherited).....	123
border (inherited).....	123
caption (inherited).....	123
caption_class (inherited).....	124
color (inherited).....	124
crosshair (inherited).....	125
cursor_position (inherited).....	125
extreme_ticks (inherited).....	126
force_static_points (inherited).....	126
formatters (inherited).....	126
grid (inherited).....	127
label_class (inherited).....	127
max_extreme (inherited).....	128
major_ticks (inherited).....	128
min_extreme (inherited).....	129
minor_ticks (inherited).....	129
size (inherited).....	130
stagger_ticks (inherited).....	130
visible (inherited).....	130
zero_plane (inherited).....	131
Methods	131
addBin (inherited).....	131
getExtremes (inherited).....	132
getZoom (inherited).....	132

getZoomBoxCoordinates (inherited).....	132
hide (inherited).....	133
hideGrid (inherited).....	133
pixelToPoint (inherited).....	133
pointToPixel (inherited).....	133
removeBin (inherited).....	134
resetZoom (inherited).....	134
setCaption (inherited).....	134
setCrosshair (inherited).....	134
setExtremes (inherited).....	135
setZoom (inherited).....	135
show (inherited).....	135
showGrid (inherited).....	135
Events	136
onHideCrosshair (inherited).....	136
onHideCursorPosition (inherited).....	136
onNeedsTicks (inherited).....	136
onShowCrosshair (inherited).....	137
onShowCursorPosition (inherited).....	137
4 Series Types.....	138
EJSC.AlarmSeries	138
Properties.....	139
autosort (inherited).....	139
axis	139
color (inherited).....	139
coloredLegend (inherited).....	140
delayLoad (inherited).....	140
fill	140
fillTo	140
flag	140
hint_string (inherited).....	141
legendsVisible (inherited).....	141
lineOpacity (inherited).....	142
lineWidth (inherited).....	142
opacity (inherited).....	142
padding (inherited).....	142
title (inherited)	143
visible (inherited).....	143
Methods	143
findClosestByPixel (inherited).....	143
findClosestByPoint (inherited).....	144
getDataHandler (inherited)	144
getPadding (inherited).....	144
getVisibility (inherited).....	145
hide (inherited).....	145
hideLegend (inherited).....	145
reload (inherited).....	145
setColor (inherited).....	145
setColoredLegend (inherited).....	146
setDataHandler (inherited).....	146
setLineOpacity (inherited).....	146
setLineWidth (inherited)	146
setOpacity (inherited).....	146
setPadding (inherited).....	147

setTitle (inherited).....	147
show (inherited).....	147
showLegend (inherited).....	147
Events	148
onAfterDataAvailable (inherited).....	148
onAfterVisibilityChange (inherited).....	148
onBeforeVisibilityChange (inherited).....	148
onShowHint (inherited).....	148
EJSC.AnalogGaugeSeries	149
Properties.....	150
anchor	150
axis	150
delayLoad (inherited).....	151
fillColor	151
fillOpacity	151
height	151
label	152
legendIsVisible (inherited).....	152
lock	152
marker_position.....	153
max	153
min	153
minorTick	154
needle	154
position	155
range	155
range_degrees.....	156
ranges	156
start_degree	156
tick	156
tickCount	157
title (inherited)	157
visible (inherited).....	157
width	157
x_axis_formatter (inherited).....	158
Methods	158
getDataHandler (inherited).....	158
getVisibility (inherited).....	158
hide (inherited).....	158
hideLegend (inherited).....	159
reload (inherited).....	159
setDataHandler (inherited).....	159
setTitle (inherited).....	159
show (inherited).....	159
showLegend (inherited).....	160
Events	160
onAfterDataAvailable (inherited).....	160
onAfterVisibilityChange (inherited).....	160
onBeforeVisibilityChange (inherited).....	160
Data Formats.....	161
EJSC.AreaSeries	162
Properties.....	163
autosort (inherited).....	163
closeLine	163

color (inherited).....	163
coloredLegend (inherited).....	164
delayLoad (inherited).....	164
drawPoints (inherited).....	164
hint_string (inherited).....	164
legendIsVisible (inherited).....	165
lineOpacity (inherited).....	165
lineWidth (inherited).....	165
opacity (inherited).....	165
padding (inherited).....	166
pointBorderColor (inherited).....	166
pointBorderSize (inherited).....	166
pointColor (inherited).....	167
pointSize (inherited).....	167
title (inherited)	167
visible (inherited).....	167
x_axis (inherited).....	168
x_axis_formatter (inherited).....	168
y_axis (inherited).....	168
y_axis_formatter (inherited).....	168
Methods	169
findClosestByPixel (inherited).....	169
findClosestByPoint (inherited).....	169
getDataHandler (inherited).....	169
getPadding (inherited).....	169
getVisibility (inherited).....	170
hide (inherited).....	170
hideLegend (inherited).....	170
reload (inherited).....	170
setColor (inherited).....	171
setColoredLegend (inherited).....	171
setDataHandler (inherited).....	171
setLineOpacity (inherited).....	171
setLineWidth (inherited)	171
setOpacity (inherited).....	172
setPadding (inherited).....	172
setTitle (inherited).....	172
show (inherited).....	172
showLegend (inherited).....	173
Events	173
onAfterDataAvailable (inherited).....	173
onAfterVisibilityChange (inherited).....	173
onBeforeVisibilityChange (inherited).....	173
onShowHint (inherited).....	173
Data Formats	174
Text Replacement Options.....	175
EJSC.BarSeries	176
Properties.....	177
autosort (inherited).....	177
color (inherited).....	177
coloredLegend (inherited).....	177
defaultColors	177
delayLoad (inherited).....	178
groupedBars	178

hint_string (inherited).....	178
intervalOffset	179
legendsVisible (inherited).....	179
lineOpacity (inherited).....	179
lineWidth (inherited).....	179
opacity (inherited).....	180
padding (inherited).....	180
orientation	181
ranges	181
title (inherited)	181
treeLegend	182
useColorArray	182
visible (inherited).....	182
x_axis (inherited).....	183
x_axis_formatter (inherited).....	183
y_axis (inherited).....	183
y_axis_formatter (inherited).....	183
Methods	184
addRange	184
clearRanges	184
deleteRange	184
findClosestByPixel (inherited).....	184
findClosestByPoint (inherited).....	185
getBarSize	185
getBarSizeInPoints.....	185
getDataHandler (inherited).....	185
getPadding (inherited).....	185
getPoints	186
getVisibility (inherited).....	186
hide (inherited).....	186
hideLegend (inherited).....	186
reload (inherited).....	187
setColor (inherited).....	187
setColoredLegend (inherited).....	187
setDataHandler (inherited).....	187
setDefaultColors.....	187
setGroupedBars	188
setIntervalOffset	188
setLineWidth (inherited)	188
setOpacity (inherited).....	189
setPadding (inherited)	189
setTitle (inherited).....	189
show (inherited).....	189
showLegend (inherited).....	189
Events	190
onAfterDataAvailable (inherited).....	190
onAfterVisibilityChange (inherited).....	190
onBarNeedsColor.....	190
onBeforeVisibilityChange (inherited).....	191
onShowHint (inherited).....	191
Data Formats.....	191
Text Replacement Options.....	193
EJSC.CandlestickSeries	193
Properties.....	194

autosort (inherited).....	194
color (inherited).....	194
coloredLegend (inherited).....	195
delayLoad (inherited).....	195
gain	195
hint_string (inherited).....	195
intervalOffset	196
legendIsVisible (inherited).....	196
lineOpacity (inherited).....	196
lineWidth (inherited).....	197
loss	197
opacity (inherited).....	197
padding (inherited).....	198
title (inherited)	198
visible (inherited).....	198
x_axis (inherited).....	199
x_axis_formatter (inherited).....	199
y_axis (inherited).....	199
y_axis_formatter (inherited).....	199
Methods	200
findClosestByPixel (inherited).....	200
findClosestByPoint (inherited).....	200
getDataHandler (inherited).....	200
getPadding (inherited).....	200
getVisibility (inherited).....	201
hide (inherited).....	201
hideLegend (inherited).....	201
reload (inherited).....	201
setColor (inherited).....	202
setColoredLegend (inherited).....	202
setDataHandler (inherited).....	202
setLineOpacity (inherited).....	202
setLineWidth (inherited)	202
setOpacity (inherited).....	203
setPadding (inherited)	203
setTitle (inherited).....	203
show (inherited).....	203
showLegend (inherited).....	204
Events	204
onAfterDataAvailable (inherited).....	204
onAfterVisibilityChange (inherited).....	204
onBeforeVisibilityChange (inherited).....	204
onShowHint (inherited).....	204
Data Formats	205
Text Replacement Options	206
EJSC.DoughnutSeries	207
Properties.....	208
defaultColors (inherited).....	208
delayLoad (inherited).....	208
doughnutOffset.....	208
height (inherited).....	208
hint_string (inherited).....	209
legendIsVisible (inherited).....	209
lineOpacity (inherited).....	209

lineWidth (inherited).....	209
opacity (inherited).....	210
position (inherited).....	210
title (inherited)	210
total_value (inherited).....	210
treeLegend (inherited).....	211
visible (inherited).....	211
width (inherited).....	211
x_axis_formatter (inherited).....	211
Methods	212
findCenter (inherited).....	212
findCenterOfCurve (inherited).....	212
getDataHandler (inherited).....	212
getPoints (inherited).....	212
getTotalValue (inherited).....	213
getVisibility (inherited).....	213
hide (inherited).....	213
hideLegend (inherited).....	213
reload (inherited).....	214
resetTotalValue (inherited).....	214
setDataHandler (inherited).....	214
setDefaultColors (inherited).....	214
setLineOpacity (inherited).....	215
setLineWidth (inherited).....	215
setOpacity (inherited).....	215
setTitle (inherited).....	215
setTotalValue (inherited).....	216
show (inherited).....	216
showLegend (inherited).....	216
Events	216
onAfterDataAvailable (inherited).....	216
onAfterVisibilityChange (inherited).....	216
onBeforeVisibilityChange (inherited).....	217
onPieceNeedsColor (inherited).....	217
onShowHint (inherited).....	217
Data Formats.....	217
Text Replacement Options.....	219
EJSC.ErrorSeries	219
Properties.....	220
avgSize	220
autosort (inherited).....	220
capSize	221
capDist	221
color (inherited).....	221
coloredLegend (inherited).....	221
delayLoad (inherited).....	222
hint_string (inherited).....	222
legendsIsVisible (inherited).....	222
lineOpacity (inherited).....	222
lineWidth (inherited).....	223
orientation	223
opacity (inherited).....	223
padding (inherited).....	223
title (inherited)	224

visible (inherited).....	224
x_axis (inherited).....	224
x_axis_formatter (inherited).....	225
y_axis (inherited).....	225
y_axis_formatter (inherited).....	225
Methods.....	225
findClosestByPixel (inherited).....	225
findClosestByPoint (inherited).....	226
getDataHandler (inherited).....	226
getPadding (inherited).....	226
getVisibility (inherited).....	227
hide (inherited).....	227
hideLegend (inherited).....	227
reload (inherited).....	227
setColor (inherited).....	227
setColoredLegend (inherited).....	228
setDataHandler (inherited).....	228
setLineOpacity (inherited).....	228
setLineWidth (inherited).....	228
setOpacity (inherited).....	228
setPadding (inherited).....	229
setTitle (inherited).....	229
show (inherited).....	229
showLegend (inherited).....	229
Events.....	230
onAfterDataAvailable (inherited).....	230
onAfterVisibilityChange (inherited).....	230
onBeforeVisibilityChange (inherited).....	230
onShowHint (inherited).....	230
EJSC.FloatingBarSeries	231
Properties.....	232
autosort (inherited).....	232
color (inherited).....	232
coloredLegend (inherited).....	232
defaultColors (inherited).....	232
delayLoad (inherited).....	233
groupedBars (inherited).....	233
hint_string (inherited).....	233
intervalOffset (inherited).....	234
legendIsVisible (inherited).....	234
lineOpacity (inherited).....	234
linewidth (inherited).....	234
opacity (inherited).....	235
orientation (inherited).....	235
padding (inherited).....	235
ranges (inherited).....	236
title (inherited).....	236
treeLegend (inherited).....	237
useColorArray (inherited).....	237
visible (inherited).....	237
x_axis (inherited).....	238
x_axis_formatter (inherited).....	238
y_axis (inherited).....	238
y_axis_formatter (inherited).....	238

Methods	239
addRange (inherited).....	239
clearRanges (inherited).....	239
deleteRange (inherited).....	239
findClosestByPixel (inherited).....	239
findClosestByPoint (inherited).....	240
getBarSize (inherited).....	240
getBarSizeInPoints (inherited).....	240
getDataHandler (inherited).....	240
getPadding (inherited).....	240
getPoints (inherited).....	241
getVisibility (inherited).....	241
hide (inherited).....	241
hideLegend (inherited).....	241
reload (inherited).....	242
setColor (inherited).....	242
setColoredLegend (inherited).....	242
setDataHandler (inherited).....	242
setDefaultColors (inherited).....	242
setIntervalOffset (inherited).....	243
setLineWidth (inherited).....	243
setOpacity (inherited).....	243
setPadding (inherited).....	243
setTitle (inherited).....	244
show (inherited).....	244
showLegend (inherited).....	244
Events	244
onAfterDataAvailable (inherited).....	244
onAfterVisibilityChange (inherited).....	245
onBarNeedsColor (inherited).....	245
onBeforeVisibilityChange (inherited).....	245
onShowHint (inherited).....	246
Data Formats	246
Text Replacement Options	248
EJSC.FunctionSeries	249
Properties.....	250
color (inherited).....	250
coloredLegend (inherited).....	250
hint_string (inherited).....	250
legendsIsVisible (inherited).....	250
lineOpacity (inherited).....	251
linewidth (inherited).....	251
padding (inherited).....	251
title (inherited).....	252
visible (inherited).....	252
x_axis (inherited).....	252
x_axis_formatter (inherited).....	252
y_axis (inherited).....	253
y_axis_formatter (inherited).....	253
Methods	253
findClosestByPixel (inherited).....	253
findClosestByPoint (inherited).....	253
getPadding (inherited).....	254
getVisibility (inherited).....	254

hide (inherited).....	254
hideLegend (inherited).....	254
reload (inherited).....	255
setColor (inherited).....	255
setColoredLegend (inherited).....	255
setLineWidth (inherited).....	255
setPadding (inherited).....	255
setTitle (inherited).....	256
show (inherited).....	256
showLegend (inherited).....	256
Events	256
onAfterVisibilityChange (inherited).....	256
onBeforeVisibilityChange (inherited).....	257
onShowHint (inherited).....	257
Text Replacement Options.....	257
EJSC.LineSeries	258
Properties.....	259
autosort (inherited).....	259
color (inherited).....	259
coloredLegend (inherited).....	259
delayLoad (inherited).....	260
drawPoints	260
hint_string (inherited).....	260
legendsIsVisible (inherited).....	260
lineOpacity (inherited).....	261
linewidth (inherited).....	261
padding (inherited).....	261
pointBorderColor.....	262
pointBorderSize.....	262
pointColor	262
pointSize	262
title (inherited)	263
visible (inherited).....	263
x_axis (inherited).....	263
x_axis_formatter (inherited).....	263
y_axis (inherited).....	263
y_axis_formatter (inherited).....	264
Methods	264
findClosestByPixel (inherited).....	264
findClosestByPoint (inherited).....	264
getDataHandler (inherited).....	265
getPadding (inherited).....	265
getVisibility (inherited).....	265
hide (inherited).....	265
hideLegend (inherited).....	266
reload (inherited).....	266
setColor (inherited).....	266
setColoredLegend (inherited).....	266
setDataHandler (inherited).....	266
setLineOpacity (inherited).....	267
setLineWidth (inherited).....	267
setPadding (inherited).....	267
setTitle (inherited).....	267
show (inherited).....	267

showLegend (inherited).....	268
Events	268
onAfterDataAvailable (inherited).....	268
onAfterVisibilityChange (inherited).....	268
onBeforeVisibilityChange (inherited).....	268
onShowHint (inherited).....	269
Text Replacement Options.....	269
Data Formats.....	269
EJSC.OpenHighLowCloseSeries	271
Properties.....	272
autosort (inherited).....	272
color (inherited).....	272
coloredLegend (inherited).....	272
delayLoad (inherited).....	272
gain	273
hint_string (inherited).....	273
intervalOffset	273
legendIsVisible (inherited).....	274
lineOpacity (inherited).....	274
lineWidth (inherited).....	274
loss	274
opacity (inherited).....	275
padding (inherited).....	275
title (inherited)	276
visible (inherited).....	276
x_axis (inherited).....	276
x_axis_formatter (inherited).....	276
y_axis (inherited).....	277
y_axis_formatter (inherited).....	277
Methods	277
findClosestByPixel (inherited).....	277
findClosestByPoint (inherited).....	277
getDataHandler (inherited).....	278
getPadding (inherited).....	278
getVisibility (inherited).....	278
hide (inherited).....	279
hideLegend (inherited).....	279
reload (inherited).....	279
setColor (inherited).....	279
setColoredLegend (inherited).....	279
setDataHandler (inherited).....	280
setLineOpacity (inherited).....	280
setLineWidth (inherited)	280
setOpacity (inherited).....	280
setPadding (inherited)	280
setTitle (inherited).....	281
show (inherited).....	281
showLegend (inherited).....	281
Events	281
onAfterDataAvailable (inherited).....	281
onAfterVisibilityChange (inherited).....	282
onBeforeVisibilityChange (inherited).....	282
onShowHint (inherited).....	282
Data Formats.....	282

Text Replacement Options	284
EJSC.OverUnderSeries	284
Properties.....	285
autosort (inherited).....	285
color (inherited).....	285
coloredLegend (inherited).....	285
exceed	286
delayLoad (inherited).....	286
hint_string (inherited).....	286
legendsVisible (inherited).....	287
lineOpacity (inherited).....	287
lineWidth (inherited).....	287
opacity (inherited).....	287
padding (inherited).....	288
subsede	288
title (inherited)	289
visible (inherited).....	289
x_axis (inherited).....	289
x_axis_formatter (inherited).....	289
y_axis (inherited).....	289
y_axis_formatter (inherited).....	290
Methods	290
findClosestByPixel (inherited).....	290
findClosestByPoint (inherited).....	290
getDataHandler (inherited).....	291
getPadding (inherited).....	291
getVisibility (inherited).....	291
hide (inherited).....	291
hideLegend (inherited).....	292
reload (inherited).....	292
setColor (inherited).....	292
setColoredLegend (inherited).....	292
setDataHandler (inherited).....	292
setLineOpacity (inherited).....	293
setLineWidth (inherited).....	293
setOpacity (inherited).....	293
setPadding (inherited).....	293
setTitle (inherited).....	293
show (inherited).....	294
showLegend (inherited).....	294
Events	294
onAfterDataAvailable (inherited).....	294
onAfterVisibilityChange (inherited).....	294
onBeforeVisibilityChange (inherited).....	295
onShowHint (inherited).....	295
EJSC.PieSeries	295
Properties.....	296
defaultColors	296
delayLoad (inherited).....	296
height	297
hint_string (inherited).....	297
legendsVisible (inherited).....	297
lineOpacity (inherited).....	297
lineWidth (inherited).....	298

opacity (inherited).....	298
position	298
title (inherited)	298
total_value	299
treeLegend	299
visible (inherited).....	299
width	299
x_axis_formatter (inherited).....	300
Methods	300
findCenter	300
findCenterOfCurve.....	300
getDataHandler (inherited).....	300
getPoints	301
getTotalValue	301
getVisibility (inherited).....	301
hide (inherited).....	301
hideLegend (inherited).....	302
reload (inherited).....	302
resetTotalValue.....	302
setDataHandler (inherited).....	302
setDefaultColors.....	302
setLineWidth (inherited).....	303
setOpacity (inherited).....	303
setTitle (inherited).....	303
setTotalValue	304
show (inherited).....	304
showLegend (inherited).....	304
Events	304
onAfterDataAvailable (inherited).....	304
onAfterVisibilityChange (inherited).....	304
onBeforeVisibilityChange (inherited).....	305
onPieceNeedsColor.....	305
onShowHint (inherited).....	305
Data Formats.....	305
Text Replacement Options.....	307
EJSC.ScatterSeries	307
Properties.....	308
autosort (inherited).....	308
color (inherited).....	308
coloredLegend (inherited).....	309
delayLoad (inherited).....	309
hint_string (inherited).....	309
legendIsVisible (inherited).....	309
lineOpacity (inherited).....	310
linewidth (inherited).....	310
opacity (inherited).....	310
padding (inherited).....	310
pointSize	311
pointStyle	311
title (inherited)	311
visible (inherited).....	312
x_axis (inherited).....	312
x_axis_formatter (inherited).....	312
y_axis (inherited).....	312

y_axis_formatter (inherited).....	312
Methods	313
findClosestByPixel (inherited).....	313
findClosestByPoint (inherited).....	313
getDataHandler (inherited).....	313
getPadding (inherited).....	314
getVisibility (inherited).....	314
hide (inherited).....	314
hideLegend (inherited).....	314
reload (inherited).....	315
setColor (inherited).....	315
setColoredLegend (inherited).....	315
setDataHandler (inherited).....	315
setLineOpacity (inherited).....	315
setLineWidth (inherited).....	316
setOpacity (inherited).....	316
setPadding (inherited).....	316
setPointStyle	316
setTitle (inherited).....	317
show (inherited).....	317
showLegend (inherited).....	317
Events	317
onAfterDataAvailable (inherited).....	317
onAfterVisibilityChange (inherited).....	318
onBeforeVisibilityChange (inherited).....	318
onShowHint (inherited).....	318
Data Formats.....	318
Text Replacement Options.....	320
EJSC.StepSeries	320
Properties.....	321
autosort (inherited).....	321
color (inherited).....	321
coloredLegend (inherited).....	321
drawPoints (inherited).....	322
hint_string (inherited).....	322
legendIsVisible (inherited).....	322
lineOpacity (inherited).....	322
linewidth (inherited).....	323
padding (inherited).....	323
pointBorderColor (inherited).....	324
pointBorderSize (inherited).....	324
pointColor (inherited).....	324
pointSize (inherited).....	324
title (inherited)	324
visible (inherited).....	325
x_axis (inherited).....	325
x_axis_formatter (inherited).....	325
y_axis (inherited).....	325
y_axis_formatter (inherited).....	326
Methods	326
findClosestByPoint (inherited).....	326
findClosestByPixel (inherited).....	326
getDataHandler (inherited).....	327
getPadding (inherited).....	327

getVisibility (inherited)	327
hide (inherited)	327
hideLegend (inherited)	328
reload (inherited)	328
setColor (inherited)	328
setColoredLegend (inherited)	328
setDataHandler (inherited)	328
setLineOpacity (inherited)	329
setLineWidth (inherited)	329
setPadding (inherited)	329
setTitle (inherited)	329
show (inherited)	329
showLegend (inherited)	330
Events	330
onAfterDataAvailable (inherited)	330
onAfterVisibilityChange (inherited)	330
onBeforeVisibilityChange (inherited)	330
onShowHint (inherited)	331
Text Replacement Options	331
Data Formats	331
EJSC.StackedBarSeries	333
Properties	334
autosort (inherited)	334
color (inherited)	334
coloredLegend (inherited)	334
defaultColors (inherited)	334
delayLoad (inherited)	335
groupedBars (inherited)	335
hint_string (inherited)	335
intervalOffset (inherited)	336
legendIsVisible (inherited)	336
lineOpacity (inherited)	336
lineWidth (inherited)	337
opacity (inherited)	337
orientation (inherited)	337
padding (inherited)	337
ranges (inherited)	338
title (inherited)	339
treeLegend (inherited)	339
treeLegendRoot	339
useColorArray (inherited)	339
visible (inherited)	340
x_axis (inherited)	340
x_axis_formatter (inherited)	340
y_axis (inherited)	340
y_axis_formatter (inherited)	340
Methods	341
addRange (inherited)	341
addSeries	341
clearRanges (inherited)	342
deleteRange (inherited)	342
findClosestByPixel (inherited)	342
findClosestByPoint (inherited)	342
getBarSize (inherited)	343

getBarSizeInPoints (inherited).....	343
getDataHandler (inherited).....	343
getPadding (inherited).....	343
getPoints (inherited).....	343
getVisibility (inherited).....	343
hide (inherited).....	344
hideLegend (inherited).....	344
reload (inherited).....	344
removeSeries	344
setColor (inherited).....	344
setColoredLegend (inherited).....	345
setDataHandler (inherited).....	345
setDefaultColors (inherited).....	345
setGroupedBars (inherited).....	346
setIntervalOffset (inherited).....	346
setLineOpacity (inherited).....	346
setLineWidth (inherited).....	346
setOpacity (inherited).....	346
setPadding (inherited).....	347
setTitle (inherited).....	347
show (inherited).....	347
showLegend (inherited).....	347
Events	348
onAfterDataAvailable (inherited).....	348
onAfterVisibilityChange (inherited).....	348
onBarNeedsColor (inherited).....	348
onBeforeVisibilityChange (inherited).....	349
onShowHint (inherited).....	349
Text Replacement Options.....	349
EJSC.TrendSeries	350
Properties.....	351
color (inherited).....	351
coloredLegend (inherited).....	351
hint_string (inherited).....	351
legendsIsVisible (inherited).....	352
lineOpacity (inherited).....	352
lineWidth (inherited).....	352
padding (inherited).....	352
title (inherited)	353
visible (inherited).....	353
x_axis (inherited).....	353
x_axis_formatter (inherited).....	353
y_axis (inherited).....	354
y_axis_formatter (inherited).....	354
Methods	354
findClosestByPixel (inherited).....	354
findClosestByPoint (inherited).....	355
getPadding (inherited).....	355
getVisibility (inherited).....	355
hide (inherited).....	355
hideLegend (inherited).....	356
reload (inherited).....	356
setColor (inherited).....	356
setColoredLegend (inherited).....	356

setLineWidth (inherited)	356
setPadding (inherited)	357
setTitle (inherited)	357
show (inherited)	357
showLegend (inherited)	357
Events	358
onAfterVisibilityChange (inherited)	358
onBeforeVisibilityChange (inherited)	358
onShowHint (inherited)	358
Text Replacement Options	358
5 Data Handlers	359
EJSC.ArrayDataHandler	359
Methods	359
getArray	359
loadData (inherited)	359
setArray	359
Events	360
onDataAvailable (inherited)	360
EJSC.CSVFileDataHandler	360
Properties	360
requestType (inherited)	360
url (inherited)	360
urlData (inherited)	361
Methods	361
getUrl (inherited)	361
loadData (inherited)	361
setRequestType (inherited)	362
setUrl (inherited)	362
setUrlData (inherited)	362
setXMLData (inherited)	362
Events	363
onDataAvailable (inherited)	363
onDataReady (inherited)	363
onNeedsData (inherited)	363
EJSC.CSVStringDataHandler	364
Methods	364
getCSV	364
loadData (inherited)	364
setCSV	364
Events	365
onDataAvailable (inherited)	365
EJSC.JSONFileDataHandler	365
Properties	365
requestType (inherited)	365
url (inherited)	365
urlData (inherited)	366
Methods	366
getUrl (inherited)	366
loadData (inherited)	366
setRequestType (inherited)	367
setUrl (inherited)	367
setUrlData (inherited)	367
setXMLData (inherited)	367
Events	368

onDataAvailable (inherited)	368
onDataReady (inherited)	368
onNeedsData (inherited)	368
EJSC.JSONStringDataHandler	369
Methods	369
getJSON	369
loadData (inherited)	369
setJSON	369
Events	370
onDataAvailable (inherited)	370
EJSC.XMLDataHandler	370
Properties	370
requestType (inherited)	370
url (inherited)	371
urlData (inherited)	371
Methods	371
getUrl (inherited)	371
loadData (inherited)	372
setRequestType (inherited)	372
setUrl (inherited)	372
setUrlData (inherited)	372
setXMLData (inherited)	373
Events	373
onDataAvailable (inherited)	373
onDataReady (inherited)	373
onNeedsData (inherited)	373
EJSC.XMLStringDataHandler	374
Methods	374
getXML	374
loadData (inherited)	375
setXML	375
Events	375
onDataAvailable (inherited)	375
6 Label Formatters	375
EJSC.DateFormatter	375
Properties	376
format_string	376
timezoneOffset	376
useUTC	377
Methods	377
format (inherited)	377
EJSC.NumberFormatter	377
Properties	378
currency_align	378
currency_position	378
currency_symbol	378
decimal_separator	378
forced_decimals	379
negative_symbol	379
thousand_separator	379
variable_decimals	379
Methods	379
format (inherited)	379
EJSC.StringFormatter	380

Properties.....	380
append	380
prefix	380
Methods.....	381
format (inherited).....	381
Events	381
onNeedsFormat.....	381
7 Base Classes.....	381
EJSC.Inheritable	381
EJSC.AjaxDataHandler	382
Properties.....	382
requestType	382
url	382
urlData	382
Methods.....	383
getUrl	383
loadData (inherited).....	383
setRequestType	383
setUrl	383
setUrlData	384
setXMLData	384
Events	384
onDataAvailable (inherited).....	384
onDataReady	384
onNeedsData	385
EJSC.Axis	385
Properties.....	385
background	385
border	386
caption	386
caption_class	387
color	387
crosshair	387
cursor_position.....	388
extremes_ticks.....	388
force_static_points.....	389
formatter	389
grid	390
hint_caption	390
label_class	390
max_extreme	391
major_ticks	391
minor_ticks	392
min_extreme	393
size	393
stagger_ticks	393
visible	394
zero_plane	394
Methods.....	395
addBin	395
getExtremes	395
getZoom	395
getZoomBoxCoordinates.....	396
hide	396

hideGrid	396
resetZoom	396
pixelToPoint	397
pointToPixel	397
removeBin	397
setCaption	398
setCrosshair	398
setExtremes	398
setZoom	398
show	399
showGrid	399
Events	399
onHideCrosshair	399
onHideCursorPosition	399
onNeedsTicks	399
onShowCrosshair	400
onShowCursorPosition	400
EJSC.DataHandler	401
Methods	401
loadData	401
Events	401
onDataAvailable	401
EJSC.GaugeSeries	401
Properties	402
color (inherited)	402
coloredLegend (inherited)	402
delayLoad (inherited)	402
legendIsVisible (inherited)	402
lineOpacity (inherited)	403
lineWidth (inherited)	403
opacity (inherited)	403
title (inherited)	403
visible (inherited)	403
x_axis_formatter (inherited)	404
Methods	404
getDataHandler (inherited)	404
getVisibility (inherited)	404
hide (inherited)	404
hideLegend (inherited)	405
reload (inherited)	405
setColor (inherited)	405
setColoredLegend (inherited)	405
setDataHandler (inherited)	405
setLineOpacity (inherited)	406
setLineWidth (inherited)	406
setOpacity (inherited)	406
setTitle (inherited)	406
show (inherited)	406
showLegend (inherited)	407
Events	407
onAfterDataAvailable (inherited)	407
onAfterVisibilityChange (inherited)	407
onBeforeVisibilityChange (inherited)	407
EJSC.Formatter	408

Properties.....	408
format_string	408
Methods.....	408
format	408
EJSC.Point	408
Properties.....	409
label	409
userdata	409
x	409
y	409
EJSC.Series	409
Properties.....	410
autosort	410
color	410
coloredLegend.....	410
delayLoad	410
hint_string	411
legendIsVisible.....	411
lineOpacity	411
lineWidth	411
opacity	412
padding	412
title	412
visible	413
x_axis	413
x_axis_formatter.....	413
y_axis	413
y_axis_formatter.....	414
Methods.....	414
findClosestByPixel.....	414
findClosestByPoint.....	414
getDataHandler.....	415
getPadding	415
getVisibility	415
hide	415
hideLegend	416
reload	416
setColor	416
setColoredLegend.....	416
setDataHandler.....	416
setLineOpacity.....	417
setLineWidth	417
setOpacity	417
setPadding	417
setTitle	417
show	418
showLegend	418
Events	418
onAfterDataAvailable.....	418
onAfterVisibilityChange.....	418
onBeforeVisibilityChange.....	419
onShowHint	419
8 Other Classes.....	419
EJSC.BarPoint	419

Properties.....	419
label (inherited).....	419
userdata (inherited).....	420
x (inherited)	420
y (inherited)	420
EJSC.FloatingBarPoint	420
Properties.....	420
label (inherited).....	420
max	421
min	421
userdata (inherited).....	421
x (inherited)	421
y (inherited)	421
EJSC.GaugePoint	422
Properties.....	422
label (inherited).....	422
userdata (inherited).....	422
x (inherited)	422
EJSC.PiePoint	423
Properties.....	423
label (inherited).....	423
userdata (inherited).....	423
x (inherited)	423
EJSC.StockPoint	424
Properties.....	424
close	424
high	424
label (inherited).....	424
low	424
open	425
userdata (inherited).....	425
x (inherited)	425
EJSC.XYPoint	425
Properties.....	425
label (inherited).....	425
userdata (inherited).....	426
x (inherited)	426
y (inherited)	426
9 Exporting To SVG.....	426
10 META Tag Configuration.....	427
11 Text Replacement Options.....	428
12 Using Colors.....	428
Part V Getting Support	428
Index	0

1 Getting Started

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.

1.1 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 How To 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"
        }
    );
</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="Emprise
JavaScript Charts :: Customization Example">
    <script type="text/javascript"
src="/EJSCChart/EJSCChart.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"  
    }  
);  
  
</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="/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;

    </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="/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); </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="/EJSCChart/EJSCChart.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:

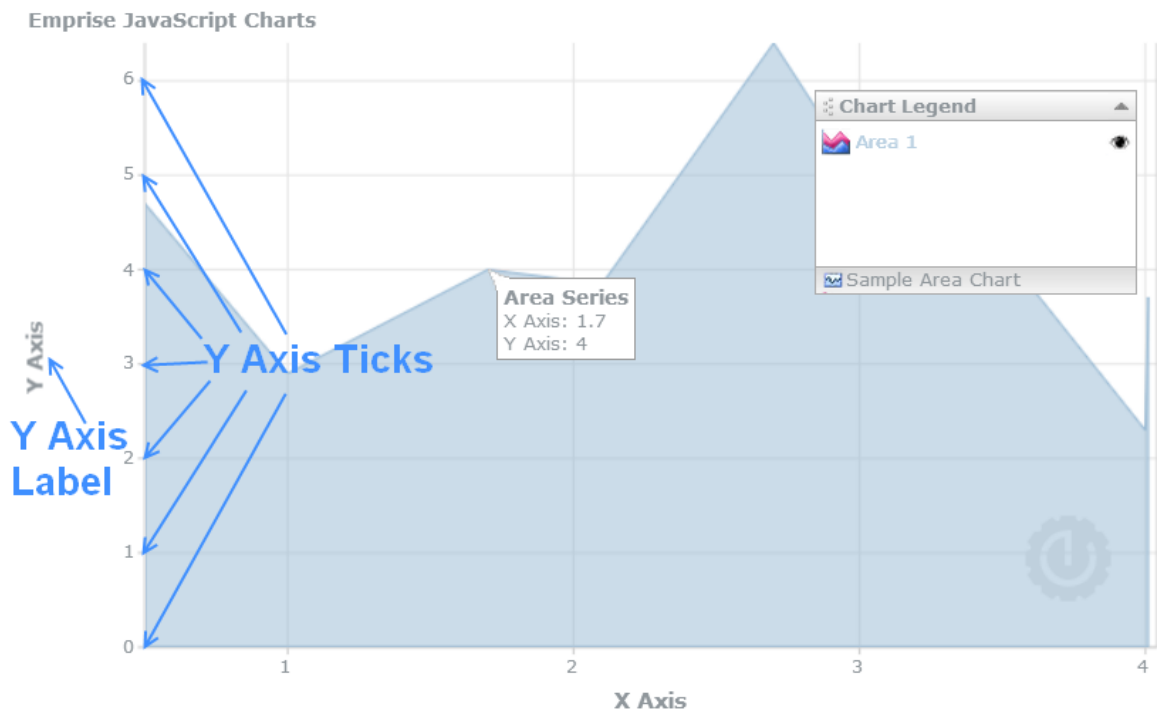
```
var myChartSeries = new EJSC.FunctionSeries(Math.sin);
myChartSeries.setLineWidth(4);
```

1.5 Anatomy of a Chat

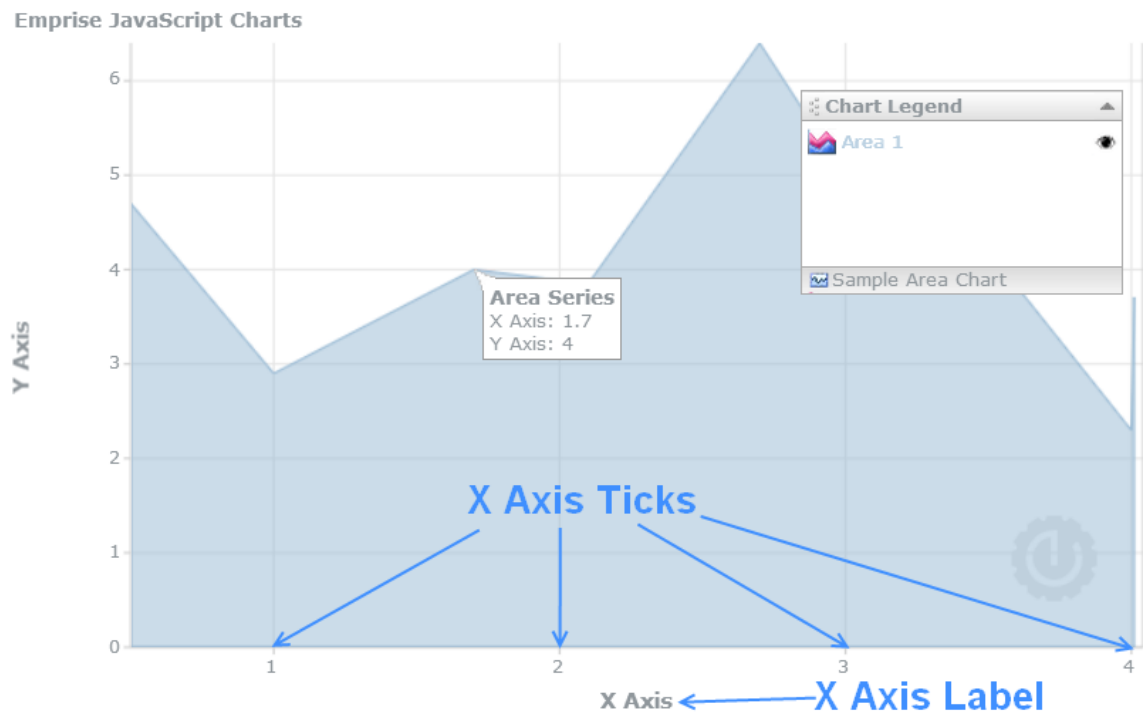
In order to fully understand the power of the properties available for customizing charts is important to become familiar with the various components of the chart.

The following diagram identify many of these components:

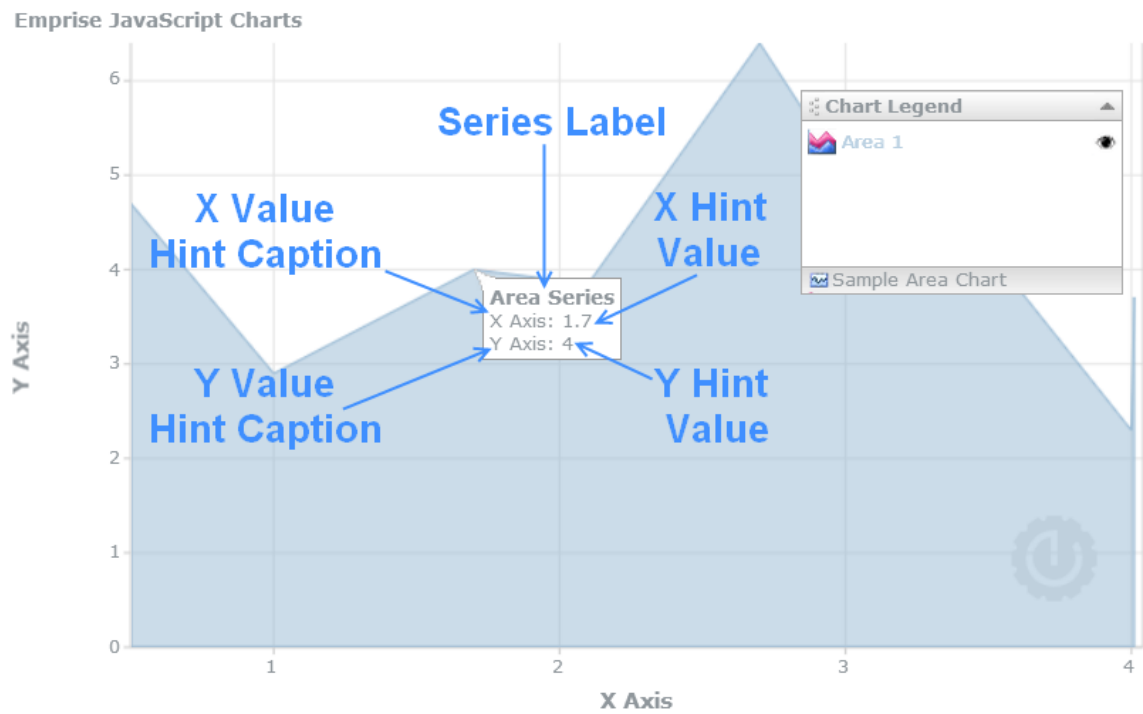
1.5.1 Y Axis



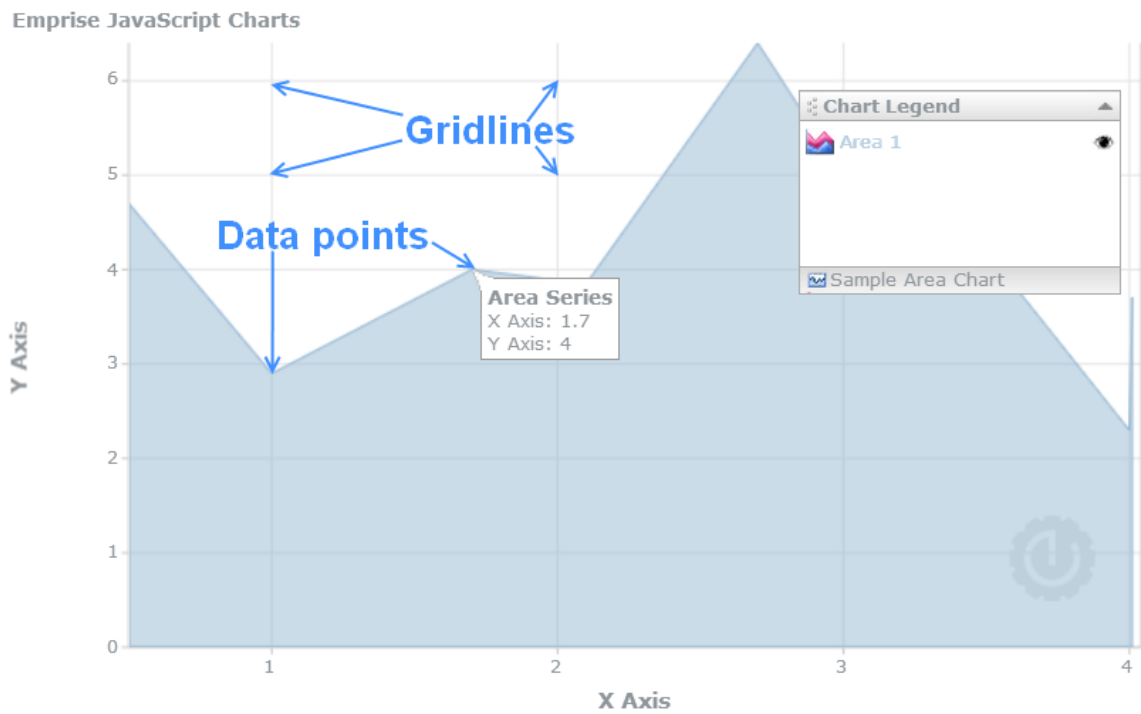
1.5.2 X Axis



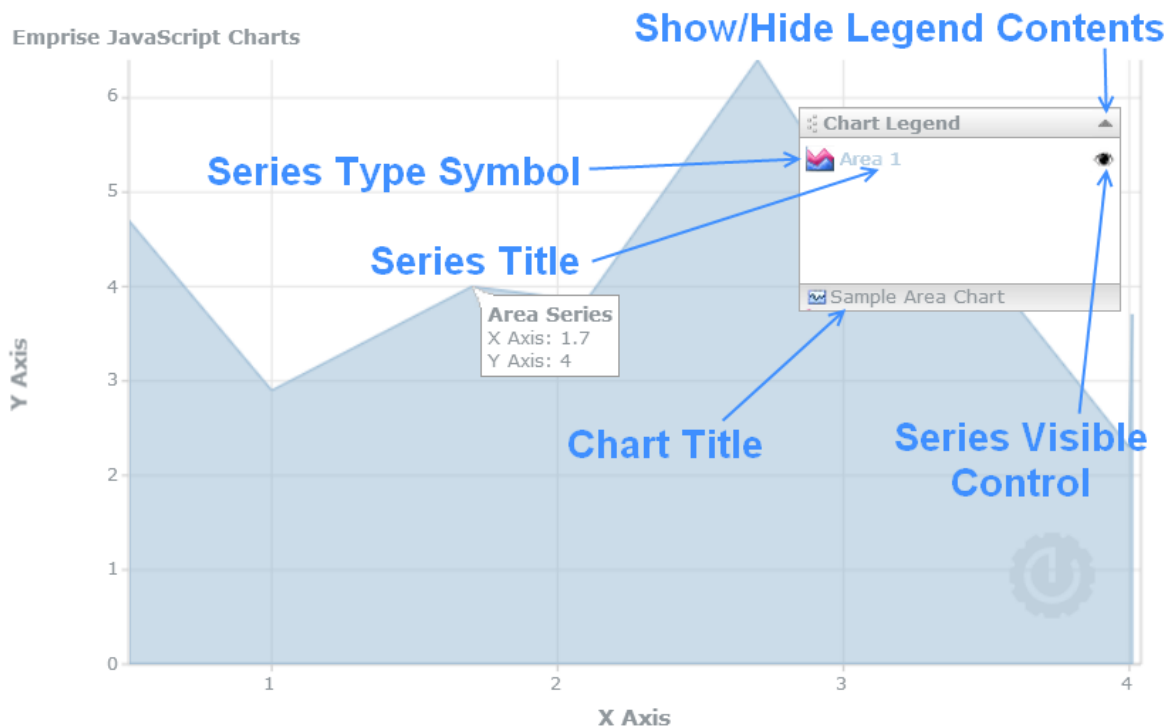
1.5.3 Hint



1.5.4 Chart Body



1.5.5 Legend



1.6 Version 1.x Compatability

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 aide you in upgrading your code.

Additional META tag options are described here.

1.6.1 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

1.7 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.

2 Changes

2.1 Changes in version 2.3

Version 2.3

New Features

- Added pointStyle property to LineSeries and AreaSeries
- Added even_shading and odd_shading properties to LinearAxis
- Added onUserEndZoom event to Chart
- Vertical Axis Captions (check cross-browser)
- Added legend_position, legend_height, and legend_width to Chart
- Added setAxis method to Chart
- New Series: ErrorSeries
- New Series: AlarmSeries
- New Series: OverUnderSeries
- New Series: StepSeries
- New Series: DoughnutSeries (DonutSeries)

- New Axis: DateAxis

Bug Fixes

- Fixed issues with displaying charts in IE9

2.2 Changes in version 2.1

Version 2.1

New Features

- Added treeLegendRoot property to StackedBarSeries
- Added left and right as valid property values for PieSeries.position and AnalogGaugeSeries.position
- Added defaultColors, useColorArray and onBarNeedsColor to StackedBarSeries
- Added the ability for StackedBarSeries to share a single color array among all owned series
- Added support for IE8 RC1 (and most previous betas) in all modes
- Added chart.setShowLegend method
- Performance increases in IE when drawing multiple series and/or series with large numbers of points
- Added support for gaps in LineSeries (use a blank string as Y)
- Added getBarSizeInPoints to BarSeries and descendant classes

Bug Fixes

- Corrected several IE related memory leaks
- Corrected memory leaks related to Ajax requests (affected all browsers)
- Corrected memory leaks when removing charts from a page repeatedly without refreshing
- Corrected parameter name for CSVStringDataHandler.setCSV
- Fixed AreaSeries draw when series contains more than 5000 points in IE
- Fixed incorrect parsing of numeric Y values in Stock Series
- Updated chart.onBeforeSelectPoint, onAfterSelectPoint, onShowHint to send the correct value for the HoverOrSelect parameter
- Corrected error when trying to get/set axis captions
- Corrected issues with certain data handlers corrupting data when Object.prototype has been modified
- Fixed issues with legend movement when the chart was inside a scrollable container
- Corrected Axis.pointToPixel so that it returns the proper value when given a string (bin)
- Corrected double loading of data when using JSONStringDataHandler
- Removed the ability for closeLine to affect drawing in LineSeries
- Corrected issues with AreaSeries drawing when closeLine was false in Safari
- Corrected AreaSeries drawing in IE when more than 5000 points exist
- Corrected issues with extremes calculations in StackedBarSeries

2.3 Changes in version 2.0.1

Version 2.0.1

New Features

- Added Series.getPadding, Series.setPadding methods
- Added Series.padding property
- Added Chart.legend_state (normal, minimized)
- Added Chart.legendMinimize, Chart.legendRestore, Chart.getLegendState methods
- Added ignoreBounds parameter to Axis.pointToPixel

Bug Fixes

- BarSeries / FloatingBarSeries - Fixed dimension calculation which caused bars to be missed if the mouse was close to an axis
- BarSeries / FloatingBarSeries - Updated point location / distance to return exact matches when the mouse is within a bar
- Fixed issue with chart files when in a directory named ejchart_
- Corrected issue with axis.force_static_points and cursor position
- Corrected issue with legend icons causing non-secure warning in https session
- Added catch to all try/finally statements to account for IE bug
- Corrected adjustment to so that crosshairs position exactly under the cursor
- Renamed variable int to interval to correct Safari 2 parse issue
- Updated default series padding to account for axis assignment
- Corrected issue with horizontal bars not drawing when rendered 1 pixel wide

Documentation Updates

- Added documentation for META tag options
- Added documentation for Series.findClosestByPoint and Series.findClosestByPixel

2.4 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

2.5 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

2.6 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

2.7 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

2.8 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 EJS.XMLStringDataHandler class
- Added drawPoints, pointSize, pointColor, pointBorderSize, pointBorderColor properties to EJS.LineSeries and EJS.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 onShowHint 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

2.9 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 overlayed 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 occuring 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 EJS 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

2.10 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

2.11 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.

3 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"}
}
```

3.1 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>
```

3.2 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=" " />
```

3.3 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.

4 API Reference

4.1 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.

4.1.1 Properties

Public Properties

DefaultImagePath
DefaultColors
DefaultBarColors
DefaultPieColors
STRINGS

Inherited Properties

none

4.1.1.1 DefaultImagePath

Definition

string **DefaultImagePath** = "images/"

Description

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

4.1.1.2 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.

4.1.1.3 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.

4.1.1.4 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)"  
]
```

```

    "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.

4.1.1.5 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.

4.2 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);
        }
    }
);
```

4.2.1 Properties

4.2.1.1 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}
);
```

4.2.1.2 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}  
);
```

4.2.1.3 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}  
);
```

4.2.1.4 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}  
);
```

4.2.1.5 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 }  
);
```

4.2.1.6 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}  
);
```

4.2.1.7 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}  
);
```

4.2.1.8 auto_zoom

Definition

string **auto_zoom** = undefined

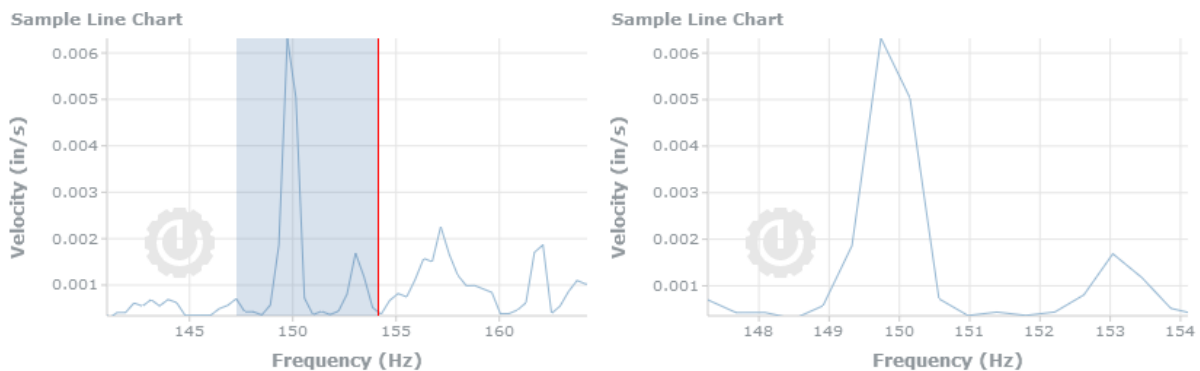
Valid property values:

'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 EJSC.Chart(
    "chart",
    { auto_zoom: 'y' }
);
```

4.2.1.9 axis_bottom

Definition

EJSC.Axis **axis_bottom** = EJSC.LinearAxis

Description

Defines the axis that will be at the bottom 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

>> *Hide the bottom axis*

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

```
        {
            axis_bottom: { visible: false }
        }
    );
```

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

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

4.2.1.10 axis_left

Definition

EJSC.Axis **axis_left** = EJSC.LinearAxis

Description

Defines the axis that will at the left 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

>> Hide the left axis

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

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

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

4.2.1.11 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 })  
    }  
);
```

4.2.1.12 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 })
        }
    );

```

4.2.1.13 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: #ffff00;"></div>
```

4.2.1.14 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.

4.2.1.15 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.

4.2.1.16 legend_state

Definition

```
string legend_state = "normal"
```

Valid options are "normal" or "minimized".

Description

Defines the initial state of the legend. This property is only applicable during chart creation. To modify the state of the legend after the chart has been created, see `Chart.legendMinimize()` and `Chart.legendRestore()`. To check the current state of the legend during chart execution, see `Chart.getLegendState()`.

4.2.1.17 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()`

4.2.1.18 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.

4.2.1.19 message_timeouts**Definition**

```
object message_timeouts = {  
    progress: 500,  
    nodata: 500,  
    info: 500,  
    error: 2000  
};
```

Description

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

4.2.1.20 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}  
);
```

4.2.1.21 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}  
);
```

4.2.1.22 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}  
);
```

4.2.1.23 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}  
);
```

4.2.1.24 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}  
);
```

4.2.1.25 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"}  
);
```

4.2.2 Methods

4.2.2.1 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.

4.2.2.2 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.AlarmSeries

EJSC.AnalogGaugeSeries

EJSC.AreaSeries

EJSC.BarSeries

EJSC.CandlestickSeries

EJSC.DoughnutSeries

EJSC.ErrorSeries

EJSC.FloatingBarSeries

EJSC.FunctionSeries

EJSC.LineSeries

EJSC.OpenHighLowCloseSeries

EJSC.OverUnderSeries

EJSC.PieSeries

EJSC.ScatterSeries

EJSC.StepSeries

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(...));
```

4.2.2.3 clearSelectedPoint

Definition

void [clearSelectedPoint](#)()

Description

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

4.2.2.4 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>

```

4.2.2.5 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 accomodate all legend items. Specifying vertical for orientation will cause the legend to match the charts 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>
```

4.2.2.6 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.

4.2.2.7 getLegendState

Definition

string [getLegendState\(\)](#)

Possible return values:

"normal"
"minimized"

Description

Returns the string which represents the current state of the legend window.

4.2.2.8 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.

4.2.2.9 getSeries

Definition

void [getSeries](#)()

Description

Returns an array of EJSC.Series object currently assigned to the chart.

4.2.2.10 getZoomBoxPixelCoordinates

Definition

object [getZoomBoxPixelCoordinates](#)(string ScreenOrChart)

RETURNS:
integer { 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").

4.2.2.11 **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.

4.2.2.12 **hideZoomBox**

Definition

void **hideZoomBox**()

Description

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

4.2.2.13 **legendMinimize**

Definition

void **legendMinimize**()

Description

Minimizes the legend window. This has no effect if the legend window is already in a minimized state.

4.2.2.14 **legendRestore**

Definition

void **legendRestore**()

Description

This method restores the legend window from its minimized state to its previous height.

4.2.2.15 **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.

4.2.2.16 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).

4.2.2.17 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.

4.2.2.18 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.findClosestByPoint() methods.

4.2.2.19 setAutoResize

Definition

void **setAutoResize**(boolean auto_resize)

Description

Enables or disables the auto_resize functionality once the chart has been created.

4.2.2.20 setLegendTitle

Definition

void **setLegendTitle**(string title)

Description

Updates the caption in the legend window title bar.

4.2.2.21 setShowLegend**Definition**

void **setShowLegend**(boolean show)

Description

Shows or hides the chart legend and updates the value of Chart.show_legend

4.2.2.22 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.

4.2.2.23 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.

4.2.2.24 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");

```

4.2.3 Events

4.2.3.1 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.

4.2.3.2 onAfterDraw

Definition

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

Description

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

4.2.3.3 onAfterDrawSeries

Definition

void [onAfterDrawSeries](#) (EJSC.Chart chart, CanvasContext ctx)

Description

Called after the series are drawn.

Sends:

chart - The chart object which finished drawing its series
 ctx - The canvas context which was just drawn to

4.2.3.4 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.

4.2.3.5 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.

4.2.3.6 onAfterUnselectPoint

Definition

void **onAfterUnselectPoint**()

Description

Called after a point has lost selection.

4.2.3.7 onAfterZoom

Definition

void **onAfterZoom**(EJSC.Chart chart)

Description

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

4.2.3.8 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.

4.2.3.9 onBeforeDbClick

Definition

boolean **onBeforeDbClick**(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.

4.2.3.10 onBeforeDraw

Definition

boolean **onBeforeDraw**()

Description

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

4.2.3.11 onBeforeDrawSeries

Definition

void **onBeforeDrawSeries**(EJSC.Chart chart, CanvasContext ctx)

Description

Called before the series are drawn.

Sends:

chart - The chart object about to draw its series

ctx - The canvas context about to be drawn to

4.2.3.12 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.

4.2.3.13 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.

4.2.3.14 onContextMenu

Definition

void **onBeforeUnselectPoint**(EJSC.Point point, Event event)

Description

Called when the context menu event is fired on the chart object.

Sends:

chart - The chart object that triggered the event

event - The JavaScript event object

4.2.3.15 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).

4.2.3.16 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.

4.2.3.17 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"

4.2.3.18 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.

4.2.3.19 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.

4.2.4 Deprecated

4.2.4.1 Properties

4.2.4.1.1 force_static_points

DEPRECATED See EJSC.Axis.force_static_points

4.2.4.1.2 force_static_points_x

DEPRECATED See EJSC.Axis.force_static_points

4.2.4.1.3 force_static_points_y

DEPRECATED See EJSC.Axis.force_static_points

4.2.4.1.4 legendTitle

DEPRECATED See EJSC.Chart.legend_title

4.2.4.1.5 show_crosshairs

DEPRECATED See EJSC.Axis.crosshair

4.2.4.1.6 show_grid

DEPRECATED See EJSC.Axis.grid

4.2.4.1.7 show_mouse_position

DEPRECATED See EJSC.Axis.cursor_position

4.2.4.1.8 show_x_axis

DEPRECATED See EJSC.Axis.visible

4.2.4.1.9 show_y_axis

DEPRECATED See EJSC.Axis.visible

4.2.4.1.10 x_axis_caption

DEPRECATED See EJSC.Axis.caption

4.2.4.1.11 x_axis_className

DEPRECATED See EJSC.Axis.caption_class

4.2.4.1.12 x_axis_extremes_ticks

DEPRECATED See EJSC.Axis.extremes_ticks

4.2.4.1.13 x_axis_formatter

DEPRECATED See EJSC.Axis.formatter

4.2.4.1.14 x_axis_max_tick_interval

DEPRECATED See EJSC.Axis.major_ticks

4.2.4.1.15 x_axis_min_tick_interval

DEPRECATED See EJSC.Axis.major_ticks

4.2.4.1.16 x_axis_minor_ticks

DEPRECATED See EJSC.Axis.minor_ticks

4.2.4.1.17 x_axis_size

DEPRECATED See EJSC.Axis.size

4.2.4.1.18 x_axis_stagger_ticks

DEPRECATED See `EJSC.Axis.stagger_ticks`

4.2.4.1.19 x_axis_tick_className

DEPRECATED See `EJSC.Axis.label_class`

4.2.4.1.20 x_axis_tick_count

DEPRECATED See `EJSC.Axis.major_ticks`

4.2.4.1.21 x_cursor_position_caption

DEPRECATED See `EJSC.Axis.cursor_position`

4.2.4.1.22 x_cursor_position_formatter

DEPRECATED See `EJSC.Axis.cursor_position`

4.2.4.1.23 x_max

DEPRECATED See `EJSC.Axis.max_extreme`

4.2.4.1.24 x_min

DEPRECATED See `EJSC.Axis.min_extreme`

4.2.4.1.25 x_value_hint_caption

DEPRECATED See `EJSC.Axis.hint_caption`

4.2.4.1.26 x_zero_plane

DEPRECATED See `EJSC.Axis.zero_plane`

4.2.4.1.27 y_axis_caption

DEPRECATED See `EJSC.Axis.caption`

4.2.4.1.28 y_axis_className

DEPRECATED See `EJSC.Axis.caption_class`

4.2.4.1.29 y_axis_extremes_ticks

DEPRECATED See `EJSC.Axis.extremes_ticks`

4.2.4.1.30 y_axis_formatter

DEPRECATED See `EJSC.Axis.formatter`

4.2.4.1.31 y_axis_max_tick_interval

DEPRECATED See `EJSC.Axis.major_ticks`

4.2.4.1.32 y_axis_min_tick_interval

DEPRECATED See `EJSC.Axis.major_ticks`

4.2.4.1.33 y_axis_minor_ticks

DEPRECATED See `EJSC.Axis.minor_ticks`

4.2.4.1.34 y_axis_size

DEPRECATED See EJSC.Axis.size

4.2.4.1.35 y_axis_tick_className

DEPRECATED See EJSC.Axis.label_class

4.2.4.1.36 y_axis_tick_count

DEPRECATED See EJSC.Axis.major_ticks

4.2.4.1.37 y_cursor_position_caption

DEPRECATED See EJSC.Axis.cursor_position

4.2.4.1.38 y_cursor_position_formatter

DEPRECATED See EJSC.Axis.cursor_position

4.2.4.1.39 y_max

DEPRECATED See EJSC.Axis.max_extreme

4.2.4.1.40 y_min

DEPRECATED See EJSC.Axis.min_extreme

4.2.4.1.41 y_value_hint_caption

DEPRECATED See EJSC.Axis.hint_caption

4.2.4.1.42 y_zero_plane

DEPRECATED See EJSC.Axis.zero_plane

4.2.4.2 Methods

4.2.4.2.1 addXAxisBin

DEPRECATED See EJSC.Axis.addBin

4.2.4.2.2 addYAxisBin

DEPRECATED See EJSC.Axis.addBin

4.2.4.2.3 convertPixelToPoint

DEPRECATED See EJSC.Axis.pixelToPoint

4.2.4.2.4 convertPointToPixel

DEPRECATED See EJSC.Axis.pointToPixel

4.2.4.2.5 findClosestPointInSeries

DEPRECATED See EJSC.Series.findClosestByPoint

4.2.4.2.6 hideGrid

DEPRECATED See EJSC.Axis.hideGrid

4.2.4.2.7 hideXAxis

DEPRECATED See `EJSC.Axis.hide`

4.2.4.2.8 hideYAxis

DEPRECATED See `EJSC.Axis.hide`

4.2.4.2.9 getXExtremes

DEPRECATED See `EJSC.Axis.getExtremes`

4.2.4.2.10 getYExtremes

DEPRECATED See `EJSC.Axis.getExtremes`

4.2.4.2.11 getZoom

DEPRECATED See `EJSC.Axis.getZoom`

4.2.4.2.12 removeXAxisBin

DEPRECATED See `EJSC.Axis.removeBin`

4.2.4.2.13 removeYAxisBin

DEPRECATED See `EJSC.Axis.removeBin`

4.2.4.2.14 selectClosestPoint

DEPRECATED See `EJSC.Chart.findClosestPoint`

4.2.4.2.15 setCrosshairs

DEPRECATED See `EJSC.Axis.setCrosshair`

4.2.4.2.16 setXAxisCaption

DEPRECATED See `EJSC.Axis.setCaption`

4.2.4.2.17 setXExtremes

DEPRECATED See `EJSC.Axis.setExtremes`

4.2.4.2.18 setYAxisCaption

DEPRECATED See `EJSC.Axis.setCaption`

4.2.4.2.19 setYExtremes

DEPRECATED See `EJSC.Axis.setExtremes`

4.2.4.2.20 setZoom

DEPRECATED See `EJSC.Axis.setZoom`

4.2.4.2.21 showGrid

DEPRECATED See `EJSC.Axis.showGrid`

4.2.4.2.22 showXAxis

DEPRECATED See `EJSC.Axis.show`

4.2.4.2.23 showYAxis

DEPRECATED See `EJSC.Axis.show`

4.2.4.3 Events

4.2.4.3.1 onAfterShowCrosshairs

DEPRECATED

There is no direct replacement for this event. See `EJSC.Axis.onShowCrosshair`

4.2.4.3.2 onBeforeBeginZoom

DEPRECATED See `EJSC.Chart.onUserBeginZoom`

4.2.4.3.3 onBeforeEndZoom

DEPRECATED See `EJSC.Chart.onUserEndZoom`

4.2.4.3.4 onShowCrosshairs

DEPRECATED

There is no direct replacement for this event. See `EJSC.Axis.onShowCrosshair`

4.2.4.3.5 onXAxisNeedsTicks

DEPRECATED See `EJSC.Axis.onNeedsTicks`

4.2.4.3.6 onYAxisNeedsTicks

DEPRECATED See `EJSC.Axis.onNeedsTicks`

4.3 Axis Types

4.3.1 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"  
    })  
});
```

4.3.1.1 Properties

4.3.1.1.1 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.

4.3.1.1.2 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" }    }  
}
```

```
    }
  } );
}
```

4.3.1.1.3 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" }
} );
```

4.3.1.1.4 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" }
} );
```

4.3.1.1.5 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" }  
} );
```

4.3.1.1.6 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 }  
    }  
} );
```

4.3.1.1.7 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"  
        }  
    }  
} );
```

4.3.1.1.8 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 }  
} );
```

4.3.1.1.9 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 }  
} );
```

4.3.1.1.10 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_c  
  }  
} );
```

4.3.1.1.11 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 }  
  }  
} );
```

4.3.1.1.12 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:"
    }
} );
```

4.3.1.1.13 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
    }
} );
```

4.3.1.1.14 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.

4.3.1.1.15 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
  }
} );
```

4.3.1.1.16 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
        }
    } );

```

4.3.1.1.17 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" }
    }
} );

```

4.3.1.1.18 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 }  
    } );
```

4.3.1.1.19 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 }  
} );
```

4.3.1.1.20 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 }  
} );
```

4.3.1.1.21 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 }
    }
} );
```

4.3.1.2 Methods

4.3.1.2.1 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(...)
);
```

4.3.1.2.2 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`

4.3.1.2.3 `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`

4.3.1.2.4 `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.

4.3.1.2.5 `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.

4.3.1.2.6 `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.

4.3.1.2.7 **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.

4.3.1.2.8 **pointToPixel** (inherited)

Definition

integer **pointToPixel**(number coordinate)

object **pointToPixel**(number coordinate, boolean ignoreBounds)

```
object result: {  
    p: integer,  
    outsideBounds: boolean  
}
```

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)

When ignoreBounds is not specified or specified as undefined, the result will be NaN if the coordinate provided is outside the currently displayed range.

4.3.1.2.9 **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.

4.3.1.2.10 `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)

4.3.1.2.11 `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`.

4.3.1.2.12 `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.

4.3.1.2.13 `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 );
```

4.3.1.2.14 setZoom (inherited)

Definition:

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

Description

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

4.3.1.2.15 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.

4.3.1.2.16 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.

4.3.1.3 Events

4.3.1.3.1 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.

4.3.1.3.2 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.

4.3.1.3.3 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;  
}
```

4.3.1.3.4 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.

4.3.1.3.5 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.

4.3.2 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"
    })
  });
```

4.3.2.1 Properties

4.3.2.1.1 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.

4.3.2.1.2 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
```

4.3.2.1.3 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" }  
    }  
} );
```

4.3.2.1.4 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" }  
} );
```

4.3.2.1.5 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" }  
} );
```

4.3.2.1.6 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" }  
} );
```

4.3.2.1.7 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 }  
    }  
} );
```

4.3.2.1.8 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"
    }
  }
});
```

4.3.2.1.9 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 }
});
```

4.3.2.1.10 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 }
    } );
```

4.3.2.1.11 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_c
    }
} );
```

4.3.2.1.12 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 }
    }
} );
```

4.3.2.1.13 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:"
    }
} );
```

4.3.2.1.14 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 EJSC.Chart( "chart", {
    axis_bottom: {
        label_class: "AxisTickLabels",
        size: 48
    }
} );
```

4.3.2.1.15 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 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.

4.3.2.1.16 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
    }
} );

```

4.3.2.1.17 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  
    }  
} );
```

4.3.2.1.18 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" }  
    }  
} );
```

4.3.2.1.19 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 }  
} );
```

4.3.2.1.20 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 }  
} );
```

4.3.2.1.21 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 }  
} );
```

4.3.2.1.22 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 EJS.C.BarSeries, EJS.C.StackedBarSeries and EJS.C.AreaSeries changed.

Example

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

```

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

```

4.3.2.2 Methods

4.3.2.2.1 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 EJS.C.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(...)  
);
```

4.3.2.2.2 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

4.3.2.2.3 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

4.3.2.2.4 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.

4.3.2.2.5 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.

4.3.2.2.6 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.

4.3.2.2.7 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.

4.3.2.2.8 pointToPixel (inherited)

Definition

integer **pointToPixel**(number coordinate)

object **pointToPixel**(number coordinate, boolean ignoreBounds)

```
object result: {  
    p: integer,  
    outsideBounds: boolean  
}
```

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)

When ignoreBounds is not specified or specified as undefined, the result will be NaN if the coordinate provided is outside the currently displayed range.

4.3.2.2.9 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.

4.3.2.2.10 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)

4.3.2.2.11 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.

4.3.2.2.12 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.

4.3.2.2.13 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 );
```

4.3.2.2.14 setZoom (inherited)

Definition:

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

Description

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

4.3.2.2.15 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.

4.3.2.2.16 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.

4.3.2.3 Events

4.3.2.3.1 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.

4.3.2.3.2 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.

4.3.2.3.3 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;
    }

```

4.3.2.3.4 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.

4.3.2.3.5 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.

4.3.3 DateAxis

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

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

Constructor

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

Example

```

var chart = new EJSC.Chart("myChart", {
    axis_bottom: new EJSC.DateAxis({
        caption: "Year/Month"
    })
});

```

Defining Properties and Events

DateAxis 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.DateAxis()
});
chart.axis_bottom.caption = "Year/Month/Day";
chart.axis_bottom.color = "#F00";

```

Setting properties in batch

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

4.3.3.1 Properties

4.3.3.1.1 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.

4.3.3.1.2 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" }  
    }  
} );
```

4.3.3.1.3 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" }  
} );
```

4.3.3.1.4 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" }  
} );
```

4.3.3.1.5 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" }  
} );
```

4.3.3.1.6 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 }  
    }  
} );
```

4.3.3.1.7 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"  
            }  
        }  
    }  
};
```

4.3.3.1.8 extreme_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 }  
});
```

4.3.3.1.9 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 }  
});
```

4.3.3.1.10 formatters (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_c  
    }  
} );
```

4.3.3.1.11 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 }  
    }  
} );
```

4.3.3.1.12 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
    }
} );

```

4.3.3.1.13 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
    }
} );

```

4.3.3.1.14 major_ticks (inherited)

Definition

```

object major_ticks = {
    thickness: 1,
    size: 4,
    color: undefined,
    opacity: 100,
    show: true,
    increment: undefined,
    offset: 0
}

```

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

increment: (values: `regex('[0-9](.*)[YMWDHNSZ]')`)

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

4.3.3.1.15 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  
    }  
} );
```

4.3.3.1.16 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" }  
    }  
} );
```

4.3.3.1.17 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 }  
} );
```

4.3.3.1.18 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 }  
} );
```

4.3.3.1.19 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 }  
} );
```

4.3.3.1.20 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 }  
    }  
} );
```

4.3.3.2 Methods

4.3.3.2.1 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(...)
);
```

4.3.3.2.2 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

4.3.3.2.3 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

4.3.3.2.4 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.

4.3.3.2.5 `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.

4.3.3.2.6 `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.

4.3.3.2.7 `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.

4.3.3.2.8 `pointToPixel` (inherited)

Definition

integer `pointToPixel`(number coordinate)

object `pointToPixel`(number coordinate, boolean ignoreBounds)

```
object result: {  
    p: integer,  
    outsideBounds: boolean  
}
```

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)

When ignoreBounds is not specified or specified as undefined, the result will be NaN if the coordinate provided is outside the currently displayed range.

4.3.3.2.9 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.

4.3.3.2.10 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)

4.3.3.2.11 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.

4.3.3.2.12 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.

4.3.3.2.13 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 );
```

4.3.3.2.14 setZoom (inherited)

Definition:

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

Description

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

4.3.3.2.15 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.

4.3.3.2.16 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.

4.3.3.3 Events

4.3.3.3.1 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.

4.3.3.3.2 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.

4.3.3.3.3 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;  
}
```

4.3.3.3.4 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.

4.3.3.3.5 onShowCursorPosition (inherited)

Definition

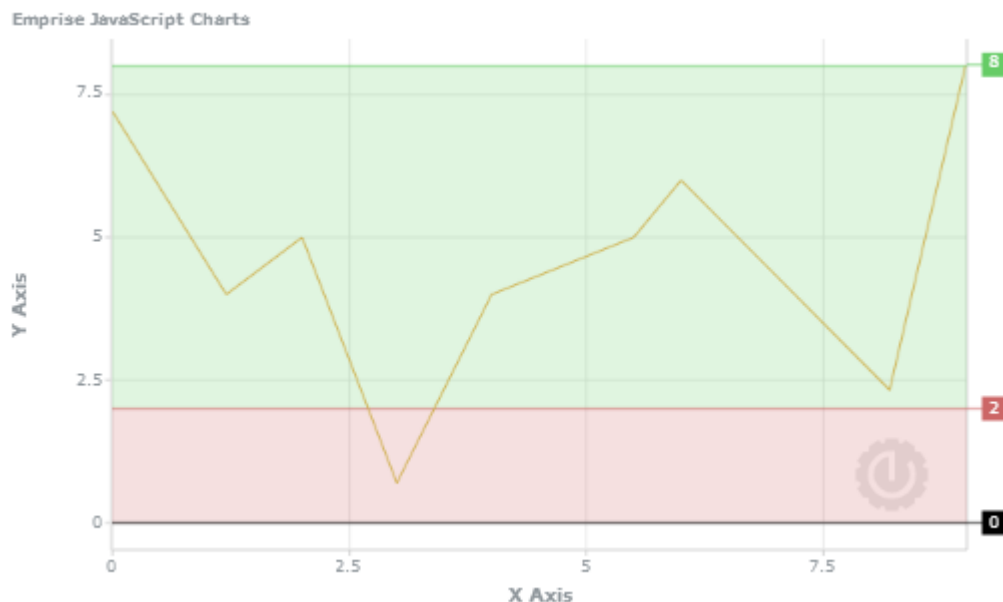
void [onShowCursorPosition](#)(float coordinate, EJSC.Axis axis, EJSC.Chart chart)

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.

4.4 Series Types

4.4.1 EJSC.AlarmSeries



The AlarmSeries is rendered as a line denoting an alarm value, with an optional fill to denote an alarm range.

The constructor expects a value and an optional set of object properties.

Constructor

```
EJSC.AlarmSeries( value, {object options} )
```

Example

```
var AlarmSeries = new EJSC.AlarmSeries( 10, {
    title: "Alarm Series"
} );
```

Defining Properties and Events

AlarmSeries 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.AlarmSeries(10);
mySeries.title = "Alarm Series";
mySeries.opacity = 20;
mySeries.fill = 'down';
mySeries.fillTo = 2;
mySeries.flag = {visible:true};
mySeries.color = '#66cc66';
```

Setting properties in batch

```
var mySeries = new EJSC.AlarmSeries( 10, {
    title: "Alarm Series",
```

```
        opacity: 20,  
        fill: 'down',  
        fillTo: 2,  
        flag: {visible:true},  
        color: '#66cc66',  
    } );
```

4.4.1.1 Properties

4.4.1.1.1 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)

4.4.1.1.2 axis

Definition

string **axis** = 'left'

Description

Defines the axis the alarm should display on.

Positions:

- top
- left
- bottom
- right

4.4.1.1.3 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)

4.4.1.1.4 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()`

4.4.1.1.5 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.

4.4.1.1.6 fill

Definition

string **fill** = undefined

Description

Defines if the alarm should fill from the value up or down

Fill Positions:

- up
- down

4.4.1.1.7 fillTo

Definition

float **fillTo** = undefined

Description

Defines the value the alarm should fill to (if undefined, fills to the min/max of the axis)

4.4.1.1.8 flag

Definition

```
object flag = {
    visible:      false,
    axis:         'opposite',
    offset:       0,
    opacity:      100,
    fontColor:    '#fff'
}
```

Description

Defines attributes for the flag attached to the alarm

Properties

Boolean	visible	– Defines if the flag is visible
string	axis	– ('same','opposite'): Defines if the flag should display on the axis it's attached to or the opposite one
integer	offset	– Defines the number of pixels to move the flag to make room if flags are too close
integer	opacity	– (0-100): The % opacity for the fill of the flag
color	fontColor	– The color to drag the text on the flag in

4.4.1.1.9 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]"
} );
```

4.4.1.1.10 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.

4.4.1.1.11 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()`

4.4.1.1.12 lineWidth (inherited)

Definition

integer **lineWidth** = 1

Description

Defines the width of the line connecting series points.

4.4.1.1.13 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()`

4.4.1.1.14 padding (inherited)

Definition

```
object padding = {  
  x_axis_min: undefined,  
  x_axis_max: undefined,  
  y_axis_min: undefined,  
  y_axis_max: undefined  
}
```

Description

The padding property may be used to override the series default padding. Padding is the amount of pixels added to the min and max series values in order to push the extremes and prevent the series from hitting the edge of the chart.

The amount of default padding is dependant upon the series type and axis configuration and may not be applicable to all series types.

This property is only applicable during series creation. To retrieve or modify the series padding after creation please see `Series.getPadding()` and `Series.setPadding()`

Example

>> Remove all padding from the BarSeries

```
var barSeries = new EJSC.BarSeries(new EJSC.ArrayDataHandler([..data..]), {
    padding: {
        x_axis_min: 0,
        x_axis_max: 0,
        y_axis_min: 0,
        y_axis_max: 0
    }
});
```

4.4.1.1.15 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.

4.4.1.1.16 visible (inherited)

Definition

boolean **visible** = true

Description

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

4.4.1.2 Methods

4.4.1.2.1 findClosestByPixel (inherited)

Definition

EJSC.Point **findClosestByPixel**(object coordinates)

```
point = {
    x: screen coordinate,
    y: screen coordinate
}
```

Description

Returns the point object closest to the screen pixel coordinates provided. The x and y properties of the

point object should be in the same scale as the x and y axes assigned to the series. To locate points based on pixel coordinates, see `Series.findClosestByPixel()`

4.4.1.2.2 `findClosestByPoint` (inherited)

Definition

EJSC.Point **`findClosestByPoint`**(object coordinate)

```
point = {  
    x: axis coordinate,  
    y: axis coordinate  
}
```

Description

Returns the point object closest to the axis coordinates provided. The x and y properties of the coordinate object should be based on the top left of the viewport and take into account the page scroll. To locate points based on axis coordinates, see `Series.findClosestByPoint()`

4.4.1.2.3 `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.

4.4.1.2.4 `getPadding` (inherited)

Definition

object **`getPadding`**()

RETURNS:

```
{  
    x_axis_min: number,  
    x_axis_max: number,  
    y_axis_min: number,  
    y_axis_max: number  
}
```

Description

Returns an object containing the series current padding. If padding has been set either by the `Series.setPadding()` method of the `Series.padding` property during construction the result of this method will be adjusted to return the most current padding values.

4.4.1.2.5 getVisibility (inherited)

Definition

boolean **getVisibility**()

Description

Returns a boolean indicating the series current visible state

4.4.1.2.6 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.

4.4.1.2.7 hideLegend (inherited)

Definition

void **hideLegend**()

Description

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

4.4.1.2.8 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.

4.4.1.2.9 setColor (inherited)

Definition

void **setColor**(string color)

Description

Changes the series color and causes the chart to redraw.

4.4.1.2.10 setColoredLegend (inherited)

Definition

void **setColoredLegend**(boolean coloredLegend)

Description

Sets the EJSC.Series.coloredLegend property and updates the legend to reflect the change.

4.4.1.2.11 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.

4.4.1.2.12 setLineOpacity (inherited)

Definition

void **setLineOpacity**(integer opacity)

Description

Sets the EJSC.Series.lineOpacity property and redraws the series to reflect the change.

4.4.1.2.13 setLineWidth (inherited)

Definition

void **setLineWidth**(integer width)

Description

Updates the lineWidth property and redraws the chart.

4.4.1.2.14 setOpacity (inherited)

Definition

void **setOpacity**(integer opacity)

Description

Sets the EJSC.Series.opacity property and redraws the series to reflect the change.

4.4.1.2.15 setPadding (inherited)

Definition

void **setPadding**(object Padding, boolean Redraw)

The padding object should be defined as when used in the series constructor. See Series.padding.

Description

This method updates the user-defined series padding and optionally recalculates the chart extremes and redraws the chart.

4.4.1.2.16 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.

4.4.1.2.17 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.

4.4.1.2.18 showLegend (inherited)

Definition

void **showLegend**()

Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

4.4.1.3 Events

4.4.1.3.1 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.

4.4.1.3.2 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.

4.4.1.3.3 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.

4.4.1.3.4 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.

4.4.2 EJSC.AnalogGaugeSeries

Emprise JavaScript Charts



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 }
```

```
);
```

4.4.2.1 Properties

4.4.2.1.1 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.

4.4.2.1.2 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	innerBorderWidth	– Defines the width of the inner border.
integer	innerBorderVisible	– Defines whether the inner border is visible or not.

integer	opacity	– Defines the opacity of the
background color of the axis.		
string	outerBorderColor	– Defines the color of the outer border.
integer	outerBorderOpacity	– Defines the opacity of the outer border.
integer	outerBorderWidth	– Defines the width of the outer border.
boolean	outerBorderVisible	– Defines whether the outer border is visible or not.
integer	thickness	– Defines the thickness of the axis.

4.4.2.1.3 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.

4.4.2.1.4 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)"

4.4.2.1.5 fillOpacity

Definition

integer **fillOpacity** = 100

Description

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

4.4.2.1.6 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;).

4.4.2.1.7 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

4.4.2.1.8 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.

4.4.2.1.9 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.

4.4.2.1.10 marker_position

Definition

string **marker_position** = "outer"

Description

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

Quadrants:

- outer
- inner

4.4.2.1.11 max

Definition

float **max** = 100

Description

Defines the maximum value to be displayed on the gauge.

4.4.2.1.12 min

Definition

float **min** = 0

Description

Defines the minimum value to be displayed on the gauge.

4.4.2.1.13 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	color	– Defines the color of the minor tick marks.
integer	count	– Defines the number of minor tick marks to display
integer	offset	– Defines the distance (in pixels) the minor tick marks
integer	opacity	– Defines the opacity of the minor tick marks.
integer	size	– Defines the width (in pixels) of the minor tick
integer	thickness	– Defines the thickness (in pixels) of the minor tick marks.

4.4.2.1.14 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	borderColor	– Defines the color of the border.
integer	borderOpacity	– Defines the opacity of the border.
integer	borderWidth	– Defines the width of the border.
string	color	– Defines the background color of the needle.
integer	opacity	– Defines the opacity of the needle.
integer	size	– Defines the width (in pixels) of the needle at its
base.		

4.4.2.1.15 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
- left
- right

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

4.4.2.1.17 range_degrees

Definition

integer **range_degrees** = 180

Description

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

4.4.2.1.18 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)' ]  
];
```

4.4.2.1.19 start_degree

Definition

integer **start_degree** = 270

Description

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

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

4.4.2.1.21 tickCount

Definition

integer **tickCount** = 11

Description

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

4.4.2.1.22 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.

4.4.2.1.23 visible (inherited)

Definition

boolean **visible** = true

Description

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

4.4.2.1.24 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;).

4.4.2.1.25 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

4.4.2.2 Methods

4.4.2.2.1 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.

4.4.2.2.2 getVisibility (inherited)

Definition

boolean **getVisibility**()

Description

Returns a boolean indicating the series current visible state

4.4.2.2.3 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.

4.4.2.2.4 hideLegend (inherited)

Definition

void **hideLegend**()

Description

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

4.4.2.2.5 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.

4.4.2.2.6 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.

4.4.2.2.7 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.

4.4.2.2.8 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.

4.4.2.2.9 showLegend (inherited)

Definition

void **showLegend**()

Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

4.4.2.3 Events

4.4.2.3.1 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.

4.4.2.3.2 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.

4.4.2.3.3 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.

4.4.2.4 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

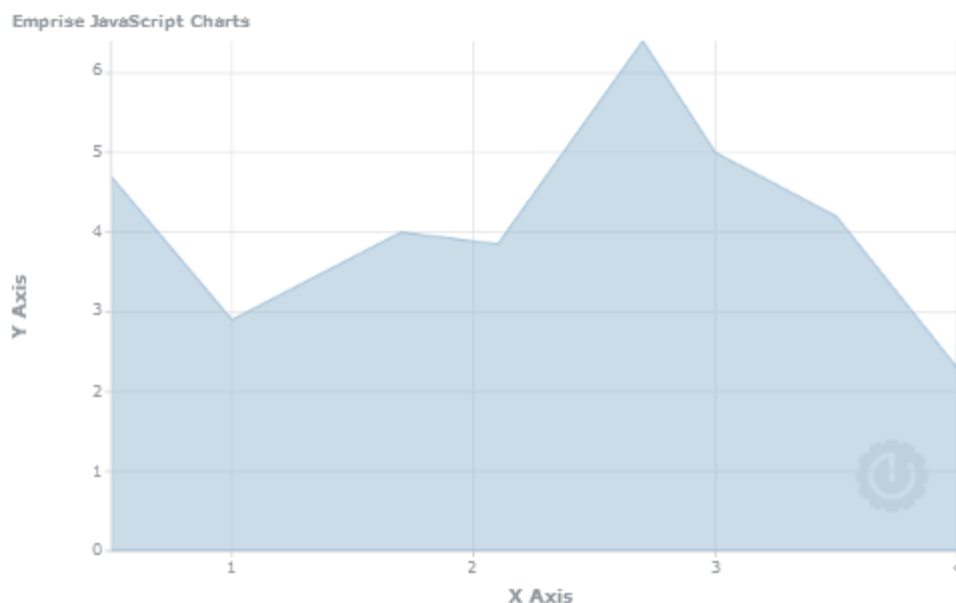
"x|null|userdata"

JSON (EJSC.JSONFileDataHandler, EJSC.JSONStringDataHandler):

Unused parameters may be omitted.

```
[{"x":"number or string","label":"string","userdata":"string"}]
```

4.4.3 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" }
);
```

4.4.3.1 Properties

4.4.3.1.1 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)

4.4.3.1.2 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).

4.4.3.1.3 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)

4.4.3.1.4 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()`

4.4.3.1.5 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.

4.4.3.1.6 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.

4.4.3.1.7 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]"  
} );
```

4.4.3.1.8 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.

4.4.3.1.9 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()

4.4.3.1.10 lineWidth (inherited)

Definition

integer **lineWidth** = 1

Description

Defines the width of the line connecting series points.

4.4.3.1.11 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()

4.4.3.1.12 padding (inherited)

Definition

```
object padding = {  
    x_axis_min: undefined,  
    x_axis_max: undefined,  
    y_axis_min: undefined,  
    y_axis_max: undefined  
}
```

Description

The padding property may be used to override the series default padding. Padding is the amount of pixels added to the min and max series values in order to push the extremes and prevent the series from hitting the edge of the chart.

The amount of default padding is dependant upon the series type and axis configuration and may not be applicable to all series types.

This property is only applicable during series creation. To retrieve or modify the series padding after creation please see `Series.getPadding()` and `Series.setPadding()`

Example

>> Remove all padding from the BarSeries

```
var barSeries = new EJSC.BarSeries(new EJSC.ArrayDataHandler([..data..]), {  
    padding: {  
        x_axis_min: 0,  
        x_axis_max: 0,  
        y_axis_min: 0,  
        y_axis_max: 0  
    }  
});
```

4.4.3.1.13 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.

4.4.3.1.14 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.

4.4.3.1.15 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.

4.4.3.1.16 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.

4.4.3.1.17 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.

4.4.3.1.18 visible (inherited)

Definition

boolean **visible** = true

Description

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

4.4.3.1.19 x_axis (inherited)

Definition

string **x_axis** = "bottom"

Description

Defines the horizontal axis to use for X values. Valid options are "top" or "bottom".

4.4.3.1.20 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

4.4.3.1.21 y_axis (inherited)

Definition

string **y_axis** = "left"

Description

Defines the vertical axis to use for Y values. Valid options are "left" or "right".

4.4.3.1.22 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

4.4.3.2 Methods

4.4.3.2.1 findClosestByPixel (inherited)

Definition

EJSC.Point **findClosestByPixel**(object coordinates)

```
point = {  
  x: screen coordinate,  
  y: screen coordinate  
}
```

Description

Returns the point object closest to the screen pixel coordinates provided. The x and y properties of the point object should be in the same scale as the x and y axes assigned to the series. To locate points based on pixel coordinates, see `Series.findClosestByPixel()`

4.4.3.2.2 findClosestByPoint (inherited)

Definition

EJSC.Point **findClosestByPoint**(object coordinate)

```
point = {  
  x: axis coordinate,  
  y: axis coordinate  
}
```

Description

Returns the point object closest to the axis coordinates provided. The x and y properties of the coordinate object should be based on the top left of the viewport and take into account the page scroll. To locate points based on axis coordinates, see `Series.findClosestByPoint()`

4.4.3.2.3 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.

4.4.3.2.4 getPadding (inherited)

Definition

object **getPadding**()

RETURNS:

```
{  
    x_axis_min: number,  
    x_axis_max: number,  
    y_axis_min: number,  
    y_axis_max: number  
}
```

Description

Returns an object containing the series current padding. If padding has been set either by the `Series.setPadding()` method of the `Series.padding` property during construction the result of this method will be adjusted to return the most current padding values.

4.4.3.2.5 `getVisibility` (inherited)

Definition

boolean `getVisibility()`

Description

Returns a boolean indicating the series current visible state

4.4.3.2.6 `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.

4.4.3.2.7 `hideLegend` (inherited)

Definition

void `hideLegend()`

Description

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

4.4.3.2.8 `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.

4.4.3.2.9 setColor (inherited)

Definition

```
void setColor( string color )
```

Description

Changes the series color and causes the chart to redraw.

4.4.3.2.10 setColoredLegend (inherited)

Definition

```
void setColoredLegend( boolean coloredLegend )
```

Description

Sets the EJSCT.Series.coloredLegend property and updates the legend to reflect the change.

4.4.3.2.11 setDataHandler (inherited)

Definition

```
void setDataHandler( EJSCT.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.

4.4.3.2.12 setLineOpacity (inherited)

Definition

```
void setLineOpacity( integer opacity )
```

Description

Sets the EJSCT.Series.lineOpacity property and redraws the series to reflect the change.

4.4.3.2.13 setLineWidth (inherited)

Definition

```
void setLineWidth( integer width )
```

Description

Updates the lineWidth property and redraws the chart.

4.4.3.2.14 setOpacity (inherited)

Definition

void **setOpacity**(integer opacity)

Description

Sets the EJSC.Series.opacity property and redraws the series to reflect the change.

4.4.3.2.15 setPadding (inherited)

Definition

void **setPadding**(object Padding, boolean Redraw)

The padding object should be defined as when used in the series constructor. See Series.padding.

Description

This method updates the user-defined series padding and optionally recalculates the chart extremes and redraws the chart.

4.4.3.2.16 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.

4.4.3.2.17 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.

4.4.3.2.18 showLegend (inherited)

Definition

void **showLegend**()

Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

4.4.3.3 Events

4.4.3.3.1 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.

4.4.3.3.2 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.

4.4.3.3.3 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.

4.4.3.3.4 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.

4.4.3.4 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>
or string" label="string" userdata="string"/>
            <point x="number or string" y="number
or string" label="string" userdata="string"/>
            <point x="number or string" y="number
            </plot>
        </graph>

```

Short:

```

        <G>
            <L>
string" label="string" userdata="string"/>
            <P x="number or string" y="number or
string" label="string" userdata="string"/>
            <P x="number or string" y="number or
            </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):

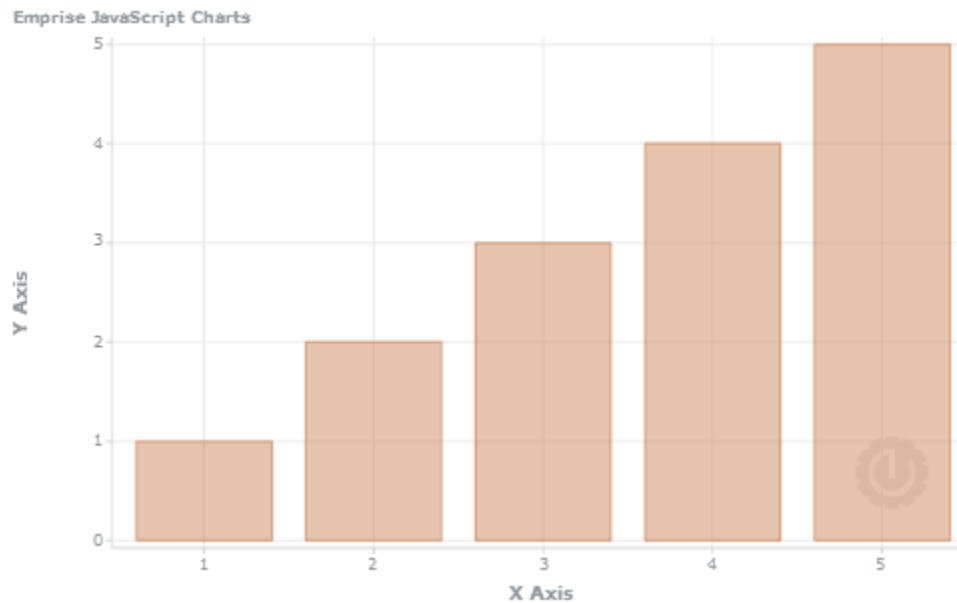
Unused parameters may be omitted.

```
[
    {
        "x": "number or string",
        "y": "number or string",
        "label": "string",
        "userdata": "string"
    },
    {
        "x": "number or string",
        "y": "number or string",
        "label": "string",
        "userdata": "string"
    }
]
```

4.4.3.5 Text Replacement Options

String	With
[chart_title]	title of the chart
[series_title]	title of the series the selected point resides in
[xaxis]	caption of the series x axis
[yaxis]	caption of the series y axis
[x]	preformatted X value of the point
[y]	preformatted Y value of the point
[label]	label property of the current point

4.4.4 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" }
```



```
);
```

4.4.4.1 Properties

4.4.4.1.1 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)

4.4.4.1.2 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)

4.4.4.1.3 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()

4.4.4.1.4 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.

4.4.4.1.5 `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.

4.4.4.1.6 `groupedBars`

Definition

boolean `groupedBars` = true

Description

Defines whether the bars from different bar series in a single chart are grouped or overlayed.

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 overlayed 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));
```

4.4.4.1.7 `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]"  
} );
```

4.4.4.1.8 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  
    }  
));
```

4.4.4.1.9 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.

4.4.4.1.10 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()

4.4.4.1.11 lineWidth (inherited)

Definition

integer **lineWidth** = 1

Description

Defines the width of the line connecting series points.

4.4.4.1.12 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()`

4.4.4.1.13 padding (inherited)

Definition

```
object padding = {  
  x_axis_min: undefined,  
  x_axis_max: undefined,  
  y_axis_min: undefined,  
  y_axis_max: undefined  
}
```

Description

The padding property may be used to override the series default padding. Padding is the amount of pixels added to the min and max series values in order to push the extremes and prevent the series from hitting the edge of the chart.

The amount of default padding is dependant upon the series type and axis configuration and may not be applicable to all series types.

This property is only applicable during series creation. To retrieve or modify the series padding after creation please see `Series.getPadding()` and `Series.setPadding()`

Example

>> Remove all padding from the BarSeries

```
var barSeries = new EJSC.BarSeries(new EJSC.ArrayDataHandler([..data..]), {  
  padding: {  
    x_axis_min: 0,  
    x_axis_max: 0,  
    y_axis_min: 0,  
    y_axis_max: 0  
  }  
});
```

4.4.4.1.14 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.

4.4.4.1.15 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),
    opacity: integer, // Represents percent, i.e. 50 = 50%
    lineOpacity: integer, // Represents percent, i.e. 50 = 50%
    lineWidth: integer // Represents the line width in pixels
}
ex: "rgb(255,0,0)"

```

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 }
        ]
    }
));

```

4.4.4.1.16 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.

4.4.4.1.17 `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.

4.4.4.1.18 `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
    }
));
```

4.4.4.1.19 `visible` (inherited)**Definition**

```
boolean visible = true
```

Description

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

4.4.4.1.20 x_axis (inherited)

Definition

string **x_axis** = "bottom"

Description

Defines the horizontal axis to use for X values. Valid options are "top" or "bottom".

4.4.4.1.21 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

4.4.4.1.22 y_axis (inherited)

Definition

string **y_axis** = "left"

Description

Defines the vertical axis to use for Y values. Valid options are "left" or "right".

4.4.4.1.23 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

4.4.4.2 Methods

4.4.4.2.1 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.

4.4.4.2.2 clearRanges

Definition

void **clearRanges**(boolean redraw)

Description

Clears all ranges defined for the series and optionally redraws the the chart.

4.4.4.2.3 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.

4.4.4.2.4 findClosestByPixel (inherited)

Definition

EJSC.Point **findClosestByPixel**(object coordinates)

```
point = {  
  x: screen coordinate,  
  y: screen coordinate  
}
```

Description

Returns the point object closest to the screen pixel coordinates provided. The x and y properties of the point object should be in the same scale as the x and y axes assigned to the series. To locate points based on pixel coordinates, see `Series.findClosestByPixel()`

4.4.4.2.5 findClosestByPoint (inherited)

Definition

EJSC.Point **findClosestByPoint**(object coordinate)

```
point = {  
  x: axis coordinate,  
  y: axis coordinate  
}
```

Description

Returns the point object closest to the axis coordinates provided. The x and y properties of the coordinate object should be based on the top left of the viewport and take into account the page scroll. To locate points based on axis coordinates, see `Series.findClosestByPoint()`

4.4.4.2.6 getBarSize

Definition

integer **getBarSize**()

Description

This method returns the size in pixels (width or height depending on orientation) of a single bar.

4.4.4.2.7 getBarSizeInPoints

Definition

integer **getBarSizeInPoints**()

Description

This method returns the size in chart points (width or height depending on orientation) of a single bar.

4.4.4.2.8 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.

4.4.4.2.9 getPadding (inherited)

Definition

object **getPadding**()

RETURNS:

```
{  
    x_axis_min: number,  
    x_axis_max: number,  
    y_axis_min: number,  
    y_axis_max: number  
}
```

Description

Returns an object containing the series current padding. If padding has been set either by the `Series.setPadding()` method of the `Series.padding` property during construction the result of this method will be adjusted to return the most current padding values.

4.4.4.2.10 `getPoints`**Definition**

array `getPoints()`

Description

This method returns an array of all `EJSC.BarPoint` objects in the series.

4.4.4.2.11 `getVisibility` (inherited)**Definition**

boolean `getVisibility()`

Description

Returns a boolean indicating the series current visible state

4.4.4.2.12 `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.

4.4.4.2.13 `hideLegend` (inherited)**Definition**

void `hideLegend()`

Description

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

4.4.4.2.14 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.

4.4.4.2.15 setColor (inherited)

Definition

void **setColor**(string color)

Description

Changes the series color and causes the chart to redraw.

4.4.4.2.16 setColoredLegend (inherited)

Definition

void **setColoredLegend**(boolean coloredLegend)

Description

Sets the EJSC.Series.coloredLegend property and updates the legend to reflect the change.

4.4.4.2.17 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.

4.4.4.2.18 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);
```

4.4.4.2.19 `setGroupedBars`**Definition**

void **setGroupedBars**(boolean grouped, boolean redraw)

Description

Changes the chart between grouped and overlaid bars and optionally triggers a redraw.

4.4.4.2.20 `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.

4.4.4.2.21 `setLineWidth` (inherited)**Definition**

void **setLineWidth**(integer width)

Description

Updates the `lineWidth` property and redraws the chart.

4.4.4.2.22 setOpacity (inherited)

Definition

void **setOpacity**(integer opacity)

Description

Sets the EJSCT.Series.opacity property and redraws the series to reflect the change.

4.4.4.2.23 setPadding (inherited)

Definition

void **setPadding**(object Padding, boolean Redraw)

The padding object should be defined as when used in the series constructor. See Series.padding.

Description

This method updates the user-defined series padding and optionally recalculates the chart extremes and redraws the chart.

4.4.4.2.24 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.

4.4.4.2.25 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.

4.4.4.2.26 showLegend (inherited)

Definition

void **showLegend**()

Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

4.4.4.3 Events**4.4.4.3.1 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.

4.4.4.3.2 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.

4.4.4.3.3 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)";
        }
    }
} ));
```

4.4.4.3.4 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.

4.4.4.3.5 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.

4.4.4.4 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> or string" label="string" userdata="string"/>	<point x="number or string" y="number <point x="number or string" y="number
--	--

```

or string" label="string" userdata="string"/>
                                </plot>
    <graph>

```

Short:

```

    <G>
                                <L>
string" label="string" userdata="string"/>
                                <P x="number or string" y="number or
string" label="string" userdata="string"/>
                                <P x="number or string" y="number or
                                </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" }
]

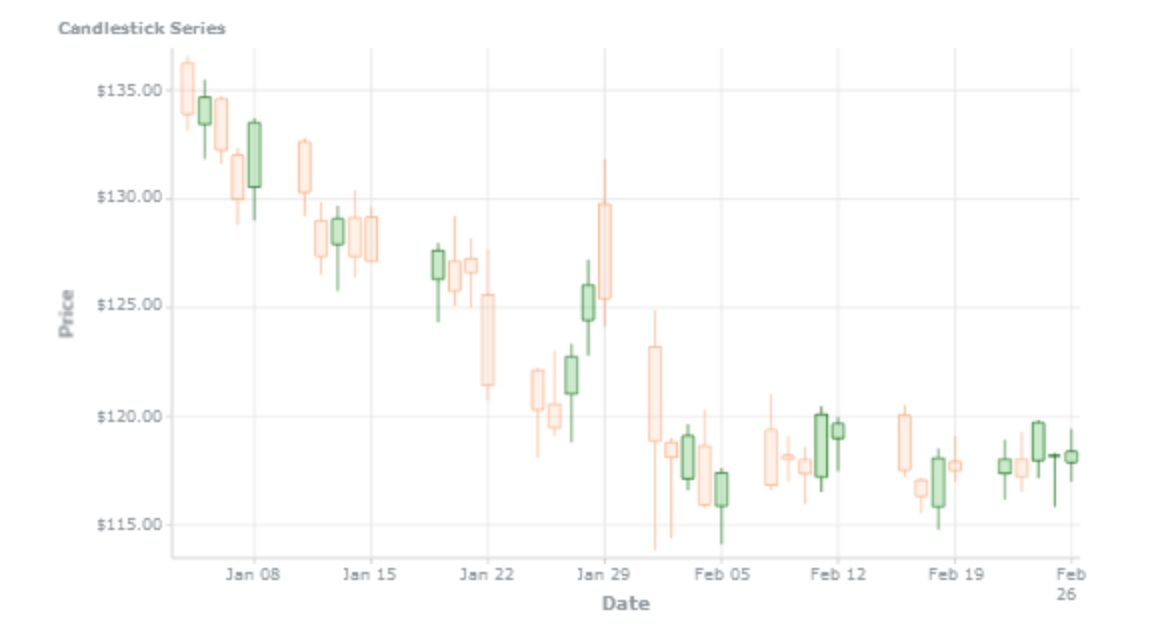
```


]

4.4.4.5 Text Replacement Options

String	With
[chart_title]	title of the chart
[series_title]	title of the series the selected
	point resides in
[xaxis]	caption of the series x axis
[yaxis]	caption of the series y axis
[x]	preformatted X value of the
	point
[y]	preformatted Y value of the
	point
[label]	label property of the current
	point

4.4.5 EJSC.CandlestickSeries



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" }
);
```

4.4.5.1 Properties

4.4.5.1.1 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)

4.4.5.1.2 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)

4.4.5.1.3 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()`

4.4.5.1.4 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.

4.4.5.1.5 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  
    }  
});
```

4.4.5.1.6 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]"
});
```

4.4.5.1.7 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
    }
));
```

4.4.5.1.8 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.

4.4.5.1.9 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()`

4.4.5.1.10 `lineWidth` (inherited)

Definition

integer `lineWidth` = 1

Description

Defines the width of the line connecting series points.

4.4.5.1.11 `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  
        }  
    }  
);
```

4.4.5.1.12 `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()`

4.4.5.1.13 padding (inherited)

Definition

```
object padding = {  
    x_axis_min: undefined,  
    x_axis_max: undefined,  
    y_axis_min: undefined,  
    y_axis_max: undefined  
}
```

Description

The padding property may be used to override the series default padding. Padding is the amount of pixels added to the min and max series values in order to push the extremes and prevent the series from hitting the edge of the chart.

The amount of default padding is dependant upon the series type and axis configuration and may not be applicable to all series types.

This property is only applicable during series creation. To retrieve or modify the series padding after creation please see `Series.getPadding()` and `Series.setPadding()`

Example

>> Remove all padding from the BarSeries

```
var barSeries = new EJSC.BarSeries(new EJSC.ArrayDataHandler([..data..]), {  
    padding: {  
        x_axis_min: 0,  
        x_axis_max: 0,  
        y_axis_min: 0,  
        y_axis_max: 0  
    }  
});
```

4.4.5.1.14 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.

4.4.5.1.15 visible (inherited)

Definition

```
boolean visible = true
```

Description

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

4.4.5.1.16 `x_axis` (inherited)

Definition

string `x_axis` = "bottom"

Description

Defines the horizontal axis to use for X values. Valid options are "top" or "bottom".

4.4.5.1.17 `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

4.4.5.1.18 `y_axis` (inherited)

Definition

string `y_axis` = "left"

Description

Defines the vertical axis to use for Y values. Valid options are "left" or "right".

4.4.5.1.19 `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

4.4.5.2 Methods

4.4.5.2.1 findClosestByPixel (inherited)

Definition

EJSC.Point **findClosestByPixel**(object coordinates)

```
point = {  
  x: screen coordinate,  
  y: screen coordinate  
}
```

Description

Returns the point object closest to the screen pixel coordinates provided. The x and y properties of the point object should be in the same scale as the x and y axes assigned to the series. To locate points based on pixel coordinates, see `Series.findClosestByPixel()`

4.4.5.2.2 findClosestByPoint (inherited)

Definition

EJSC.Point **findClosestByPoint**(object coordinate)

```
point = {  
  x: axis coordinate,  
  y: axis coordinate  
}
```

Description

Returns the point object closest to the axis coordinates provided. The x and y properties of the coordinate object should be based on the top left of the viewport and take into account the page scroll. To locate points based on axis coordinates, see `Series.findClosestByPoint()`

4.4.5.2.3 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.

4.4.5.2.4 getPadding (inherited)

Definition

object **getPadding**()

RETURNS:

```
{  
    x_axis_min: number,  
    x_axis_max: number,  
    y_axis_min: number,  
    y_axis_max: number  
}
```

Description

Returns an object containing the series current padding. If padding has been set either by the `Series.setPadding()` method of the `Series.padding` property during construction the result of this method will be adjusted to return the most current padding values.

4.4.5.2.5 `getVisibility` (inherited)**Definition**

boolean `getVisibility()`

Description

Returns a boolean indicating the series current visible state

4.4.5.2.6 `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.

4.4.5.2.7 `hideLegend` (inherited)**Definition**

void `hideLegend()`

Description

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

4.4.5.2.8 `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.

4.4.5.2.9 setColor (inherited)

Definition

void **setColor**(string color)

Description

Changes the series color and causes the chart to redraw.

4.4.5.2.10 setColoredLegend (inherited)

Definition

void **setColoredLegend**(boolean coloredLegend)

Description

Sets the EJSC.Series.coloredLegend property and updates the legend to reflect the change.

4.4.5.2.11 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.

4.4.5.2.12 setLineOpacity (inherited)

Definition

void **setLineOpacity**(integer opacity)

Description

Sets the EJSC.Series.lineOpacity property and redraws the series to reflect the change.

4.4.5.2.13 setLineWidth (inherited)

Definition

void **setLineWidth**(integer width)

Description

Updates the lineWidth property and redraws the chart.

4.4.5.2.14 setOpacity (inherited)

Definition

void **setOpacity**(integer opacity)

Description

Sets the EJSC.Series.opacity property and redraws the series to reflect the change.

4.4.5.2.15 setPadding (inherited)

Definition

void **setPadding**(object Padding, boolean Redraw)

The padding object should be defined as when used in the series constructor. See Series.padding.

Description

This method updates the user-defined series padding and optionally recalculates the chart extremes and redraws the chart.

4.4.5.2.16 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.

4.4.5.2.17 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.

4.4.5.2.18 showLegend (inherited)

Definition

void **showLegend**()

Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

4.4.5.3 Events

4.4.5.3.1 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.

4.4.5.3.2 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.

4.4.5.3.3 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.

4.4.5.3.4 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.

4.4.5.4 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"
    }
]

```

4.4.5.5 Text Replacement Options

String	With
[chart_title]	title of the chart
[series_title]	title of the series the selected point resides in
[xaxis]	caption of the series x axis
[yaxis]	caption of the series y axis
[x]	preformatted X value of the point
[high]	preformatted high value of the point
[low]	preformatted low value of the point
[open]	preformatted open value of the point
[close]	preformatted close value of the point
[label]	label property of the current point

4.4.6 EJSC.DoughnutSeries

Emprise JavaScript Charts



The DoughnutSeries is rendered similar to the PieSeries, but with part of the center missing. You can use "Donut" instead of Doughnut for shorthand.

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

Example

```
var mySeries = new EJSC.DoughnutSeries(
    new EJSC.ArrayDataHandler([[10,"Label 1"],[20,"Label 2"],[120,"Label3"]]),
    { title: "New Doughnut Series" }
);
```

Defining Properties and Events

DoughnutSeries 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 Doughnut Series";
```

Setting properties in batch

```
var mySeries = new EJSC.PieSeries(
```

```
        ArrayDataHandler([[10,"Label 1"],[20,"Label 2"],[120,"Label 3"]]),  
        { title: "New Doughnut Series" }  
    );
```

4.4.6.1 Properties

4.4.6.1.1 defaultColors (inherited)

Definition

array **defaultColors** = EJSC.DefaultPieColors

Description

Defines the pool of default colors available for pie pieces.

4.4.6.1.2 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.

4.4.6.1.3 doughnutOffset

Definition

float **doughnutOffset** = .5

Description

Float (0-1): Defines the amount of the radius that the donut inner radius should be (0 being like Pie, 1 being inner radius = outer radius)

4.4.6.1.4 height (inherited)

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).

4.4.6.1.5 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]"  
} );
```

4.4.6.1.6 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.

4.4.6.1.7 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()

4.4.6.1.8 lineWidth (inherited)

Definition

integer **lineWidth** = 1

Description

Defines the width of the line connecting series points.

4.4.6.1.9 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()`

4.4.6.1.10 position (inherited)

Definition

string **position** = "center"

Description

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

Quadrants:

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

4.4.6.1.11 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.

4.4.6.1.12 total_value (inherited)

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.

4.4.6.1.13 treeLegend (inherited)

Definition

boolean **treeLegend** = true

Description

Defines whether to display each pie piece individually in the legend.

4.4.6.1.14 visible (inherited)

Definition

boolean **visible** = true

Description

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

4.4.6.1.15 width (inherited)

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`).

4.4.6.1.16 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

4.4.6.2 Methods

4.4.6.2.1 findCenter (inherited)

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.

4.4.6.2.2 findCenterOfCurve (inherited)

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.

4.4.6.2.3 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.

4.4.6.2.4 getPoints (inherited)

Definition

array **getPoints**()

Description

This method returns an array of all EJSC.PiePoint objects in the series.

4.4.6.2.5 getTotalValue (inherited)

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.

4.4.6.2.6 getVisibility (inherited)

Definition

boolean **getVisibility**()

Description

Returns a boolean indicating the series current visible state

4.4.6.2.7 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.

4.4.6.2.8 hideLegend (inherited)

Definition

void **hideLegend**()

Description

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

4.4.6.2.9 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.

4.4.6.2.10 resetTotalValue (inherited)

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.

4.4.6.2.11 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.

4.4.6.2.12 setDefaultColors (inherited)

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);
```

4.4.6.2.13 setLineOpacity (inherited)

Definition

void **setLineOpacity**(integer opacity)

Description

Sets the EJSCT.Series.lineOpacity property and redraws the series to reflect the change.

4.4.6.2.14 setLineWidth (inherited)

Definition

void **setLineWidth**(integer width)

Description

Updates the lineWidth property and redraws the chart.

4.4.6.2.15 setOpacity (inherited)

Definition

void **setOpacity**(integer opacity)

Description

Sets the EJSCT.Series.opacity property and redraws the series to reflect the change.

4.4.6.2.16 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.

4.4.6.2.17 setTotalValue (inherited)

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.

4.4.6.2.18 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.

4.4.6.2.19 showLegend (inherited)

Definition

void **showLegend**()

Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

4.4.6.3 Events

4.4.6.3.1 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.

4.4.6.3.2 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.

4.4.6.3.3 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.

4.4.6.3.4 onPieceNeedsColor (inherited)

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)"

4.4.6.3.5 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.

4.4.6.4 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):

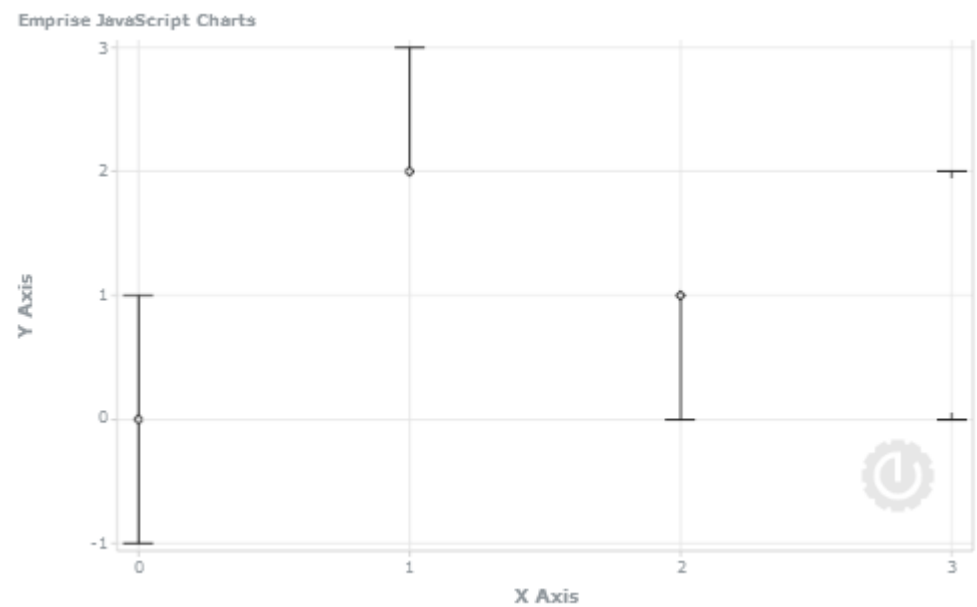
Unused parameters may be omitted.

```
[
    { "x": "number or
string", "label": "string", "userdata": "string" },
    { "x": "number or
string", "label": "string", "userdata": "string" }
]
```

4.4.6.5 Text Replacement Options

String	With
[chart_title]	title of the chart
[series_title]	title of the series the selected
[x]	point resides in
	preformatted value of the
[total]	point
	total value represented by the
	sum of all piece values
[percent]	percent of the total value the
	current point represents
[label]	label property of the current
	point

4.4.7 EJSC.ErrorSeries



The ErrorSeries is rendered as a set of markers which may include an average (or expected) value, a high error, and a low error.

The average is denoted by a diamond while the errors are denoted by a "T". Connecting line(s) will be drawn between the average (if defined) and the error markers.

The constructor expects a value and an optional set of object properties.

Constructor

```
EJSC.ErrorSeries( EJSC.DataHandler dataHandler, {object options} )
```

Example

```
var ErrorSeries = new EJSC.ErrorSeries( new EJSC.ArrayDataHandler( [[0,0],[1,2],[2,1]] ) , {  
    title: 'Error Series'  
} );
```

Defining Properties and Events

ErrorSeries 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.ErrorSeries(new EJSC.ArrayDataHandler( [[0,0],[1,2],[2,1]] ) );  
mySeries.title = "Error Series";  
mySeries.capSize = 5;  
mySeries.capDist = 6;  
mySeries.avgSize = 1;  
mySeries.orientation = 'horizontal';
```

Setting properties in batch

```
var mySeries = new EJSC.ErrorSeries(new EJSC.ArrayDataHandler( [[0,0],[1,2],[2,1]] ), {  
    title: "Error Series",  
    capSize: 5,  
    capDist: 6,  
    avgSize: 1,  
    orientation: 'horizontal'  
} );
```

4.4.7.1 Properties

4.4.7.1.1 avgSize

Definition

integer **avgSize** = 3

Description

Defines the radius of the average marker

4.4.7.1.2 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. EJS.C.PieSeries, EJS.C.AnalogGaugeSeries)

4.4.7.1.3 capSize

Definition

integer **capSize** = 10

Description

Defines the length of the cap arms

4.4.7.1.4 capDist

Definition

integer **capDist** = 5

Description

Defines the length of the cap connector if average is NULL

4.4.7.1.5 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. EJS.C.PieSeries)

4.4.7.1.6 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 EJS.C.Series.setColoredLegend()

4.4.7.1.7 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.

4.4.7.1.8 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]"  
} );
```

4.4.7.1.9 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.

4.4.7.1.10 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()

4.4.7.1.11 lineWidth (inherited)

Definition

integer **lineWidth** = 1

Description

Defines the width of the line connecting series points.

4.4.7.1.12 orientation

Definition

string **orientation** = 'vertical'

Description

Defines the orientation of the bars

Options:

- vertical
- horizontal

4.4.7.1.13 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()

4.4.7.1.14 padding (inherited)

Definition

```
object padding = {  
  x_axis_min: undefined,  
  x_axis_max: undefined,  
  y_axis_min: undefined,  
  y_axis_max: undefined  
}
```

Description

The padding property may be used to override the series default padding. Padding is the amount of

pixels added to the min and max series values in order to push the extremes and prevent the series from hitting the edge of the chart.

The amount of default padding is dependant upon the series type and axis configuration and may not be applicable to all series types.

This property is only applicable during series creation. To retrieve or modify the series padding after creation please see `Series.getPadding()` and `Series.setPadding()`

Example

>> Remove all padding from the BarSeries

```
var barSeries = new EJSC.BarSeries(new EJSC.ArrayDataHandler([..data..]), {
    padding: {
        x_axis_min: 0,
        x_axis_max: 0,
        y_axis_min: 0,
        y_axis_max: 0
    }
});
```

4.4.7.1.15 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.

4.4.7.1.16 visible (inherited)

Definition

boolean **visible** = true

Description

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

4.4.7.1.17 x_axis (inherited)

Definition

string **x_axis** = "bottom"

Description

Defines the horizontal axis to use for X values. Valid options are "top" or "bottom".

4.4.7.1.18 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

4.4.7.1.19 y_axis (inherited)

Definition

string **y_axis** = "left"

Description

Defines the vertical axis to use for Y values. Valid options are "left" or "right".

4.4.7.1.20 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

4.4.7.2 Methods

4.4.7.2.1 findClosestByPixel (inherited)

Definition

EJSC.Point **findClosestByPixel**(object coordinates)

```
point = {  
  x: screen coordinate,  
  y: screen coordinate  
}
```

Description

Returns the point object closest to the screen pixel coordinates provided. The x and y properties of the point object should be in the same scale as the x and y axes assigned to the series. To locate points based on pixel coordinates, see `Series.findClosestByPixel()`

4.4.7.2.2 `findClosestByPoint` (inherited)

Definition

EJSC.Point **`findClosestByPoint`**(object coordinate)

```
point = {  
    x: axis coordinate,  
    y: axis coordinate  
}
```

Description

Returns the point object closest to the axis coordinates provided. The x and y properties of the coordinate object should be based on the top left of the viewport and take into account the page scroll. To locate points based on axis coordinates, see `Series.findClosestByPoint()`

4.4.7.2.3 `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.

4.4.7.2.4 `getPadding` (inherited)

Definition

object **`getPadding`**()

RETURNS:

```
{  
    x_axis_min: number,  
    x_axis_max: number,  
    y_axis_min: number,  
    y_axis_max: number  
}
```

Description

Returns an object containing the series current padding. If padding has been set either by the `Series.setPadding()` method of the `Series.padding` property during construction the result of this method will be adjusted to return the most current padding values.

4.4.7.2.5 getVisibility (inherited)

Definition

boolean **getVisibility**()

Description

Returns a boolean indicating the series current visible state

4.4.7.2.6 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.

4.4.7.2.7 hideLegend (inherited)

Definition

void **hideLegend**()

Description

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

4.4.7.2.8 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.

4.4.7.2.9 setColor (inherited)

Definition

void **setColor**(string color)

Description

Changes the series color and causes the chart to redraw.

4.4.7.2.10 setColoredLegend (inherited)

Definition

void **setColoredLegend**(boolean coloredLegend)

Description

Sets the EJSC.Series.coloredLegend property and updates the legend to reflect the change.

4.4.7.2.11 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.

4.4.7.2.12 setLineOpacity (inherited)

Definition

void **setLineOpacity**(integer opacity)

Description

Sets the EJSC.Series.lineOpacity property and redraws the series to reflect the change.

4.4.7.2.13 setLineWidth (inherited)

Definition

void **setLineWidth**(integer width)

Description

Updates the lineWidth property and redraws the chart.

4.4.7.2.14 setOpacity (inherited)

Definition

void **setOpacity**(integer opacity)

Description

Sets the EJSC.Series.opacity property and redraws the series to reflect the change.

4.4.7.2.15 setPadding (inherited)

Definition

void **setPadding**(object Padding, boolean Redraw)

The padding object should be defined as when used in the series constructor. See Series.padding.

Description

This method updates the user-defined series padding and optionally recalculates the chart extremes and redraws the chart.

4.4.7.2.16 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.

4.4.7.2.17 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.

4.4.7.2.18 showLegend (inherited)

Definition

void **showLegend**()

Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

4.4.7.3 Events

4.4.7.3.1 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.

4.4.7.3.2 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.

4.4.7.3.3 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.

4.4.7.3.4 onShowHint (inherited)

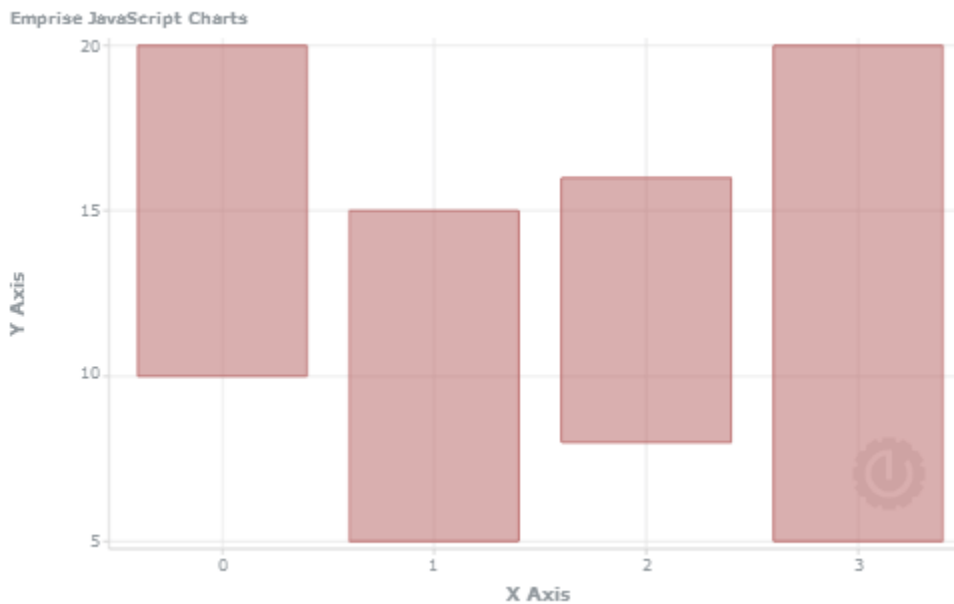
Definition

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.

4.4.8 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" }
);
```

4.4.8.1 Properties

4.4.8.1.1 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)

4.4.8.1.2 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)

4.4.8.1.3 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()

4.4.8.1.4 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.

4.4.8.1.5 `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.

4.4.8.1.6 `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));
```

4.4.8.1.7 `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]"  
} );
```

4.4.8.1.8 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  
    }  
));
```

4.4.8.1.9 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.

4.4.8.1.10 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()

4.4.8.1.11 lineWidth (inherited)

Definition

integer **lineWidth** = 1

Description

Defines the width of the line connecting series points.

4.4.8.1.12 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()`

4.4.8.1.13 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.

4.4.8.1.14 padding (inherited)

Definition

```
object padding = {  
  x_axis_min: undefined,  
  x_axis_max: undefined,  
  y_axis_min: undefined,  
  y_axis_max: undefined  
}
```

Description

The padding property may be used to override the series default padding. Padding is the amount of pixels added to the min and max series values in order to push the extremes and prevent the series from hitting the edge of the chart.

The amount of default padding is dependant upon the series type and axis configuration and may not be applicable to all series types.

This property is only applicable during series creation. To retrieve or modify the series padding after creation please see `Series.getPadding()` and `Series.setPadding()`

Example

>> Remove all padding from the BarSeries

```
var barSeries = new EJSC.BarSeries(new EJSC.ArrayDataHandler([..data..]), {
  padding: {
    x_axis_min: 0,
    x_axis_max: 0,
    y_axis_min: 0,
    y_axis_max: 0
  }
});
```

4.4.8.1.15 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, // <
ex: "rgb(255,0,0)"	color:	string, // Defined as rgb(RED, GREEN, BLUE),
	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 }
    ]
  }
));
```

4.4.8.1.16 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.

4.4.8.1.17 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.

4.4.8.1.18 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
    }
));
```

4.4.8.1.19 visible (inherited)

Definition

boolean **visible** = true

Description

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

4.4.8.1.20 x_axis (inherited)

Definition

string **x_axis** = "bottom"

Description

Defines the horizontal axis to use for X values. Valid options are "top" or "bottom".

4.4.8.1.21 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

4.4.8.1.22 y_axis (inherited)

Definition

string **y_axis** = "left"

Description

Defines the vertical axis to use for Y values. Valid options are "left" or "right".

4.4.8.1.23 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

4.4.8.2 Methods

4.4.8.2.1 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.

4.4.8.2.2 clearRanges (inherited)

Definition

void **clearRanges**(boolean redraw)

Description

Clears all ranges defined for the series and optionally redraws the the chart.

4.4.8.2.3 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.

4.4.8.2.4 findClosestByPixel (inherited)

Definition

EJSC.Point **findClosestByPixel**(object coordinates)

```
point = {  
  x: screen coordinate,  
  y: screen coordinate  
}
```

Description

Returns the point object closest to the screen pixel coordinates provided. The x and y properties of the point object should be in the same scale as the x and y axes assigned to the series. To locate points based on pixel coordinates, see `Series.findClosestByPixel()`

4.4.8.2.5 findClosestByPoint (inherited)

Definition

EJSC.Point **findClosestByPoint**(object coordinate)

```
point = {  
    x: axis coordinate,  
    y: axis coordinate  
}
```

Description

Returns the point object closest to the axis coordinates provided. The x and y properties of the coordinate object should be based on the top left of the viewport and take into account the page scroll. To locate points based on axis coordinates, see `Series.findClosestByPoint()`

4.4.8.2.6 getBarSize (inherited)

Definition

integer **getBarSize**()

Description

This method returns the size in pixels (width or height depending on orientation) of a single bar.

4.4.8.2.7 getBarSizeInPoints (inherited)

Definition

integer **getBarSizeInPoints**()

Description

This method returns the size in chart points (width or height depending on orientation) of a single bar.

4.4.8.2.8 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.

4.4.8.2.9 getPadding (inherited)

Definition

object **getPadding**()

RETURNS:

```

{
    x_axis_min: number,
    x_axis_max: number,
    y_axis_min: number,
    y_axis_max: number
}

```

Description

Returns an object containing the series current padding. If padding has been set either by the `Series.setPadding()` method of the `Series.padding` property during construction the result of this method will be adjusted to return the most current padding values.

4.4.8.2.10 `getPoints` (inherited)**Definition**

array `getPoints()`

Description

This method returns an array of all `EJSC.BarPoint` objects in the series.

4.4.8.2.11 `getVisibility` (inherited)4.4.8.2.12 `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.

4.4.8.2.13 `hideLegend` (inherited)**Definition**

void `hideLegend()`

Description

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

4.4.8.2.14 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.

4.4.8.2.15 setColor (inherited)

Definition

void **setColor**(string color)

Description

Changes the series color and causes the chart to redraw.

4.4.8.2.16 setColoredLegend (inherited)

Definition

void **setColoredLegend**(boolean coloredLegend)

Description

Sets the EJSC.Series.coloredLegend property and updates the legend to reflect the change.

4.4.8.2.17 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.

4.4.8.2.18 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);
```

4.4.8.2.19 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.

4.4.8.2.20 setLineWidth (inherited)

Definition

void [setLineWidth](#)(integer width)

Description

Updates the lineWidth property and redraws the chart.

4.4.8.2.21 setOpacity (inherited)

Definition

void [setOpacity](#)(integer opacity)

Description

Sets the `EJSC.Series.opacity` property and redraws the series to reflect the change.

4.4.8.2.22 setPadding (inherited)

Definition

void [setPadding](#)(object Padding, boolean Redraw)

The padding object should be defined as when used in the series constructor. See `Series.padding`.

Description

This method updates the user-defined series padding and optionally recalculates the chart extremes and redraws the chart.

4.4.8.2.23 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.

4.4.8.2.24 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.

4.4.8.2.25 showLegend (inherited)

Definition

void **showLegend**()

Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

4.4.8.3 Events

4.4.8.3.1 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.

4.4.8.3.2 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.

4.4.8.3.3 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)";
        }
    }
} ));
```

4.4.8.3.4 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.

4.4.8.3.5 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.

4.4.8.4 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>
max="number" label="string" userdata="string"/>
max="number" label="string" userdata="string"/>
            </L>
        </G>

Orientation == "horizontal"

```

```

        <G>
            <L>
max="number" label="string" userdata="string"/>
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" }
]
```

4.4.8.5 Text Replacement Options**String**

[chart_title]
[series_title]

[xaxis]
[yaxis]
[x]

[y]

[label]

[min]

[max]

With

title of the chart

title of the series the selected

point resides in

caption of the series x axis

caption of the series y axis

preformatted X value of the

point

preformatted Y value of the

point

label property of the current

point

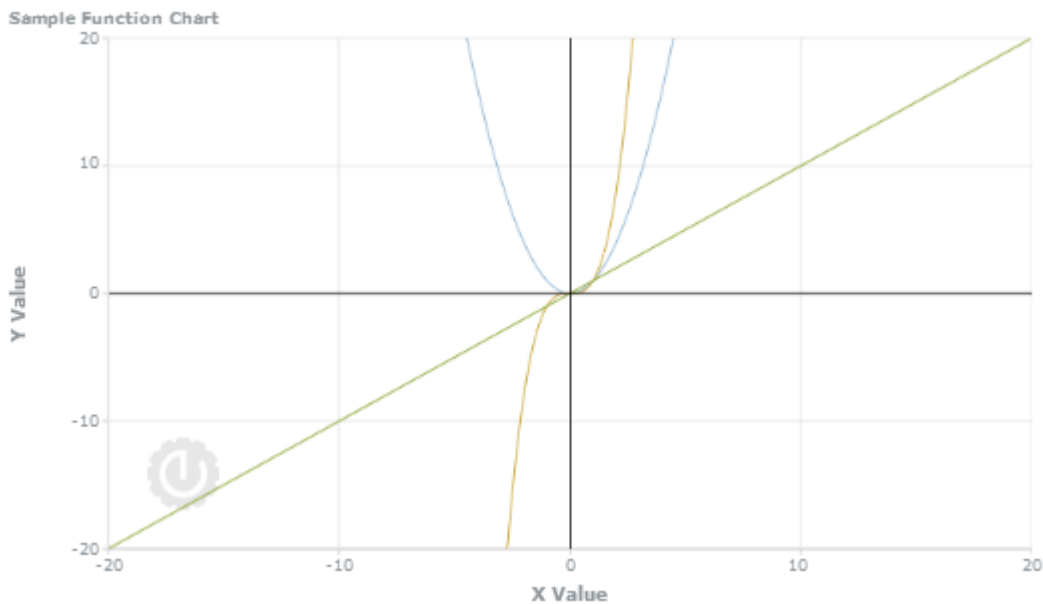
preformatted min value of the

current point

preformatted max value of the

currnt point

4.4.9 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" }  
);
```

4.4.9.1 Properties

4.4.9.1.1 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)

4.4.9.1.2 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()

4.4.9.1.3 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]"  
} );
```

4.4.9.1.4 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.

4.4.9.1.5 `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()`

4.4.9.1.6 `lineWidth` (inherited)

Definition

integer `lineWidth` = 1

Description

Defines the width of the line connecting series points.

4.4.9.1.7 `padding` (inherited)

Definition

```
object padding = {  
  x_axis_min: undefined,  
  x_axis_max: undefined,  
  y_axis_min: undefined,  
  y_axis_max: undefined  
}
```

Description

The padding property may be used to override the series default padding. Padding is the amount of pixels added to the min and max series values in order to push the extremes and prevent the series from hitting the edge of the chart.

The amount of default padding is dependant upon the series type and axis configuration and may not be applicable to all series types.

This property is only applicable during series creation. To retrieve or modify the series padding after creation please see `Series.getPadding()` and `Series.setPadding()`

Example

>> Remove all padding from the BarSeries

```
var barSeries = new EJSC.BarSeries(new EJSC.ArrayDataHandler([..data..]), {  
  padding: {
```

```

        x_axis_min: 0,
        x_axis_max: 0,
        y_axis_min: 0,
        y_axis_max: 0
    }
    } );

```

4.4.9.1.8 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.

4.4.9.1.9 visible (inherited)

Definition

boolean **visible** = true

Description

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

4.4.9.1.10 x_axis (inherited)

Definition

string **x_axis** = "bottom"

Description

Defines the horizontal axis to use for X values. Valid options are "top" or "bottom".

4.4.9.1.11 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

4.4.9.1.12 `y_axis` (inherited)**Definition**

string `y_axis` = "left"

Description

Defines the vertical axis to use for Y values. Valid options are "left" or "right".

4.4.9.1.13 `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

4.4.9.2 Methods4.4.9.2.1 `findClosestByPixel` (inherited)**Definition**

EJSC.Point `findClosestByPixel`(object coordinates)

```
point = {  
  x: screen coordinate,  
  y: screen coordinate  
}
```

Description

Returns the point object closest to the screen pixel coordinates provided. The x and y properties of the point object should be in the same scale as the x and y axes assigned to the series. To locate points based on pixel coordinates, see `Series.findClosestByPixel()`

4.4.9.2.2 `findClosestByPoint` (inherited)**Definition**

EJSC.Point `findClosestByPoint`(object coordinate)

```
point = {  
  x: axis coordinate,  
  y: axis coordinate  
}
```

Description

Returns the point object closest to the axis coordinates provided. The x and y properties of the coordinate object should be based on the top left of the viewport and take into account the page scroll. To locate points based on axis coordinates, see `Series.findClosestByPoint()`

4.4.9.2.3 `getPadding` (inherited)**Definition**

object `getPadding`()

RETURNS:

```
{
    x_axis_min: number,
    x_axis_max: number,
    y_axis_min: number,
    y_axis_max: number
}
```

Description

Returns an object containing the series current padding. If padding has been set either by the `Series.setPadding()` method of the `Series.padding` property during construction the result of this method will be adjusted to return the most current padding values.

4.4.9.2.4 `getVisibility` (inherited)**Definition**

boolean `getVisibility`()

Description

Returns a boolean indicating the series current visible state

4.4.9.2.5 `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.

4.4.9.2.6 `hideLegend` (inherited)**Definition**

void **hideLegend**()

Description

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

4.4.9.2.7 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.

4.4.9.2.8 setColor (inherited)

Definition

void **setColor**(string color)

Description

Changes the series color and causes the chart to redraw.

4.4.9.2.9 setColoredLegend (inherited)

Definition

void **setColoredLegend**(boolean coloredLegend)

Description

Sets the EJSC.Series.coloredLegend property and updates the legend to reflect the change.

4.4.9.2.10 setLineWidth (inherited)

Definition

void **setLineWidth**(integer width)

Description

Updates the lineWidth property and redraws the chart.

4.4.9.2.11 setPadding (inherited)

Definition

void **setPadding**(object Padding, boolean Redraw)

The padding object should be defined as when used in the series constructor. See Series.padding.

Description

This method updates the user-defined series padding and optionally recalculates the chart extremes and redraws the chart.

4.4.9.2.12 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.

4.4.9.2.13 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.

4.4.9.2.14 showLegend (inherited)

Definition

void **showLegend**()

Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

4.4.9.3 Events

4.4.9.3.1 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.

4.4.9.3.2 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.

4.4.9.3.3 onShowHint (inherited)

Definition

string onShowHint(EJSC.Point point, EJSC.Series series, EJSC.Chart chart, string hint_string)

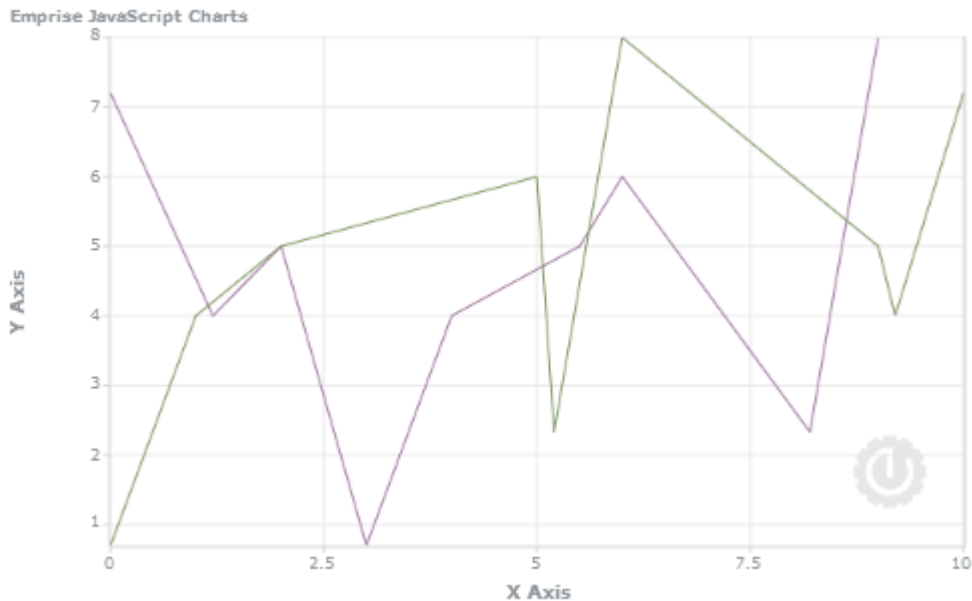
Description

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.

4.4.9.4 Text Replacement Options

String	With
[chart_title]	title of the chart
[series_title]	title of the series the selected
[xaxis]	point resides in
[yaxis]	caption of the series x axis
[x]	caption of the series y axis
	preformatted X value of the
[y]	point
	preformatted Y value of the
[label]	point
	label property of the current
	point

4.4.10 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" }
);
```

Gaps in line series

Gaps are supported in the line series and may be enabled by providing a blank string for a given X and/or Y value:

```
var mySeries = new EJSC.LineSeries(  
    new  
    EJSC.ArrayDataHandler([[1,1],[2,2],[3,""],[4,3],[5,2],[6,""],[7,2.5],[8,1.75]]),  
    { lineWidth: 2, title: "New Line Series" }  
);
```

4.4.10.1 Properties

4.4.10.1.1 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)

4.4.10.1.2 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)

4.4.10.1.3 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()

4.4.10.1.4 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.

4.4.10.1.5 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.

4.4.10.1.6 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]"  
} );
```

4.4.10.1.7 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.

4.4.10.1.8 `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()`

4.4.10.1.9 `lineWidth` (inherited)

Definition

integer `lineWidth` = 1

Description

Defines the width of the line connecting series points.

4.4.10.1.10 `padding` (inherited)

Definition

```
object padding = {  
    x_axis_min: undefined,  
    x_axis_max: undefined,  
    y_axis_min: undefined,  
    y_axis_max: undefined  
}
```

Description

The padding property may be used to override the series default padding. Padding is the amount of pixels added to the min and max series values in order to push the extremes and prevent the series from hitting the edge of the chart.

The amount of default padding is dependant upon the series type and axis configuration and may not be applicable to all series types.

This property is only applicable during series creation. To retrieve or modify the series padding after creation please see `Series.getPadding()` and `Series.setPadding()`

Example

>> Remove all padding from the BarSeries

```
var barSeries = new EJSC.BarSeries(new EJSC.ArrayDataHandler([..data..]), {
```

```
padding: {  
    x_axis_min: 0,  
    x_axis_max: 0,  
    y_axis_min: 0,  
    y_axis_max: 0  
}  
} );
```

4.4.10.1.11 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.

4.4.10.1.12 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.

4.4.10.1.13 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.

4.4.10.1.14 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`.

4.4.10.1.15 `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.

4.4.10.1.16 `visible` (inherited)

Definition

boolean `visible` = `true`

Description

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

4.4.10.1.17 `x_axis` (inherited)

Definition

string `x_axis` = "bottom"

Description

Defines the horizontal axis to use for X values. Valid options are "top" or "bottom".

4.4.10.1.18 `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`

4.4.10.1.19 `y_axis` (inherited)

Definition

```
string y_axis = "left"
```

Description

Defines the vertical axis to use for Y values. Valid options are "left" or "right".

4.4.10.1.20 `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

4.4.10.2 Methods

4.4.10.2.1 `findClosestByPixel` (inherited)

Definition

```
EJSC.Point findClosestByPixel( object coordinates )
```

```
point = {  
  x: screen coordinate,  
  y: screen coordinate  
}
```

Description

Returns the point object closest to the screen pixel coordinates provided. The x and y properties of the point object should be in the same scale as the x and y axes assigned to the series. To locate points based on pixel coordinates, see `Series.findClosestByPixel()`

4.4.10.2.2 `findClosestByPoint` (inherited)

Definition

```
EJSC.Point findClosestByPoint( object coordinate )
```

```
point = {  
  x: axis coordinate,  
  y: axis coordinate  
}
```

Description

Returns the point object closest to the axis coordinates provided. The x and y properties of the coordinate object should be based on the top left of the viewport and take into account the page scroll. To locate points based on axis coordinates, see `Series.findClosestByPoint()`

4.4.10.2.3 `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.

4.4.10.2.4 `getPadding` (inherited)

Definition

object `getPadding()`

RETURNS:

```
{
  x_axis_min: number,
  x_axis_max: number,
  y_axis_min: number,
  y_axis_max: number
}
```

Description

Returns an object containing the series current padding. If padding has been set either by the `Series.setPadding()` method of the `Series.padding` property during construction the result of this method will be adjusted to return the most current padding values.

4.4.10.2.5 `getVisibility` (inherited)

Definition

boolean `getVisibility()`

Description

Returns a boolean indicating the series current visible state

4.4.10.2.6 `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.

4.4.10.2.7 hideLegend (inherited)

Definition

void **hideLegend**()

Description

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

4.4.10.2.8 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.

4.4.10.2.9 setColor (inherited)

Definition

void **setColor**(string color)

Description

Changes the series color and causes the chart to redraw.

4.4.10.2.10 setColoredLegend (inherited)

Definition

void **setColoredLegend**(boolean coloredLegend)

Description

Sets the EJSC.Series.coloredLegend property and updates the legend to reflect the change.

4.4.10.2.11 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.

4.4.10.2.12 `setLineOpacity` (inherited)

Definition

void `setLineOpacity`(integer opacity)

Description

Sets the `EJSC.Series.lineOpacity` property and redraws the series to reflect the change.

4.4.10.2.13 `setLineWidth` (inherited)

Definition

void `setLineWidth`(integer width)

Description

Updates the `lineWidth` property and redraws the chart.

4.4.10.2.14 `setPadding` (inherited)

Definition

void `setPadding`(object Padding, boolean Redraw)

The padding object should be defined as when used in the series constructor. See `Series.padding`.

Description

This method updates the user-defined series padding and optionally recalculates the chart extremes and redraws the chart.

4.4.10.2.15 `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.

4.4.10.2.16 `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.

4.4.10.2.17 showLegend (inherited)

Definition

void **showLegend**()

Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

4.4.10.3 Events

4.4.10.3.1 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.

4.4.10.3.2 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.

4.4.10.3.3 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.

4.4.10.3.4 onShowHint (inherited)

Definition

string **onShowHint**(EJSCLPoint point, EJSCLSeries series, EJSCLChart 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.

4.4.10.4 Text Replacement Options

String	With
[chart_title]	title of the chart
[series_title]	title of the series the selected
	point resides in
[xaxis]	caption of the series x axis
[yaxis]	caption of the series y axis
[x]	preformatted X value of the
	point
[y]	preformatted Y value of the
	point
[label]	label property of the current
	point

4.4.10.5 Data Formats

The x and y parameters are required, label and userdata are both optional.

XML (EJSCLXMLDataHandler, EJSCLXMLStringDataHandler):

Unused parameters may be omitted

Full:

<graph>

<plot>

or string" label="string" userdata="string"/>

or string" label="string" userdata="string"/>

</plot>

</graph>

<point x="number or string" y="number

<point x="number or string" y="number

Short:

<G>

<L>

```

string" label="string" userdata="string"/>
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"}
]

```

4.4.11 EJSC.OpenHighLowCloseSeries



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" }  
    );
```

4.4.11.1 Properties

4.4.11.1.1 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)

4.4.11.1.2 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)

4.4.11.1.3 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()

4.4.11.1.4 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.

4.4.11.1.5 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  
        }  
    }  
);
```

4.4.11.1.6 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]"  
} );
```

4.4.11.1.7 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
    }
));
```

4.4.11.1.8 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.

4.4.11.1.9 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()

4.4.11.1.10 lineWidth (inherited)

Definition

integer **lineWidth** = 1

Description

Defines the width of the line connecting series points.

4.4.11.1.11 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
        }
    }
);
```

4.4.11.1.12 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()`

4.4.11.1.13 padding (inherited)

Definition

```
object padding = {
    x_axis_min: undefined,
    x_axis_max: undefined,
    y_axis_min: undefined,
    y_axis_max: undefined
}
```

Description

The padding property may be used to override the series default padding. Padding is the amount of pixels added to the min and max series values in order to push the extremes and prevent the series from hitting the edge of the chart.

The amount of default padding is dependant upon the series type and axis configuration and may not be applicable to all series types.

This property is only applicable during series creation. To retrieve or modify the series padding after creation please see `Series.getPadding()` and `Series.setPadding()`

Example

>> Remove all padding from the BarSeries

```
var barSeries = new EJSC.BarSeries(new EJSC.ArrayDataHandler([...data...]), {  
    padding: {  
        x_axis_min: 0,  
        x_axis_max: 0,  
        y_axis_min: 0,  
        y_axis_max: 0  
    }  
});
```

4.4.11.1.14 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.

4.4.11.1.15 visible (inherited)

Definition

boolean **visible** = true

Description

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

4.4.11.1.16 x_axis (inherited)

Definition

string **x_axis** = "bottom"

Description

Defines the horizontal axis to use for X values. Valid options are "top" or "bottom".

4.4.11.1.17 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

4.4.11.1.18 y_axis (inherited)

Definition

string **y_axis** = "left"

Description

Defines the vertical axis to use for Y values. Valid options are "left" or "right".

4.4.11.1.19 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

4.4.11.2 Methods

4.4.11.2.1 findClosestByPixel (inherited)

Definition

EJSC.Point **findClosestByPixel**(object coordinates)

```
point = {  
  x: screen coordinate,  
  y: screen coordinate  
}
```

Description

Returns the point object closest to the screen pixel coordinates provided. The x and y properties of the point object should be in the same scale as the x and y axes assigned to the series. To locate points based on pixel coordinates, see `Series.findClosestByPixel()`

4.4.11.2.2 findClosestByPoint (inherited)

Definition

EJSC.Point **findClosestByPoint**(object coordinate)

```
point = {  
  x: axis coordinate,  
  y: axis coordinate  
}
```

Description

Returns the point object closest to the axis coordinates provided. The x and y properties of the coordinate object should be based on the top left of the viewport and take into account the page scroll. To locate points based on axis coordinates, see `Series.findClosestByPoint()`

4.4.11.2.3 `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.

4.4.11.2.4 `getPadding` (inherited)**Definition**

object `getPadding()`

RETURNS:

```
{  
  x_axis_min: number,  
  x_axis_max: number,  
  y_axis_min: number,  
  y_axis_max: number  
}
```

Description

Returns an object containing the series current padding. If padding has been set either by the `Series.setPadding()` method of the `Series.padding` property during construction the result of this method will be adjusted to return the most current padding values.

4.4.11.2.5 `getVisibility` (inherited)**Definition**

boolean `getVisibility()`

Description

Returns a boolean indicating the series current visible state

4.4.11.2.6 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.

4.4.11.2.7 hideLegend (inherited)

Definition

```
void hideLegend( )
```

Description

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

4.4.11.2.8 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.

4.4.11.2.9 setColor (inherited)

Definition

```
void setColor( string color )
```

Description

Changes the series color and causes the chart to redraw.

4.4.11.2.10 setColoredLegend (inherited)

Definition

```
void setColoredLegend( boolean coloredLegend )
```

Description

Sets the `EJSC.Series.coloredLegend` property and updates the legend to reflect the change.

4.4.11.2.11 `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.

4.4.11.2.12 `setLineOpacity` (inherited)

Definition

void **setLineOpacity**(integer opacity)

Description

Sets the `EJSC.Series.lineOpacity` property and redraws the series to reflect the change.

4.4.11.2.13 `setLineWidth` (inherited)

Definition

void **setLineWidth**(integer width)

Description

Updates the `lineWidth` property and redraws the chart.

4.4.11.2.14 `setOpacity` (inherited)

Definition

void **setOpacity**(integer opacity)

Description

Sets the `EJSC.Series.opacity` property and redraws the series to reflect the change.

4.4.11.2.15 `setPadding` (inherited)

Definition

void **setPadding**(object Padding, boolean Redraw)

The padding object should be defined as when used in the series constructor. See `Series.padding`.

Description

This method updates the user-defined series padding and optionally recalculates the chart extremes and redraws the chart.

4.4.11.2.16 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.

4.4.11.2.17 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.

4.4.11.2.18 showLegend (inherited)

Definition

void **showLegend**()

Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

4.4.11.3 Events

4.4.11.3.1 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.

4.4.11.3.2 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.

4.4.11.3.3 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.

4.4.11.3.4 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.

4.4.11.4 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):

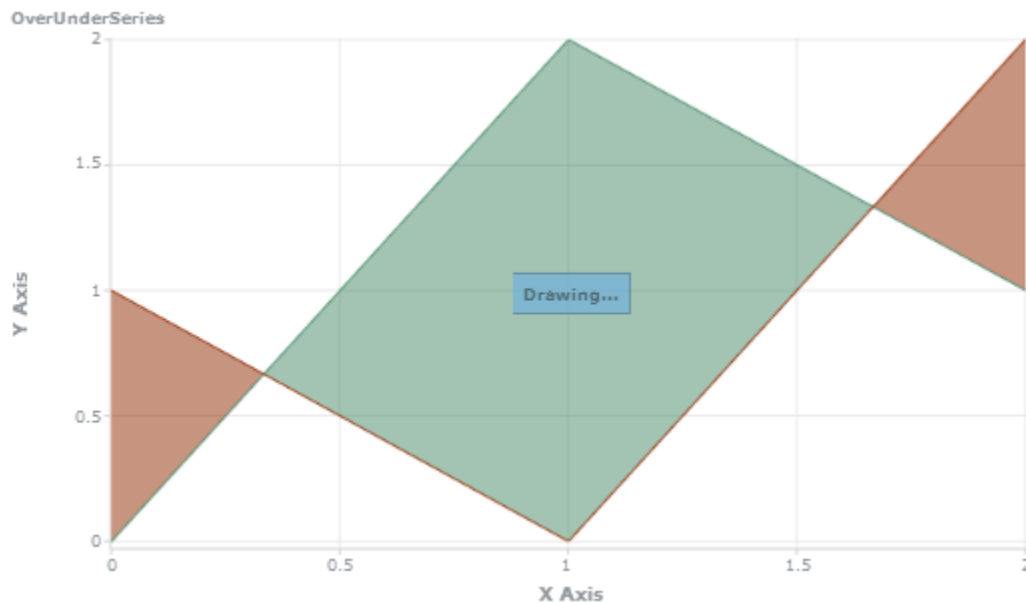
Unused parameters may be omitted.

```
[
    {
        "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"
    }
]
```

4.4.11.5 Text Replacement Options

String	With
[chart_title]	title of the chart
[series_title]	title of the series the selected point resides in
[xaxis]	caption of the series x axis
[yaxis]	caption of the series y axis
[x]	preformatted X value of the point
[high]	preformatted high value of the point
[low]	preformatted low value of the point
[open]	preformatted open value of the point
[close]	preformatted close value of the point
[label]	label property of the current point

4.4.12 EJSC.OverUnderSeries



The OverUnderSeries is drawn as a fill between two LineSeries. Different colors are used to tell which referenced series' Y values are higher at any given X position.

Constructor

```
EJSC.OverUnderSeries( EJSC.Series baseSeries, EJSC.Series compareSeries, {object options} )
```

Example

```
var mySeries = new EJSC.OverUnderSeriesSeries(lineSeries1, lineSeries2 , {
    title: "New OverUnderSeries Series"
} );
```

Defining Properties and Events

OverUnderSeries Series 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.OverUnderSeries(lineSeries1, lineSeries2);
mySeries.exceed = {visible: false};
mySeries.subcede = {visible: false};
mySeries.title = "New OverUnderSeries Series";
```

Setting properties in batch

```
var mySeries = new EJSC.OpenHighLowCloseSeries(lineSeries1, lineSeries2,{
    exceed: {visible: false},
    subcede: {visible: false},
    title: "New OverUnderSeries Series"
} );
```

4.4.12.1 Properties

4.4.12.1.1 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)

4.4.12.1.2 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)

4.4.12.1.3 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()`

4.4.12.1.4 exceed

Definition

```
object exceed = {
  visible: true,
  color: '#00ff00',
  opacity: 20
}
```

Description

Defines the attributes for drawing the exceedences

Properties

boolean	visible	– Defines if the <u>exceedences</u> are visible
color	color	– The color to drag the text on the <u>exceedences</u> in
integer	opacity	– Integer (0-100): The % opacity for the fill of the
<u>exceedences</u>		

4.4.12.1.5 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.

4.4.12.1.6 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]"  
} );
```

4.4.12.1.7 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.

4.4.12.1.8 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()`

4.4.12.1.9 lineWidth (inherited)

Definition

integer **lineWidth** = 1

Description

Defines the width of the line connecting series points.

4.4.12.1.10 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()`

4.4.12.1.11 padding (inherited)

Definition

```
object padding = {
    x_axis_min: undefined,
    x_axis_max: undefined,
    y_axis_min: undefined,
    y_axis_max: undefined
}
```

Description

The padding property may be used to override the series default padding. Padding is the amount of pixels added to the min and max series values in order to push the extremes and prevent the series from hitting the edge of the chart.

The amount of default padding is dependant upon the series type and axis configuration and may not be applicable to all series types.

This property is only applicable during series creation. To retrieve or modify the series padding after creation please see `Series.getPadding()` and `Series.setPadding()`

Example

>> Remove all padding from the BarSeries

```
var barSeries = new EJSC.BarSeries(new EJSC.ArrayDataHandler([..data..]), {
    padding: {
        x_axis_min: 0,
        x_axis_max: 0,
        y_axis_min: 0,
        y_axis_max: 0
    }
});
```

4.4.12.1.12 subsede

Definition

```
object subcede = {
    visible: true,
    color: '#ff0000',
    opacity: 20
}
```

Description

Defines the attributes for drawing the subsedences

Properties

boolean	visible	– Defines if the subsedences are visible
color	color	– The color to drag the text on the <u>subsedences</u> in
integer	opacity	– Integer (0-100): The % opacity for the fill of the
<u>subsedences</u>		

4.4.12.1.13 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.

4.4.12.1.14 visible (inherited)

Definition

boolean **visible** = true

Description

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

4.4.12.1.15 x_axis (inherited)

Definition

string **x_axis** = "bottom"

Description

Defines the horizontal axis to use for X values. Valid options are "top" or "bottom".

4.4.12.1.16 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

4.4.12.1.17 y_axis (inherited)

Definition

string **y_axis** = "left"

Description

Defines the vertical axis to use for Y values. Valid options are "left" or "right".

4.4.12.1.18 `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

4.4.12.2 Methods

4.4.12.2.1 `findClosestByPixel` (inherited)

Definition

EJSC.Point `findClosestByPixel`(object coordinates)

```
point = {  
  x: screen coordinate,  
  y: screen coordinate  
}
```

Description

Returns the point object closest to the screen pixel coordinates provided. The x and y properties of the point object should be in the same scale as the x and y axes assigned to the series. To locate points based on pixel coordinates, see `Series.findClosestByPixel()`

4.4.12.2.2 `findClosestByPoint` (inherited)

Definition

EJSC.Point `findClosestByPoint`(object coordinate)

```
point = {  
  x: axis coordinate,  
  y: axis coordinate  
}
```

Description

Returns the point object closest to the axis coordinates provided. The x and y properties of the coordinate object should be based on the top left of the viewport and take into account the page scroll.

To locate points based on axis coordinates, see `Series.findClosestByPoint()`

4.4.12.2.3 `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.

4.4.12.2.4 `getPadding` (inherited)

Definition

object `getPadding()`

RETURNS:

```
{
    x_axis_min: number,
    x_axis_max: number,
    y_axis_min: number,
    y_axis_max: number
}
```

Description

Returns an object containing the series current padding. If padding has been set either by the `Series.setPadding()` method of the `Series.padding` property during construction the result of this method will be adjusted to return the most current padding values.

4.4.12.2.5 `getVisibility` (inherited)

Definition

boolean `getVisibility()`

Description

Returns a boolean indicating the series current visible state

4.4.12.2.6 `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.

4.4.12.2.7 hideLegend (inherited)

Definition

```
void hideLegend( )
```

Description

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

4.4.12.2.8 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.

4.4.12.2.9 setColor (inherited)

Definition

```
void setColor( string color )
```

Description

Changes the series color and causes the chart to redraw.

4.4.12.2.10 setColoredLegend (inherited)

Definition

```
void setColoredLegend( boolean coloredLegend )
```

Description

Sets the EJSC.Series.coloredLegend property and updates the legend to reflect the change.

4.4.12.2.11 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.

4.4.12.2.12 setLineOpacity (inherited)

Definition

void **setLineOpacity**(integer opacity)

Description

Sets the EJSCT.Series.lineOpacity property and redraws the series to reflect the change.

4.4.12.2.13 setLineWidth (inherited)

Definition

void **setLineWidth**(integer width)

Description

Updates the lineWidth property and redraws the chart.

4.4.12.2.14 setOpacity (inherited)

Definition

void **setOpacity**(integer opacity)

Description

Sets the EJSCT.Series.opacity property and redraws the series to reflect the change.

4.4.12.2.15 setPadding (inherited)

Definition

void **setPadding**(object Padding, boolean Redraw)

The padding object should be defined as when used in the series constructor. See Series.padding.

Description

This method updates the user-defined series padding and optionally recalculates the chart extremes and redraws the chart.

4.4.12.2.16 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.

4.4.12.2.17 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.

4.4.12.2.18 showLegend (inherited)

Definition

void **showLegend**()

Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

4.4.12.3 Events

4.4.12.3.1 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.

4.4.12.3.2 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.

4.4.12.3.3 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.

4.4.12.3.4 onShowHint (inherited)

Definition

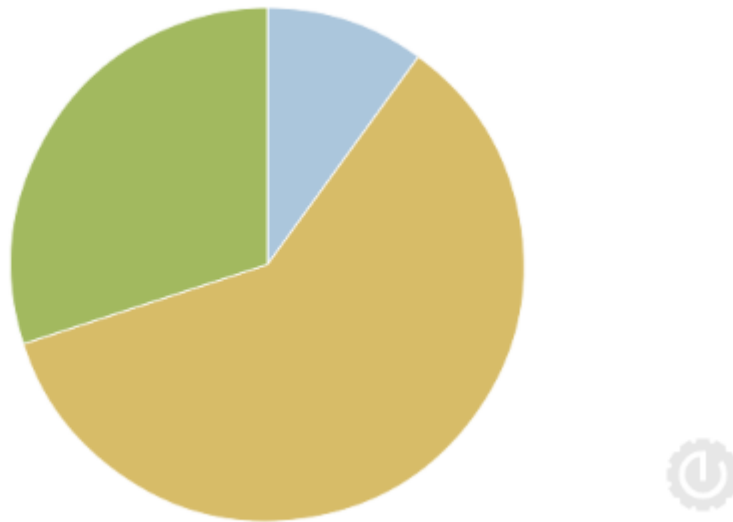
string **onShowHint**(EJSC.Point point, EJSC.Series series, EJSC.Chart chart, string hint_string)

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.

4.4.13 EJSC.PieSeries

Emprise JavaScript Charts



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" }
);
```

4.4.13.1 Properties

4.4.13.1.1 defaultColors

Definition

array **defaultColors** = EJSC.DefaultPieColors

Description

Defines the pool of default colors available for pie pieces.

4.4.13.1.2 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.

4.4.13.1.3 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).

4.4.13.1.4 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]"  
} );
```

4.4.13.1.5 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.

4.4.13.1.6 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()

4.4.13.1.7 lineWidth (inherited)

Definition

integer **lineWidth** = 1

Description

Defines the width of the line connecting series points.

4.4.13.1.8 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()`

4.4.13.1.9 position

Definition

string **position** = "center"

Description

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

Quadrants:

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

4.4.13.1.10 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.

4.4.13.1.11 `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.

4.4.13.1.12 `treeLegend`

Definition

boolean `treeLegend` = true

Description

Defines whether to display each pie piece individually in the legend.

4.4.13.1.13 `visible` (inherited)

Definition

boolean `visible` = true

Description

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

4.4.13.1.14 `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`).

4.4.13.1.15 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

4.4.13.2 Methods

4.4.13.2.1 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.

4.4.13.2.2 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.

4.4.13.2.3 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.

4.4.13.2.4 getPoints

Definition

array **getPoints**()

Description

This method returns an array of all EJSC.PiePoint objects in the series.

4.4.13.2.5 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.

4.4.13.2.6 getVisibility (inherited)

Definition

boolean **getVisibility**()

Description

Returns a boolean indicating the series current visible state

4.4.13.2.7 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.

4.4.13.2.8 hideLegend (inherited)

Definition

```
void hideLegend( )
```

Description

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

4.4.13.2.9 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.

4.4.13.2.10 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.

4.4.13.2.11 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.

4.4.13.2.12 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);
```

4.4.13.2.13 setLineWidth (inherited)

Definition

void **setLineWidth**(integer width)

Description

Updates the lineWidth property and redraws the chart.

4.4.13.2.14 setOpacity (inherited)

Definition

void **setOpacity**(integer opacity)

Description

Sets the EJSCT.Series.opacity property and redraws the series to reflect the change.

4.4.13.2.15 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.

4.4.13.2.16 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`.

4.4.13.2.17 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.

4.4.13.2.18 showLegend (inherited)

Definition

void **showLegend**()

Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

4.4.13.3 Events

4.4.13.3.1 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.

4.4.13.3.2 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.

4.4.13.3.3 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.

4.4.13.3.4 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)"

4.4.13.3.5 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.

4.4.13.4 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):

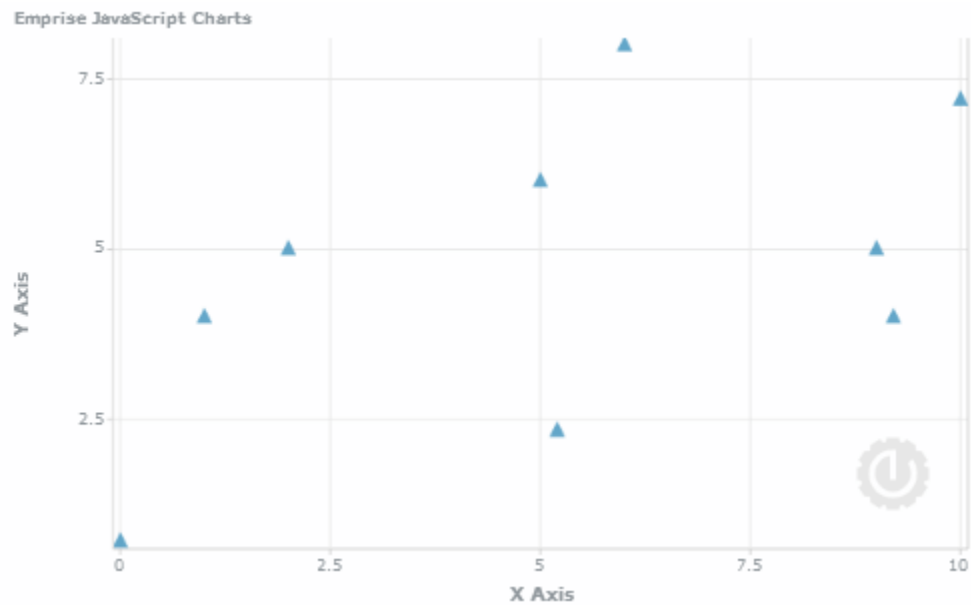
Unused parameters may be omitted.

```
[
    { "x": "number or
string", "label": "string", "userdata": "string" },
    { "x": "number or
string", "label": "string", "userdata": "string" }
]
```

4.4.13.5 Text Replacement Options

String	With
[chart_title]	title of the chart
[series_title]	title of the series the selected
[x]	point resides in
	preformatted value of the
[total]	point
	total value represented by the
[percent]	sum of all piece values
	percent of the total value the
[label]	current point represents
	label property of the current
	point

4.4.14 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" }  
);
```

4.4.14.1 Properties

4.4.14.1.1 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)

4.4.14.1.2 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)

4.4.14.1.3 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()`

4.4.14.1.4 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.

4.4.14.1.5 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]"  
} );
```

4.4.14.1.6 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.

4.4.14.1.7 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()`

4.4.14.1.8 lineWidth (inherited)

Definition

integer **lineWidth** = 1

Description

Defines the width of the line connecting series points.

4.4.14.1.9 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()`

4.4.14.1.10 padding (inherited)

Definition

```
object padding = {  
  x_axis_min: undefined,  
  x_axis_max: undefined,  
  y_axis_min: undefined,  
  y_axis_max: undefined  
}
```

Description

The padding property may be used to override the series default padding. Padding is the amount of pixels added to the min and max series values in order to push the extremes and prevent the series from hitting the edge of the chart.

The amount of default padding is dependant upon the series type and axis configuration and may not

be applicable to all series types.

This property is only applicable during series creation. To retrieve or modify the series padding after creation please see `Series.getPadding()` and `Series.setPadding()`

Example

>> Remove all padding from the BarSeries

```
var barSeries = new EJSC.BarSeries(new EJSC.ArrayDataHandler([..data..]), {
    padding: {
        x_axis_min: 0,
        x_axis_max: 0,
        y_axis_min: 0,
        y_axis_max: 0
    }
});
```

4.4.14.1.11 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`

4.4.14.1.12 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`

4.4.14.1.13 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.

4.4.14.1.14 `visible` (inherited)

Definition

boolean `visible` = true

Description

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

4.4.14.1.15 `x_axis` (inherited)

Definition

string `x_axis` = "bottom"

Description

Defines the horizontal axis to use for X values. Valid options are "top" or "bottom".

4.4.14.1.16 `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

4.4.14.1.17 `y_axis` (inherited)

Definition

string `y_axis` = "left"

Description

Defines the vertical axis to use for Y values. Valid options are "left" or "right".

4.4.14.1.18 `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

4.4.14.2 Methods

4.4.14.2.1 findClosestByPixel (inherited)

Definition

EJSC.Point **findClosestByPixel**(object coordinates)

```
point = {
  x: screen coordinate,
  y: screen coordinate
}
```

Description

Returns the point object closest to the screen pixel coordinates provided. The x and y properties of the point object should be in the same scale as the x and y axes assigned to the series. To locate points based on pixel coordinates, see `Series.findClosestByPixel()`

4.4.14.2.2 findClosestByPoint (inherited)

Definition

EJSC.Point **findClosestByPoint**(object coordinate)

```
point = {
  x: axis coordinate,
  y: axis coordinate
}
```

Description

Returns the point object closest to the axis coordinates provided. The x and y properties of the coordinate object should be based on the top left of the viewport and take into account the page scroll. To locate points based on axis coordinates, see `Series.findClosestByPoint()`

4.4.14.2.3 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.

4.4.14.2.4 getPadding (inherited)

Definition

object **getPadding**()

RETURNS:

```
{
    x_axis_min: number,
    x_axis_max: number,
    y_axis_min: number,
    y_axis_max: number
}
```

Description

Returns an object containing the series current padding. If padding has been set either by the Series.setPadding() method of the Series.padding property during construction the result of this method will be adjusted to return the most current padding values.

4.4.14.2.5 getVisibility (inherited)

Definition

boolean **getVisibility**()

Description

Returns a boolean indicating the series current visible state

4.4.14.2.6 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.

4.4.14.2.7 hideLegend (inherited)

Definition

void **hideLegend**()

Description

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

4.4.14.2.8 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.

4.4.14.2.9 setColor (inherited)

Definition

void **setColor**(string color)

Description

Changes the series color and causes the chart to redraw.

4.4.14.2.10 setColoredLegend (inherited)

Definition

void **setColoredLegend**(boolean coloredLegend)

Description

Sets the EJSCT.Series.coloredLegend property and updates the legend to reflect the change.

4.4.14.2.11 setDataHandler (inherited)

Definition

void **setDataHandler**(EJSCT.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.

4.4.14.2.12 setLineOpacity (inherited)

Definition

void **setLineOpacity**(integer opacity)

Description

Sets the EJSK.Series.lineOpacity property and redraws the series to reflect the change.

4.4.14.2.13 setLineWidth (inherited)

Definition

void **setLineWidth**(integer width)

Description

Updates the lineWidth property and redraws the chart.

4.4.14.2.14 setOpacity (inherited)

Definition

void **setOpacity**(integer opacity)

Description

Sets the EJSK.Series.opacity property and redraws the series to reflect the change.

4.4.14.2.15 setPadding (inherited)

Definition

void **setPadding**(object Padding, boolean Redraw)

The padding object should be defined as when used in the series constructor. See Series.padding.

Description

This method updates the user-defined series padding and optionally recalculates the chart extremes and redraws the chart.

4.4.14.2.16 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 EJSK.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" );
```

4.4.14.2.17 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.

4.4.14.2.18 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.

4.4.14.2.19 showLegend (inherited)

Definition

```
void showLegend( )
```

Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

4.4.14.3 Events

4.4.14.3.1 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.

4.4.14.3.2 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.

4.4.14.3.3 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.

4.4.14.3.4 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.

4.4.14.4 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>
string" label="string" userdata="string"/>        <P x="number or string" y="number or
string" label="string" userdata="string"/>        <P x="number or string" y="number or
            </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" }
]

```

4.4.14.5 Text Replacement Options

String

[chart_title]
[series_title]

[xaxis]
[yaxis]
[x]

[y]

[label]

With

title of the chart
title of the series the selected

point resides in

caption of the series x axis
caption of the series y axis
preformatted X value of the

point

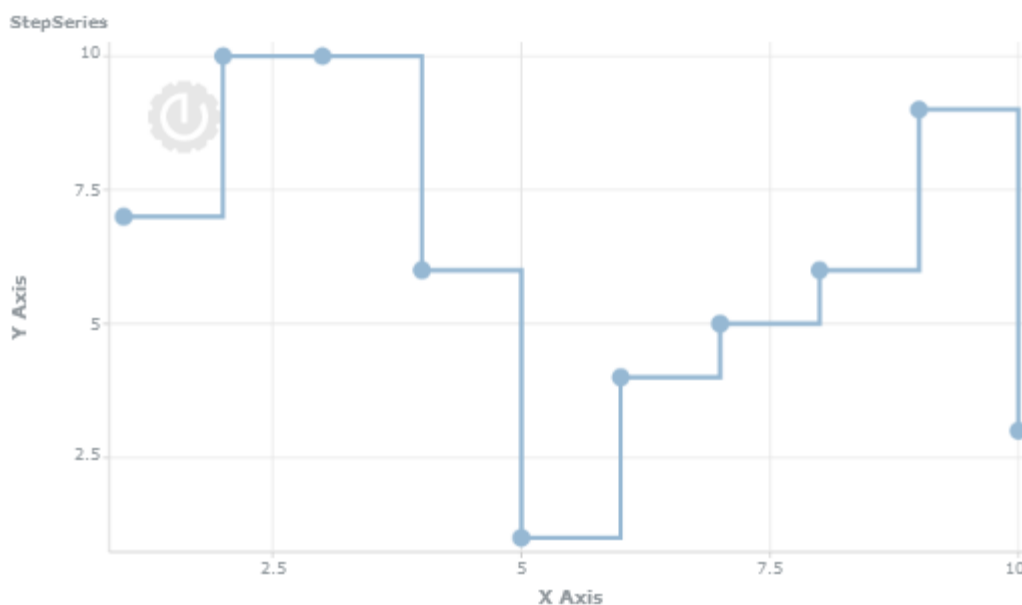
preformatted Y value of the

point

label property of the current

point

4.4.15 EJSC.StepSeries



The StepSeries is rendered similar to the LineSeries, but the line "steps" (ie it only changes Y values at the next point's X value).

The constructor expects an instantiated EJSC.DataHandler descendant and an optional set of object properties.

Constructor

```
EJSC.StepSeries( EJSC.DataHandler dataHandler, {object options} )
```

Example

```
var mySeries = new EJSC.StepSeries(
    new EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,3],[5,2],[6,4]]),
    { title: "New Step Series" }
);
```


Defining Properties and Events

StepSeries 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.StepSeries(new
EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,3],[5,2],[6,4]]));
StepSeries.lineWidth = 2;
StepSeries.title = "New Step Series";
```

Setting properties in batch

```
var mySeries = new EJSC.StepSeries(
    new EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,3],[5,2],[6,4]]),
    { StepSeries: 2, title: "New Step Series" }
);
```

4.4.15.1 Properties

4.4.15.1.1 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)

4.4.15.1.2 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)

4.4.15.1.3 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()`

4.4.15.1.4 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.

4.4.15.1.5 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]"  
} );
```

4.4.15.1.6 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.

4.4.15.1.7 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()`

4.4.15.1.8 lineWidth (inherited)

Definition

integer **lineWidth** = 1

Description

Defines the width of the line connecting series points.

4.4.15.1.9 padding (inherited)

Definition

```
object padding = {  
    x_axis_min: undefined,  
    x_axis_max: undefined,  
    y_axis_min: undefined,  
    y_axis_max: undefined  
}
```

Description

The padding property may be used to override the series default padding. Padding is the amount of pixels added to the min and max series values in order to push the extremes and prevent the series from hitting the edge of the chart.

The amount of default padding is dependant upon the series type and axis configuration and may not be applicable to all series types.

This property is only applicable during series creation. To retrieve or modify the series padding after creation please see `Series.getPadding()` and `Series.setPadding()`

Example

>> Remove all padding from the BarSeries

```
var barSeries = new EJSC.BarSeries(new EJSC.ArrayDataHandler([..data..]), {  
    padding: {  
        x_axis_min: 0,  
        x_axis_max: 0,  
        y_axis_min: 0,  
        y_axis_max: 0  
    }  
});
```

4.4.15.1.10 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.

4.4.15.1.11 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.

4.4.15.1.12 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.

4.4.15.1.13 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.

4.4.15.1.14 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.

4.4.15.1.15 `visible` (inherited)

Definition

boolean **visible** = true

Description

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

4.4.15.1.16 `x_axis` (inherited)

Definition

string **x_axis** = "bottom"

Description

Defines the horizontal axis to use for X values. Valid options are "top" or "bottom".

4.4.15.1.17 `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

4.4.15.1.18 `y_axis` (inherited)

Definition

string **y_axis** = "left"

Description

Defines the vertical axis to use for Y values. Valid options are "left" or "right".

4.4.15.1.19 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

4.4.15.2 Methods

4.4.15.2.1 findClosestByPoint (inherited)

Definition

EJSC.Point **findClosestByPoint**(object coordinate)

```
point = {  
  x: axis coordinate,  
  y: axis coordinate  
}
```

Description

Returns the point object closest to the axis coordinates provided. The x and y properties of the coordinate object should be based on the top left of the viewport and take into account the page scroll. To locate points based on axis coordinates, see `Series.findClosestByPoint()`

4.4.15.2.2 findClosestByPixel (inherited)

Definition

EJSC.Point **findClosestByPixel**(object coordinates)

```
point = {  
  x: screen coordinate,  
  y: screen coordinate  
}
```

Description

Returns the point object closest to the screen pixel coordinates provided. The x and y properties of the point object should be in the same scale as the x and y axes assigned to the series. To locate points based on pixel coordinates, see `Series.findClosestByPixel()`

4.4.15.2.3 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.

4.4.15.2.4 getPadding (inherited)

Definition

object [getPadding](#)()

RETURNS:

```
{  
    x_axis_min: number,  
    x_axis_max: number,  
    y_axis_min: number,  
    y_axis_max: number  
}
```

Description

Returns an object containing the series current padding. If padding has been set either by the Series.setPadding() method of the Series.padding property during construction the result of this method will be adjusted to return the most current padding values.

4.4.15.2.5 getVisibility (inherited)

Definition

boolean [getVisibility](#)()

Description

Returns a boolean indicating the series current visible state

4.4.15.2.6 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.

4.4.15.2.7 hideLegend (inherited)

Definition

void **hideLegend**()

Description

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

4.4.15.2.8 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.

4.4.15.2.9 setColor (inherited)

Definition

void **setColor**(string color)

Description

Changes the series color and causes the chart to redraw.

4.4.15.2.10 setColoredLegend (inherited)

Definition

void **setColoredLegend**(boolean coloredLegend)

Description

Sets the EJSC.Series.coloredLegend property and updates the legend to reflect the change.

4.4.15.2.11 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.

4.4.15.2.12 setLineOpacity (inherited)

Definition

void **setLineOpacity**(integer opacity)

Description

Sets the EJSC.Series.lineOpacity property and redraws the series to reflect the change.

4.4.15.2.13 setLineWidth (inherited)

Definition

void **setLineWidth**(integer width)

Description

Updates the lineWidth property and redraws the chart.

4.4.15.2.14 setPadding (inherited)

Definition

void **setPadding**(object Padding, boolean Redraw)

The padding object should be defined as when used in the series constructor. See Series.padding.

Description

This method updates the user-defined series padding and optionally recalculates the chart extremes and redraws the chart.

4.4.15.2.15 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.

4.4.15.2.16 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.

4.4.15.2.17 showLegend (inherited)

Definition

void [showLegend](#)()

Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

4.4.15.3 Events

4.4.15.3.1 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.

4.4.15.3.2 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.

4.4.15.3.3 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.

4.4.15.3.4 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.

4.4.15.4 Text Replacement Options

String	With
[chart_title]	title of the chart
[series_title]	title of the series the selected
[xaxis]	point resides in
[yaxis]	caption of the series x axis
[x]	caption of the series y axis
	preformatted X value of the
[y]	point
	preformatted Y value of the
[label]	point
	label property of the current
	point

4.4.15.5 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

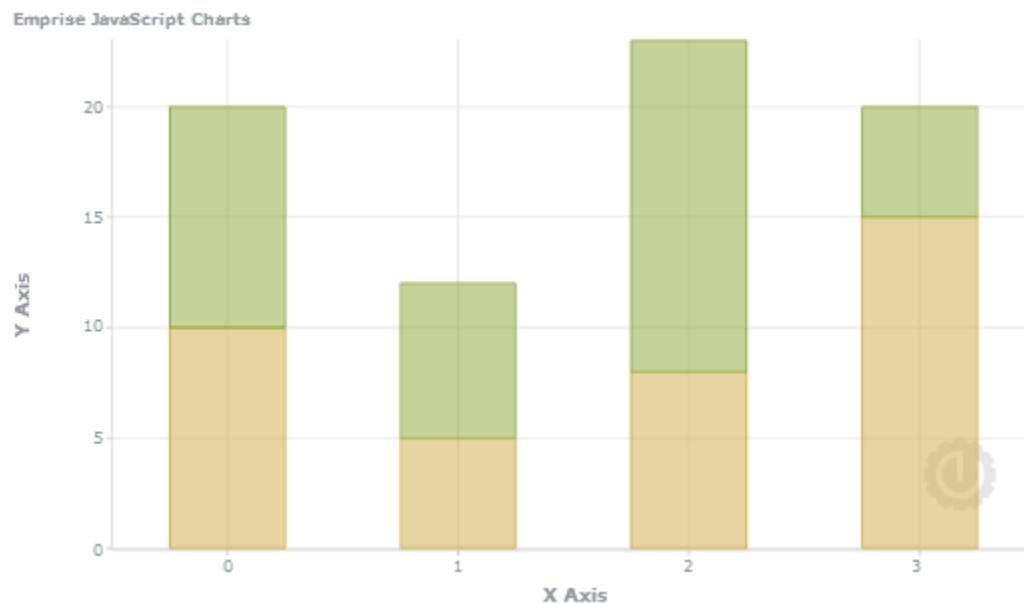
```
"x|y|null|userdata,x|y|null|userdata"
```

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"}
]
```

4.4.16 EJSC.StackedBarSeries



StackedBarSeries renders its points as vertical or horizontal bars which are stacked on the previous series.

The constructor expects an instantiated EJSC.DataHandler descendant and an optional set of object properties.

Constructor

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

Example

```
var mySeries = new EJSC.StackedBarSeries(
    new EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,3],[5,2],[6,4]]),
    { title: "New Stacked 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 myStackedSeries1 = new EJSC.StackedBarSeries(new EJSC.ArrayDataHandler([[1,1],[2,2],[3,3]]),
    myStackedSeries1.lineWidth = 1;
    myStackedSeries1.title = "Bottom Stack";

var myStackedSeries2 = new EJSC.StackedBarSeries(new EJSC.ArrayDataHandler([[1,1],[2,2],[3,3]]),
    myStackedSeries2.lineWidth = 1;
    myStackedSeries2.title = "Top Stack";
```

Setting properties in batch

```
var myStackedSeries1 = new EJSC.StackedBarSeries(
    new EJSC.ArrayDataHandler( [ [0,10],[1,5],[2,8],[3,15] ] ),
```

```

        { lineWidth: 1, title: "Bottom Stack" }
    );

    var myStackedSeries2 = new EJSC.StackedBarSeries(
        new EJSC.ArrayDataHandler( [ [0,10],[1,7],[2,15],[3,5] ] )
        { lineWidth: 1, title: "Top Stack" }
    );

```

4.4.16.1 Properties

4.4.16.1.1 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)

4.4.16.1.2 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)

4.4.16.1.3 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()

4.4.16.1.4 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.

4.4.16.1.5 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.

4.4.16.1.6 groupedBars (inherited)

Definition

boolean **groupedBars** = true

Description

Defines whether the bars from different bar series in a single chart are grouped or overlayed.

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 overlayed 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));
```

4.4.16.1.7 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]"
});
```

4.4.16.1.8 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
    }
));
```

4.4.16.1.9 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.

4.4.16.1.10 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()`

4.4.16.1.11 `lineWidth` (inherited)

Definition

integer `lineWidth` = 1

Description

Defines the width of the line connecting series points.

4.4.16.1.12 `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()`

4.4.16.1.13 `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.

4.4.16.1.14 `padding` (inherited)

Definition

```
object padding = {  
  x_axis_min: undefined,  
  x_axis_max: undefined,  
  y_axis_min: undefined,  
  y_axis_max: undefined  
}
```

Description

The padding property may be used to override the series default padding. Padding is the amount of

pixels added to the min and max series values in order to push the extremes and prevent the series from hitting the edge of the chart.

The amount of default padding is dependant upon the series type and axis configuration and may not be applicable to all series types.

This property is only applicable during series creation. To retrieve or modify the series padding after creation please see `Series.getPadding()` and `Series.setPadding()`

Example

>> Remove all padding from the BarSeries

```
var barSeries = new EJSC.BarSeries(new EJSC.ArrayDataHandler([..data..]), {
    padding: {
        x_axis_min: 0,
        x_axis_max: 0,
        y_axis_min: 0,
        y_axis_max: 0
    }
});
```

4.4.16.1.15 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 }
        ]
    }
));
```

```
));
```

4.4.16.1.16 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.

4.4.16.1.17 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.

4.4.16.1.18 treeLegendRoot

Definition

boolean **treeLegendRoot** = false

Description

Defines whether to display the stacked series as a top level node in the legend with each bar series shown as a child below.

4.4.16.1.19 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  
  }  
));
```

4.4.16.1.20 `visible` (inherited)

Definition

boolean `visible` = true

Description

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

4.4.16.1.21 `x_axis` (inherited)

Definition

string `x_axis` = "bottom"

Description

Defines the horizontal axis to use for X values. Valid options are "top" or "bottom".

4.4.16.1.22 `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

4.4.16.1.23 `y_axis` (inherited)

Definition

string `y_axis` = "left"

Description

Defines the vertical axis to use for Y values. Valid options are "left" or "right".

4.4.16.1.24 `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

4.4.16.2 Methods

4.4.16.2.1 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.

4.4.16.2.2 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(...));
```

4.4.16.2.3 clearRanges (inherited)

Definition

void **clearRanges**(boolean redraw)

Description

Clears all ranges defined for the series and optionally redraws the the chart.

4.4.16.2.4 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.

4.4.16.2.5 findClosestByPixel (inherited)

Definition

EJSC.Point **findClosestByPixel**(object coordinates)

```
point = {  
    x: screen coordinate,  
    y: screen coordinate  
}
```

Description

Returns the point object closest to the screen pixel coordinates provided. The x and y properties of the point object should be in the same scale as the x and y axes assigned to the series. To locate points based on pixel coordinates, see `Series.findClosestByPixel()`

4.4.16.2.6 findClosestByPoint (inherited)

Definition

EJSC.Point **findClosestByPoint**(object coordinate)

```
point = {  
    x: axis coordinate,  
    y: axis coordinate  
}
```

Description

Returns the point object closest to the axis coordinates provided. The x and y properties of the coordinate object should be based on the top left of the viewport and take into account the page scroll.

To locate points based on axis coordinates, see `Series.findClosestByPoint()`

4.4.16.2.7 `getBarSize` (inherited)

Definition

integer `getBarSize()`

Description

This method returns the size in pixels (width or height depending on orientation) of a single bar.

4.4.16.2.8 `getBarSizeInPoints` (inherited)

Definition

integer `getBarSizeInPoints()`

Description

This method returns the size in chart points (width or height depending on orientation) of a single bar.

4.4.16.2.9 `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.

4.4.16.2.10 `getPadding` (inherited)

Enter topic text here.

4.4.16.2.11 `getPoints` (inherited)

Definition

array `getPoints()`

Description

This method returns an array of all `EJSC.BarPoint` objects in the series.

4.4.16.2.12 `getVisibility` (inherited)

Definition

boolean `getVisibility()`

Description

Returns a boolean indicating the series current visible state

4.4.16.2.13 `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.

4.4.16.2.14 `hideLegend` (inherited)

Definition

void `hideLegend`()

Description

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

4.4.16.2.15 `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.

4.4.16.2.16 `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.

4.4.16.2.17 `setColor` (inherited)

Definition

void **setColor**(string color)

Description

Changes the series color and causes the chart to redraw.

4.4.16.2.18 **setColoredLegend** (inherited)

Definition

void **setColoredLegend**(boolean coloredLegend)

Description

Sets the `EJSC.Series.coloredLegend` property and updates the legend to reflect the change.

4.4.16.2.19 **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.

4.4.16.2.20 **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);
```

4.4.16.2.21 `setGroupedBars` (inherited)**Definition**

void `setGroupedBars`(boolean grouped, boolean redraw)

Description

Changes the chart between grouped and overlaid bars and optionally triggers a redraw.

4.4.16.2.22 `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.

4.4.16.2.23 `setLineOpacity` (inherited)**Definition**

void `setLineOpacity`(integer opacity)

Description

Sets the `EJSC.Series.lineOpacity` property and redraws the series to reflect the change.

4.4.16.2.24 `setLineWidth` (inherited)**Definition**

void `setLineWidth`(integer width)

Description

Updates the `lineWidth` property and redraws the chart.

4.4.16.2.25 `setOpacity` (inherited)**Definition**

void `setOpacity`(integer opacity)

Description

Sets the `EJSC.Series.opacity` property and redraws the series to reflect the change.

4.4.16.2.26 setPadding (inherited)

Definition

void **setPadding**(object Padding, boolean Redraw)

The padding object should be defined as when used in the series constructor. See Series.padding.

Description

This method updates the user-defined series padding and optionally recalculates the chart extremes and redraws the chart.

4.4.16.2.27 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.

4.4.16.2.28 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.

4.4.16.2.29 showLegend (inherited)

Definition

void **showLegend**()

Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

4.4.16.3 Events

4.4.16.3.1 onAfterDataAvailable (inherited)

Definition

boolean **onAfterDataAvailable**(EJSChart chart, EJSCSeries 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.

4.4.16.3.2 onAfterVisibilityChange (inherited)

Definition

boolean **onAfterVisibilityChange**(EJSCSeries series, EJSChart 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.

4.4.16.3.3 onBarNeedsColor (inherited)

Definition

string **onBarNeedsColor**(EJSCBarPoint point, EJSCSeries series, EJSChart chart)
 object **onBarNeedsColor**(EJSCBarPoint point, EJSCSeries series, EJSChart 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 EJSChart("chart");
var bar = chart.addSeries(new EJSCBarSeries( 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)";
        }
    }
}
}
));
```

4.4.16.3.4 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.

4.4.16.3.5 onShowHint (inherited)

Definition

string **onShowHint**(EJSC.Point point, EJSC.Series series, EJSC.Chart chart, string hint_string)

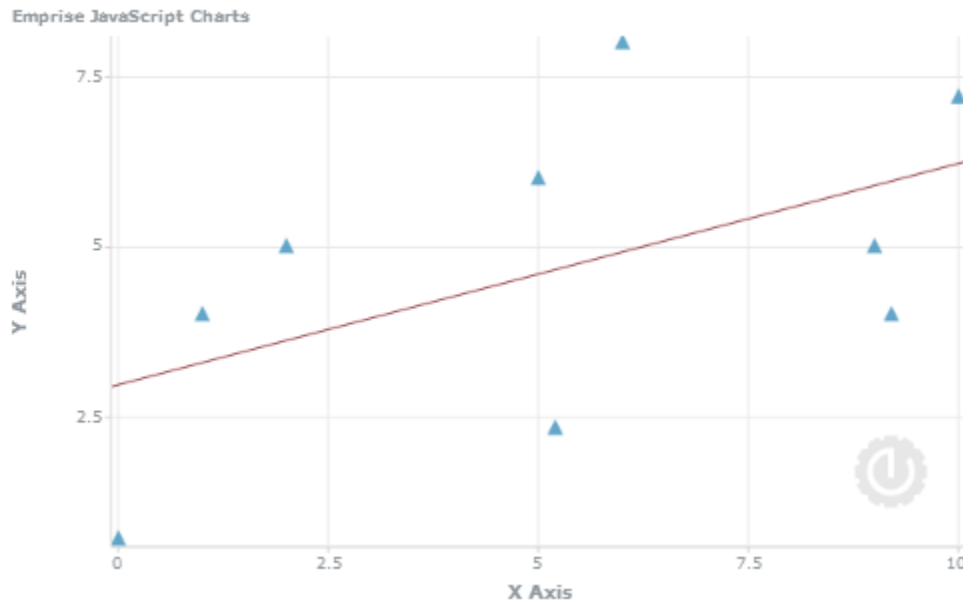
Description

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.

4.4.16.4 Text Replacement Options

String	With
[chart_title]	title of the chart
[series_title]	title of the series the selected
	point resides in
[xaxis]	caption of the series x axis
[yaxis]	caption of the series y axis
[x]	preformatted X value of the
	point
[y]	preformatted Y value of the
	point
[label]	label property of the current
	point

4.4.17 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" }  
);
```

4.4.17.1 Properties

4.4.17.1.1 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)

4.4.17.1.2 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()

4.4.17.1.3 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]"
```

```
} );
```

4.4.17.1.4 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.

4.4.17.1.5 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()`

4.4.17.1.6 lineWidth (inherited)

Definition

integer **lineWidth** = 1

Description

Defines the width of the line connecting series points.

4.4.17.1.7 padding (inherited)

Definition

```
object padding = {  
  x_axis_min: undefined,  
  x_axis_max: undefined,  
  y_axis_min: undefined,  
  y_axis_max: undefined  
}
```

Description

The padding property may be used to override the series default padding. Padding is the amount of pixels added to the min and max series values in order to push the extremes and prevent the series from hitting the edge of the chart.

The amount of default padding is dependant upon the series type and axis configuration and may not be applicable to all series types.

This property is only applicable during series creation. To retrieve or modify the series padding after creation please see `Series.getPadding()` and `Series.setPadding()`

Example

>> *Remove all padding from the BarSeries*

```
var barSeries = new EJSC.BarSeries(new EJSC.ArrayDataHandler([..data..]), {
    padding: {
        x_axis_min: 0,
        x_axis_max: 0,
        y_axis_min: 0,
        y_axis_max: 0
    }
});
```

4.4.17.1.8 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.

4.4.17.1.9 visible (inherited)

Definition

boolean **visible** = true

Description

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

4.4.17.1.10 x_axis (inherited)

Definition

string **x_axis** = "bottom"

Description

Defines the horizontal axis to use for X values. Valid options are "top" or "bottom".

4.4.17.1.11 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

4.4.17.1.12 `y_axis` (inherited)

Definition

string `y_axis` = "left"

Description

Defines the vertical axis to use for Y values. Valid options are "left" or "right".

4.4.17.1.13 `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

4.4.17.2 Methods

4.4.17.2.1 `findClosestByPixel` (inherited)

Definition

EJSC.Point `findClosestByPixel`(object coordinates)

```
point = {  
  x: screen coordinate,  
  y: screen coordinate  
}
```

Description

Returns the point object closest to the screen pixel coordinates provided. The x and y properties of the point object should be in the same scale as the x and y axes assigned to the series. To locate points based on pixel coordinates, see `Series.findClosestByPixel()`

4.4.17.2.2 findClosestByPoint (inherited)

Definition

EJSC.Point **findClosestByPoint**(object coordinate)

```
point = {  
    x: axis coordinate,  
    y: axis coordinate  
}
```

Description

Returns the point object closest to the axis coordinates provided. The x and y properties of the coordinate object should be based on the top left of the viewport and take into account the page scroll. To locate points based on axis coordinates, see `Series.findClosestByPoint()`

4.4.17.2.3 getPadding (inherited)

Definition

object **getPadding**()

RETURNS:

```
{  
    x_axis_min: number,  
    x_axis_max: number,  
    y_axis_min: number,  
    y_axis_max: number  
}
```

Description

Returns an object containing the series current padding. If padding has been set either by the `Series.setPadding()` method of the `Series.padding` property during construction the result of this method will be adjusted to return the most current padding values.

4.4.17.2.4 getVisibility (inherited)

Definition

boolean **getVisibility**()

Description

Returns a boolean indicating the series current visible state

4.4.17.2.5 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.

4.4.17.2.6 `hideLegend` (inherited)

Definition

```
void hideLegend( )
```

Description

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

4.4.17.2.7 `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.

4.4.17.2.8 `setColor` (inherited)

Definition

```
void setColor( string color )
```

Description

Changes the series color and causes the chart to redraw.

4.4.17.2.9 `setColoredLegend` (inherited)

Definition

```
void setColoredLegend( boolean coloredLegend )
```

Description

Sets the `EJSC.Series.coloredLegend` property and updates the legend to reflect the change.

4.4.17.2.10 `setLineWidth` (inherited)

Definition

void **setLineWidth**(integer width)

Description

Updates the lineWidth property and redraws the chart.

4.4.17.2.11 setPadding (inherited)

Definition

void **setPadding**(object Padding, boolean Redraw)

The padding object should be defined as when used in the series constructor. See Series.padding.

Description

This method updates the user-defined series padding and optionally recalculates the chart extremes and redraws the chart.

4.4.17.2.12 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.

4.4.17.2.13 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.

4.4.17.2.14 showLegend (inherited)

Definition

void **showLegend**()

Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

4.4.17.3 Events

4.4.17.3.1 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.

4.4.17.3.2 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.

4.4.17.3.3 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.

4.4.17.4 Text Replacement Options

String

[chart_title]
[series_title]

[xaxis]
[yaxis]
[x]

[y]

[label]

With

title of the chart
title of the series the selected

point resides in
caption of the series x axis
caption of the series y axis
preformatted X value of the

point
preformatted Y value of the

point
label property of the current

point

4.5 Data Handlers

4.5.1 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] ]  
);
```

4.5.1.1 Methods

4.5.1.1.1 `getArray`

Definition

array [getArray](#)()

Description

Returns the data array currently stored in the ArrayDataHandler.

4.5.1.1.2 `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.

4.5.1.1.3 `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.

4.5.1.2 Events

4.5.1.2.1 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.

4.5.2 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"  
);
```

4.5.2.1 Properties

4.5.2.1.1 requestType (inherited)

Definition

string `requestType` = "GET"

Description

Defines the type of HTTP request to be made. Valid options are "GET" and "POST"

4.5.2.1.2 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.

4.5.2.1.3 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"
```

4.5.2.2 Methods

4.5.2.2.1 getUrl (inherited)

Definition

string `getUrl`()

Description

Returns the url string currently stored in the AjaxDataHandler descendant.

DEPRECATED, see `EJSC.AjaxDataHandler.url`

4.5.2.2.2 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.

4.5.2.2.3 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()`

4.5.2.2.4 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()`

4.5.2.2.5 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()`

4.5.2.2.6 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.

4.5.2.3 Events

4.5.2.3.1 `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 `EJSC.Series` object.

4.5.2.3.2 `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.

4.5.2.3.3 `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.

4.5.3 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"
);
```

4.5.3.1 Methods

4.5.3.1.1 getCSV

Definition

string **getCSV**()

Description

Returns the csv string currently stored in the CSVStringDataHandler.

4.5.3.1.2 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.

4.5.3.1.3 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.

4.5.3.2 Events

4.5.3.2.1 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.

4.5.4 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.

4.5.4.1 Properties

4.5.4.1.1 requestType (inherited)

Definition

string `requestType` = "GET"

Description

Defines the type of HTTP request to be made. Valid options are "GET" and "POST"

4.5.4.1.2 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.

4.5.4.1.3 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"
```

4.5.4.2 Methods

4.5.4.2.1 getUrl (inherited)

Definition

string **getUrl**()

Description

Returns the url string currently stored in the AjaxDataHandler descendant.

DEPRECATED, see [EJSC.AjaxDataHandler.url](#)

4.5.4.2.2 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.

4.5.4.2.3 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()`

4.5.4.2.4 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()`

4.5.4.2.5 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()`

4.5.4.2.6 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.

4.5.4.3 Events

4.5.4.3.1 `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 `EJSC.Series` object.

4.5.4.3.2 `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.

4.5.4.3.3 `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`.

4.5.5 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" } }'  
);
```

4.5.5.1 Methods

4.5.5.1.1 getJSON

Definition

string **getJSON**()

Description

Returns the JSON string currently stored in the `JSONStringDataHandler`.

4.5.5.1.2 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.

4.5.5.1.3 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.

4.5.5.2 Events

4.5.5.2.1 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.

4.5.6 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

4.5.6.1 Properties

4.5.6.1.1 requestType (inherited)

Definition

string `requestType` = "GET"

Description

Defines the type of HTTP request to be made. Valid options are "GET" and "POST"

4.5.6.1.2 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.

4.5.6.1.3 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"
```

4.5.6.2 Methods

4.5.6.2.1 getUrl (inherited)

Definition

string **getUrl**()

Description

Returns the url string currently stored in the AjaxDataHandler descendant.

DEPRECATED, see *EJSC.AjaxDataHandler.url*

4.5.6.2.2 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.

4.5.6.2.3 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()`

4.5.6.2.4 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()`

4.5.6.2.5 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()`

4.5.6.2.6 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.

4.5.6.3 Events

4.5.6.3.1 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 EJSCT.Series object.

4.5.6.3.2 onDataReady (inherited)

Definition

boolean [onDataReady](#)(XMLHttpRequest response, EJSCT.AjaxDataHandler data_handler, EJSCT.Series series, EJSCT.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.

4.5.6.3.3 onNeedsData (inherited)

Definition

boolean [onNeedsData](#)(EJSCT.AjaxDataHandler dataHandler, EJSCT.Series series, EJSCT.Chart chart)
XMLHttpRequest [onNeedsData](#)(EJSCT.AjaxDataHandler dataHandler, EJSCT.Series series, EJSCT.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.

4.5.7 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" />  
');
```

Formats:

The XMLStringDataHandler supports three formats:

- Full
- Short
- Compact

4.5.7.1 Methods

4.5.7.1.1 getXML

Definition

string [getXML](#)()

Description

Returns the xml string currently stored in the XMLStringDataHandler.

4.5.7.1.2 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.

4.5.7.1.3 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.

4.5.7.2 Events

4.5.7.2.1 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.

4.6 Label Formatters**4.6.1 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"
});
```

4.6.1.1 Properties

4.6.1.1.1 format_string

Definition

string **format_string** = ""

Description

The following format options are available:

Format	
YYYY	(4) digit year (i.e. 2007)
YY	(2) digit year (i.e. 07)
MMMM	month name (i.e. January)
MMM	month name (i.e. Jan)
MM	(2) digit month of the year
	with leading zeros (i.e. 01)
M	of the year without leading
	zeros
DDDD	day of the week (i.e. Monday)
DDD	day of the week (i.e. Mon)
DD	(2) digit day of the month with
	leading zeros (i.e. 04)
D	of the month without leading
	zeros
HH	of the day in 24-hour format
	with leading zeros
H	of the day in 24-hour format
	without leading zeros
hh	of the day in 12-hour format
	with leading zeros
h	of the day in 12-hour format
	without leading zeros
NN	with leading zeros
N	without leading zeros
SS	with leading zeros
S	without leading zeros
ZZZ	(3 places) with leading zeros
ZZ	(2 places) with leading zeros
Z	without leading zeros
AA	/PM notation
A	/P notation
aa	/pm notation
a	/p notation

4.6.1.1.2 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
});
```

4.6.1.1.3 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.

4.6.1.2 Methods

4.6.1.2.1 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.

4.6.2 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: "()"   
});
```

4.6.2.1 Properties4.6.2.1.1 `currency_align`**Definition**

string `currency_align` = "left"

Description

Valid options are "left" and "right".

4.6.2.1.2 `currency_position`**Definition**

string `currency_position` = "inner"

Description

Valid options are "inner" and "outer".

4.6.2.1.3 `currency_symbol`**Definition**

string `currency_symbol` = ""

Description

Specifies the symbol used to indicate currency.

Note: If blank "" then no currency marker is used.

4.6.2.1.4 `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 ","

4.6.2.1.5 forced_decimals

Definition

integer **forced_decimals** = undefined

Description

Specifies the number of decimal places which MUST be displayed (0's are appended if necessary).

4.6.2.1.6 negative_symbol

Definition

string **negative_symbol** = "-"

Description

Specifies the formatting of negative numbers.

Valid options are "-" and "()"

4.6.2.1.7 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

4.6.2.1.8 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.

4.6.2.2 Methods

4.6.2.2.1 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.

4.6.3 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"  
});
```

4.6.3.1 Properties

4.6.3.1.1 append

Definition

string **append** = undefined

Description

The string stored in the append property will be applied to every value processed by the formatter.

4.6.3.1.2 prefix

Definition

string **prefix** = undefined

Description

The string stored in the prefix property will be applied to every value processed by the formatter.

4.6.3.2 Methods

4.6.3.2.1 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.

4.6.3.3 Events

4.6.3.3.1 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;
  }
});
```

4.7 Base Classes

4.7.1 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

4.7.2 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

4.7.2.1 Properties

4.7.2.1.1 requestType

Definition

string **requestType** = "GET"

Description

Defines the type of HTTP request to be made. Valid options are "GET" and "POST"

4.7.2.1.2 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.

4.7.2.1.3 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"
```

4.7.2.2 Methods

4.7.2.2.1 getUrl

Definition

string **getUrl**()

Description

Returns the url string currently stored in the AjaxDataHandler descendant.

DEPRECATED, see *EJSC.AjaxDataHandler.url*

4.7.2.2.2 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.

4.7.2.2.3 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()`

4.7.2.2.4 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()`

4.7.2.2.5 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()`

4.7.2.2.6 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.

4.7.2.3 Events

4.7.2.3.1 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 `EJSC.Series` object.

4.7.2.3.2 onDataReady

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.

4.7.2.3.3 onNeedsData

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.

4.7.3 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

4.7.3.1 Properties

4.7.3.1.1 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.

4.7.3.1.2 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" }  
    }  
} );
```

4.7.3.1.3 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" }  
} );
```

4.7.3.1.4 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" }
} );
```

4.7.3.1.5 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" }
} );
```

4.7.3.1.6 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 }
    }
} );
```

4.7.3.1.7 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"
        }
    }
} );
```

4.7.3.1.8 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 }  
} );
```

4.7.3.1.9 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 }  
} );
```

4.7.3.1.10 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_d  
    }  
} );
```

4.7.3.1.11 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 }  
    }  
} );
```

4.7.3.1.12 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:"  
    }  
} );
```

4.7.3.1.13 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
  }
} );
```

4.7.3.1.14 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
  }
} );
```

4.7.3.1.15 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.

4.7.3.1.16 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" }
  }
} );
```


4.7.3.1.17 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  
    }  
} );
```

4.7.3.1.18 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 }  
} );
```

4.7.3.1.19 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 }  
} );
```

4.7.3.1.20 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 }  
} );
```

4.7.3.1.21 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 }  
    }  
} );
```

4.7.3.2 Methods

4.7.3.2.1 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(...)
);
```

4.7.3.2.2 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

4.7.3.2.3 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`

4.7.3.2.4 `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.

4.7.3.2.5 `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.

4.7.3.2.6 `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.

4.7.3.2.7 `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)

4.7.3.2.8 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.

4.7.3.2.9 pointToPixel

Definition

integer **pointToPixel**(number coordinate)
 object **pointToPixel**(number coordinate, boolean ignoreBounds)

object result: {
 p: integer,
 outsideBounds: boolean
 }

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)

When ignoreBounds is not specified or specified as undefined, the result will be NaN if the coordinate provided is outside the currently displayed range.

4.7.3.2.10 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.

4.7.3.2.11 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`.

4.7.3.2.12 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.

4.7.3.2.13 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 );
```

4.7.3.2.14 setZoom

Definition:

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

Description

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

4.7.3.2.15 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.

4.7.3.2.16 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.

4.7.3.3 Events

4.7.3.3.1 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.

4.7.3.3.2 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.

4.7.3.3.3 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;
}
```

4.7.3.3.4 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.

4.7.3.3.5 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.

4.7.4 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

4.7.4.1 Methods

4.7.4.1.1 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.

4.7.4.2 Events

4.7.4.2.1 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.

4.7.5 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

4.7.5.1 Properties

4.7.5.1.1 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. EJSeries.PieSeries)

4.7.5.1.2 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 `EJSeries.Series.setColoredLegend()`

4.7.5.1.3 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.

4.7.5.1.4 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.

4.7.5.1.5 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()`

4.7.5.1.6 lineWidth (inherited)

Definition

integer **lineWidth** = 1

Description

Defines the width of the line connecting series points.

4.7.5.1.7 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()`

4.7.5.1.8 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.

4.7.5.1.9 visible (inherited)

Definition

boolean **visible** = true

Description

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

4.7.5.1.10 `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

4.7.5.2 Methods

4.7.5.2.1 `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.

4.7.5.2.2 `getVisibility` (inherited)

Definition

boolean `getVisibility`()

Description

Returns a boolean indicating the series current visible state

4.7.5.2.3 `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.

4.7.5.2.4 hideLegend (inherited)

Definition

```
void hideLegend( )
```

Description

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

4.7.5.2.5 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.

4.7.5.2.6 setColor (inherited)

Definition

```
void setColor( string color )
```

Description

Changes the series color and causes the chart to redraw.

4.7.5.2.7 setColoredLegend (inherited)

Definition

```
void setColoredLegend( boolean coloredLegend )
```

Description

Sets the EJSC.Series.coloredLegend property and updates the legend to reflect the change.

4.7.5.2.8 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.

4.7.5.2.9 setLineOpacity (inherited)

Definition

void **setLineOpacity**(integer opacity)

Description

Sets the EJSC.Series.lineOpacity property and redraws the series to reflect the change.

4.7.5.2.10 setLineWidth (inherited)

Definition

void **setLineWidth**(integer width)

Description

Updates the lineWidth property and redraws the chart.

4.7.5.2.11 setOpacity (inherited)

Definition

void **setOpacity**(integer opacity)

Description

Sets the EJSC.Series.opacity property and redraws the series to reflect the change.

4.7.5.2.12 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.

4.7.5.2.13 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.

4.7.5.2.14 showLegend (inherited)

Definition

void **showLegend**()

Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

4.7.5.3 Events

4.7.5.3.1 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.

4.7.5.3.2 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.

4.7.5.3.3 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.

4.7.6 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()`

4.7.6.1 Properties

4.7.6.1.1 `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.

4.7.6.2 Methods

4.7.6.2.1 `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.

4.7.7 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`

4.7.7.1 Properties

4.7.7.1.1 label

Definition

string **label** = null

Description

Defines the point label. Currently this is only implemented in EJSCEPieSeries EJSCEPiePoint objects.

4.7.7.1.2 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 EJSCEXMLDataHandler (full and short formats).

4.7.7.1.3 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.

4.7.7.1.4 y

Definition

float **y** = null

Description

Defines point Y value.

4.7.8 EJSCESeries

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 EJSCEChart class.

Constructor

none

4.7.8.1 Properties

4.7.8.1.1 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)

4.7.8.1.2 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)

4.7.8.1.3 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()

4.7.8.1.4 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.

4.7.8.1.5 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<br /><strong>X: </strong>[x]<br/><strong>Y: </strong>[y]"  
});
```

4.7.8.1.6 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.

4.7.8.1.7 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()

4.7.8.1.8 lineWidth

Definition

integer **lineWidth** = 1

Description

Defines the width of the line connecting series points.

4.7.8.1.9 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()`

4.7.8.1.10 padding

Definition

```
object padding = {  
    x_axis_min: undefined,  
    x_axis_max: undefined,  
    y_axis_min: undefined,  
    y_axis_max: undefined  
}
```

Description

The padding property may be used to override the series default padding. Padding is the amount of pixels added to the min and max series values in order to push the extremes and prevent the series from hitting the edge of the chart.

The amount of default padding is dependant upon the series type and axis configuration and may not be applicable to all series types.

This property is only applicable during series creation. To retrieve or modify the series padding after creation please see `Series.getPadding()` and `Series.setPadding()`

Example

>> Remove all padding from the BarSeries

```
var barSeries = new EJSC.BarSeries(new EJSC.ArrayDataHandler([..data..]), {  
    padding: {  
        x_axis_min: 0,  
        x_axis_max: 0,  
        y_axis_min: 0,  
        y_axis_max: 0  
    }  
});
```

4.7.8.1.11 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.

4.7.8.1.12 `visible`

Definition

boolean `visible` = true

Description

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

4.7.8.1.13 `x_axis`

Definition

string `x_axis` = "bottom"

Description

Defines the horizontal axis to use for X values. Valid options are "top" or "bottom".

4.7.8.1.14 `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

4.7.8.1.15 `y_axis`

Definition

string `y_axis` = "left"

Description

Defines the vertical axis to use for Y values. Valid options are "left" or "right".

4.7.8.1.16 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

4.7.8.2 Methods

4.7.8.2.1 findClosestByPixel

Definition

EJSC.Point **findClosestByPixel**(object coordinates)

```
point = {  
  x: screen coordinate,  
  y: screen coordinate  
}
```

Description

Returns the point object closest to the screen pixel coordinates provided. The x and y properties of the point object should be in the same scale as the x and y axes assigned to the series. To locate points based on pixel coordinates, see `Series.findClosestByPixel()`

4.7.8.2.2 findClosestByPoint

Definition

EJSC.Point **findClosestByPoint**(object coordinate)

```
point = {  
  x: axis coordinate,  
  y: axis coordinate  
}
```

Description

Returns the point object closest to the axis coordinates provided. The x and y properties of the coordinate object should be based on the top left of the viewport and take into account the page scroll. To locate points based on axis coordinates, see `Series.findClosestByPoint()`

4.7.8.2.3 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.

4.7.8.2.4 getPadding

Definition

object [getPadding](#)()

RETURNS:

```
{
    x_axis_min: number,
    x_axis_max: number,
    y_axis_min: number,
    y_axis_max: number
}
```

Description

Returns an object containing the series current padding. If padding has been set either by the Series.setPadding() method of the Series.padding property during construction the result of this method will be adjusted to return the most current padding values.

4.7.8.2.5 getVisibility

Definition

boolean [getVisibility](#)()

Description

Returns a boolean indicating the series current visible state

4.7.8.2.6 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.

4.7.8.2.7 hideLegend

Definition

void **hideLegend**()

Description

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

4.7.8.2.8 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.

4.7.8.2.9 setColor

Definition

void **setColor**(string color)

Description

Changes the series color and causes the chart to redraw.

4.7.8.2.10 setColoredLegend

Definition

void **setColoredLegend**(boolean coloredLegend)

Description

Sets the EJSC.Series.coloredLegend property and updates the legend to reflect the change.

4.7.8.2.11 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.

4.7.8.2.12 setLineOpacity

Definition

void **setLineOpacity**(integer opacity)

Description

Sets the EJSCT.Series.lineOpacity property and redraws the series to reflect the change.

4.7.8.2.13 setLineWidth

Definition

void **setLineWidth**(integer width)

Description

Updates the lineWidth property and redraws the chart.

4.7.8.2.14 setOpacity

Definition

void **setOpacity**(integer opacity)

Description

Sets the EJSCT.Series.opacity property and redraws the series to reflect the change.

4.7.8.2.15 setPadding

Definition

void **setPadding**(object Padding, boolean Redraw)

The padding object should be defined as when used in the series constructor. See Series.padding.

Description

This method updates the user-defined series padding and optionally recalculates the chart extremes and redraws the chart.

4.7.8.2.16 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.

4.7.8.2.17 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.

4.7.8.2.18 showLegend

Definition

void **showLegend**()

Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

4.7.8.3 Events

4.7.8.3.1 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.

4.7.8.3.2 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.

4.7.8.3.3 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.

4.7.8.3.4 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.

4.8 Other Classes

4.8.1 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 )
```

4.8.1.1 Properties

4.8.1.1.1 label (inherited)

Definition

string **label** = null

Description

Defines the point label. Currently this is only implemented in EJSC.PieSeries EJSC.PiePoint objects.

4.8.1.1.2 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).

4.8.1.1.3 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.

4.8.1.1.4 y (inherited)

Definition

float **y** = null

Description

Defines point Y value.

4.8.2 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 )
```

4.8.2.1 Properties

4.8.2.1.1 label (inherited)

Definition

string **label** = null

Description

Defines the point label. Currently this is only implemented in EJSCEPieSeries EJSCEPiePoint objects.

4.8.2.1.2 max

Definition

float **max** = null

Description

Defines point max value (x or y depending on the series orientation).

4.8.2.1.3 min

Definition

float **min** = null

Description

Defines point min value (x or y depending on the series orientation).

4.8.2.1.4 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 EJSCEXMLDataHandler (full and short formats).

4.8.2.1.5 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.

4.8.2.1.6 y (inherited)

Definition

float **y** = null

Description

Defines point Y value.

4.8.3 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 )
```

4.8.3.1 Properties

4.8.3.1.1 label (inherited)

Definition

string **label** = null

Description

Defines the point label. Currently this is only implemented in EJSC.PieSeries EJSC.PiePoint objects.

4.8.3.1.2 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).

4.8.3.1.3 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.

4.8.4 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 )
```

4.8.4.1 Properties

4.8.4.1.1 label (inherited)

Definition

string **label** = null

Description

Defines the point label. Currently this is only implemented in EJSC.PieSeries EJSC.PiePoint objects.

4.8.4.1.2 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).

4.8.4.1.3 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.

4.8.5 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 )
```

4.8.5.1 Properties

4.8.5.1.1 close

Definition

Description

4.8.5.1.2 high

Definition

Description

4.8.5.1.3 label (inherited)

Definition

string **label** = null

Description

Defines the point label. Currently this is only implemented in EJSC.PieSeries EJSC.PiePoint objects.

4.8.5.1.4 low

Definition

Description

4.8.5.1.5 open

Definition**Description**

4.8.5.1.6 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).

4.8.5.1.7 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.

4.8.6 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 )
```

4.8.6.1 Properties

4.8.6.1.1 label (inherited)

Definition

string **label** = null

Description

Defines the point label. Currently this is only implemented in EJSChart.PieSeries EJSChart.PiePoint objects.

4.8.6.1.2 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 EJSChart.XMLDataHandler (full and short formats).

4.8.6.1.3 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.

4.8.6.1.4 y (inherited)

Definition

float **y** = null

Description

Defines point Y value.

4.9 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>
<button onclick="buttonClick();">Export SVG</button>
```

4.10 META Tag Configuration

META tags must be included in the page header (between `<head></head>`), should be placed before the `<script>` tag which imports the `EJSCChart.js` library and are formatted as follows:

```
<meta name="OPTION-NAME" content="OPTION-VALUE"/>
```

The options described below affect the loading/initialization stage of the library.

ejsc-src-path

Specifies the base path of the `EJSCChart.js` and other library files. Setting this option is sometimes necessary when the library cannot find the `<script>` tag in the page header due to dynamic loading or inclusion in the `<body>` of the page.

META "content" attribute: Path to directory containing `EJSCChart` files, i.e. `content="/EJSCChart/"`

ejsc-auto-load-support-files

Specifies whether the library should automatically load its supporting `.js` and `.css` files. This option can be turned off to allow for manual inclusion of the necessary files when site organization necessitates the placing of all `.css`, `.js` and images in separate files.

META "content" attribute: `true` or `false`, i.e. `content="false"`

The default, if this tag is not included, is `true`.

We recommend keeping all EJSCChart related files in their original directory structure as version upgrades will be much less work.

ejsc-v1-compatibility

Automatically loads the compatibility file.

META "content" attribute: true or false, i.e. content="true"

The default, if this tag is not included, is false.

4.11 Text Replacement Options

Each series has its own set of text replacement options.

4.12 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 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

Sales Related Inquiries

sales@ejschart.com

Technical Support

support@ejschart.com

Phone

Sales: +1 (860) 464-8555

Support: (See support contract)

Mailing Address:

PO Box 129

Ledyard, CT 06339

USA