Globals /

API Documentation for SciChart.js - v3.5.727

Go to JavaScript Charting Documentation Go to JavaScript Chart Examples Go to SciChart.js on Github

Index

Enumerations

Classes

- AdornerLayer
- AnimationFiniteStateMachine
- AnimationToken
- AnnotationBase
- AnnotationClickEventArgs
- AnnotationDragDeltaEventArgs
- AnnotationHoverEventArgs
- AnnotationHoverModifier
- AxisBase2D
- AxisBase3D
- AxisBase3DLabelStyle
- AxisCore
- AxisCubeDescriptor
- AxisCubeEntity
- AxisLayoutState
- AxisMarkerAnnotation
- AxisRenderer
- AxisTitleRenderer
- BandAnimation
- BandAnimationStyle
- BandSeriesDataLabelProvider
- BandSeriesDrawingProvider
- BandSeriesHitTestProvider
- BaseAnimationStyle
- BaseAxisLayoutStrategy
- BaseBandRenderableSeries
- BaseCenteredAxisLayoutStrategy
- BaseDataLabelProvider
- BaseDataSeries
- BaseDataSeries3D
- BaseGridDataSeries3D
- BaseHeatmapDataSeries
- BaseHeatmapRenderableSeries
- BaseHitTestProvider
- BaseLineRenderableSeries
- BaseMeshPointMarker3D
- BaseMountainRenderableSeries
- BaseOhlcRenderableSeries
- BasePaletteProvider
- BasePointMarker
- BasePointMarker3D
- BasePointMarkerStyle
- BasePointSeriesResampled
- BasePointSeriesWrapped
- BaseRenderDataTransform
- BaseRenderableSeries
- BaseRenderableSeries3D
- BaseSceneEntity3D
- BaseSeriesDrawingProvider
- BaseStackedCollection
- BaseStackedMountainRenderableSeries
- BaseStackedRenderableSeries
- BaseTexturePointMarker3D
- BatchRenderContext
- BezierRenderDataTransform
- BottomAlignedInnerAxisLayoutStrategy
- BottomAlignedOuterAxisLayoutStrategy
- BottomAlignedOuterHorizontallyStackedAxisLayoutStrategy
- FlippedCategoryCoordinateCalculator
- FlippedNumericCoordinateCalculator
- FontKey
- GenericAnimation
- GizmoEntity
- GlowEffect
- GradientColorPalette
- GradientParams
- Guard
- HeatMapDataLabelProvider
- HeatmapColorMap
- HeatmapLegend
- HeatmapSeriesInfo
- HitTestInfo
- HitTestInfo3D
- HlcCustomFilter
- HlcDataSeries
- HlcFilterBase
- HlcPointSeriesWrapped
- HlcScaleOffsetFilter
- HlcSeriesInfo
- HorizontalLineAnnotation
- HoveredChangedArgs
- ImpulseSeriesHitTestProvider
- LabelInfo
- LabelProvider
- LabelProviderBase2D
- LayoutManager
- LeftAlignedInnerAxisLayoutStrategy
- LeftAlignedOuterAxisLayoutStrategy
- LeftAlignedOuterVerticallyStackedAxisLayoutStrategy
- LegendModifier
- LineAnimation
- LineAnnotation
- LineSeriesDataLabelProvider
- LineSeriesDrawingProvider
- LineSeriesHitTestProvider
- List
- LogarithmicAxis
- LogarithmicCoordinateCalculator
- LogarithmicDeltaCalculator
- LogarithmicLabelProvider
- LogarithmicTickProvider
- Logger
- ManualLegend
- MemoryUsageHelper
- MeshColorPalette
- ModifierArgsBase
- ModifierMouseArgs
- MountainAnimation
- MountainAnimationStyle
- MountainSeriesDrawingProvider
- MountainSeriesHitTestProvider
- MouseManager
- MouseWheelZoomModifier
- MouseWheelZoomModifier3D
- NativeTextAnnotation
- NonUniformHeatMapDataLabelProvider
- ScaleAnimation
- ScatterAnimation
- ScatterPointsSceneEntity
- ScatterRenderableSeries3D
- ScatterSeriesHitTestProvider
- SceneDescriptor
- SciChart3DRenderer
- SciChart3DSurface
- SciChartDefaults
- SciChartHorizontalGroup
- SciChartJSDarkTheme
- SciChartJSDarkv2Theme
- SciChartJSLightTheme
- SciChartJsNavyTheme
- SciChartLegend
- SciChartLegendBase
- SciChartOverview
- SciChartPieLegend
- SciChartPieSurface
- SciChartRenderer
- SciChartSubSurface
- SciChartSurface
- SciChartSurfaceBase
- SciChartVerticalGroup
- SelectionChangedArgs
- SeriesAnimation
- SeriesAnimationFiniteStateMachine
- SeriesHoveredArgs
- SeriesInfo
- SeriesInfo3D
- SeriesSelectedArgs
- SeriesSelectionModifier
- SeriesVisibleChangedArgs
- ShaderEffect
- ShadowEffect
- Size
- SmartDateLabelProvider
- SmoothStackedMountainRenderableSeries
- SmoothStackedRenderDataTransform
- SolidColorBrushPalette
- SpherePointMarker3D
- SplineBandRenderableSeries
- SplineLineRenderableSeries
- SplineMountainRenderableSeries
- SplineRenderDataTransform
- SpritePointMarker
- SquarePointMarker
- StackedCollectionDataLabelProvider
- StackedColumnCollection
- StackedColumnRenderableSeries
- StackedColumnSeriesDataLabelProvider
- StackedColumnSeriesHitTestProvider
- StackedMountainCollection
- StackedMountainRenderableSeries
- StackedMountainSeriesHitTestProvider
- StackedXySeriesInfo
- StaticTickCoordinatesProvider

- EDrawMeshAs
- EErrorDirection
- EErrorMode
- EExecuteOn
- EFillPaletteMode
- EHorizontalAlignment
- EHorizontalAnchorPoint
- EHorizontalTextPosition
- EHoverMode
- EInnerAxisPlacementCoordinateMode
- ELabelAlignment
- ELabelPlacement
- ELabelProviderType
- ELayoutManagerType
- ELayoutStrategyType
- ELegendOrientation
- ELegendPlacement
- ELegendType
- ELineDrawMode
- ELineType
- ELogarithmicMajorTickMode
- ELogarithmicMinorTickMode
- EMarkerType
- EMeshPaletteMode
- EMeshResolution

- BoxAnnotation
- BubbleAnimation
- BubbleSeriesDataLabelProvider
- BubbleSeriesDrawingProvider
- BubbleSeriesHitTestProvider
- CameraController
- CandlestickAnimation
- CandlestickAnimationStyle
- CanvasTexture
- CategoryAxis
- CategoryAxisBase
- CategoryCoordinateCalculator
- CategoryDeltaCalculator
- CentralAxesLayoutManager
- ChartModifierBase
- ChartModifierBase2D
- ChartModifierBase3D
- ChartTitleRenderer
- ColumnAnimation
- ColumnAnimationStyle
- ColumnRenderableSeries3D
- ColumnSceneEntity
- ColumnSeriesDataLabelProvider
- ColumnSeriesDrawingProvider
- ColumnSeriesHitTestProvider
- ContoursDataLabelProvider
- CoordinateCalculatorBase
- CrossPointMarker
- CrosshairLinesSceneEntity
- CubePointMarker3D
- CursorModifier
- CursorTooltipSvgAnnotation
- CustomAnnotation
- CustomChartModifier2D
- CustomChartModifier3D
- CustomPointMarkerStyle
- CylinderPointMarker3D
- DataDistributionCalculator
- DataLabelProvider
- DataLabelState
- DataPointInfo
- DataPointSelectionChangedArgs
- DataPointSelectionModifier
- DataPointSelectionPaletteProvider
- DateLabelProvider
- DateTimeDeltaCalculator
- DateTimeNumericAxis
- DefaultEntityIdProvider
- DefaultPaletteProvider
- DefaultSciChartLoader
- DefaultTickCoordinatesProvider
- DefaultViewportManager3D
- DeletableEntity
- DeltaCalculator
- Dictionary
- NonUniformHeatmapDataSeries
- NonUniformHeatmapDrawingProvider
- NonUniformHeatmapHitTestProvider
- NonUniformHeatmapRenderableSeries
- NumberRange
- NumberRangeAnimator
- NumberUtil
- NumericAxis
- NumericAxis3D
- NumericCoordinateCalculator
- NumericDeltaCalculator
- NumericLabelProvider
- NumericTickProvider
- ObjectRegistry
- ObservableArray
- ObservableArrayBase
- ObservableArrayChangedArgs
- OhlcAnimation
- OhlcAnimationStyle
- OhlcBaseRenderDataTransform
- OhlcCustomFilter
- OhlcDataSeries
- OhlcFilterBase
- OhlcPointSeriesResampled
- OhlcPointSeriesWrapped
- OhlcScaleOffsetFilter
- OhlcSeriesDrawingProvider
- OhlcSeriesHitTestProvider
- OhlcSeriesInfo
- OneTimePerformanceWarning
- OrbitModifier3D
- OverviewCustomResizableAnnotation
- OverviewRangeSelectionModifier
- PaletteFactory
- PerformanceDebugHelper
- PieLabelProvider
- PieSegment
- PinchZoomModifier
- PinchZoomModifier3D
- PixelPointMarker3D
- Point
- PointLine3DSceneEntity
- PointLineRenderableSeries3D
- PointMarkerDrawingProvider
- PointMarkerStyle
- PointerEventsMediatorModifier
- PriceBar
- PropertyChangedEventArgs
- PyramidPointMarker3D
- QuadPointMarker
- Rect
- RenderContext2D
- RenderContextAnnotationBase
- RenderPassData
- RenderPassDataCollection
- SurfaceMeshRenderableSeries3D
- SurfaceMeshSceneEntity
- SurfaceMeshSceneEntityState
- SurfaceMeshSeriesInfo3D
- SvgAnnotationBase
- SweepAnimation
- SynchronizedLayoutManager
- TemplateMetadataGenerator
- TextAnnotation
- TextDataLabelProvider
- TextLabelProvider
- TextSeriesHitTestProvider
- TextureManager
- ThemeProvider
- Thickness
- TickCoordinatesProvider
- TickProvider
- TitleRendererBase
- TooltipModifier3D
- TooltipSvgAnnotation3D
- TopAlignedInnerAxisLayoutStrategy
- TopAlignedOuterAxisLayoutStrategy
- TopAlignedOuterHorizontallyStackedAxisLayoutStrategy
- TrianglePointMarker
- TrianglePointMarker3D
- UniformContoursDrawingProvider
- UniformContoursRenderableSeries
- UniformGridDataSeries3D
- UniformHeatmapDataSeries
- UniformHeatmapDrawingProvider
- UniformHeatmapHitTestProvider
- UniformHeatmapRenderableSeries
- UpdateSuspender
- Vector3
- VerticalLineAnnotation
- VerticalSliceModifier
- ViewportManager3DBase
- VisibleRangeChangedArgs
- WaveAnimation
- WebGlBrush
- WebGlHelper
- WebGlPen
- WebGlRenderContext2D
- XAxisDragModifier
- XPointMarker
- XyBaseRenderDataTransform
- XyCustomFilter
- XyDataSeries
- XyFilterBase
- XyLinearTrendFilter
- XyMovingAverageFilter
- XyPointSeriesResampled
- XyPointSeriesWrapped
- XyRatioFilter
- XyScaleOffsetFilter

- DoubleAnimator
- DoubleVectorProvider
- DpiHelper
- EllipsePointMarker
- EllipsePointMarker3D
- ErrorSeriesDrawingProvider
- ErrorSeriesHitTestProvider
- EventHandler
- ExtremeResamplerHelper
- FIFOVectorProvider
- FadeAnimation
- FastBandRenderableSeries
- FastBubbleRenderable Series
- FastCandlestickRenderable Series
- FastColumnRenderable Series
- FastErrorBarsRenderable Series
- FastImpulseRenderable Series
- FastLineRenderableSeries
- FastMountainRenderable Series
- FastOhlcRenderableSeries
- FastTextRenderableSeries
- RenderPassInfo
- RenderPassInfo3D
- RenderableSeriesScene Entity
- RenderableSeriesScene EntityState
- ResamplingParams
- ResetCamera3DModifier
- RightAlignedInnerAxis LayoutStrategy
- RightAlignedOuterAxis LayoutStrategy
- RightAlignedOuter VerticallyStackedAxis LayoutStrategy
- RolloverLegendSvg Annotation
- RolloverMarkerSvg Annotation
- RolloverModifier
- RolloverModifier RenderableSeriesProps
- RolloverTooltipSvg Annotation
- RootSceneEntity
- RubberBandSvgRect
- RubberBandXyZoom Modifier
- XyScatterRenderableSeries
- XySeriesInfo
- XyTextDataSeries
- XyyBaseRenderData Transform
- XyyBezierRenderData Transform
- XyyCustomFilter
- XyyDataSeries
- XyyFilterBase
- XyyPointSeriesResampled
- XyyPointSeriesWrapped
- XyyScaleOffsetFilter
- XyySeriesInfo
- XyySplineRenderData Transform
- XyzCustomFilter
- XyzDataSeries
- XyzDataSeries3D
- XyzFilterBase
- XyzPointSeriesWrapped
- XyzScaleOffsetFilter
- XyzSeriesInfo
- XyzSeriesInfo3D
- YAxisDragModifier
- ZoomExtentsModifier
- ZoomPanModifier

Interfaces

- ColumnRenderable Series3DOptions
- I1DMetadataGenerator
- I2DMetadataGenerator
- I2DSubSurfaceOptions
- I2DSurfaceOptions
- I3DSurfaceOptions
- IAdornerProvider
- IAdvancedPaletteProvider
- IAllPaletteProviders
- IAnimation
- IAnnotation
- IAnnotationBaseOptions
- IAnnotationHoverModifier Options
- IAutoRangeAnimation Options
- IAxisBase2dOptions
- IAxisBase3dOptions
- IAxisCoreOptions
- IAxisDescriptor
- IAxisMarkerAnnotation Options
- IAxisParams
- IBandAnimationOptions
- IDateTimeNumericAxis Options
- IDeletable
- IDeltaCalculatorOptions
- IDictionary
- IDoubleVectorProvider
- IEasingMap
- IEngineeringPrefix
- IEntityIdProvider
- IEventHandler
- IEventListenerSource
- IEventSubscriptionItem
- IFadeAnimationOptions
- IFastErrorBarsRenderable SeriesOptions
- IFastLineRenderableSeries Options
- IFillPaletteProvider
- IFilterBase
- IGenericAnimation
- IGenericAnimationOptions
- IGetColorDataParams
- IGlowEffectOptions
- IGradientColorPalette Options
- IPointSeries
- IPointSeriesResampled
- IPointerEventsMediator ModifierOptions
- IRangeSelectionModifier Options
- IReceiveMouseEvents
- IRenderContext2D
- IRenderDataTransform
- IRenderableSeries
- IRenderableSeries3D
- IRenderableSeriesScene Entity
- IResamplingParamsClone Options
- IResetCamera3DOptions
- IReversableEasingMap
- IRolloverLegendSvg AnnotationOptions
- IRolloverModifier
- IRolloverModifierOptions
- IRolloverTooltipSvg AnnotationOptions
- IRubberBandXyZoom ModifierOptions

- IBandAnimationStyleOptions
- IBandRenderableSeriesOptions
- IBandSeriesDataLabelProviderOptions
- IBaseAnimationOptions
- IBaseAnimationStyleOptions
- IBaseBandRenderableSeriesOptions
- IBaseDataLabelProviderOptions
- IBaseDataSeries3DOptions
- IBaseDataSeriesOptions
- IBaseGridDataSeries3DOptions
- IBaseHeatmapSeriesOptions
- IBaseLineRenderableSeriesOptions
- IBaseMountainRenderableSeriesOptions
- IBaseOhlcRenderableSeriesOptions
- IBasePointMarker3DOptions
- IBasePointMarkerStyleOptions
- IBaseRenderableSeries3DOptions
- IBaseRenderableSeriesOptions
- IBaseSceneEntity
- IBaseStackedCollectionOptions
- IBaseStackedMountainRenderableSeriesOptions
- IBasedStackedRenderableSeriesOptions
- IBezierRenderDataTransformOptions
- IBoxAnnotationOptions
- IBrush2D
- IBubbleAnimationOptions
- IBubbleRenderableSeriesOptions
- IBubbleSeriesDataLabelProviderOptions
- ICacheable
- IGradientPaletteOptions
- IGridDataSeries3D
- IHVLineAnnotationOptions
- IHeatmapColorMapOptions
- IHeatmapDataLabelProviderOptions
- IHeatmapLegendOptions
- IHeatmapRenderableSeriesOptions
- IHeatmapSeries
- IHitTestProvider
- IHlcCustomFilterOptions
- IHlcDataSeriesOptions
- IHlcFilterOptions
- IHlcPointSeries
- IHlcScaleOffsetFilterOptions
- IHoverCallbackArgs
- IHoverOptions
- IHoverable
- IImpulseRenderableSeries
- IIncludeAxis
- IIncludeSeries
- IIncludeXAxis
- IIncludeYAxis
- IInnerAxisLayoutStrategyOptions
- ILabel2DOptions
- ILabelOptions
- ILayoutManagerOptions
- ILegendModifierOptions
- ILegendOptionsBase
- ILineAnimationOptions
- ILineAnnotationOptions
- ILineSeriesDataLabelProviderOptions
- ILineSeriesDrawingProviderProperties
- ILineStyle
- ILogarithmicAxisOptions
- IManualLegendOptions
- IMetadataGenerator
- IMountainAnimationOptions
- IMountainAnimationStyleOptions
- IMountainRenderableSeriesOptions
- IMouseWheelZoom
- IScaleAnimationOptions
- IScatterAnimationOptions
- ISciChart2DDefinition
- ISciChart3DSurfaceOptions
- ISciChartLoader
- ISciChartPieDefinition
- ISciChartSubSurfaceOptions
- ISciChartSurfaceBase
- ISciChartSurfaceOptions
- ISelectedPointOptions
- ISeriesDrawingProvider
- ISeriesSelectionModifierOptions
- IShaderEffectOptions
- IShadowEffectOptions
- ISmartDateLabelProviderOptions
- ISmoothStackedMountainRenderableSeriesOptions
- ISolidColorBrushPaletteOptions
- ISpline
- ISplineBandRenderableSeriesOptions
- ISplineLineRenderableSeriesOptions
- ISplineMountainRenderableSeriesOptions
- ISpritePointMarkerOptions
- ISpriteTextures
- IStackedColumnCollectionOptions
- IStackedColumnRenderableSeriesOptions
- IStackedColumnSeriesDataLabelProviderOptions
- IStackedMountainRenderableSeriesOptions
- IStrokePaletteProvider
- ISubChartDefinition
- ISurfaceMeshRenderableSeries3DOptions
- ISurfaceOptionsBase
- ISuspendable
- ISvgAnnotationBaseOptions
- ISweepAnimationOptions
- ITextAnnotationOptions
- ITextDataLabelProvider

- ICameraController
- ICameraOptions
- ICandlestickAnimationOptions
- ICandlestickAnimationStyleOptions
- ICandlestickRenderableSeriesOptions
- ICategoryAxisBaseOptions
- ICategoryAxisOptions
- ICentralAxesLayoutManagerOptions
- IChartModifierBase
- IChartModifierBase3DOptions
- IChartModifierBaseOptions
- IChartTitleRenderer
- IColorMapParams
- IColumnAnimationOptions
- IColumnAnimationStyleOptions
- IColumnRenderableSeriesOptions
- IColumnSeriesDataLabelProviderOptions
- IColumnSeriesDrawingProviderProperties
- IContourDrawingParams
- IContoursDataLabelProviderOptions
- IContoursRenderableSeriesOptions
- ICursorModifierOptions
- ICursorTooltipSvgAnnotationOptions
- ICustomAnnotationOptions
- ICustomPointMarkerStyleOptions
- ICustomResizableAnnotationOptions
- IDataChangeArgs
- IDataDistributionCalculator
- IDataLabelLayoutManager
- IDataLabelProviderOptions
- IDataPointSelectionModifierOptions
- IDataSeries
- IDataSeries3D
- IMouseWheelZoomModifier3DOptions
- IMouseWheelZoomModifierOptions
- INativeTextAnnotationOptions
- INonUniformHeatmapRenderableSeriesOptions
- INonUniformHeatmapSeriesOptions
- INotifyOnDpiChanged
- INumericAxis3dOptions
- INumericAxisOptions
- IOffsets
- IOhlcAnimationOptions
- IOhlcAnimationStyleOptions
- IOhlcCustomFilterOptions
- IOhlcDataSeriesOptions
- IOhlcFilterOptions
- IOhlcPointSeries
- IOhlcRenderableSeriesOptions
- IOhlcScaleOffsetFilterOptions
- IOrbitModifier3DOptions
- IOverviewOptions
- IPaletteProvider
- IPaletteProvider3D
- IPen2D
- IPenOptions
- IPieSegment
- IPieSegmentOptions
- IPieSurfaceOptions
- IPinchZoomModifier3DOptions
- IPinchZoomModifierOptions
- IPointLineRenderableSeries3DOptions
- IPointMarker
- IPointMarkerOptions
- IPointMarkerPaletteProvider
- IPointMarkerPaletteProvider3D
- IPointMarkerStyleOptions
- IPointMetadata
- IPointMetadata3D
- ITextLabelOptions
- ITextRenderableSeriesOptions
- ITextStyle
- IThemePartial
- IThemeProvider
- IThemeable
- ITitleRenderer
- ITooltipModifier3DOptions
- ITooltipSvgAnnotation3DOptions
- IUniformGridDataSeries3DOptions
- IUniformHeatmapSeriesOptions
- IUpdateSuspender
- IVerticalSliceOptions
- IWaveAnimationOptions
- IXAxisDragModifierOptions
- IXyCustomFilterOptions
- IXyDataSeriesOptions
- IXyFilterOptions
- IXyMovingAverageFilterOptions
- IXyPointSeries
- IXyRatioFilterOptions
- IXyScaleOffsetFilterOptions
- IXyScatterRenderableSeriesOptions
- IXyTextDataSeriesOptions
- IXyyCustomFilterOptions
- IXyyDataSeriesOptions
- IXyyFilterOptions
- IXyyPointSeries
- IXyyScaleOffsetFilterOptions
- IXyzCustomFilterOptions
- IXyzDataSeries3DOptions
- IXyzDataSeriesOptions
- IXyzFilterOptions
- IXyzPointSeries
- IXyzScaleOffsetFilterOptions
- IYAxisDragModifierOptions
- IZoomExtentsModifierOptions
- IZoomPanModifierOptions
- TSciChartPerformanceMark

Type aliases

- T NumberArray
- T RequiredOwnProps
- T SuspendedInstance
- T SvgStringTemplate
- T TAdvancedTextProperties
- T TFormatLabelFn
- T TGradientStop
- T TGridLineStyle
- T THeatmapLegend
- T THlcSeriesData
- T TSciChart
- T TSciChart3D
- T TSciChartConfig
- T TSciChartDestination
- T TSciChartSurfaceCanvases

- T TAdvancedTextStyle
- T TAnimationDefinition
- T TAnnotationDefinition
- T TArgb
- T TAxisCubeState
- T TAxisDefinition
- T TAxisTitleStyle
- T TAxisViewRects
- T TBorder
- T TCachedLabelStyle
- T TCameraState
- T TCategoryCoordCalc
- T TCellSizeMapper
- T TChartTitleStyle
- T TCheckedChangedArgs
- T TCommonTextStyle
- T TContourLineStyle
- T TCoord
- T TCursorTooltipData Template
- T TCursorTooltipSvgTemplate
- T TDataLabel
- T TDataLabelProvider Definition
- T TDataLabelStyle
- T TDataSeriesDefinition
- T TDpiChangedEventArgs
- T TDrawFunction
- T TEasingFn
- T TEffectDefinition
- T TElement
- T TFilterDefinition
- T THoverCallback
- T THoveredChangedCallback
- T TInnerLayoutStrategy Definition
- T TLabelProviderDefinition
- T TLayoutAxisPartsWith StrategyFunc
- T TLayoutManagerDefinition
- T TLegendItem
- T TLinearColorMap
- T TModifierDefinition
- T TModifierKeys
- T TModifierMouseArgs Params
- T TNativeCache
- T TNativeTextStyle
- T TOhlcSeriesData
- T TOuterLayoutStrategy Definition
- T TPaletteProviderDefinition
- T TPalettingState
- T TPerformanceDetailType
- T TPointMarkerArgb
- T TPointMarkerDefinition
- T TPositionPoperties
- T TPriceBar
- T TRenderLayers
- T TRolloverLegendSvg Template
- T TRolloverTooltipData Template
- T TRolloverTooltipSvg Template
- T TSelectionChanged Callback
- T TSeriesDataDefinition
- T TSeriesDefinition
- T TSeriesHoverChanged Callback
- T TSeriesRenderPassInfo
- T TSeriesSelectionChanged Callback
- T TSeriesVisibleChanged Callback
- T TSharedDataDefinition
- T TShift
- T TSize
- T TStackedAxisLength
- T TSurfaceDefinition
- T TTargetsSelector
- T TTextStyle
- T TTextStyleBase
- T TTextStyleBase3D
- T TTextureTextStyle
- T TTickLineStyle
- T TTickObject
- T TTooltip3DDataTemplate
- T TTooltip3DSvgTemplate
- T TWebAssemblyChart
- T TWebAssemblyChart3D
- T TXySeriesData
- T TXyTextSeriesData
- T TXyySeriesData
- T TXyzSeriesData
- T TZoomExtentsCallback
- T TfilterFunction

Variables

- AUTO_COLOR
- DebugForDpi
- FIFTY_DAYS
- FIVE_DAYS
- MIN_LOG_AXIS_VALUE
- MIN_SERIES_AREA_SIZE
- ONE_HOUR
- TEN_SECONDS
- buildStamp
- cleanupWasmContext
- constructorMap
- defaultSelectionAnnotation SvgString
- defaultUnSelected AnnotationSvgString
- devCount
- hasSent
- isDev
- isStorageAvailable
- labelCacheByTextAndStyle
- lastStyleId
- libraryVersion
- licenseType
- loaderCss
- maxSize
- minAge
- objectCache
- orderId
- precision
- productCode
- rect
- result
- sciChartConfig
- sciChartConfig3D
- sessionTime
- styleCache
- superScript_map
- telemetryEnabled

Functions

- adjustTooltipPosition
- adjustTooltipPosition3D
- animateAny
- animationUpdate
- appendRangeFifo
- applyOpacityToHtmlColor
- areArraysEqual
- arrayRemove
- base64Id
- bezierTransform
- build2DChart
- buildAnnotations
- buildAxes
- buildAxis
- buildChart
- buildDataSeries
- buildFilter
- buildModifier
- buildModifiers
- buildPieChart
- buildSeries
- calcAnnotationBordersFor AxisMarker
- calcAverageForArray
- calcAverageForDouble Vector
- calcCrossProduct
- calcDistance
- calcDistanceFromLine
- calcDistanceFromLine Segment
- calcDotProduct
- calcNewApex
- calcTooltipSize
- calculateAbsoluteRender Layer
- calculateCellCoordinates
- calculateHeatmapTexture
- calculateMaxGroupSize
- calculateOffsets
- chartReviver
- checkAreEqualTextStyles
- checkBuildStamp
- checkCanDraw
- checkIsAnimationRunning
- checkIsNaN
- checkStyle
- checkTextStyleEqual
- clearCacheByStyle
- clearLicensingDebug
- configure2DSurface
- configureChart
- convertColor
- convertLabelAlignment ToTextAlignment
- convertMultiLineAlignment
- convertRgbToHexColor
- convertSearchMode
- formatUnixDateToHuman StringSSms
- formatUnixDateToHuman StringYYYY
- freeCache
- freeStyle
- fromTsrVector4
- generateBooleanHash
- generateCombinedHash
- generateGuid
- generateHash
- generateNumberHash
- generateObjectHash
- getActiveAxes
- getAdjustedRotation
- getAllFontKeys
- getArraysEqual
- getAttributeFromString
- getAxis3dById
- getAxisById
- getAxisGeneric
- getCache
- getColor
- getColorDataForTexture
- getCoordinateWith CoordinateMode
- getDataSeriesDefinition
- getDataSeriesFrom RenderableSeriesDefinition
- getDescriptorsEqual
- getDevCount
- getFontKey
- getFontString
- getFunction
- getHorizontalAxisRequired Size
- getIncludedAxis
- getInterpolatedColor
- getIsDev
- getIsHorizontal
- getIsLicenseDebug
- getIsVertical
- getKey
- getLabel
- getLabelCoordinates
- getLabelValue
- getLegendContainerHtml
- getLegendItemHtml
- getLicenseType
- getLicenseWizardMaxPort
- getLicenseWizardPort
- getLineCoordinates
- getLineStylesEqual
- getMaxSize
- getMinAge
- getMonthString
- getNativeRect
- getNativeTextSize
- interpolateNumber
- isArraySorted
- isBetween
- isConstructor
- isNumberArray
- isRealNumber
- isTypedArray
- layoutAxisParts
- layoutAxisPartsBottom Strategy
- layoutAxisPartsLeftStrategy
- layoutAxisPartsRight Strategy
- layoutAxisPartsTopStrategy
- linearColorMapLerp
- logDoubleVector
- logToBase
- makeCacheKey
- makeIncArray
- measureTextHeight
- measureTextWidth
- memoize
- numericHashCode
- parseArgbToHtmlColor
- parseCacheKey
- parseColorToHexString Abgr
- parseColorToHexString Argb
- parseColorToTArgb
- parseColorToUIntAbgr
- parseColorToUIntArgb
- parsePc
- parseSize
- parseTArgbToHtmlColor
- pruneCache
- registerFunction
- registerType
- registerWasmType
- resetAxes
- resetCache
- runIfValue
- sendTelemetry
- setDevCount
- setIsDev
- setIsLicenseDebug
- setLabel
- setLicenseType
- setMaxSize
- setMinAge
- setOrderId
- setProductCode
- setTelemetry
- setUserCookie
- shouldSendTelemetry
- storageAvailable
- stringOccurrences
- stripAutoColor

- convertToHtmlPx
- convertToPixel
- convertToRelativeHtmlSize
- copyDoubleVector
- copyVector
- countUnique
- createBrushInCache
- createChartDestination
- createColorMap
- createDataSeries
- createHitTestInfo
- createImageAsync
- createImagesArrayAsync
- createNativeRect
- createPenInCache
- createPointMarker
- createSCRTPen
- createSingle3dInternal
- createSolidBrush
- createSvg
- createType
- deleteCache
- deleteSafe
- drawAxisMarkerAnnotation
- drawBorder
- drawLineAnnotation
- drawModifiersAxisLabel
- ensureRegistrations
- fillMetadata
- fillNoisySinewave
- fitElementToViewRect
- fitSvgToViewRect
- formatNumber
- formatNumber2Digits
- formatUnixDateToHuman String
- formatUnixDateToHuman StringDD
- formatUnixDateToHuman StringDDMM
- formatUnixDateToHuman StringDDMMHHMM
- formatUnixDateToHuman StringDDMMYY
- formatUnixDateToHuman StringHHMM
- formatUnixDateToHuman StringHHMMSS
- formatUnixDateToHuman StringMMM
- formatUnixDateToHuman StringMMMDD
- getNearestNonUniform HeatmapPoint
- getNearestPoint
- getNearestUniform HeatmapPoint
- getNearestXPoint
- getNearestXyPoint
- getNearestXyyPoint
- getNextRandomPriceBar Factory
- getNoisySinewave
- getOHLCYRange
- getOrderId
- getProductCode
- getRandomInRange
- getRubberBandRect
- getScrtBrushFromCache
- getScrtPenFromCache
- getSharedWasmContext
- getSize
- getStocksDataFactory
- getStyleId
- getSubTypes
- getTArgbEqual
- getTelemetry
- getTextBounds
- getTextStylesEqual
- getUniqueValues
- getUserCookie
- getValueWithCoordinate Mode
- getVector4
- getVectorColorVertex
- getVectorRectVertex
- getVertex
- getVerticalAxisRequired Size
- getWebGlBrushFromCache
- getWebGlPenFromCache
- getWindowedYRange
- getXRange
- getYyYRange
- guardSameLengthZValues AndMetadata
- handleInvalidAxis Alignment
- hasAllProperties
- hasOwnProperty
- htmlToElement
- initializeChartEngine2D
- interpolateColor
- interpolateLinear
- subArray
- switchData
- testHasExcluded
- testIsHitForBand
- testIsHitForColumn
- testIsHitForErrorBars
- testIsHitForImpulse
- testIsHitForLine
- testIsHitForMountain
- testIsHitForOHLC
- testIsHitForPoint
- testIsInBounds
- testIsInInterval
- testIsInXBounds
- testIsOverAxes
- toEngineering
- toHex
- toScientific
- toSuperScript
- translateDataValueRect ToAbsolute
- translateFromCanvas ToSeriesViewRect
- translateFromCanvas ToSeriesViewRectX
- translateFromCanvas ToSeriesViewRectY
- translateFromSeriesView RectToCanvas
- translateFromSeriesView RectToCanvasX
- translateFromSeriesView RectToCanvasY
- translateToNotScaled
- uintArgbColor IsTransparent
- uintArgbColorLerp
- uintArgbColorLerp24bit
- uintArgbColorMultiply Opacity
- uintArgbColorOverride Opacity
- uintArgbColorToAbgr
- updateAxisLayoutState
- updateLeftAndRightChart LayoutState
- updateTopAndBottom ChartLayoutState
- updateTsrVector4
- wrapNativeText
- wrapText
- zeroArray2D

Object literals

- animationHelpers
- annotationHelpers
- autoReverseEasing
- chartBuilder
- easing
- handler
- hashUtils
- hitTestHelpers
- labelCache
- licenseManager2dState
- localStorageApi
- testLayoutManager

Type aliases

NumberArray

`T` `NumberArray: number[] | Float64Array`

Defined in src/types/NumberArray.ts

RequiredOwnProps

```
T RequiredOwnProps<Type, BaseType>: Required<Omit<Type, keyof BaseType>>
```

Defined in src/types/HelperTypes.ts

Copies own properties of an interface or copies all of its props if base type is not provided. Makes all of the props required.

Type parameters

- **Type**: *BaseType*
- **BaseType**

SuspendedInstance

```
T SuspendedInstance: { id: string; suspendCount: number; suspendable: ISuspendable }
```

Defined in src/Charting/Visuals/UpdateSuspender.ts

Internal type used to track suspendable instances and number of nested suspend calls

Type declaration

- **id**: *string*
- **suspendCount**: *number*
- **suspendable**: *ISuspendable*

SvgStringTemplate

```
T SvgStringTemplate: (x1: number, y1: number, x2: number, y2: number) => string
```

Defined in src/Charting/Visuals/Annotations/OverviewCustomResizableAnnotation.ts

Function signature for the SVG builder function

Type declaration

```
(x1: number, y1: number, x2: number, y2: number): string
```

Parameters

- **x1:** *number*
- **y1:** *number*
- **x2:** *number*
- **y2:** *number*

Returns *string*

TAdvancedTextProperties

```
T TAdvancedTextProperties: { multilineAlignment?: EMultiLineAlignment; rotation?: number
}
```

Defined in src/types/TextStyle.ts

Type declaration

- Optional **multilineAlignment**?: *EMultiLineAlignment*
  Horizontal text alignment for multiline text.
- Optional **rotation**?: *number*
  Text rotation in degrees.

TAdvancedTextStyle

T TAdvancedTextStyle: *TCommonTextStyle* & *TAdvancedTextProperties*

Defined in src/types/TextStyle.ts

Defines text style with advanced options

TAnimationDefinition

T TAnimationDefinition: { options?: *IFadeAnimationOptions*; type: *Fade* } | { options?: *IScaleAnimationOptions*; type: *Scale* } | { options?: *ISweepAnimationOptions*; type: *Sweep* } | { options?: *IWaveAnimationOptions*; type: *Wave* } | { customType?: *string*; options?: *IBaseAnimationOptions*; type: *Custom* }

Defined in src/Builder/buildSeries.ts

Definition of an animation, comprising a EAnimationType and the relevant options

TAnnotationDefinition

T TAnnotationDefinition: { options?: *IAxisMarkerAnnotationOptions*; type: *RenderContextAxisMarkerAnnotation* } | { options?: *IBoxAnnotationOptions*; type: *RenderContextBoxAnnotation* } | { options?: *ILineAnnotationOptions*; type: *RenderContextHorizontalLineAnnotation* } | { options?: *ILineAnnotationOptions*; type: *RenderContextLineAnnotation* } | { options?: *ILineAnnotationOptions*; type: *RenderContextVerticalLineAnnotation* } | { options?: *ITextAnnotationOptions*; type: *SVGTextAnnotation* } | { options?: *ICustomAnnotationOptions*; type: *SVGCustomAnnotation* } | { options?: *INativeTextAnnotationOptions*; type: *RenderContextNativeTextAnnotation* }

Defined in src/Builder/buildAnnotations.ts

Definition of an annotation, comprising a EAnnotationType and the relevant options

TArgb

T TArgb: { blue: *number*; green: *number*; opacity: *number*; red: *number* }

Defined in src/utils/parseColor.ts

Type declaration

- **blue**: *number*
- **green**: *number*
- **opacity**: *number*
- **red**: *number*

TAxisCubeState

T TAxisCubeState: { xVisibleMax: *number*; xVisibleMin: *number*; xWorldDimension: *number*; yVisibleMax: *number*; yVisibleMin: *number*; yWorldDimension: *number*; zVisibleMax: *number*; zVisibleMin: *number*; zWorldDimension: *number* }

Defined in src/Charting3D/Visuals/Primitives/RenderableSeriesSceneEntityState.ts

A type class to contain information about the current state of 3D Axis Cube, such is Visible Range or World Dimensions.

Type declaration

- **xVisibleMax**: *number*
- **xVisibleMin**: *number*
- **xWorldDimension**: *number*
- **yVisibleMax**: *number*
- **yVisibleMin**: *number*
- **yWorldDimension**: *number*

- **zVisibleMax**: *number*
- **zVisibleMin**: *number*
- **zWorldDimension**: *number*

TAxisDefinition

```
T TAxisDefinition: { options?: INumericAxisOptions; type: NumericAxis } | { options?:
  ILogarithmicAxisOptions; type: LogarithmicAxis } | { options?: ICategoryAxisOptions;
  type: CategoryAxis } | { options?: IDateTimeNumericAxisOptions; type:
  DateTimeNumericAxis }
```

Defined in src/Builder/buildAxis.ts

Definition of an AxisBase2D, comprising a EAxisType and the relevant options

TAxisTitleStyle

```
T TAxisTitleStyle: TTextStyle & { rotation?: number }
```

Defined in src/Charting/Visuals/Axis/AxisCore.ts

TAxisViewRects

```
T TAxisViewRects: { axisRendererViewRect: Rect; axisTitleRendererViewRect: Rect }
```

Defined in src/Charting/LayoutManager/AxisLayoutHelpers.ts

Type declaration

- **axisRendererViewRect**: *Rect*
- **axisTitleRendererViewRect**: *Rect*

TBorder

```
T TBorder: { border?: number; borderBottom?: number; borderLeft?: number; borderRight?:
  number; borderTop?: number; color?: string }
```

Defined in src/types/TBorder.ts

A Border applied to the series viewport, axes or the entire chart surface

Type declaration

- Optional **border**?: *number*
- Optional **borderBottom**?: *number*
- Optional **borderLeft**?: *number*
- Optional **borderRight**?: *number*
- Optional **borderTop**?: *number*
- Optional **color**?: *string*

TCachedLabelStyle

```
T TCachedLabelStyle: TTextStyle & { extras?: string; providerId: string; rotation?:
  number }
```

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelCache.ts

TCameraState

```
T TCameraState: { height?: number; pitch?: number; radius?: number; width?: number; yaw?:
```

number }

Defined in src/Charting3D/ChartModifiers/ResetCamera3DModifier.ts

Type declaration

- Optional **height**?: *number*
- Optional **pitch**?: *number*
- Optional **radius**?: *number*
- Optional **width**?: *number*
- Optional **yaw**?: *number*

TCategoryCoordCalc

T TCategoryCoordCalc: *CategoryCoordinateCalculator* | *FlippedCategoryCoordinateCalculator*

Defined in src/Charting/Visuals/Axis/CategoryAxisBase.ts

TCellSizeMapper

T TCellSizeMapper: (index: *number*) => *number*

Defined in src/Charting/Model/NonUniformHeatmapDataSeries.ts

Type declaration

(index: *number*): *number*

Parameters

- **index:** *number*

Returns *number*

TChartTitleStyle

T TChartTitleStyle: *TAdvancedTextStyle* & { alignment?: *ETextAlignment*; placeWithinChart?: *boolean*; position?: *ETitlePosition* }

Defined in src/types/TextStyle.ts

Defines text style and positioning options used for Chart Title rendering

TCheckedChangedArgs

T TCheckedChangedArgs: { isChecked: *boolean*; series: *IRenderableSeries* }

Defined in src/Charting/ChartModifiers/LegendModifier.ts

Type args for the LegendModifier.isCheckedChanged callback

Type declaration

- **isChecked**: *boolean*

  Whether the corresponding legend item is checked or not (by default, this corresponds to IRenderableSeries.isVisible)

- **series**: *IRenderableSeries*

  The series which was checked or unchecked

TCommonTextStyle

T TCommonTextStyle: *TNativeTextStyle* | *TTextureTextStyle*

Defined in src/types/TextStyle.ts

Defines text style which allows to switch between Native / Non-native text rendering

TContourLineStyle

```
T TContourLineStyle: { color?: string; strokeThickness?: number }
```

Defined in src/Charting/Visuals/RenderableSeries/UniformContoursRenderableSeries.ts

A type class to contain information about contour line styles

remarks A contour line is drawn using the UniformContoursRenderableSeries with a 2D array of data

- Set the color as an HTML Color code to define the color
- Set the strokeThickness to change the thickness of the contour line

Type declaration

- Optional **color**?: *string*
- Optional **strokeThickness**?: *number*

TCoord

```
T TCoord: xCoord | yCoord
```

Defined in src/utils/tooltip.ts

TCursorTooltipDataTemplate

```
T TCursorTooltipDataTemplate: (seriesInfos: SeriesInfo[], tooltipTitle: string) =>
string[]
```

Defined in src/Charting/ChartModifiers/CursorModifier.ts

Type declaration

```
(seriesInfos: SeriesInfo[], tooltipTitle: string): string[]
```

Parameters

- **seriesInfos:** *SeriesInfo*[]
- **tooltipTitle:** *string*

Returns *string*[]

TCursorTooltipSvgTemplate

```
T TCursorTooltipSvgTemplate: (seriesInfos: SeriesInfo[], svgAnnotation:
CursorTooltipSvgAnnotation) => string
```

Defined in src/Charting/ChartModifiers/CursorModifier.ts

Type declaration

```
(seriesInfos: SeriesInfo[], svgAnnotation: CursorTooltipSvgAnnotation): string
```

Parameters

- **seriesInfos:** *SeriesInfo*[]
- **svgAnnotation:** *CursorTooltipSvgAnnotation*

Returns *string*

TDataLabel

```
T TDataLabel: { color?: number; dataX: number; dataY: number; position: Point; rect:
  Rect; text: string }
```

Defined in src/Charting/Visuals/RenderableSeries/DataLabels/BaseDataLabelProvider.ts

Type declaration

- Optional **color**?: *number*
- **dataX**: *number*
- **dataY**: *number*
- **position**: *Point*

  The start point for text drawing. This is not the top left of the text rectangle, since the y value for text is on the alphabetic baseline Difference is bounds.GetLineBounds(0).m_fHeight
- **rect**: *Rect*

  The Rectangle that contains the label, relative to the seriesViewRect
- **text**: *string*

TDataLabelProviderDefinition

```
T TDataLabelProviderDefinition: { options?: IDataLabelProviderOptions; type: Default } |
  { options?: ILineSeriesDataLabelProviderOptions; type: Line } | { options?:
  IColumnSeriesDataLabelProviderOptions; type: Column } | { options?:
  IBaseDataLabelProviderOptions; type: Text } | { options?:
  IHeatmapDataLabelProviderOptions; type: Heatmap } | { options?:
  IContoursDataLabelProviderOptions; type: Contours } | { options?:
  IBandSeriesDataLabelProviderOptions; type: Band } | { options?:
  IBubbleSeriesDataLabelProviderOptions; type: Bubble } | { options?:
  IHeatmapDataLabelProviderOptions; type: NonUniformHeatmap } | { options?:
  IBaseStackedCollectionOptions; type: StackedCollection } | { customType: string;
  options?: IBaseDataLabelProviderOptions; type: Custom }
```

Defined in src/Builder/buildSeries.ts

TDataLabelStyle

```
T TDataLabelStyle: { fontFamily?: string; fontSize?: number; lineSpacing?: number;
  multiLineAlignment?: EMultiLineAlignment; padding?: Thickness }
```

Defined in src/types/TDataLabelStyle.ts

A type class to contain information about data label styles

remarks

- Set the fontFamily as a string to set the font
- Set the fontSize as you would in HTML/CSS
- Set the color as an HTML Color code to define the color

Type declaration

- Optional **fontFamily**?: *string*
- Optional **fontSize**?: *number*
- Optional **lineSpacing**?: *number*
- Optional **multiLineAlignment**?: *EMultiLineAlignment*
- Optional **padding**?: *Thickness*

TDataSeriesDefinition

```
T TDataSeriesDefinition: { options: TSeriesDataDefinition; type: EDataSeriesType }
```

Defined in src/Builder/buildDataSeries.ts

Type declaration

- **options**: *TSeriesDataDefinition*
- **type**: *EDataSeriesType*

TDpiChangedEventArgs

`T TDpiChangedEventArgs: { newValue: number; oldValue: number }`

Defined in src/Charting/Visuals/TextureManager/DpiHelper.ts

Type arguments to DpiHelper.dpiChanged event

Type declaration

- **newValue**: *number*
  The new Dpi scaling factor
- **oldValue**: *number*
  The previous Dpi scaling factor

TDrawFunction

`T TDrawFunction: () => void`

Defined in src/Charting/Drawing/WebGlRenderContext2D.ts

Type declaration

```
(): void
```

Returns *void*

TEasingFn

`T TEasingFn: (time: number) => number`

Defined in src/Core/Animations/EasingFunctions.ts

An easing function used in animations through SciChart. See easing for a list of values

Type declaration

```
(time: number): number
```

Parameters

- **time:** *number*

Returns *number*

TEffectDefinition

`T TEffectDefinition: { options?: IGlowEffectOptions; type: Glow } | { options?: IShadowEffectOptions; type: Shadow }`

Defined in src/Builder/buildSeries.ts

Definition of a shader effect, comprising a EShaderEffectType and the relevant options

TElement

`T TElement: { arg: NumberArray | any[]; name: string }`

Defined in src/Core/Guard.ts

Type used by Guard.arraysSameLengthArr

Type declaration

- **arg**: *NumberArray* | *any*[]
- **name**: *string*

TFilterDefinition

```
T TFilterDefinition: { options?: IXyFilterOptions; type: XyLinearTrend } | { options?:
  IXyMovingAverageFilterOptions; type: XyMovingAverage } | { options:
  IXyRatioFilterOptions; type: XyRatio } | { options?: IXyScaleOffsetFilterOptions; type:
  XyScaleOffset } | { options?: IXyyScaleOffsetFilterOptions; type: XyyScaleOffset } | {
  options?: IXyzScaleOffsetFilterOptions; type: XyzScaleOffset } | { options?:
  IHlcScaleOffsetFilterOptions; type: HlcScaleOffset } | { options?:
  IOhlcScaleOffsetFilterOptions; type: OhlcScaleOffset }
```

Defined in src/Builder/buildDataSeries.ts

Definition of a data filter

TFormatLabelFn

```
T TFormatLabelFn: (dataValue: number) => string
```

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelProvider.ts

Formats a data-value into a string for display

param The data-value (e.g. a numeric value)

Type declaration

```
(dataValue: number): string
```

Parameters

- **dataValue:** *number*

Returns *string*

TGradientStop

```
T TGradientStop: { color: string; offset: number }
```

Defined in src/types/TGradientStop.ts

A Gradient stop applied to gradients on lines, fills thoughout SciChart

Type declaration

- **color**: *string*

  The gradient stop color as an HTML color code
- **offset**: *number*

  The gradient offset. Allowable values are in the range 0 to 1

TGridLineStyle

```
T TGridLineStyle: { color?: string; strokeDashArray?: number[]; strokeThickness?: number
  }
```

Defined in src/Charting/Visuals/Axis/AxisCore.ts

A type class to contain information about gridline styles

remarks A grid line is the X Y axis grid inside the chart

- Set the color as an HTML Color code to define the color
- Set the strokeThickness to change the thickness of the grid line
- Set the strokeDashArray to define dash pattern, e.g. [2,2] will have a 2-pixel long dash every 2 pixels

Type declaration

- Optional **color**?: *string*
- Optional **strokeDashArray**?: *number*[]
- Optional **strokeThickness**?: *number*

THeatmapLegend

```
T THeatmapLegend: { heatmapLegend: HeatmapLegend; wasmContext: TSciChart }
```

Defined in src/Charting/Visuals/HeatmapLegend.ts

Type declaration

- **heatmapLegend**: *HeatmapLegend*
- **wasmContext**: *TSciChart*

THlcSeriesData

```
T THlcSeriesData: { closeDataId?: number | string; highDataId?: number | string;
  lowDataId?: number | string; xDataId?: number | string } & IHlcDataSeriesOptions & {
  filter?: TFilterDefinition }
```

Defined in src/Builder/buildDataSeries.ts

Definition of Open, High, Low, Close data

THoverCallback

```
T THoverCallback<TEntityType>: (args: IHoverCallbackArgs<TEntityType>) => void
```

Defined in src/Charting/ChartModifiers/PointerEventsMediatorModifier.ts

Type parameters

- **TEntityType**: *IHoverable*

Type declaration

```
(args: IHoverCallbackArgs<TEntityType>): void
```

Parameters

- **args:** *IHoverCallbackArgs*<*TEntityType*>

Returns *void*

THoveredChangedCallback

```
T THoveredChangedCallback: (args: HoveredChangedArgs) => void
```

Defined in src/Charting/ChartModifiers/SeriesSelectionModifier.ts

The type of the callback function

Type declaration

```
(args: HoveredChangedArgs): void
```

Parameters

- **args:** *HoveredChangedArgs*

Returns *void*

TInnerLayoutStrategyDefinition

```
T TInnerLayoutStrategyDefinition: { customType?: string; options?:
  IInnerAxisLayoutStrategyOptions; type: ELayoutStrategyType }
```

Defined in src/Charting/LayoutManager/LayoutManager.ts

Type declaration

- Optional **customType**?: *string*
- Optional **options**?: *IInnerAxisLayoutStrategyOptions*
- **type**: *ELayoutStrategyType*

TLabelProviderDefinition

```
T TLabelProviderDefinition: { options?: ILabelOptions; type: Numeric } | { options?:
  ILabelOptions; type: Date } | { options?: ILabelOptions; type: Logarithmic } | {
  options?: {}; type: SmartDate } | { options?: ITextLabelOptions; type: Text } | {
  options?: ILabelOptions; type: Pie }
```

Defined in src/Builder/buildAxis.ts

Definition of a LabelProviderBase2D, comprising a ELabelProviderType and the relevant options

TLayoutAxisPartsWithStrategyFunc

```
T TLayoutAxisPartsWithStrategyFunc: (axisRendererWidth: number, axisRendererHeight:
  number, axisTitleRendererWidth: number, axisTitleRendererHeight: number, axisRect:
  Rect, border: TBorder) => TAxisViewRects
```

Defined in src/Charting/LayoutManager/AxisLayoutHelpers.ts

Type declaration

```
(axisRendererWidth: number, axisRendererHeight: number, axisTitleRendererWidth:
  number, axisTitleRendererHeight: number, axisRect: Rect, border: TBorder):
  TAxisViewRects
```

Parameters

- **axisRendererWidth:** *number*
- **axisRendererHeight:** *number*
- **axisTitleRendererWidth:** *number*
- **axisTitleRendererHeight:** *number*
- **axisRect:** *Rect*
- **border:** *TBorder*

Returns *TAxisViewRects*

TLayoutManagerDefinition

```
T TLayoutManagerDefinition: { options?: ILayoutManagerOptions; type: Default } | {
  options?: ICentralAxesLayoutManagerOptions; type: CentralAxes } | { options?:
  ILayoutManagerOptions; type: Synchronised }
```

Defined in src/Builder/buildSurface.ts

TLegendItem

```
T TLegendItem: { checked: boolean; color: string; gradient?: GradientParams; id: string;
  name: string; showMarker?: boolean }
```

Defined in src/Charting/Visuals/Legend/SciChartLegendBase.ts

Type declaration

- **checked**: *boolean*
- **color**: *string*
- Optional **gradient**?: *GradientParams*
- **id**: *string*
- **name**: *string*
- Optional **showMarker**?: *boolean*

TLinearColorMap

```
T TLinearColorMap: { GradientStops: TGradientStop[]; Maximum: number; Minimum: number;
  Mode: EColorMapMode }
```

Defined in src/types/TLinearColorMap.ts

Type declaration

- **GradientStops**: *TGradientStop*[]
- **Maximum**: *number*
- **Minimum**: *number*
- **Mode**: *EColorMapMode*

TModifierDefinition

```
T TModifierDefinition: { options?: ICursorModifierOptions; type: Cursor } | { options?:
  IDataPointSelectionModifierOptions; type: DataPointSelection } | { options?:
  ILegendModifierOptions; type: Legend } | { options?: IMouseWheelZoomModifierOptions;
  type: MouseWheelZoom } | { options?: IPinchZoomModifierOptions; type: PinchZoom } | {
  options?: IRolloverModifierOptions; type: Rollover } | { options?:
  IVerticalSliceOptions; type: VerticalSlice } | { options?:
  IRubberBandXyZoomModifierOptions; type: RubberBandXYZoom } | { options?:
  ISeriesSelectionModifierOptions; type: SeriesSelection } | { options?:
  IAnnotationHoverModifierOptions; type: AnnotationHover } | { options?:
  IXAxisDragModifierOptions; type: XAxisDrag } | { options?: IYAxisDragModifierOptions;
  type: YAxisDrag } | { options?: IZoomExtentsModifierOptions; type: ZoomExtents } | {
  options?: IZoomPanModifierOptions; type: ZoomPan } | { options?:
  IRangeSelectionModifierOptions; type: OverviewRangeSelection } | { customType?: string;
  options?: IChartModifierBaseOptions; type: Custom }
```

Defined in src/Builder/buildModifiers.ts

Definition of a 2d chart modifier, comprising a EChart2DModifierType and the relevant options

TModifierKeys

```
T TModifierKeys: { altKey: boolean; ctrlKey: boolean; shiftKey: boolean }
```

Defined in src/Charting/ChartModifiers/DataPointSelectionModifier.ts

Type to store whether modifier keys (Control, Shift, Alt) are pressed or not

Type declaration

- **altKey**: *boolean*
- **ctrlKey**: *boolean*
- **shiftKey**: *boolean*

TModifierMouseArgsParams

```
T TModifierMouseArgsParams: { altKey?: boolean; button?: number; ctrlKey?: boolean;
  isActiveSubChartEvent?: boolean; isMaster?: boolean; modifierGroup?: string;
  mousePoint?: Point; mouseWheelDelta?: number; nativeEvent?: MouseEvent; pointerId?:
  number; pointerType?: string; shiftKey?: boolean; target?: Element }
```

Defined in src/Charting/ChartModifiers/ModifierMouseArgs.ts

Type declaration

- Optional **altKey**?: *boolean*
- Optional **button**?: *number*
- Optional **ctrlKey**?: *boolean*
- Optional **isActiveSubChartEvent**?: *boolean*
- Optional **isMaster**?: *boolean*
- Optional **modifierGroup**?: *string*
- Optional **mousePoint**?: *Point*
- Optional **mouseWheelDelta**?: *number*
- Optional **nativeEvent**?: *MouseEvent*
- Optional **pointerId**?: *number*
- Optional **pointerType**?: *string*
- Optional **shiftKey**?: *boolean*
- Optional **target**?: *Element*

TNativeCache

```
T TNativeCache: { keyCache: Map<string, FontKey>; rect: SCRTRectVertex; textBounds:
  TSRTextBounds; vecColorVertex: VectorColorVertex; vecRects: VectorRectVertex; vector4:
  TSRVector4; vertex: SCRTColorVertex }
```

Defined in src/Charting/Visuals/Helpers/NativeObject.ts

Type declaration

- **keyCache**: *Map<string, FontKey>*
- **rect**: *SCRTRectVertex*
- **textBounds**: *TSRTextBounds*
- **vecColorVertex**: *VectorColorVertex*
- **vecRects**: *VectorRectVertex*
- **vector4**: *TSRVector4*
- **vertex**: *SCRTColorVertex*

TNativeTextStyle

```
T TNativeTextStyle: TTextStyleBase & { useNativeText: true }
```

Defined in src/types/TextStyle.ts

Defines properties of text rendered via Native Text API

TOhlcSeriesData

```
T TOhlcSeriesData: { closeDataId?: number | string; highDataId?: number | string;
  lowDataId?: number | string; openDataId?: number | string; xDataId?: number | string }
  & IOhlcDataSeriesOptions & { filter?: TFilterDefinition }
```

Defined in src/Builder/buildDataSeries.ts

Definition of Open, High, Low, Close data

TOuterLayoutStrategyDefinition

```
T TOuterLayoutStrategyDefinition: { customType?: string; options?: any; type: ELayoutStrategyType }
```

Defined in src/Charting/LayoutManager/LayoutManager.ts

Type declaration

- Optional **customType**?: *string*
- Optional **options**?: *any*
- **type**: *ELayoutStrategyType*

TPaletteProviderDefinition

```
T TPaletteProviderDefinition: { options: GradientParams; type: Gradient } | { options: ISelectedPointOptions; type: DataPointSelection } | { customType: string; options?: any; type: Custom }
```

Defined in src/Builder/buildSeries.ts

Definition of a palette provider, comprising a EPaletteProviderType and the relevant options

TPalettingState

```
T TPalettingState: { gradientPaletting: boolean; lastCount?: number; lastResamplingHash?: number; lastStartIndex?: number; originalBrushColor?: number; originalPenColor?: number; originalPenGradient?: boolean; paletteStartIndex?: number; paletteTextureCache?: PaletteCache; palettedColors: UIntVector; palettedColorsHashCode: number; requiresUpdate: boolean }
```

Defined in src/Charting/Visuals/RenderableSeries/DrawingProviders/BaseSeriesDrawingProvider.ts

Type declaration

- **gradientPaletting**: *boolean*
- Optional **lastCount**?: *number*
- Optional **lastResamplingHash**?: *number*
- Optional **lastStartIndex**?: *number*
- Optional **originalBrushColor**?: *number*
- Optional **originalPenColor**?: *number*
- Optional **originalPenGradient**?: *boolean*
- Optional **paletteStartIndex**?: *number*
- Optional **paletteTextureCache**?: *PaletteCache*
- **palettedColors**: *UIntVector*
- **palettedColorsHashCode**: *number*
- **requiresUpdate**: *boolean*

TPerformanceDetailType

```
T TPerformanceDetailType: { contextId?: string; relatedId?: string }
```

Defined in src/utils/perfomance.ts

Type declaration

- Optional **contextId**?: *string*
- Optional **relatedId**?: *string*

TPointMarkerArgb

```
T TPointMarkerArgb: { fill: number; stroke: number }
```

Defined in src/Charting/Model/IPaletteProvider.ts

Type declaration

- **fill**: *number*
- **stroke**: *number*

TPointMarkerDefinition

```
T TPointMarkerDefinition: { options?: IPointMarkerOptions; type: Cross } | { options?:
  IPointMarkerOptions; type: Ellipse } | { options?: ISpritePointMarkerOptions; type:
  Sprite } | { options?: IPointMarkerOptions; type: Square } | { options?:
  IPointMarkerOptions; type: Triangle } | { options?: IPointMarkerOptions; type: X } | {
  customType?: string; options?: IPointMarkerOptions; type: Custom }
```

Defined in src/Builder/buildSeries.ts

Definition of a pointmarker, comprising a EPointMarkerType and the relevant options

TPositionPoperties

```
T TPositionPoperties: { coordPropertyName: TCoord; shiftPropertyName: TShift;
  sizePropertyName: TSize }
```

Defined in src/utils/tooltip.ts

Type declaration

- **coordPropertyName**: *TCoord*
- **shiftPropertyName**: *TShift*
- **sizePropertyName**: *TSize*

TPriceBar

```
T TPriceBar: {}
```

Defined in src/utils/randomPricesDataSource.ts

Type declaration

TRenderLayers

```
T TRenderLayers: {}
```

Defined in src/Charting/Drawing/WebGlRenderContext2D.ts

Defines Render Layers 1,2,3,4 which are used to compose rendering

Type declaration

TRolloverLegendSvgTemplate

```
T TRolloverLegendSvgTemplate: (seriesInfos: SeriesInfo[], svgAnnotation:
  RolloverLegendSvgAnnotation) => string
```

Defined in src/Charting/ChartModifiers/RolloverModifier.ts

Type declaration

```
(seriesInfos: SeriesInfo[], svgAnnotation: RolloverLegendSvgAnnotation): string
```

Parameters

- **seriesInfos:** *SeriesInfo*[]
- **svgAnnotation:** *RolloverLegendSvgAnnotation*

Returns *string*

TRolloverTooltipDataTemplate

```
T TRolloverTooltipDataTemplate: (seriesInfo: SeriesInfo, tooltipTitle: string,
tooltipLabelX: string, tooltipLabelY: string) => string[]
```

Defined in src/Charting/ChartModifiers/RolloverModifier.ts

Type declaration

```
(seriesInfo: SeriesInfo, tooltipTitle: string, tooltipLabelX: string,
tooltipLabelY: string): string[]
```

Parameters

- **seriesInfo:** *SeriesInfo*
- **tooltipTitle:** *string*
- **tooltipLabelX:** *string*
- **tooltipLabelY:** *string*

Returns *string*[]

TRolloverTooltipSvgTemplate

```
T TRolloverTooltipSvgTemplate: (id: string, seriesInfo: SeriesInfo, rolloverTooltip:
RolloverTooltipSvgAnnotation) => string
```

Defined in src/Charting/Visuals/RenderableSeries/RolloverModifier/RolloverModifierRenderableSeriesProps.ts

Type declaration

```
(id: string, seriesInfo: SeriesInfo, rolloverTooltip:
RolloverTooltipSvgAnnotation): string
```

Parameters

- **id:** *string*
- **seriesInfo:** *SeriesInfo*
- **rolloverTooltip:** *RolloverTooltipSvgAnnotation*

Returns *string*

TSciChart

```
T TSciChart: TWasmContext
```

Defined in src/Charting/Visuals/SciChartSurface.ts

TSciChart3D

```
T TSciChart3D: TWasmContext
```

Defined in src/Charting3D/Visuals/SciChart3DSurface.ts

TSciChartConfig

```
T TSciChartConfig: { dataUrl?: string; testWasm?: TSciChart | TSciChart3D; wasmUrl?: string }
```

Defined in src/Charting/Visuals/SciChartSurfaceBase.ts

Type declaration

- Optional **dataUrl**?: *string*
- Optional **testWasm**?: *TSciChart* | *TSciChart3D*

  Internal testing use only
- Optional **wasmUrl**?: *string*

TSciChartDestination

```
T TSciChartDestination: { canvasElementId: string; height: number; sciChartSurface: ISciChartSurfaceBase; width: number }
```

Defined in src/Charting/Visuals/SciChartSurfaceBase.ts

Type declaration

- **canvasElementId**: *string*
- **height**: *number*
- **sciChartSurface**: *ISciChartSurfaceBase*
- **width**: *number*

TSciChartSurfaceCanvases

```
T TSciChartSurfaceCanvases: { aspect?: number; disableAspect?: boolean; domBackgroundSvgContainer?: SVGSVGElement; domCanvas2D?: HTMLCanvasElement; domCanvasWebGL?: HTMLCanvasElement; domChartRoot?: HTMLDivElement; domDivContainer?: HTMLDivElement; domSeriesBackground?: HTMLDivElement; domSvgAdornerLayer?: SVGSVGElement; domSvgContainer?: SVGSVGElement }
```

Defined in src/types/TSciChartSurfaceCanvases.ts

Type declaration

- Optional **aspect**?: *number*
- Optional **disableAspect**?: *boolean*
- Optional **domBackgroundSvgContainer**?: *SVGSVGElement*
- Optional **domCanvas2D**?: *HTMLCanvasElement*
- Optional **domCanvasWebGL**?: *HTMLCanvasElement*
- Optional **domChartRoot**?: *HTMLDivElement*
- Optional **domDivContainer**?: *HTMLDivElement*
- Optional **domSeriesBackground**?: *HTMLDivElement*
- Optional **domSvgAdornerLayer**?: *SVGSVGElement*
- Optional **domSvgContainer**?: *SVGSVGElement*

TSelectionChangedCallback

```
T TSelectionChangedCallback: (args: SelectionChangedArgs) => void
```

Defined in src/Charting/ChartModifiers/SeriesSelectionModifier.ts

The type of the callback function

Type declaration

```
(args: SelectionChangedArgs): void
```

Parameters

- **args:** *SelectionChangedArgs*

Returns *void*

TSeriesDataDefinition

```
T TSeriesDataDefinition: TXySeriesData | TXyySeriesData | TXyzSeriesData | THlcSeriesData
  | TOhlcSeriesData | TXyTextSeriesData | IUniformHeatmapSeriesOptions |
  INonUniformHeatmapSeriesOptions
```

Defined in src/Builder/buildDataSeries.ts

Definition of series data, which can take various shapes

TSeriesDefinition

```
T TSeriesDefinition: { options?: IBandRenderableSeriesOptions; type: BandSeries;
  xyyData?: TXyySeriesData } | { options?: IBubbleRenderableSeriesOptions; type:
  BubbleSeries; xyzData?: TXyzSeriesData } | { options?: IColumnRenderableSeriesOptions;
  type: ColumnSeries; xyData?: TXySeriesData } | { options?: IImpulseRenderableSeries;
  type: ImpulseSeries; xyData?: TXySeriesData } | { ohlcData?: TOhlcSeriesData; options?:
  ICandlestickRenderableSeriesOptions; type: CandlestickSeries } | { options?:
  IFastLineRenderableSeriesOptions; type: LineSeries; xyData?: TXySeriesData } | {
  options?: IMountainRenderableSeriesOptions; type: MountainSeries; xyData?:
  TXySeriesData } | { hlcData?: THlcSeriesData; options?:
  IFastErrorBarsRenderableSeriesOptions; type: ErrorBarsSeries } | { ohlcData?:
  TOhlcSeriesData; options?: IOhlcRenderableSeriesOptions; type: OhlcSeries } | {
  options?: IXyScatterRenderableSeriesOptions; type: ScatterSeries; xyData?:
  TXySeriesData } | { options?: IXyTextDataSeriesOptions; type: TextSeries; xyTextData?:
  TXyTextSeriesData } | { options?: ISplineBandRenderableSeriesOptions; type:
  SplineBandSeries; xyyData?: TXyySeriesData } | { options?:
  ISplineLineRenderableSeriesOptions; type: SplineLineSeries; xyData?: TXySeriesData } |
  { options?: ISplineMountainRenderableSeriesOptions; type: SplineMountainSeries;
  xyData?: TXySeriesData } | { options?: ISmoothStackedMountainRenderableSeriesOptions;
  type: SmoothStackedMountainSeries; xyData?: TXySeriesData } | { heatmapData?:
  IUniformHeatmapSeriesOptions; options?: IHeatmapRenderableSeriesOptions; type:
  UniformHeatmapSeries } | { heatmapData?: INonUniformHeatmapSeriesOptions; options?:
  INonUniformHeatmapRenderableSeriesOptions; type: NonUniformHeatmapSeries } | {
  heatmapData?: IUniformHeatmapSeriesOptions; options?: IContoursRenderableSeriesOptions;
  type: UniformContoursSeries } | { options?: IStackedColumnRenderableSeriesOptions;
  type: StackedColumnSeries; xyData?: TXySeriesData } | { options?:
  IStackedMountainRenderableSeriesOptions; type: StackedMountainSeries; xyData?:
  TXySeriesData } | { options?: IStackedColumnCollectionOptions; series?:
  TSeriesDefinition[]; type: StackedColumnCollection } | { options?:
  IBaseStackedCollectionOptions; series?: TSeriesDefinition[]; type:
  StackedMountainCollection } | { customType?: string; options?:
  IBaseRenderableSeriesOptions; type: Custom }
```

Defined in src/Builder/buildSeries.ts

Definition of a renderable series, comprising a ESeriesType, the relevant options, and an optional data object whose type depends on the series type

TSeriesHoverChangedCallback

```
T TSeriesHoverChangedCallback: (sourceSeries: IRenderableSeries, isHovered: boolean) =>
  void
```

Defined in src/Charting/Visuals/RenderableSeries/IBaseRenderableSeriesOptions.ts

The type of the callback

Type declaration

```
(sourceSeries: IRenderableSeries, isHovered: boolean): void
```

Parameters

- **sourceSeries:** *IRenderableSeries*
- **isHovered:** *boolean*

Returns *void*

TSeriesRenderPassInfo

```
T TSeriesRenderPassInfo: { indicesRange: NumberRange; pointSeries: IPointSeries;
  renderableSeries: IRenderableSeries; resamplingHash: number }
```

Defined in src/Charting/Services/RenderPassInfo.ts

Type declaration

- **indicesRange**: *NumberRange*
- **pointSeries**: *IPointSeries*
- **renderableSeries**: *IRenderableSeries*
- **resamplingHash**: *number*

TSeriesSelectionChangedCallback

```
T TSeriesSelectionChangedCallback: (sourceSeries: IRenderableSeries, isSelected: boolean)
  => void
```

Defined in src/Charting/Visuals/RenderableSeries/IBaseRenderableSeriesOptions.ts

The type of the callback

Type declaration

```
(sourceSeries: IRenderableSeries, isSelected: boolean): void
```

Parameters

- **sourceSeries:** *IRenderableSeries*
- **isSelected:** *boolean*

Returns *void*

TSeriesVisibleChangedCallback

```
T TSeriesVisibleChangedCallback: (sourceSeries: IRenderableSeries, isVisible: boolean) =>
  void
```

Defined in src/Charting/Visuals/RenderableSeries/IBaseRenderableSeriesOptions.ts

The type of the callback

Type declaration

```
(sourceSeries: IRenderableSeries, isVisible: boolean): void
```

Parameters

- **sourceSeries:** *IRenderableSeries*
- **isVisible:** *boolean*

Returns *void*

TSharedDataDefinition

```
T TSharedDataDefinition: Record<number | string, number[]>
```

Defined in src/Builder/buildDataSeries.ts

Shared data that can be used in ISciChart2DDefinition or directly in chartBuilder.buildSeries or chartBuilder.buildDataSeries

TShift

```
T TShift: xCoordShift | yCoordShift
```

Defined in src/utils/tooltip.ts

TSize

```
T TSize: width | height
```

Defined in src/utils/tooltip.ts

TStackedAxisLength

```
T TStackedAxisLength: number | string
```

Defined in src/types/TStackedAxisLength.ts

Length defined as an absolute number or percentage

TSurfaceDefinition

```
T TSurfaceDefinition: ISciChart2DDefinition | { options?: ISciChart2DDefinition; type?:
  Default2D } | { options?: ISciChartPieDefinition; type?: Pie2D }
```

Defined in src/Builder/chartBuilder.ts

TTargetsSelector

```
T TTargetsSelector<TEntityType>: (modifier: PointerEventsMediatorModifier<TEntityType>)
  => TEntityType[]
```

Defined in src/Charting/ChartModifiers/PointerEventsMediatorModifier.ts

Type parameters

- **TEntityType**: *IHoverable*

Type declaration

```
(modifier: PointerEventsMediatorModifier<TEntityType>): TEntityType[]
```

Parameters

- **modifier:** *PointerEventsMediatorModifier*<*TEntityType*>

Returns *TEntityType*[]

TTextStyle

```
T TTextStyle: { alignment?: ELabelAlignment; color?: string; fontFamily?: string;
  fontSize?: number; fontStyle?: string; fontWeight?: string; multilineAlignment?:
  EMultiLineAlignment; padding?: Thickness }
```

Defined in src/Charting/Visuals/Axis/AxisCore.ts

A type class to contain information about Axis Label text styles

remarks

- Set the fontFamily as a string to set the font
- Set the fontSize as you would in HTML/CSS
- Set the fontWeight and fontStyle as you would in HTML/CSS
- Set the color as an HTML Color code to define the color

Type declaration

- Optional **alignment**?: *ELabelAlignment*

  Horizontal label alignment for vertical axes. Default Auto.

  privateremarks This property should only be used for axis labels. So the current type definition should be changed in future versions, specifically this property may be renamed or removed. In other cases, e.g. for multiline text alignment use TTextStyle.multilineAlignment

- Optional **color**?: *string*
- Optional **fontFamily**?: *string*
- Optional **fontSize**?: *number*
- Optional **fontStyle**?: *string*
- Optional **fontWeight**?: *string*
- Optional **multilineAlignment**?: *EMultiLineAlignment*

  Horizontal text alignment for multiline text.

- Optional **padding**?: *Thickness*

  Padding is left 4, right 4, top 2, bottom 0 by default. This is because there is natural space below the text baseline. If you are using text labels rather than just numbers, or when using native text, you may want to increase the bottom padding.

TTextStyleBase

```
T TTextStyleBase: { color?: string; fontFamily?: string; fontSize?: number; lineSpacing?: number; padding?: Thickness }
```

Defined in src/types/TextStyle.ts

Defines common properties of text to render

Type declaration

- Optional **color**?: *string*
- Optional **fontFamily**?: *string*
- Optional **fontSize**?: *number*
- Optional **lineSpacing**?: *number*

  Line spacing to use if text is wrapped. This is a multiple of the line height and defaults to 1.1

- Optional **padding**?: *Thickness*

TTextStyleBase3D

```
T TTextStyleBase3D: { color?: string; fontFamily?: string; fontSize?: number }
```

Defined in src/types/TextStyle3D.ts

Type declaration

- Optional **color**?: *string*
- Optional **fontFamily**?: *string*
- Optional **fontSize**?: *number*

TTextureTextStyle

```
T TTextureTextStyle: TTextStyleBase & { fontStyle?: string; fontWeight?: string; useNativeText?: false }
```

Defined in src/types/TextStyle.ts

Defines properties of text rendered as a texture

TTickLineStyle

```
T TTickLineStyle: { color?: string; strokeThickness?: number; tickSize?: number }
```

Defined in src/Charting/Visuals/Axis/AxisCore.ts

A type class to contain information about Tick line styles

remarks A tick line is the small 3 pixel line outside the axis.

- Set the tickSize to define the size of this tick in pixels.
- Set the color as an HTML Color code to define the color
- Set the strokeThickness to change the thickness of the tick line

Type declaration

- Optional **color**?: *string*
- Optional **strokeThickness**?: *number*
- Optional **tickSize**?: *number*

TTickObject

```
T TTickObject: { majorTickCoords: number[]; majorTickLabels: string[]; majorTicks:
number[]; minorTickCoords: number[]; minorTicks: number[] }
```

Defined in src/Charting/Visuals/Axis/AxisBase2D.ts

A type which contains info about major, minor tick coordinates, labels and values. Used when drawing the axis gridlines

Type declaration

- **majorTickCoords**: *number*[]

  The major tick coordinates
- **majorTickLabels**: *string*[]

  The major tick label strings
- **majorTicks**: *number*[]

  The major tick numeric values
- **minorTickCoords**: *number*[]

  The minor tick coordinates
- **minorTicks**: *number*[]

  The minor tick numeric values

TTooltip3DDataTemplate

```
T TTooltip3DDataTemplate: (seriesInfo: SeriesInfo3D, svgAnnotation:
TooltipSvgAnnotation3D) => string[]
```

Defined in src/Charting3D/Visuals/Annotations/TooltipSvgAnnotation3D.ts

Type declaration

```
(seriesInfo: SeriesInfo3D, svgAnnotation: TooltipSvgAnnotation3D): string[]
```

Parameters

- **seriesInfo:** *SeriesInfo3D*
- **svgAnnotation:** *TooltipSvgAnnotation3D*

Returns *string*[]

TTooltip3DSvgTemplate

T TTooltip3DSvgTemplate: (seriesInfo: *SeriesInfo3D*, svgAnnotation: *TooltipSvgAnnotation3D*) => *string*

Defined in src/Charting3D/Visuals/Annotations/TooltipSvgAnnotation3D.ts

Type declaration

(seriesInfo: *SeriesInfo3D*, svgAnnotation: *TooltipSvgAnnotation3D*): *string*

Parameters

- **seriesInfo:** *SeriesInfo3D*
- **svgAnnotation:** *TooltipSvgAnnotation3D*

Returns *string*

TWebAssemblyChart

T TWebAssemblyChart: { sciChartSurface: *SciChartSurface*; wasmContext: *TSciChart* }

Defined in src/Charting/Visuals/SciChartSurface.ts

Type declaration

- **sciChartSurface**: *SciChartSurface*
- **wasmContext**: *TSciChart*

TWebAssemblyChart3D

T TWebAssemblyChart3D: { sciChart3DSurface: *SciChart3DSurface*; wasmContext: *TSciChart3D* }

Defined in src/Charting3D/Visuals/SciChart3DSurface.ts

Type declaration

- **sciChart3DSurface**: *SciChart3DSurface*
- **wasmContext**: *TSciChart3D*

TXySeriesData

T TXySeriesData: { xDataId?: *number* | *string*; yDataId?: *number* | *string* } & *IXyDataSeriesOptions* & { filter?: *TFilterDefinition* }

Defined in src/Builder/buildDataSeries.ts

Definition of XY data

TXyTextSeriesData

T TXyTextSeriesData: { xDataId?: *number* | *string*; yDataId?: *number* | *string* } & *IXyTextDataSeriesOptions* & { filter?: *TFilterDefinition* }

Defined in src/Builder/buildDataSeries.ts

Definition of XYText data

TXyySeriesData

T TXyySeriesData: { xDataId?: *number* | *string*; y1DataId?: *number* | *string*; yDataId?: *number* | *string* } & *IXyyDataSeriesOptions* & { filter?: *TFilterDefinition* }

Defined in src/Builder/buildDataSeries.ts

Definition of XYY data

TXyzSeriesData

```
T TXyzSeriesData: { xDataId?: number | string; yDataId?: number | string; zDataId?: number | string } & IXyzDataSeriesOptions & { filter?: TFilterDefinition }
```

Defined in src/Builder/buildDataSeries.ts

Definition of XYZ data

TZoomExtentsCallback

```
T TZoomExtentsCallback: (sciChartSurface: SciChartSurface) => boolean
```

Defined in src/Charting/ChartModifiers/ZoomExtentsModifier.ts

A function to execute when zoomExtents is activated. If this exists and returns false, the builtin behaviour is ignored;

Type declaration

```
(sciChartSurface: SciChartSurface): boolean
```

Parameters

- **sciChartSurface:** *SciChartSurface*

Returns *boolean*

TfilterFunction

```
T TfilterFunction: (index: number, y: number) => number
```

Defined in src/Charting/Model/Filters/XyCustomFilter.ts

Type declaration

```
(index: number, y: number): number
```

Parameters

- **index:** *number*
- **y:** *number*

Returns *number*

Variables

Const AUTO_COLOR

```
AUTO_COLOR: "auto" = "auto"
```

Defined in src/Charting/Themes/IThemeProvider.ts

Const DebugForDpi

```
DebugForDpi: boolean = false
```

Defined in src/Charting/Visuals/SciChartSurfaceBase.ts

Const FIFTY_DAYS

● FIFTY_DAYS: *number* = 60 * 60 * 24 * 50

Defined in src/Charting/Visuals/Axis/LabelProvider/SmartDateLabelProvider.ts

Const FIVE_DAYS

● FIVE_DAYS: *number* = 60 * 60 * 24 * 5

Defined in src/Charting/Visuals/Axis/LabelProvider/SmartDateLabelProvider.ts

Const MIN_LOG_AXIS_VALUE

● MIN_LOG_AXIS_VALUE: *1e-10* = 1e-10

Defined in src/Charting/Visuals/Axis/LogarithmicAxis.ts

Const MIN_SERIES_AREA_SIZE

● MIN_SERIES_AREA_SIZE: *10* = 10

Defined in src/Charting/LayoutManager/LayoutManager.ts

Const ONE_HOUR

● ONE_HOUR: *number* = 60 * 60

Defined in src/Charting/Visuals/Axis/LabelProvider/SmartDateLabelProvider.ts

Const TEN_SECONDS

● TEN_SECONDS: *10* = 10

Defined in src/Charting/Visuals/Axis/LabelProvider/SmartDateLabelProvider.ts

Const buildStamp

● buildStamp: *"2024-10-02T00:00:00"* = "2024-10-02T00:00:00"

Defined in src/Core/BuildStamp.ts

Let cleanupWasmContext

● cleanupWasmContext: () => *void*

Defined in src/Charting/Visuals/createMaster.ts
Defined in src/Charting3D/Visuals/createMaster3d.ts

Type declaration

▸ (): *void*

Returns *void*

Const constructorMap

● constructorMap: *Map*<*string*, { func: *Function*; wasm?: *boolean* }> = new Map<string,{ wasm?: boolean; func: Function }>()

Defined in src/Builder/classFactory.ts

Const defaultSelectionAnnotationSvgString

- defaultSelectionAnnotationSvgString: *"<svg width="50" height="50" preserveAspectRatio="none" xmlns="http://www.w3.org/2000/svg"><rect width="100%" height="100%" style="fill:transparent"></rect></svg>"* = `<svg width="50" height="50" preserveAspectRatio="none" xmlns="http://www.w3.org/2000/svg"><rect width="100%" height="100%" style="fill:transparent"></rect></svg>`

Defined in src/Charting/ChartModifiers/OverviewRangeSelectionModifier.ts

Const defaultUnSelectedAnnotationSvgString

- defaultUnSelectedAnnotationSvgString: *"<svg width="50" height="50" preserveAspectRatio="none" xmlns="http://www.w3.org/2000/svg"><rect width="100%" height="100%" style="fill:black"></rect></svg>"* = `<svg width="50" height="50" preserveAspectRatio="none" xmlns="http://www.w3.org/2000/svg"><rect width="100%" height="100%" style="fill:black"></rect></svg>`

Defined in src/Charting/ChartModifiers/OverviewRangeSelectionModifier.ts

Let devCount

- devCount: *number* = 1

Defined in src/Charting/Visuals/licenseManager2dState.ts

Let hasSent

- hasSent: *boolean* = false

Defined in src/Core/Telemetry.ts

Let isDev

- isDev: *boolean* = false

Defined in src/Charting/Visuals/licenseManager2dState.ts

Let isStorageAvailable

- isStorageAvailable: *boolean* = undefined

Defined in src/Core/storage/localStorageApi.ts

Const labelCacheByTextAndStyle

- labelCacheByTextAndStyle: *Map<string, LabelInfo>* = new Map<string, LabelInfo>()

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelCache.ts

Let lastStyleId

- lastStyleId: *number* = 0

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelCache.ts

Const libraryVersion

- libraryVersion: *"3.5.0000"* = "3.5.0000"

Defined in src/Core/BuildStamp.ts

Let licenseType

- licenseType: *LicenseType* = LicenseType.NoLicense

Defined in src/Charting/Visuals/licenseManager2dState.ts

Const loaderCss

- loaderCss: *".scichart_loader {display: inline-block;position: relative;width: 80px;height: 80px;top: 50%;transform: translateY(-50%);}.scichart_loader div {animation: scichart_loader 1.2s cubic-bezier(0.5, 0, 0.5, 1) infinite;transform-origin: 40px 40px;}.scichart_loader div span {display: block;position: absolute;width: 7px;height: 7px;border-radius: 50%;margin: -4px 0 0 -4px;}.scichart_loader div:nth-child(1) {animation-delay: -0.036s;}.scichart_loader div:nth-child(1) span {top: 63px;left: 63px;}.scichart_loader div:nth-child(2) {animation-delay: -0.072s;}.scichart_loader div:nth-child(2) span {top: 68px;left: 56px;}.scichart_loader div:nth-child(3) {animation-delay: -0.108s;}.scichart_loader div:nth-child(3) span {top: 71px;left: 48px;}.scichart_loader div:nth-child(4) {animation-delay: -0.144s;}.scichart_loader div:nth-child(4) span {top: 72px;left: 40px;}.scichart_loader div:nth-child(5) {animation-delay: -0.18s;}.scichart_loader div:nth-child(5) span {top: 71px;left: 32px;}.scichart_loader div:nth-child(6) {animation-delay: -0.216s;}.scichart_loader div:nth-child(6) span {top: 68px;left: 24px;}.scichart_loader div:nth-child(7) {animation-delay: -0.252s;}.scichart_loader div:nth-child(7) span {top: 63px;left: 17px;}.scichart_loader div:nth-child(8) {animation-delay: -0.288s;}.scichart_loader div:nth-child(8) span {top: 56px;left: 12px;}@keyframes scichart_loader {0% {transform: rotate(0deg);}100% {transform: rotate(360deg);}}"* = `.scichart_loader {display: inline-block;position: relative;width: 80px;height: 80px;top: 50%;transform: translateY(-50%);}.scichart_loader div {animation: scichart_loader 1.2s cubic-bezier(0.5, 0, 0.5, 1) infinite;transform-origin: 40px 40px;}.scichart_loader div span {display: block;position: absolute;width: 7px;height: 7px;border-radius: 50%;margin: -4px 0 0 -4px;}.scichart_loader div:nth-child(1) {animation-delay: -0.036s;}.scichart_loader div:nth-child(1) span {top: 63px;left: 63px;}.scichart_loader div:nth-child(2) {animation-delay: -0.072s;}.scichart_loader div:nth-child(2) span {top: 68px;left: 56px;}.scichart_loader div:nth-child(3) {animation-delay: -0.108s;}.scichart_loader div:nth-child(3) span {top: 71px;left: 48px;}.scichart_loader div:nth-child(4) {animation-delay: -0.144s;}.scichart_loader div:nth-child(4) span {top: 72px;left: 40px;}.scichart_loader div:nth-child(5) {animation-delay: -0.18s;}.scichart_loader div:nth-child(5) span {top: 71px;left: 32px;}.scichart_loader div:nth-child(6) {animation-delay: -0.216s;}.scichart_loader div:nth-child(6) span {top: 68px;left: 24px;}.scichart_loader div:nth-child(7) {animation-delay: -0.252s;}.scichart_loader div:nth-child(7) span {top: 63px;left: 17px;}.scichart_loader div:nth-child(8) {animation-delay: -0.288s;}.scichart_loader div:nth-child(8) span {top: 56px;left: 12px;}@keyframes scichart_loader {0% {transform: rotate(0deg);}100% {transform: rotate(360deg);}}`

Defined in src/Charting/Visuals/loader.ts

Let maxSize

- maxSize: *number* = 200

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelCache.ts

Let minAge

- minAge: *number* = 1000 * 60

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelCache.ts

Const objectCache

- objectCache: *Map<string, TNativeCache>* = new Map<string, TNativeCache>()

Defined in src/Charting/Visuals/Helpers/NativeObject.ts

Let orderId

- orderId: *string* = ""

Defined in src/Charting/Visuals/licenseManager2dState.ts

Const precision

- precision: *100* = 100

Defined in src/Charting/Visuals/RenderableSeries/DrawingProviders/NonUniformHeatmapDrawingProvider.ts

Let productCode

- productCode: *string* = ""

Defined in src/Charting/Visuals/licenseManager2dState.ts

Let rect

- rect: *SCRTRectVertex*

Defined in src/Charting/Visuals/Helpers/createNativeRect.ts

Let result

- result: *boolean*

Defined in src/Core/BuildStamp.ts

Const sciChartConfig

- sciChartConfig: *TSciChartConfig*

Defined in src/Charting/Visuals/SciChartSurface.ts

Const sciChartConfig3D

- sciChartConfig3D: *TSciChartConfig*

Defined in src/Charting3D/Visuals/SciChart3DSurface.ts

Const sessionTime

- sessionTime: *number* = 60 * 60 * 1000

Defined in src/Core/Telemetry.ts

Const styleCache

- styleCache: *Record<string*, { style: *TCachedLabelStyle*; uses: *number* }>

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelCache.ts

Const superScript_map

- superScript_map: *string*[] =
  ["\u2070","\u00B9","\u00B2","\u00B3","\u2074","\u2075","\u2076","\u2077","\u2078","\u207

Defined in src/utils/number.ts

Let telemetryEnabled

telemetryEnabled: *boolean* = false

Defined in src/Charting/Visuals/licenseManager2dState.ts

Functions

Const adjustTooltipPosition

adjustTooltipPosition(width: *number*, height: *number*, svgAnnotation: *CursorTooltipSvgAnnotation*): *void*

Defined in src/Charting/ChartModifiers/CursorModifier.ts

Relocate the tooltip so that it is always within the seriesViewRect

Parameters

- **width:** *number*
- **height:** *number*
- **svgAnnotation:** *CursorTooltipSvgAnnotation*

Returns *void*

Const adjustTooltipPosition3D

adjustTooltipPosition3D(width: *number*, height: *number*, svgAnnotation: *TooltipSvgAnnotation3D*): *void*

Defined in src/Charting3D/ChartModifiers/TooltipModifier3D.ts

Relocate the tooltip so that it is always within the seriesViewRect

Parameters

- **width:** *number*
- **height:** *number*
- **svgAnnotation:** *TooltipSvgAnnotation3D*

Returns *void*

animateAny

animateAny<T>(durationMs: *number*, from: *T*, to: *T*, onAnimate: (intermediateValue: *T*) => *void*, interpolate: (start: *T*, end: *T*, interpolationFactor: *number*) => *T*, onCompleted: () => *void*, easingFn: *TEasingFn*): *AnimationToken*

Defined in src/Core/Animations/Animator.ts

deprecated Instead create an GenericAnimation and pass it to sciChartSurface.addAnimation

Type parameters

- **T**

Parameters

- **durationMs:** *number*
- **from:** *T*
- **to:** *T*
- **onAnimate:** (**intermediateValue**: *T*) => *void*

  (intermediateValue: *T*): *void*

  Parameters

  - **intermediateValue:** *T*

  Returns *void*

- **interpolate:** (**start**: *T*, **end**: *T*, **interpolationFactor**: *number*) => *T*

  ```
  (start: T, end: T, interpolationFactor: number): T
  ```

  Parameters

  - **start:** *T*
  - **end:** *T*
  - **interpolationFactor:** *number*

  Returns *T*

- **onCompleted:** () => *void*

  ```
  (): void
  ```

  Returns *void*

- **easingFn:** *TEasingFn*

Returns *AnimationToken*

Const animationUpdate

```
animationUpdate(animationFSM: SeriesAnimationFiniteStateMachine, timeElapsed: number,
beforeAnimationStart: () => void, afterAnimationComplete: () => void,
updateAnimationProperties: (progress: number, animationFSM:
SeriesAnimationFiniteStateMachine) => void): void
```

Defined in src/Charting/Visuals/RenderableSeries/Animations/animationHelpers.ts

Runs update for the animation

Parameters

- **animationFSM:** *SeriesAnimationFiniteStateMachine*
- **timeElapsed:** *number*
- **beforeAnimationStart:** () => *void*

  ```
  (): void
  ```

  Returns *void*

- **afterAnimationComplete:** () => *void*

  ```
  (): void
  ```

  Returns *void*

- **updateAnimationProperties:** (**progress**: *number*, **animationFSM**: *SeriesAnimationFiniteStateMachine*) => *void*

  ```
  (progress: number, animationFSM: SeriesAnimationFiniteStateMachine): void
  ```

  Parameters

  - **progress:** *number*
  - **animationFSM:** *SeriesAnimationFiniteStateMachine*

  Returns *void*

Returns *void*

appendRangeFifo

```
appendRangeFifo(source: any[], target: any[], fifoCapacity: number, startIndex:
number): appendRangeFifo
```

Defined in src/utils/array.ts

Helper function to append an array to a target array, treating the target as a circular buffer

Parameters

- **source:** *any*[]
- **target:** *any*[]
- **fifoCapacity:** *number*
- **startIndex:** *number*

Returns *appendRangeFifo*

applyOpacityToHtmlColor

`applyOpacityToHtmlColor(color: string, opacity: number): applyOpacityToHtmlColor`

Defined in src/utils/colorUtil.ts

Applies the given opacity to an html color code or name, returning an html color code.

Parameters

- **color:** *string*
- **opacity:** *number*

Returns *applyOpacityToHtmlColor*

Const areArraysEqual

`areArraysEqual(leftArray: number[], rightArray: number[]): boolean`

Defined in src/utils/array.ts

Parameters

- **leftArray:** *number*[]
- **rightArray:** *number*[]

Returns *boolean*

arrayRemove

`arrayRemove<T>(array: T[], item: T): T[]`

Defined in src/utils/array.ts

Type parameters

- **T**

Parameters

- **array:** *T*[]
- **item:** *T*

Returns *T*[]

Const base64Id

`base64Id(maxLength?: number): string`

Defined in src/utils/guid.ts

Generate random base64 id string. The default length is 22 which is 132-bits so almost the same as a GUID but as base64

Parameters

- Default value **maxLength:** *number* = 22

  Optional value to specify the length of the id to be generated, defaults to 22

Returns *string*

Const bezierTransform

bezierTransform(xValues: *SCRTDoubleVector*, yValues: *SCRTDoubleVector*, indexes: *SCRTDoubleVector*, oldX: *SCRTDoubleVector*, oldY: *SCRTDoubleVector*, iStart: *number*, iEnd: *number*, interpolationPoints: *number*, curvature: *number*, y1Values?: *SCRTDoubleVector*): *void*

Defined in src/Charting/Visuals/RenderableSeries/RenderDataTransforms/BezierRenderDataTransform.ts

Parameters

- **xValues:** *SCRTDoubleVector*
- **yValues:** *SCRTDoubleVector*
- **indexes:** *SCRTDoubleVector*
- **oldX:** *SCRTDoubleVector*
- **oldY:** *SCRTDoubleVector*
- **iStart:** *number*
- **iEnd:** *number*
- **interpolationPoints:** *number*
- **curvature:** *number*
- Optional **y1Values:** *SCRTDoubleVector*

Returns *void*

Const build2DChart

build2DChart(divElementId: *string* | *HTMLDivElement*, definition: *ISciChart2DDefinition* | *string*): *Promise*<{ sciChartSurface: *SciChartSurface*; wasmContext: { CategoryCoordinateCalculatorDouble: {}; DoubleRange: {} & {}; DoubleVector: {}; FlippedCategoryCoordinateCalculatorDouble: {}; FlippedLinearCoordinateCalculatorDouble: {}; FlippedLinearCoordinateCalculatorSingle: {}; FlippedLogarithmicCoordinateCalculator: {}; FloatVector: {}; IntVector: {}; LinearCoordinateCalculatorDouble: {}; LinearCoordinateCalculatorSingle: {}; LogarithmicCoordinateCalculator: {}; NiceDoubleScale: { CalculateTickSpacing: (min: *number*, max: *number*, minorsPerMajor: *number*, maxTicks: *number*) => *DoubleRange*; NiceNum: (range: *number*, round: *boolean*) => *number* }; NiceLogScale: { CalculateLowPrecisionTickSpacing: (min: *number*, max: *number*, logBase: *number*, minorsPerMajor: *number*, maxTicks: *number*) => *DoubleRange*; CalculateTickSpacing: (min: *number*, max: *number*, logBase: *number*, minorsPerMajor: *number*, maxTicks: *number*) => *DoubleRange* }; NumberUtil: { Constrain: (value: *number*, lowerBound: *number*, upperBound: *number*) => *number*; FindIndex: (inputValues: *SCRTDoubleVector*, value: *number*, searchMode: *SCRTFindIndexSearchMode*, dataIsSortedAscending: *boolean*) => *number*; IsDivisibleBy: (value: *number*, divisor: *number*) => *boolean*; IsPowerOf: (value: *number*, power: *number*, logBase: *number*) => *boolean*; LinearInterpolateI: (from: *number*, to: *number*, ratio: *number*) => *number*; Log: (value: *number*, logBase: *number*) => *number*; MinMax: (inputValues: *SCRTDoubleVector*) => *DoubleRange*; MinMaxWithIndex: (inputValues: *SCRTDoubleVector*, startIndex: *number*, count: *number*) => *DoubleRange*; RoundDown: (value: *number*, nearest: *number*) => *number*; RoundDownPower: (value: *number*, power: *number*, logBase: *number*) => *number*; RoundToDigits: (value: *number*, decimals: *number*) => *number*; RoundUp: (value: *number*, nearest: *number*) => *number*; RoundUpPower: (value: *number*, power: *number*, logBase: *number*) => *number* }; ResamplingArgs: {}; ResamplingData: { CategoryData: *ResamplingData*; LinearData: *ResamplingData*; UnevenlySpacedData: *ResamplingData*; UnsortedData: *ResamplingData* }; ResamplingMode: { Auto: *ResamplingMode*; Max: *ResamplingMode*; Mid: *ResamplingMode*; Min: *ResamplingMode*; MinMax: *ResamplingMode*; MinOrMax: *ResamplingMode*; None: *ResamplingMode* }; SCRTAnimationHelperFade: (yValues: *SCRTDoubleVector*, progress: *number*, ysValues: *SCRTDoubleVector*) => *number*; SCRTAnimationHelperScale: (yValues: *SCRTDoubleVector*, zeroLine: *number*, progress: *number*, ysValues: *SCRTDoubleVector*) => *number*; SCRTAnimationHelperSweep: (yValues: *SCRTDoubleVector*, progress: *number*, ysValues: *SCRTDoubleVector*) => *number*; SCRTAnimationHelperWave: (yValues: *SCRTDoubleVector*, durationFraction: *number*, zeroLine: *number*, progress: *number*, ysValues: *SCRTDoubleVector*) => *number*; SCRTBandDrawingParams: {}; SCRTBandSeriesDrawingProvider: {}; SCRTBrush: {}; SCRTBubbleSeriesDrawingProvider: {}; SCRTCandleType: { CandleStick: *SCRTCandleType*; OHLC: *SCRTCandleType* }; SCRTCandlestickSeriesDrawingProvider: {}; SCRTColorVertex: {} & {}; SCRTColumnDrawingParams: {}; SCRTColumnSeriesDrawingProvider: {}; SCRTColumnVertex: {}; SCRTContourParams: {}; SCRTCopyToDestinationInterface: { implement: (wrapper: *SCRTCopyToDestinationInterfaceWrapper*) => *SCRTCopyToDestinationInterface* }; SCRTCreateBitmapTexture: (width: *number*, height: *number*, textureFormat: *eTSRTextureFormat*) => *TSRTexture*; SCRTCreateDahedPen: (color: *number*, thickness: *number*, antialiased: *boolean*, dashPattern: *FloatVector*) => *SCRTPen*; SCRTCreatePalette: (colors: *IntVector*) => *SCRTPalette*; SCRTCredentials: { ApplyLicenseResponse: (response: *string*) => *number*; Dump: () => *string*; GetAllowDebugging: () => *boolean*; GetBuildStamp: () => *string*; GetDeveloperCount: () => *number*; GetEncrypted: (stringToEncrypt: *string*) => *string*; GetEncryptedOrderId: () => *string*; GetExpiryDate: () => *string*; GetLicenseChallenge: () => *string*; GetLicenseDaysRemaining: () => *number*; GetLicenseErrors: () => *string*; GetLicenseType: () => *SCRTLicenseType*; GetOrderId: () => *string*; GetProductCode: () => *string*; HasFeature: (feature: *string*) => *SCRTLicenseType*; Hash256Encode64: (stringToHash: *string*) => *string*; RequiresValidation: () => *boolean*; ResetRuntimeLicense: () => *void*; SetRuntimeLicenseKeyW: (licenseKey: *string*) => *boolean* }; SCRTDoLeakCheck: () => *void*; SCRTDoubleArrayOperations: {};

SCRTDoubleArraysXyResampleOutput: {}; SCRTDoubleResampler: {}; SCRTDoubleResamplerMergeIndicesParams: {}; SCRTDoubleVector: {} & {}; SCRTFifoVector: {}; SCRTFileLoadCallbackInterface: { implement: (wrapper: *SCRTFileLoadCallbackInterfaceWrapper*) => *SCRTFileLoadCallbackInterface* }; SCRTFillTextureAbgr: (texture: *TSRTexture*, width: *number*, height: *number*, pixels: *IntVector*) => *void*; SCRTFillTextureFloat32: (texture: *TSRTexture*, width: *number*, height: *number*, pixels: *SCRTFloatVector*) => *TSRVector4*; SCRTFillVectorSequential: (SCRTFillVectorSequential: *SCRTDoubleVector*, count: *number*) => *void*; SCRTFindIndexSearchMode: { Exact: *SCRTFindIndexSearchMode*; Nearest: *SCRTFindIndexSearchMode*; RoundDown: *SCRTFindIndexSearchMode*; RoundUp: *SCRTFindIndexSearchMode* }; SCRTFloatVector: {}; SCRTFontKey: {}; SCRTFrameRenderer2D: {}; SCRTGetGlobalSampleChartInterface: () => *SCRTSampleChartInterface*; SCRTGetMainRenderContext2D: () => *SCRTRenderContext*; SCRTGetScreenHeight: () => *number*; SCRTGetScreenWidth: () => *number*; SCRTGlowEffect: {}; SCRTHeatmapDrawingParams: {}; SCRTHeatmapSeriesDrawingProvider: {}; SCRTHitTestHelper: { GetNearestXyPoint: (xCoordinateCalculator: *CoordinateCalculator*, yCoordinateCalculator: *CoordinateCalculator*, xValues: *SCRTDoubleVector*, yValues: *SCRTDoubleVector*, isSorted: *boolean*, xHitCoord: *number*, yHitCoord: *number*, hitTestRadius: *number*) => *DoubleRange* }; SCRTInitEngine2D: () => *void*; SCRTLicenseType: { LICENSE_TYPE_COMMUNITY: *SCRTLicenseType*; LICENSE_TYPE_FULL: *SCRTLicenseType*; LICENSE_TYPE_FULL_EXPIRED: *SCRTLicenseType*; LICENSE_TYPE_INVALID_DEVELOPER_LICENSE: *SCRTLicenseType*; LICENSE_TYPE_INVALID_LICENSE: *SCRTLicenseType*; LICENSE_TYPE_NO_LICENSE: *SCRTLicenseType*; LICENSE_TYPE_REQUIRES_VALIDATION: *SCRTLicenseType*; LICENSE_TYPE_SUBSCRIPTION_EXPIRED: *SCRTLicenseType*; LICENSE_TYPE_TRIAL: *SCRTLicenseType*; LICENSE_TYPE_TRIAL_EXPIRED: *SCRTLicenseType* }; SCRTLineDrawingParams: {}; SCRTLineGapMode: { CloseGaps: *SCRTLineGapMode*; Default: *SCRTLineGapMode*; DrawGaps: *SCRTLineGapMode* }; SCRTLineSeriesDrawingProvider: {}; SCRTLineType: { Digital: *SCRTLineType*; List: *SCRTLineType*; Nan: *SCRTLineType*; Strip: *SCRTLineType* }; SCRTMemCopy: (destPtr: *number*, sourcePtr: *number*, count: *number*) => *void*; SCRTMemMove: (destPtr: *number*, sourcePtr: *number*, count: *number*) => *void*; SCRTMountainDrawingParams: {}; SCRTMountainSeriesDrawingProvider: {}; SCRTMultiplyColorVectorOpacity: (originalVector: *IntVector*, resultVector: *IntVector*, factor: *number*) => *void*; SCRTOhlcDrawingParams: {}; SCRTPalette: {} & { GetNoOverrideColorCode: () => *number* }; SCRTPen: {}; SCRTPointDrawingParams: {}; SCRTRectVertex: {} & {}; SCRTRegisterFile: (fileName: *string*, url: *string*, callback: *SCRTFileLoadCallbackInterface*) => *void*; SCRTSampleChartInterface: { implement: (wrapper: *SCRTSampleChartInterfaceWrapper*) => *SCRTSampleChartInterface* }; SCRTScatterSeriesDrawingProvider: {}; SCRTSeriesEffectType: { Glow: *SCRTSeriesEffectType* }; SCRTSetActiveDoubleVector: (SCRTSetActiveDoubleVector: *SCRTDoubleVector*, doubleVector: *number*) => *void*; SCRTSetActiveTexture: (texture: *TSRTexture*) => *void*; SCRTSetClearAlphaParams: (enabled: *boolean*, alpha: *number*) => *void*; SCRTSetGlobalCopyToDestinationInterface: (param0: *SCRTCopyToDestinationInterface*) => *void*; SCRTSetGlobalSampleChartInterface: (param0: *SCRTSampleChartInterface*) => *void*; SCRTSetMainWindowSize: (width: *number*, height: *number*) => *void*; SCRTSetTextureLinearSamplerEnabled: (texture: *TSRTexture*, enabled: *boolean*) => *void*; SCRTSetWaterMarkProperties: (properties: *SCRTWaterMarkProperties*) => *void*; SCRTShadowEffect: {}; SCRTShutdownEngine2D: () => *void*; SCRTSolidBrush: {}; SCRTSplineHelperCubicSpline: (xValues: *SCRTDoubleVector*, yValues: *SCRTDoubleVector*, xsValues: *SCRTDoubleVector*, ysValues: *SCRTDoubleVector*, initialSize: *number*, interpolationPoints: *number*, containsNAN: *boolean*) => *void*; SCRTSpriteType: { FixedSize: *SCRTSpriteType*; Normal: *SCRTSpriteType* }; SCRTStackedColumnDrawingParams: {}; SCRTStackedColumnSeriesDrawingProvider: {}; SCRTSurfaceDestination: { implement: (wrapper: *SCRTSurfaceDestinationWrapper*) => *SCRTSurfaceDestination* }; SCRTTextureBrush: {}; SCRTWaterMarkProperties: {}; SCRTXvaluesProvider: {}; StringVector: {}; TSRCamera: {}; TSRRequestCanvasDraw: (canvasID: *string*) => *void*; TSRRequestDraw: () => *void*; TSRRequestExit: () => *void*; TSRSetDrawRequestsEnabled: (enabled: *boolean*) => *void*; TSRTextBounds: {}; TSRTextLineBounds: {}; TSRVector2: {} & {}; TSRVector3: {} & {}; TSRVector4: {} & {}; UIntVector: {}; VectorColorVertex: {}; VectorColumnVertex: {}; VectorRectVertex: {}; WStringVector: {}; canvas: *HTMLCanvasElement*; canvas2D: *HTMLCanvasElement*; eSCRTBlendMode: { BlendAdditiveAlpha: *eSCRTBlendMode*; BlendAdditiveColor: *eSCRTBlendMode*; BlendAdditiveOneAlpha: *eSCRTBlendMode*; BlendDefault: *eSCRTBlendMode*; BlendDisabled: *eSCRTBlendMode* }; eSCRTBrushMappingMode: { PerPrimitive: *eSCRTBrushMappingMode*; PerScreen: *eSCRTBrushMappingMode* }; eTSRCameraProjectionMode: { CAMERA_PROJECTIONMODE_ORTHOGONAL: *eTSRCameraProjectionMode*; CAMERA_PROJECTIONMODE_PERSPECTIVE: *eTSRCameraProjectionMode* }; eTSRMetaDataType: { BitFlags: *eTSRMetaDataType*; Core: *eTSRMetaDataType*; Defined: *eTSRMetaDataType*; DynamicDefined: *eTSRMetaDataType*; Enum: *eTSRMetaDataType*; Unknown: *eTSRMetaDataType* }; eTSRPlatform: { Android: *eTSRPlatform*; Linux: *eTSRPlatform*; Mac: *eTSRPlatform*; Undefined: *eTSRPlatform*; Web: *eTSRPlatform*; Windows: *eTSRPlatform*; iOS: *eTSRPlatform* }; eTSRRendererType: { TSR_RENDERER_TYPE_D3D11: *eTSRRendererType*; TSR_RENDERER_TYPE_D3D11_LEVEL10: *eTSRRendererType*; TSR_RENDERER_TYPE_D3D12: *eTSRRendererType*; TSR_RENDERER_TYPE_D3D9: *eTSRRendererType*; TSR_RENDERER_TYPE_GL: *eTSRRendererType*; TSR_RENDERER_TYPE_GLES2: *eTSRRendererType*; TSR_RENDERER_TYPE_GLES3: *eTSRRendererType*; TSR_RENDERER_TYPE_METAL: *eTSRRendererType*; TSR_RENDERER_TYPE_UNDEFINED: *eTSRRendererType*; TSR_RENDERER_TYPE_VULKAN: *eTSRRendererType* }; eTSRTextAlignMode: { Center: *eTSRTextAlignMode*; Left: *eTSRTextAlignMode*; Right: *eTSRTextAlignMode* }; eTSRTextureFormat: { TSR_TEXTUREFORMAT_A8B8G8R8: *eTSRTextureFormat*; TSR_TEXTUREFORMAT_R32F: *eTSRTextureFormat* }; eVariableUsage: { Array: *eVariableUsage*; Blob: *eVariableUsage*; DynamicArray: *eVariableUsage*; Normal: *eVariableUsage*; Pointer: *eVariableUsage*; Vector: *eVariableUsage*; VectorOfPointers: *eVariableUsage* } }>

Defined in src/Builder/buildSurface.ts

Construct a chart with SciChartSurface using a ISciChart2DDefinition which can be pure data.

remarks This method is async and must be awaited

Parameters

- **divElementId:** *string* | *HTMLDivElement*

  The Div Element ID where the SciChartSurface will reside

- **definition:** *ISciChart2DDefinition* | *string*

the ISciChart2DDefinition

Returns *Promise*<{ **sciChartSurface**: *SciChartSurface*; **wasmContext**: { **CategoryCoordinateCalculatorDouble**: {}; **DoubleRange**: {} & {}; **DoubleVector**: {}; **FlippedCategoryCoordinateCalculatorDouble**: {}; **FlippedLinearCoordinateCalculatorDouble**: {}; **FlippedLinearCoordinateCalculatorSingle**: {}; **FlippedLogarithmicCoordinateCalculator**: {}; **FloatVector**: {}; **IntVector**: {}; **LinearCoordinateCalculatorDouble**: {}; **LinearCoordinateCalculatorSingle**: {}; **LogarithmicCoordinateCalculator**: {}; **NiceDoubleScale**: { **CalculateTickSpacing**: (**min**: *number*, **max**: *number*, **minorsPerMajor**: *number*, **maxTicks**: *number*) => *DoubleRange*; **NiceNum**: (**range**: *number*, **round**: *boolean*) => *number* }; **NiceLogScale**: { **CalculateLowPrecisionTickSpacing**: (**min**: *number*, **max**: *number*, **logBase**: *number*, **minorsPerMajor**: *number*, **maxTicks**: *number*) => *DoubleRange*; **CalculateTickSpacing**: (**min**: *number*, **max**: *number*, **logBase**: *number*, **minorsPerMajor**: *number*, **maxTicks**: *number*) => *DoubleRange* }; **NumberUtil**: { **Constrain**: (**value**: *number*, **lowerBound**: *number*, **upperBound**: *number*) => *number*; **FindIndex**: (**inputValues**: *SCRTDoubleVector*, **value**: *number*, **searchMode**: *SCRTFindIndexSearchMode*, **dataIsSortedAscending**: *boolean*) => *number*; **IsDivisibleBy**: (**value**: *number*, **divisor**: *number*) => *boolean*; **IsPowerOf**: (**value**: *number*, **power**: *number*, **logBase**: *number*) => *boolean*; **LinearInterpolateI**: (**from**: *number*, **to**: *number*, **ratio**: *number*) => *number*; **Log**: (**value**: *number*, **logBase**: *number*) => *number*; **MinMax**: (**inputValues**: *SCRTDoubleVector*) => *DoubleRange*; **MinMaxWithIndex**: (**inputValues**: *SCRTDoubleVector*, **startIndex**: *number*, **count**: *number*) => *DoubleRange*; **RoundDown**: (**value**: *number*, **nearest**: *number*) => *number*; **RoundDownPower**: (**value**: *number*, **power**: *number*, **logBase**: *number*) => *number*; **RoundToDigits**: (**value**: *number*, **decimals**: *number*) => *number*; **RoundUp**: (**value**: *number*, **nearest**: *number*) => *number*; **RoundUpPower**: (**value**: *number*, **power**: *number*, **logBase**: *number*) => *number* }; **ResamplingArgs**: {}; **ResamplingData**: { **CategoryData**: *ResamplingData*; **LinearData**: *ResamplingData*; **UnevenlySpacedData**: *ResamplingData*; **UnsortedData**: *ResamplingData* }; **ResamplingMode**: { **Auto**: *ResamplingMode*; **Max**: *ResamplingMode*; **Mid**: *ResamplingMode*; **Min**: *ResamplingMode*; **MinMax**: *ResamplingMode*; **MinOrMax**: *ResamplingMode*; **None**: *ResamplingMode* }; **SCRTAnimationHelperFade**: (**yValues**: *SCRTDoubleVector*, **progress**: *number*, **ysValues**: *SCRTDoubleVector*) => *number*; **SCRTAnimationHelperScale**: (**yValues**: *SCRTDoubleVector*, **zeroLine**: *number*, **progress**: *number*, **ysValues**: *SCRTDoubleVector*) => *number*; **SCRTAnimationHelperSweep**: (**yValues**: *SCRTDoubleVector*, **progress**: *number*, **ysValues**: *SCRTDoubleVector*) => *number*; **SCRTAnimationHelperWave**: (**yValues**: *SCRTDoubleVector*, **durationFraction**: *number*, **zeroLine**: *number*, **progress**: *number*, **ysValues**: *SCRTDoubleVector*) => *number*; **SCRTBandDrawingParams**: {}; **SCRTBandSeriesDrawingProvider**: {}; **SCRTBrush**: {}; **SCRTBubbleSeriesDrawingProvider**: {}; **SCRTCandleType**: { **CandleStick**: *SCRTCandleType*; **OHLC**: *SCRTCandleType* }; **SCRTCandlestickSeriesDrawingProvider**: {}; **SCRTColorVertex**: {} & {}; **SCRTColumnDrawingParams**: {}; **SCRTColumnSeriesDrawingProvider**: {}; **SCRTColumnVertex**: {}; **SCRTContourParams**: {}; **SCRTCopyToDestinationInterface**: { **implement**: (**wrapper**: *SCRTCopyToDestinationInterfaceWrapper*) => *SCRTCopyToDestinationInterface* }; **SCRTCreateBitmapTexture**: (**width**: *number*, **height**: *number*, **textureFormat**: *eTSRTextureFormat*) => *TSRTexture*; **SCRTCreateDahedPen**: (**color**: *number*, **thickness**: *number*, **antialiased**: *boolean*, **dashPattern**: *FloatVector*) => *SCRTPen*; **SCRTCreatePalette**: (**colors**: *IntVector*) => *SCRTPalette*; **SCRTCredentials**: { **ApplyLicenseResponse**: (**response**: *string*) => *number*; **Dump**: () => *string*; **GetAllowDebugging**: () => *boolean*; **GetBuildStamp**: () => *string*; **GetDeveloperCount**: () => *number*; **GetEncrypted**: (**stringToEncrypt**: *string*) => *string*; **GetEncryptedOrderId**: () => *string*; **GetExpiryDate**: () => *string*; **GetLicenseChallenge**: () => *string*; **GetLicenseDaysRemaining**: () => *number*; **GetLicenseErrors**: () => *string*; **GetLicenseType**: () => *SCRTLicenseType*; **GetOrderId**: () => *string*; **GetProductCode**: () => *string*; **HasFeature**: (**feature**: *string*) => *SCRTLicenseType*; **Hash256Encode64**: (**stringToHash**: *string*) => *string*; **RequiresValidation**: () => *boolean*; **ResetRuntimeLicense**: () => *void*; **SetRuntimeLicenseKeyW**: (**licenseKey**: *string*) => *boolean* }; **SCRTDoLeakCheck**: () => *void*; **SCRTDoubleArrayOperations**: {}; **SCRTDoubleArraysXyResampleOutput**: {}; **SCRTDoubleResampler**: {}; **SCRTDoubleResamplerMergeIndicesParams**: {}; **SCRTDoubleVector**: {} & {}; **SCRTFifoVector**: {}; **SCRTFileLoadCallbackInterface**: { **implement**: (**wrapper**: *SCRTFileLoadCallbackInterfaceWrapper*) => *SCRTFileLoadCallbackInterface* }; **SCRTFillTextureAbgr**: (**texture**: *TSRTexture*, **width**: *number*, **height**: *number*, **pixels**: *IntVector*) => *void*; **SCRTFillTextureFloat32**: (**texture**: *TSRTexture*, **width**: *number*, **height**: *number*, **pixels**: *SCRTFloatVector*) => *TSRVector4*; **SCRTFillVectorSequential**: (**SCRTFillVectorSequential**: *SCRTDoubleVector*, **count**: *number*) => *void*; **SCRTFindIndexSearchMode**: { **Exact**: *SCRTFindIndexSearchMode*; **Nearest**: *SCRTFindIndexSearchMode*; **RoundDown**: *SCRTFindIndexSearchMode*; **RoundUp**: *SCRTFindIndexSearchMode* }; **SCRTFloatVector**: {}; **SCRTFontKey**: {}; **SCRTFrameRenderer2D**: {}; **SCRTGetGlobalSampleChartInterface**: () => *SCRTSampleChartInterface*; **SCRTGetMainRenderContext2D**: () => *SCRTRenderContext*; **SCRTGetScreenHeight**: () => *number*; **SCRTGetScreenWidth**: () => *number*; **SCRTGlowEffect**: {}; **SCRTHeatmapDrawingParams**: {}; **SCRTHeatmapSeriesDrawingProvider**: {}; **SCRTHitTestHelper**: { **GetNearestXyPoint**: (**xCoordinateCalculator**: *CoordinateCalculator*, **yCoordinateCalculator**: *CoordinateCalculator*, **xValues**: *SCRTDoubleVector*, **yValues**: *SCRTDoubleVector*, **isSorted**: *boolean*, **xHitCoord**: *number*, **yHitCoord**: *number*, **hitTestRadius**: *number*) => *DoubleRange* }; **SCRTInitEngine2D**: () => *void*; **SCRTLicenseType**: { **LICENSE_TYPE_COMMUNITY**: *SCRTLicenseType*; **LICENSE_TYPE_FULL**: *SCRTLicenseType*; **LICENSE_TYPE_FULL_EXPIRED**: *SCRTLicenseType*; **LICENSE_TYPE_INVALID_DEVELOPER_LICENSE**: *SCRTLicenseType*; **LICENSE_TYPE_INVALID_LICENSE**: *SCRTLicenseType*; **LICENSE_TYPE_NO_LICENSE**: *SCRTLicenseType*; **LICENSE_TYPE_REQUIRES_VALIDATION**: *SCRTLicenseType*; **LICENSE_TYPE_SUBSCRIPTION_EXPIRED**: *SCRTLicenseType*; **LICENSE_TYPE_TRIAL**:

SCRTLicenseType; **LICENSE_TYPE_TRIAL_EXPIRED**: *SCRTLicenseType* };
SCRTLineDrawingParams: {}; **SCRTLineGapMode**: { **CloseGaps**: *SCRTLineGapMode*; **Default**:
SCRTLineGapMode; **DrawGaps**: *SCRTLineGapMode* }; **SCRTLineSeriesDrawingProvider**: {};
SCRTLineType: { **Digital**: *SCRTLineType*; **List**: *SCRTLineType*; **Nan**: *SCRTLineType*; **Strip**:
SCRTLineType }; **SCRTMemCopy**: (**destPtr**: *number*, **sourcePtr**: *number*, **count**: *number*) => *void*;
SCRTMemMove: (**destPtr**: *number*, **sourcePtr**: *number*, **count**: *number*) => *void*;
SCRTMountainDrawingParams: {}; **SCRTMountainSeriesDrawingProvider**: {};
SCRTMultiplyColorVectorOpacity: (**originalVector**: *IntVector*, **resultVector**: *IntVector*, **factor**:
number) => *void*; **SCRTOhlcDrawingParams**: {}; **SCRTPalette**: {} & { **GetNoOverrideColorCode**:
() => *number* }; **SCRTPen**: {}; **SCRTPointDrawingParams**: {}; **SCRTRectVertex**: {} & {};
SCRTRegisterFile: (**fileName**: *string*, **url**: *string*, **callback**: *SCRTFileLoadCallbackInterface*) =>
void; **SCRTSampleChartInterface**: { **implement**: (**wrapper**: *SCRTSampleChartInterfaceWrapper*)
=> *SCRTSampleChartInterface* }; **SCRTScatterSeriesDrawingProvider**: {};
SCRTSeriesEffectType: { **Glow**: *SCRTSeriesEffectType* }; **SCRTSetActiveDoubleVector**:
(**SCRTSetActiveDoubleVector**: *SCRTDoubleVector*, **doubleVector**: *number*) => *void*;
SCRTSetActiveTexture: (**texture**: *TSRTexture*) => *void*; **SCRTSetClearAlphaParams**: (**enabled**:
boolean, **alpha**: *number*) => *void*; **SCRTSetGlobalCopyToDestinationInterface**: (**param0**:
SCRTCopyToDestinationInterface) => *void*; **SCRTSetGlobalSampleChartInterface**: (**param0**:
SCRTSampleChartInterface) => *void*; **SCRTSetMainWindowSize**: (**width**: *number*, **height**:
number) => *void*; **SCRTSetTextureLinearSamplerEnabled**: (**texture**: *TSRTexture*, **enabled**:
boolean) => *void*; **SCRTSetWaterMarkProperties**: (**properties**: *SCRTWaterMarkProperties*) =>
void; **SCRTShadowEffect**: {}; **SCRTShutdownEngine2D**: () => *void*; **SCRTSolidBrush**: {};
SCRTSplineHelperCubicSpline: (**xValues**: *SCRTDoubleVector*, **yValues**: *SCRTDoubleVector*,
xsValues: *SCRTDoubleVector*, **ysValues**: *SCRTDoubleVector*, **initialSize**: *number*,
interpolationPoints: *number*, **containsNAN**: *boolean*) => *void*; **SCRTSpriteType**: { **FixedSize**:
SCRTSpriteType; **Normal**: *SCRTSpriteType* }; **SCRTStackedColumnDrawingParams**: {};
SCRTStackedColumnSeriesDrawingProvider: {}; **SCRTSurfaceDestination**: { **implement**:
(**wrapper**: *SCRTSurfaceDestinationWrapper*) => *SCRTSurfaceDestination* }; **SCRTTextureBrush**: {};
SCRTWaterMarkProperties: {}; **SCRTXvaluesProvider**: {}; **StringVector**: {}; **TSRCamera**: {};
TSRRequestCanvasDraw: (**canvasID**: *string*) => *void*; **TSRRequestDraw**: () => *void*;
TSRRequestExit: () => *void*; **TSRSetDrawRequestsEnabled**: (**enabled**: *boolean*) => *void*;
TSRTextBounds: {}; **TSRTextLineBounds**: {}; **TSRVector2**: {} & {}; **TSRVector3**: {} & {};
TSRVector4: {} & {}; **UIntVector**: {}; **VectorColorVertex**: {}; **VectorColumnVertex**: {};
VectorRectVertex: {}; **WStringVector**: {}; **canvas**: *HTMLCanvasElement*; **canvas2D**:
HTMLCanvasElement; **eSCRTBlendMode**: { **BlendAdditiveAlpha**: *eSCRTBlendMode*;
BlendAdditiveColor: *eSCRTBlendMode*; **BlendAdditiveOneAlpha**: *eSCRTBlendMode*;
BlendDefault: *eSCRTBlendMode*; **BlendDisabled**: *eSCRTBlendMode* };
eSCRTBrushMappingMode: { **PerPrimitive**: *eSCRTBrushMappingMode*; **PerScreen**:
eSCRTBrushMappingMode }; **eTSRCameraProjectionMode**: {
CAMERA_PROJECTIONMODE_ORTHOGONAL: *eTSRCameraProjectionMode*;
CAMERA_PROJECTIONMODE_PERSPECTIVE: *eTSRCameraProjectionMode* };
eTSRMetaDataType: { **BitFlags**: *eTSRMetaDataType*; **Core**: *eTSRMetaDataType*; **Defined**:
eTSRMetaDataType; **DynamicDefined**: *eTSRMetaDataType*; **Enum**: *eTSRMetaDataType*;
Unknown: *eTSRMetaDataType* }; **eTSRPlatform**: { **Android**: *eTSRPlatform*; **Linux**: *eTSRPlatform*;
Mac: *eTSRPlatform*; **Undefined**: *eTSRPlatform*; **Web**: *eTSRPlatform*; **Windows**: *eTSRPlatform*; **iOS**:
eTSRPlatform }; **eTSRRendererType**: { **TSR_RENDERER_TYPE_D3D11**: *eTSRRendererType*;
TSR_RENDERER_TYPE_D3D11_LEVEL10: *eTSRRendererType*; **TSR_RENDERER_TYPE_D3D12**:
eTSRRendererType; **TSR_RENDERER_TYPE_D3D9**: *eTSRRendererType*; **TSR_RENDERER_TYPE_GL**:
eTSRRendererType; **TSR_RENDERER_TYPE_GLES2**: *eTSRRendererType*;
TSR_RENDERER_TYPE_GLES3: *eTSRRendererType*; **TSR_RENDERER_TYPE_METAL**:
eTSRRendererType; **TSR_RENDERER_TYPE_UNDEFINED**: *eTSRRendererType*;
TSR_RENDERER_TYPE_VULKAN: *eTSRRendererType* }; **eTSRTextAlignMode**: { **Center**:
eTSRTextAlignMode; **Left**: *eTSRTextAlignMode*; **Right**: *eTSRTextAlignMode* }; **eTSRTextureFormat**:
{ **TSR_TEXTUREFORMAT_A8B8G8R8**: *eTSRTextureFormat*; **TSR_TEXTUREFORMAT_R32F**:
eTSRTextureFormat }; **eVariableUsage**: { **Array**: *eVariableUsage*; **Blob**: *eVariableUsage*;
DynamicArray: *eVariableUsage*; **Normal**: *eVariableUsage*; **Pointer**: *eVariableUsage*; **Vector**:
eVariableUsage; **VectorOfPointers**: *eVariableUsage* } } }>

Const buildAnnotations

```
buildAnnotations(definition: TAnnotationDefinition | TAnnotationDefinition[]): any[]
```

Defined in src/Builder/buildAnnotations.ts

Build one or more annotations from a definition that can be pure data.

Parameters

- **definition:** *TAnnotationDefinition* | *TAnnotationDefinition*[]

  One or an array of TAnnotationDefinition

Returns *any*[]

An array of annotations

Const buildAxes

`buildAxes(wasmContext: TSciChart, definition: TAxisDefinition | TAxisDefinition[]): AxisBase2D[]`

Defined in src/Builder/buildAxis.ts

Build one or more axes from a definition that can be pure data.

Parameters

- **wasmContext:** *TSciChart*
  A SciChart 2D WebAssembly Context or SciChart 3D WebAssembly Context
- **definition:** *TAxisDefinition* | *TAxisDefinition*[]
  One or an array of TAxisDefinition

Returns *AxisBase2D*[]

An array of AxisBase2D.

Const buildAxis

`buildAxis(wasmContext: TSciChart, definition: TAxisDefinition): AxisBase2D`

Defined in src/Builder/buildAxis.ts

Parameters

- **wasmContext:** *TSciChart*
- **definition:** *TAxisDefinition*

Returns *AxisBase2D*

Const buildChart

`buildChart(divElementId: string | HTMLDivElement, definition: string | TSurfaceDefinition): Promise<TWebAssemblyChart | SciChartPieSurface>`

Defined in src/Builder/chartBuilder.ts

Builds an entire chart from a definition that can be pure data.

Parameters

- **divElementId:** *string* | *HTMLDivElement*
  The Div Element ID where the SciChartSurface will reside
- **definition:** *string* | *TSurfaceDefinition*
  a or a string which will be parsed to it.

Returns *Promise*<*TWebAssemblyChart* | *SciChartPieSurface*>

Const buildDataSeries

`buildDataSeries(wasmContext: TSciChart, dataSeriesDefinition: TDataSeriesDefinition, sharedData?: TSharedDataDefinition, originalDataSeries?: IDataSeries): IDataSeries`

Defined in src/Builder/buildDataSeries.ts

Build a data series from a definition that can be pure data.

Parameters

- **wasmContext:** *TSciChart*
  A SciChart 2D WebAssembly Context or SciChart 3D WebAssembly Context
- **dataSeriesDefinition:** *TDataSeriesDefinition*
- Optional **sharedData:** *TSharedDataDefinition*
  Optional TSharedDataDefinition to define shared data which can be referenced by the data series
- Optional **originalDataSeries:** *IDataSeries*

Optional IDataSeries to define original data for filter

Returns *IDataSeries*

An IDataSeries

Const buildFilter

```
buildFilter(dataSeries: BaseDataSeries, definition: TFilterDefinition):
HlcScaleOffsetFilter | OhlcScaleOffsetFilter | XyLinearTrendFilter |
XyMovingAverageFilter | XyRatioFilter | XyScaleOffsetFilter | XyyScaleOffsetFilter |
XyzScaleOffsetFilter
```

Defined in src/Builder/buildDataSeries.ts

Parameters

- **dataSeries:** *BaseDataSeries*
- **definition:** *TFilterDefinition*

Returns *HlcScaleOffsetFilter* | *OhlcScaleOffsetFilter* | *XyLinearTrendFilter* | *XyMovingAverageFilter* | *XyRatioFilter* | *XyScaleOffsetFilter* | *XyyScaleOffsetFilter* | *XyzScaleOffsetFilter*

Const buildModifier

```
buildModifier(definition: TModifierDefinition): ChartModifierBase2D
```

Defined in src/Builder/buildModifiers.ts

Parameters

- **definition:** *TModifierDefinition*

Returns *ChartModifierBase2D*

Const buildModifiers

```
buildModifiers(definition: TModifierDefinition | TModifierDefinition[]):
ChartModifierBase2D[]
```

Defined in src/Builder/buildModifiers.ts

Build one or more chart modifiers from a definition that can be pure data.

Parameters

- **definition:** *TModifierDefinition* | *TModifierDefinition*[]

  One or an array of TModifierDefinition

Returns *ChartModifierBase2D*[]

An array of modifiers

Const buildPieChart

```
buildPieChart(divElementId: string | HTMLDivElement, definition: ISciChartPieDefinition
| string): Promise<SciChartPieSurface>
```

Defined in src/Builder/buildSurface.ts

Construct a chart with SciChartPieSurface using a ISciChartPieDefinition which can be pure data.

remarks This method is async and must be awaited

Parameters

- **divElementId:** *string* | *HTMLDivElement*

  The Div Element ID where the SciChartPieSurface will reside

- **definition:** *ISciChartPieDefinition* | *string*

  the ISciChartPieDefinition

Returns *Promise*<*SciChartPieSurface*>

Const buildSeries

```
buildSeries(wasmContext: TSciChart, definition: TSeriesDefinition |
TSeriesDefinition[], sharedData?: TSharedDataDefinition): IRenderableSeries[]
```

Defined in src/Builder/buildSeries.ts

Build one or more renderable series from a definition that can be pure data.

Parameters

- **wasmContext:** *TSciChart*

  A SciChart 2D WebAssembly Context or SciChart 3D WebAssembly Context

- **definition:** *TSeriesDefinition* | *TSeriesDefinition*[]

  One or an array of TSeriesDefinition

- Optional **sharedData:** *TSharedDataDefinition*

  Optional TSharedDataDefinition to define shared data which can be referenced by the renderable series

Returns *IRenderableSeries*[]

An array of IRenderableSeries.

Const calcAnnotationBordersForAxisMarker

```
calcAnnotationBordersForAxisMarker(isVerticalChart: boolean, x1: number, y1: number,
horizontalAxis: AxisBase2D, verticalAxis: AxisBase2D, textureWidth: number, viewRect:
Rect, xPosition: number, yPosition: number, textureHeight: number,
annotationMarginXDirection: number, annotationMarginYDirection: number,
isHorizontalAxisAlignmentReversed: boolean, isVerticalAxisAlignmentReversed: boolean):
{ x1: number; x2: number; y1: number; y2: number }
```

Defined in src/utils/pointUtil.ts

Calculates annotation borders for AxisMarkerAnnotation

Parameters

- **isVerticalChart:** *boolean*

  the vertical chart flag

- **x1:** *number*

  the X1 data value of the annotation

- **y1:** *number*

  the Y1 data value of the annotation

- **horizontalAxis:** *AxisBase2D*

  the horizontal axis

- **verticalAxis:** *AxisBase2D*

  the vertical axis

- **textureWidth:** *number*

  the texture width

- **viewRect:** *Rect*

  the seriesViewRect

- **xPosition:** *number*

  the X position of the texture on the SciChartSurface, the left-top corner position on the canvas

- **yPosition:** *number*

  the Y position of the texture on the SciChartSurface, the left-top corner position on the canvas

- **textureHeight:** *number*

  the texture width

- **annotationMarginXDirection:** *number*

the texture margin in X direction

- **annotationMarginYDirection:** *number*
  the texture margin in Y direction
- **isHorizontalAxisAlignmentReversed:** *boolean*
  if true EAxisAlignment.Top, otherwise EAxisAlignment.Bottom
- **isVerticalAxisAlignmentReversed:** *boolean*
  if true EAxisAlignment.Left, otherwise EAxisAlignment.Right

Returns { **x1**: *number*; **x2**: *number*; **y1**: *number*; **y2**: *number* }

- **x1**: *number*
- **x2**: *number*
- **y1**: *number*
- **y2**: *number*

Const calcAverageForArray

```
calcAverageForArray(ar: number[], averageNum: number, index?: number): number
```

Defined in src/utils/calcAverage.ts

Parameters

- **ar:** *number*[]
- **averageNum:** *number*
- Optional **index:** *number*

Returns *number*

Const calcAverageForDoubleVector

```
calcAverageForDoubleVector(doubleVector: SCRTDoubleVector, averageNum: number, index?: number): number
```

Defined in src/utils/calcAverage.ts

description Calculates average for DoubleVector

Parameters

- **doubleVector:** *SCRTDoubleVector*
- **averageNum:** *number*
  number of values to respect for the average
- Optional **index:** *number*
  index of the latest value to include, by default equals to length-1 of DoubleVector

Returns *number*

Const calcCrossProduct

```
calcCrossProduct(xA: number, yA: number, xB: number, yB: number, xC: number, yC: number): number
```

Defined in src/utils/pointUtil.ts

Parameters

- **xA:** *number*
- **yA:** *number*
- **xB:** *number*
- **yB:** *number*

- **xC:** *number*
- **yC:** *number*

Returns *number*

Const calcDistance

`calcDistance(x1: number, y1: number, x2: number, y2: number): number`

Defined in src/utils/pointUtil.ts

Parameters

- **x1:** *number*
- **y1:** *number*
- **x2:** *number*
- **y2:** *number*

Returns *number*

Const calcDistanceFromLine

`calcDistanceFromLine(x: number, y: number, startX: number, startY: number, endX: number, endY: number): number`

Defined in src/utils/pointUtil.ts

Parameters

- **x:** *number*
- **y:** *number*
- **startX:** *number*
- **startY:** *number*
- **endX:** *number*
- **endY:** *number*

Returns *number*

Const calcDistanceFromLineSegment

`calcDistanceFromLineSegment(x: number, y: number, startX: number, startY: number, endX: number, endY: number): number`

Defined in src/utils/pointUtil.ts

Parameters

- **x:** *number*
- **y:** *number*
- **startX:** *number*
- **startY:** *number*
- **endX:** *number*
- **endY:** *number*

Returns *number*

Const calcDotProduct

`calcDotProduct(v1x: number, v1y: number, v2x: number, v2y: number, v3x: number, v3y: number): number`

Defined in src/utils/pointUtil.ts

Parameters

- **v1x:** *number*
- **v1y:** *number*
- **v2x:** *number*
- **v2y:** *number*
- **v3x:** *number*
- **v3y:** *number*

Returns *number*

Const calcNewApex

```
calcNewApex(x1: number, y1: number, x2: number, y2: number, isVertical: boolean): { x1y1: { x: number; y: number }; x1y2: { x: number; y: number }; x2y1: { x: number; y: number }; x2y2: { x: number; y: number } }
```

Defined in src/Charting/Visuals/Annotations/annotationHelpers.ts

Parameters

- **x1:** *number*
- **y1:** *number*
- **x2:** *number*
- **y2:** *number*
- **isVertical:** *boolean*

Returns { **x1y1**: { **x**: *number*; **y**: *number* }; **x1y2**: { **x**: *number*; **y**: *number* }; **x2y1**: { **x**: *number*; **y**: *number* }; **x2y2**: { **x**: *number*; **y**: *number* } }

- **x1y1**: { **x**: *number*; **y**: *number* }
  - **x**: *number*
  - **y**: *number*
- **x1y2**: { **x**: *number*; **y**: *number* }
  - **x**: *number*
  - **y**: *number*
- **x2y1**: { **x**: *number*; **y**: *number* }
  - **x**: *number*
  - **y**: *number*
- **x2y2**: { **x**: *number*; **y**: *number* }
  - **x**: *number*
  - **y**: *number*

Const calcTooltipSize

```
calcTooltipSize(valuesWithLabels: string[], fontSize?: number): { height: number; width: number }
```

Defined in src/Charting/ChartModifiers/CursorModifier.ts

Calculate the width and height of the tooltip based on the content array

Parameters

- **valuesWithLabels:** *string*[]
- Default value **fontSize:** *number* = 13

Returns { **height**: *number*; **width**: *number* }

- **height**: *number*

- **width**: *number*

Const calculateAbsoluteRenderLayer

```
calculateAbsoluteRenderLayer(offset: number, step: number, relativeRenderLayer: EDefaultRenderLayer): number
```

Defined in src/Charting/Drawing/WebGlRenderContext2D.ts

Parameters

- **offset:** *number*

  layer z-order offset of the surface

- **step:** *number*

  specifies the capacity of layers that could be potentially added between the default chart layers

- **relativeRenderLayer:** *EDefaultRenderLayer*

  layer number relative to the specific surface layers

Returns *number*

absolute order of the layer on the chart (considering parent chart and previous subChart surface layers)

Const calculateCellCoordinates

```
calculateCellCoordinates(inputArr: number[], dimension: number, startInd: number, count: number, inc: number, offset: number): any[]
```

Defined in src/Charting/Visuals/RenderableSeries/DrawingProviders/HeatmapHelpers.ts

Calculates absolute coordinates of the heatmap cells

Parameters

- **inputArr:** *number*[]

  relative cell sizes

- **dimension:** *number*

  texture size

- **startInd:** *number*

- **count:** *number*

- **inc:** *number*

- **offset:** *number*

Returns *any*[]

Const calculateHeatmapTexture

```
calculateHeatmapTexture(colorDataParams: IGetColorDataParams, intVector: UIntVector, heatTextureCache: TextureCache, precision: number): TSRTexture
```

Defined in src/Charting/Visuals/RenderableSeries/DrawingProviders/HeatmapHelpers.ts

Parameters

- **colorDataParams:** *IGetColorDataParams*

- **intVector:** *UIntVector*

- **heatTextureCache:** *TextureCache*

- **precision:** *number*

Returns *TSRTexture*

Const calculateMaxGroupSize

calculateMaxGroupSize(outerLayoutSizes: *Record*<*string*, *number*>): *number*

calculateMaxGroupSize(outerLayoutSizes: *Record*<*string*, *number*>): *number*

Defined in src/Charting/LayoutManager/SciChartVerticalGroup.ts

Parameters

- **outerLayoutSizes:** *Record*<*string*, *number*>

Returns *number*

Const calculateOffsets

calculateOffsets(heatmapRect: *Rect*, isVerticalChart: *boolean*, xCellSizes: *number*[], yCellSizes: *number*[], horStartInd: *number*, horCellCount: *number*, horInc: *number*, vertStartInd: *number*, vertCellCount: *number*, vertInc: *number*, seriesViewRect: *Rect*): { horCellOffsets: *any*[]; vertCellOffsets: *any*[] }

Defined in src/Charting/Visuals/RenderableSeries/DrawingProviders/HeatmapHelpers.ts

Parameters

- **heatmapRect:** *Rect*
- **isVerticalChart:** *boolean*
- **xCellSizes:** *number*[]
- **yCellSizes:** *number*[]
- **horStartInd:** *number*
- **horCellCount:** *number*
- **horInc:** *number*
- **vertStartInd:** *number*
- **vertCellCount:** *number*
- **vertInc:** *number*
- **seriesViewRect:** *Rect*

Returns { **horCellOffsets**: *any*[]; **vertCellOffsets**: *any*[] }

- **horCellOffsets**: *any*[]
- **vertCellOffsets**: *any*[]

chartReviver

chartReviver(key: *string*, value: *any*): *chartReviver*

Defined in src/Builder/chartBuilder.ts

The reviver function needed when parsing definitions to JSON

Parameters

- **key:** *string*
- **value:** *any*

Returns *chartReviver*

Const checkAreEqualTextStyles

checkAreEqualTextStyles(style1: *TAxisTitleStyle*, style2: *TAxisTitleStyle*): *boolean*

Defined in src/Charting/Visuals/Axis/AxisTitleRenderer.ts

Parameters

- **style1:** *TAxisTitleStyle*
- **style2:** *TAxisTitleStyle*

Returns *boolean*

Const checkBuildStamp

checkBuildStamp(wasmContext: *TSciChart* | *TSciChart3D*): *boolean*

Defined in src/Core/BuildStamp.ts

Parameters

- **wasmContext:** *TSciChart* | *TSciChart3D*

Returns *boolean*

Const checkCanDraw

checkCanDraw(animationFSM: *AnimationFiniteStateMachine*): *boolean*

Defined in src/Charting/Visuals/RenderableSeries/Animations/animationHelpers.ts

Checks if can draw, is used not to draw when {@link BaseAnimation} has delay

Parameters

- **animationFSM:** *AnimationFiniteStateMachine*

Returns *boolean*

Const checkIsAnimationRunning

checkIsAnimationRunning(animationQueue: *IAnimation*[], animationFSM: *AnimationFiniteStateMachine*): *boolean*

Defined in src/Charting/Visuals/RenderableSeries/Animations/animationHelpers.ts

Checks if the animation is running

Parameters

- **animationQueue:** *IAnimation*[]

  The animation queue

- **animationFSM:** *AnimationFiniteStateMachine*

  The animation finate state machine

Returns *boolean*

Const checkIsNaN

checkIsNaN(value: *number*): *boolean*

Defined in src/utils/number.ts

Parameters

- **value:** *number*

Returns *boolean*

Const checkStyle

checkStyle(currentStyleId: *string*, newStyle: *TCachedLabelStyle*): *boolean*

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelCache.ts

Parameters

- **currentStyleId:** *string*
- **newStyle:** *TCachedLabelStyle*

Returns *boolean*

Const checkTextStyleEqual

checkTextStyleEqual(style1: *TCachedLabelStyle*, style2: *TCachedLabelStyle*): *boolean*

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelCache.ts

Parameters

- **style1:** *TCachedLabelStyle*
- **style2:** *TCachedLabelStyle*

Returns *boolean*

Const clearCacheByStyle

clearCacheByStyle(styleId: *string*): *void*

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelCache.ts

Parameters

- **styleId:** *string*

Returns *void*

clearLicensingDebug

clearLicensingDebug(): *clearLicensingDebug*

Defined in src/Core/storage/localStorageApi.ts

Returns *clearLicensingDebug*

configure2DSurface

configure2DSurface(definition: *ISciChart2DDefinition*, sciChartSurface: *SciChartSurface*, wasmContext: *TSciChart*): *configure2DSurface*

Defined in src/Builder/buildSurface.ts

Parameters

- **definition:** *ISciChart2DDefinition*
- **sciChartSurface:** *SciChartSurface*
- **wasmContext:** *TSciChart*

Returns *configure2DSurface*

Const configureChart

configureChart(sciChartSurface: *SciChartSurfaceBase*, wasmContext: *TSciChart*, definition: *string* | *TSurfaceDefinition*): *void*

Defined in src/Builder/chartBuilder.ts

Configures an existing surface using a definition. This is useful if you need to use the wasmContext in methods or classes you use in your definition

Parameters

- **sciChartSurface:** *SciChartSurfaceBase*
- **wasmContext:** *TSciChart*

  The webassemply context. Pass undefined for a pie surface.
- **definition:** *string* | *TSurfaceDefinition*

Returns *void*

Const convertColor

convertColor(htmlColor: *string*, opacity?: *number*): *string*

Defined in src/utils/convertColor.ts

Parameters

- **htmlColor:** *string*
- Default value **opacity:** *number* = 1

Returns *string*

rgbColor, for example "0xff00ff00" - green

Const convertLabelAlignmentToTextAlignment

convertLabelAlignmentToTextAlignment(alignment: *ELabelAlignment*): *ETextAlignment*

Defined in src/Charting/Visuals/Axis/AxisTitleRenderer.ts

Parameters

- **alignment:** *ELabelAlignment*

Returns *ETextAlignment*

Const convertMultiLineAlignment

convertMultiLineAlignment(multiLineAlignment: *EMultiLineAlignment*, webAssemblyContext: *TSciChart*): *eTSRTextAlignMode*

convertMultiLineAlignment(multiLineAlignment: *ELabelAlignment*, webAssemblyContext: *TSciChart*): *eTSRTextAlignMode*

Defined in src/types/TextPosition.ts

Parameters

- **multiLineAlignment:** *EMultiLineAlignment*
- **webAssemblyContext:** *TSciChart*

Returns *eTSRTextAlignMode*

Const convertRgbToHexColor

convertRgbToHexColor(r: *number*, g: *number*, b: *number*): *string*

Defined in src/utils/convertColor.ts

Converts individual R,G, and B components to HEX Color

Parameters

- **r:** *number*
- **g:** *number*
- **b:** *number*

Returns *string*

argbColor, for example "0xff00ff00" - green

Const convertSearchMode

convertSearchMode(wasmContext: *TSciChart*, mode: *ESearchMode*): *SCRTFindIndexSearchMode*

Defined in src/types/SearchMode.ts

Converts ESearchMode (typescript friendly Enum) to SCRTFindIndexSearchMode which is

required by the webassembly engine

Parameters

- **wasmContext:** *TSciChart*
- **mode:** *ESearchMode*

Returns *SCRTFindIndexSearchMode*

Const convertToHtmlPx

`convertToHtmlPx(value: number): string`

Defined in src/utils/translate.ts

Parameters

- **value:** *number*

Returns *string*

Const convertToPixel

`convertToPixel(red: number, green: number, blue: number, opacity: number): number`

Defined in src/utils/convertToPixel.ts

Parameters

- **red:** *number*
  number value from 0 to 255
- **green:** *number*
  number value from 0 to 255
- **blue:** *number*
  number value from 0 to 255
- **opacity:** *number*
  number value from 0 to 255

Returns *number*

pixel in hex format: opacity, red, green, blue. For example: "0xff0000ff" - blue pixel with no opacity

Const convertToRelativeHtmlSize

`convertToRelativeHtmlSize(value: number): string`

Defined in src/utils/translate.ts

Parameters

- **value:** *number*

Returns *string*

Const copyDoubleVector

`copyDoubleVector(source: SCRTDoubleVector, target: SCRTDoubleVector, wasmContext: TSciChart): void`

Defined in src/utils/copyVector.ts

Parameters

- **source:** *SCRTDoubleVector*
- **target:** *SCRTDoubleVector*
- **wasmContext:** *TSciChart*

Returns *void*

Const copyVector

```
copyVector(sourceVector: SCRTDoubleVector, targetVector: SCRTDoubleVector): void
```

Defined in src/Charting/Visuals/RenderableSeries/Animations/animationHelpers.ts

deprecated Use copyDoubleVector instead

Parameters

- **sourceVector:** *SCRTDoubleVector*
- **targetVector:** *SCRTDoubleVector*

Returns *void*

countUnique

```
countUnique(iterable: string[]): countUnique
```

Defined in src/utils/array.ts

Parameters

- **iterable:** *string*[]

Returns *countUnique*

Const createBrushInCache

```
createBrushInCache(cache: BrushCache, fill: string, opacity: number,
textureHeightRatio?: number, textureWidthRatio?: number, fillGradientLinear?:
GradientParams): SCRTBrush
```

Defined in src/Charting/Drawing/BrushCache.ts

Creates a native {@link SCRTBrush} brush from html color code string passed in and caches it

Parameters

- **cache:** *BrushCache*
  The object that will store a brush
- **fill:** *string*
  The HTML Color code
- **opacity:** *number*
  The opacity factor.
- Optional **textureHeightRatio:** *number*
  The height ratio of the main canvas to the WebGl canvas.
- Optional **textureWidthRatio:** *number*
  The width ratio of the main canvas to the WebGl canvas.
- Optional **fillGradientLinear:** *GradientParams*
  The gradient params.

Returns *SCRTBrush*

new or existing instance of {@link SCRTBrush}}

Const createChartDestination

```
createChartDestination(canvas: HTMLCanvasElement): { canvas: HTMLCanvasElement;
GetHeight: any; GetID: any; GetWidth: any }
```

Defined in src/Charting/Visuals/SciChartSurfaceBase.ts

Parameters

- **canvas:** *HTMLCanvasElement*

Returns { **canvas**: *HTMLCanvasElement*; **GetHeight**: *any*; **GetID**: *any*; **GetWidth**: *any* }

- **canvas**: *HTMLCanvasElement*
- **GetHeight**: **function**

```
GetHeight(): any
```

Defined in src/Charting/Visuals/SciChartSurfaceBase.ts

Returns *any*

- **GetID**: **function**

```
GetID(): any
```

Defined in src/Charting/Visuals/SciChartSurfaceBase.ts

Returns *any*

- **GetWidth**: **function**

```
GetWidth(): any
```

Defined in src/Charting/Visuals/SciChartSurfaceBase.ts

Returns *any*

Const createColorMap

```
createColorMap(originalGradientStops: TGradientStop[], precision: number): any[]
```

Defined in src/Charting/Visuals/RenderableSeries/DrawingProviders/HeatmapHelpers.ts

Parameters

- **originalGradientStops:** *TGradientStop*[]
- **precision:** *number*

Returns *any*[]

Const createDataSeries

```
createDataSeries(wasmContext: TSciChart, dataSeriesDefinition: TDataSeriesDefinition,
sharedData?: TSharedDataDefinition): UniformHeatmapDataSeries | XyyDataSeries |
OhlcDataSeries | NonUniformHeatmapDataSeries | XyDataSeries | HlcDataSeries |
XyTextDataSeries | XyzDataSeries
```

Defined in src/Builder/buildDataSeries.ts

Parameters

- **wasmContext:** *TSciChart*
- **dataSeriesDefinition:** *TDataSeriesDefinition*
- Optional **sharedData:** *TSharedDataDefinition*

Returns *UniformHeatmapDataSeries* | *XyyDataSeries* | *OhlcDataSeries* | *NonUniformHeatmapDataSeries* | *XyDataSeries* | *HlcDataSeries* | *XyTextDataSeries* | *XyzDataSeries*

Const createHitTestInfo

```
createHitTestInfo(renderableSeries: IRenderableSeries, xCoordinateCalculator:
CoordinateCalculatorBase, yCoordinateCalculator: CoordinateCalculatorBase,
isVerticalChart: boolean, dataSeries: BaseDataSeries, xNativeValues: SCRTDoubleVector,
yNativeValues: SCRTDoubleVector, xHitCoord: number, yHitCoord: number,
nearestPointIndex: number, hitTestRadius: number, distance?: number): HitTestInfo
```

Defined in src/Charting/Visuals/RenderableSeries/HitTest/hitTestHelpers.ts

Parameters

- **renderableSeries:** *IRenderableSeries*

- **xCoordinateCalculator:** *CoordinateCalculatorBase*
- **yCoordinateCalculator:** *CoordinateCalculatorBase*
- **isVerticalChart:** *boolean*
- **dataSeries:** *BaseDataSeries*
- **xNativeValues:** *SCRTDoubleVector*
- **yNativeValues:** *SCRTDoubleVector*
- **xHitCoord:** *number*

  the X coordinate on the screen relative to seriesViewRect, X and Y swapped for vertical charts
- **yHitCoord:** *number*

  the Y coordinate on the screen relative to seriesViewRect, X and Y swapped for vertical charts
- **nearestPointIndex:** *number*
- **hitTestRadius:** *number*
- Optional **distance:** *number*

Returns *HitTestInfo*

createImageAsync

```
createImageAsync(src: string): Promise<HTMLImageElement>
```

Defined in src/utils/imageUtil.ts

Helper function to create an HTML Image {@link HTMLImageElement} asychronously by passing in the string image source

remarks Returns a promise, await this to get the image element

Parameters

- **src:** *string*

Returns *Promise<HTMLImageElement>*

createImagesArrayAsync

```
createImagesArrayAsync(images: string[]): Promise<HTMLImageElement[]>
```

Defined in src/utils/imageUtil.ts

Helper function to create an HTML Images {@link HTMLImageElement} asychronously by passing in the string array

Parameters

- **images:** *string[]*

Returns *Promise<HTMLImageElement[]>*

Const createNativeRect

```
createNativeRect(webAssemblyContext: TSciChart, xTopLeft: number, yTopLeft: number,
xBottomRight: number, yBottomRight: number): SCRTRectVertex
```

Defined in src/Charting/Visuals/Helpers/createNativeRect.ts

Helper function to create a {@link SCRTRectVertex} native rectangle vertex

Parameters

- **webAssemblyContext:** *TSciChart*

  The SciChart 2D WebAssembly Context containing native methods and access to our WebGL2 Engine and WebAssembly numerical methods
- **xTopLeft:** *number*
- **yTopLeft:** *number*

- **xBottomRight:** *number*
- **yBottomRight:** *number*

Returns *SCRTRectVertex*

Const createPenInCache

```
createPenInCache(penCache: Pen2DCache, stroke: string, strokeThickness: number,
opacity: number, strokeDashArray?: number[], antiAliased?: boolean): SCRTPen
```

Defined in src/Charting/Drawing/Pen2DCache.ts

Creates a native {@link SCRTPen} Pen from html color code string passed in and caches it

Parameters

- **penCache:** *Pen2DCache*

  The object that will store a pen

- **stroke:** *string*

  The HTML Color code

- **strokeThickness:** *number*

  The strokethickness in pixels

- **opacity:** *number*

  The opacity factor

- Optional **strokeDashArray:** *number*[]

  the StrokeDashArray which defines any dash e.g. [2,2] means dash for 2pts, gap for 2pts (or undefined = solid line).

- Optional **antiAliased:** *boolean*

Returns *SCRTPen*

the new or existing instance of {@link SCRTPen}}

Const createPointMarker

```
createPointMarker(wasmContext: TSciChart, pointMarkerStyle: BasePointMarkerStyle):
EllipsePointMarker | SpritePointMarker | CrossPointMarker | XPointMarker |
SquarePointMarker | TrianglePointMarker
```

Defined in src/Charting/Visuals/RenderableSeries/Animations/animationHelpers.ts

Creates the point marker

Parameters

- **wasmContext:** *TSciChart*
- **pointMarkerStyle:** *BasePointMarkerStyle*

Returns *EllipsePointMarker* | *SpritePointMarker* | *CrossPointMarker* | *XPointMarker* | *SquarePointMarker* | *TrianglePointMarker*

Const createSCRTPen

```
createSCRTPen(wasmContext: TSciChart, htmlColorCode: string, strokeThickness: number,
opacity: number, strokeDashArray?: number[], antiAliased?: boolean): SCRTPen
```

Defined in src/Charting/Visuals/Helpers/createPen.ts

Helper function to create a {@link SCRTPen} native pen access to our WebGL2 Engine and WebAssembly numerical methods

Parameters

- **wasmContext:** *TSciChart*

  The SciChart WebAssembly Context containing native methods and access to our WebGL2 WebAssembly Drawing Engine

- **htmlColorCode:** *string*

Html color code in the format "#fff", "#ff0000", "rgba(255,255,0,1)", "#11333333"

- **strokeThickness:** *number*

  the stroke thickness of the pen in pixels

- **opacity:** *number*

  The opacity factor

- Optional **strokeDashArray:** *number*[]

  the StrokeDashArray which defines any dash e.g. [2,2] means dash for 2pts, gap for 2pts (or undefined = solid line).

- Default value **antiAliased:** *boolean* = true

Returns *SCRTPen*

Const createSingle3dInternal

```
createSingle3dInternal(divElement: string | HTMLDivElement, options?: I2DSurfaceOptions): Promise<TWebAssemblyChart3D>
```

Defined in src/Charting3D/Visuals/createSingle3d.ts

Parameters

- **divElement:** *string* | *HTMLDivElement*
- Optional **options:** *I2DSurfaceOptions*

Returns *Promise<TWebAssemblyChart3D>*

Const createSolidBrush

```
createSolidBrush(wasmContext: TSciChart, htmlColorCode: string, opacity: number): SCRTBrush
```

Defined in src/Charting/Visuals/Helpers/createSolidBrush.ts

Helper function to create a {@link SCRTBrush} native solid color brushes

Parameters

- **wasmContext:** *TSciChart*
- **htmlColorCode:** *string*

  Html color code in the format "#fff", "#ff0000", "rgba(255,255,0,1)", "#11333333"

- **opacity:** *number*

Returns *SCRTBrush*

Const createSvg

```
createSvg(svgString: string, svgRoot: Node, nextElement?: Element): SVGElement
```

Defined in src/Charting/Visuals/Annotations/annotationHelpers.ts

All svg creation is run through this funciton so it can be mocked for tests

Parameters

- **svgString:** *string*
- **svgRoot:** *Node*
- Optional **nextElement:** *Element*

Returns *SVGElement*

Const createType

```
createType(baseType: EBaseType, type: string, wasmContext: TSciChart | TSciChart3D, options: any): any
```

Defined in src/Builder/classFactory.ts

Parameters

- **baseType:** *EBaseType*
- **type:** *string*
- **wasmContext:** *TSciChart* | *TSciChart3D*
- **options:** *any*

Returns *any*

Const deleteCache

```
deleteCache(wasmContext: TSciChart | TSciChart3D): void
```

Defined in src/Charting/Visuals/Helpers/NativeObject.ts

Parameters

- **wasmContext:** *TSciChart* | *TSciChart3D*

Returns *void*

deleteSafe

```
deleteSafe<T>(object: T): T
```

Defined in src/Core/Deleter.ts

Checks if an object implements IDeletable, then calls IDeletable.delete

Type parameters

- **T**: *IDeletable*

Parameters

- **object:** *T*

Returns *T*

Const drawAxisMarkerAnnotation

```
drawAxisMarkerAnnotation(currentAxis: AxisBase2D, renderContext: WebGlRenderContext2D,
displayValue: string, markerCoordinate: number, x1Coord: number, y1Coord: number,
textStyle: TTextStyle, fill: string, opacity: number, image: HTMLImageElement,
imageWidth: number, imageHeight: number): { textureHeight: number; textureWidth:
number; xPosition: number; yPosition: number }
```

Defined in src/Charting/Visuals/Helpers/drawLabel.ts

Parameters

- **currentAxis:** *AxisBase2D*
- **renderContext:** *WebGlRenderContext2D*
- **displayValue:** *string*
- **markerCoordinate:** *number*
- **x1Coord:** *number*
- **y1Coord:** *number*
- **textStyle:** *TTextStyle*
- **fill:** *string*
- **opacity:** *number*
- **image:** *HTMLImageElement*
- **imageWidth:** *number*
- **imageHeight:** *number*

Returns { **textureHeight**: *number*; **textureWidth**: *number*; **xPosition**: *number*; **yPosition**: *number* }

- **textureHeight**: *number*
- **textureWidth**: *number*
- **xPosition**: *number*
- **yPosition**: *number*

Const drawBorder

```
drawBorder(renderContext: WebGlRenderContext2D, webAssemblyContext2D: TSciChart,
solidBrushCacheBorder: SolidBrushCache, borderRect: Rect, leftBorder: number,
topBorder: number, rightBorder: number, bottomBorder: number, color: string): void
```

Defined in src/Charting/Visuals/Helpers/drawBorder.ts

Parameters

- **renderContext:** *WebGlRenderContext2D*
- **webAssemblyContext2D:** *TSciChart*
- **solidBrushCacheBorder:** *SolidBrushCache*
- **borderRect:** *Rect*
- **leftBorder:** *number*
- **topBorder:** *number*
- **rightBorder:** *number*
- **bottomBorder:** *number*
- **color:** *string*

Returns *void*

Const drawLineAnnotation

```
drawLineAnnotation(currentAxis: AxisBase2D, renderContext: WebGlRenderContext2D,
labelPlacement: ELabelPlacement, displayValue: string, x1Coord: number, x2Coord:
number, y1Coord: number, y2Coord: number, textStyle: TTextStyle, fill: string,
strokePen: IPen2D, viewRect: Rect, showLabel: boolean, opacity: number,
horizontalAlignment?: EHorizontalAlignment, verticalAlignment?: EVerticalAlignment):
Rect
```

Defined in src/Charting/Visuals/Helpers/drawLabel.ts

Function to draw Vertical or Horizontal Line annotations with labels

Parameters

- **currentAxis:** *AxisBase2D*
- **renderContext:** *WebGlRenderContext2D*
- **labelPlacement:** *ELabelPlacement*
- **displayValue:** *string*
- **x1Coord:** *number*
- **x2Coord:** *number*
- **y1Coord:** *number*
- **y2Coord:** *number*
- **textStyle:** *TTextStyle*
- **fill:** *string*
- **strokePen:** *IPen2D*
- **viewRect:** *Rect*
- **showLabel:** *boolean*
- **opacity:** *number*

- Optional **horizontalAlignment:** *EHorizontalAlignment*
- Optional **verticalAlignment:** *EVerticalAlignment*

Returns *Rect*

Const drawModifiersAxisLabel

```
drawModifiersAxisLabel(currentAxis: AxisBase2D, renderContext: WebGlRenderContext2D,
labelCoord: number, fill: string, stroke: string): Rect
```

Defined in src/Charting/Visuals/Helpers/drawLabel.ts

Parameters

- **currentAxis:** *AxisBase2D*
- **renderContext:** *WebGlRenderContext2D*
- **labelCoord:** *number*
- **fill:** *string*
- **stroke:** *string*

Returns *Rect*

Const ensureRegistrations

```
ensureRegistrations(): void
```

Defined in src/Builder/chartBuilder.ts

This is just something to call to ensure that all the registrations are run before a surface is created

Returns *void*

Const fillMetadata

```
fillMetadata(width: number, height: number, metadata: IPointMetadata[][]):
IPointMetadata[][]
```

Defined in src/Charting/Model/BaseHeatmapDataSeries.ts

Parameters

- **width:** *number*
- **height:** *number*
- **metadata:** *IPointMetadata*[][]

Returns *IPointMetadata*[][]

Const fillNoisySinewave

```
fillNoisySinewave(pointCount: number, xMax: number, frequency: number, amplitude:
number, noiseAmplitude: number, dataSeries: XyDataSeries): void
```

Defined in src/utils/math.ts

Parameters

- **pointCount:** *number*
- **xMax:** *number*
- **frequency:** *number*
- **amplitude:** *number*
- **noiseAmplitude:** *number*
- **dataSeries:** *XyDataSeries*

Returns *void*

Const fitElementToViewRect

```
fitElementToViewRect(element: HTMLElement | SVGSVGElement, viewRect: Rect): void
```

Defined in src/utils/translate.ts

Parameters

- **element:** *HTMLElement | SVGSVGElement*
- **viewRect:** *Rect*

Returns *void*

Const fitSvgToViewRect

```
fitSvgToViewRect(svgElement: SVGSVGElement, viewRect: Rect): void
```

Defined in src/utils/translate.ts

Parameters

- **svgElement:** *SVGSVGElement*
- **viewRect:** *Rect*

Returns *void*

Const formatNumber

```
formatNumber(dataValue: number, numericFormat: ENumericFormat, precision: number,
engineeringPrefix?: IEngineeringPrefix): string
```

Defined in src/utils/number.ts

Parameters

- **dataValue:** *number*
- **numericFormat:** *ENumericFormat*
- **precision:** *number*
- Optional **engineeringPrefix:** *IEngineeringPrefix*

Returns *string*

Const formatNumber2Digits

```
formatNumber2Digits(value: number): string
```

Defined in src/utils/number.ts

description Formats value always to have 2 decimal digits, e.g. 12.45, 14.20, 17.00

Parameters

- **value:** *number*

Returns *string*

Const formatUnixDateToHumanString

```
formatUnixDateToHumanString(unixTimestamp: number, locale?: string): string
```

Defined in src/utils/date.ts

Result 11/23/2018

Parameters

- **unixTimestamp:** *number*
- Default value **locale:** *string* = "en-US"

Returns *string*

Const formatUnixDateToHumanStringDD

`formatUnixDateToHumanStringDD(unixTimestamp: number): string`

Defined in src/utils/date.ts

Parameters

- **unixTimestamp:** *number*

Returns *string*

Const formatUnixDateToHumanStringDDMM

`formatUnixDateToHumanStringDDMM(unixTimestamp: number): string`

Defined in src/utils/date.ts

Parameters

- **unixTimestamp:** *number*

Returns *string*

Const formatUnixDateToHumanStringDDMMHHMM

`formatUnixDateToHumanStringDDMMHHMM(unixTimestamp: number): string`

Defined in src/utils/date.ts

Parameters

- **unixTimestamp:** *number*

Returns *string*

Const formatUnixDateToHumanStringDDMMYY

`formatUnixDateToHumanStringDDMMYY(unixTimestamp: number): string`

Defined in src/utils/date.ts

Parameters

- **unixTimestamp:** *number*

Returns *string*

Const formatUnixDateToHumanStringHHMM

`formatUnixDateToHumanStringHHMM(unixTimestamp: number): string`

Defined in src/utils/date.ts

Parameters

- **unixTimestamp:** *number*

Returns *string*

Const formatUnixDateToHumanStringHHMMSS

`formatUnixDateToHumanStringHHMMSS(unixTimestamp: number): string`

Defined in src/utils/date.ts

Parameters

- **unixTimestamp:** *number*

Returns *string*

Const formatUnixDateToHumanStringMMM

`formatUnixDateToHumanStringMMM(unixTimestamp: number): string`

Defined in src/utils/date.ts

Parameters

- **unixTimestamp:** *number*

Returns *string*

Const formatUnixDateToHumanStringMMMDD

`formatUnixDateToHumanStringMMMDD(unixTimestamp: number): string`

Defined in src/utils/date.ts

Parameters

- **unixTimestamp:** *number*

Returns *string*

Const formatUnixDateToHumanStringSSms

`formatUnixDateToHumanStringSSms(unixTimestamp: number): string`

Defined in src/utils/date.ts

Parameters

- **unixTimestamp:** *number*

Returns *string*

Const formatUnixDateToHumanStringYYYY

`formatUnixDateToHumanStringYYYY(unixTimestamp: number): string`

Defined in src/utils/date.ts

Parameters

- **unixTimestamp:** *number*

Returns *string*

Const freeCache

`freeCache(wasmContext: TSciChart): IDeletable`

Defined in src/Charting/Visuals/Helpers/NativeObject.ts

Parameters

- **wasmContext:** *TSciChart*

Returns *IDeletable*

Const freeStyle

freeStyle(styleId: *string*): *void*

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelCache.ts

Parameters

- **styleId:** *string*

Returns *void*

Const fromTsrVector4

fromTsrVector4(tsrColor: *TSRVector4*): *TArgb*

Defined in src/utils/tsrExtensions.ts

Parameters

- **tsrColor:** *TSRVector4*

Returns *TArgb*

Const generateBooleanHash

generateBooleanHash(value: *boolean*): *number*

Defined in src/utils/hash.ts

Parameters

- **value:** *boolean*

Returns *number*

Const generateCombinedHash

generateCombinedHash(hashes: *number*[]): *number*

Defined in src/utils/hash.ts

Parameters

- **hashes:** *number*[]

Returns *number*

Const generateGuid

generateGuid(): *string*

Defined in src/utils/guid.ts

description Generates GUID/UUID RFC4122 version 4 compliant

Returns *string*

Const generateHash

generateHash(s: *string*): *number*

Defined in src/utils/hash.ts

Parameters

- **s:** *string*

Returns *number*

Const generateNumberHash

generateNumberHash(value: *number*): *number*

Defined in src/utils/hash.ts

Parameters

- **value:** *number*

Returns *number*

Const generateObjectHash

generateObjectHash(obj: *any*): *number*

Defined in src/utils/hash.ts

Parameters

- **obj:** *any*

Returns *number*

Const getActiveAxes

getActiveAxes(xAxisArr: *AxisBase2D*[], mousePoint: *Point*): *AxisBase2D*[]

Defined in src/Charting/ChartModifiers/ChartModifierBase2D.ts

Parameters

- **xAxisArr:** *AxisBase2D*[]
- **mousePoint:** *Point*

Returns *AxisBase2D*[]

Const getAdjustedRotation

getAdjustedRotation(rotation: *number*, position: *ETitlePosition*): *number*

Defined in src/Charting/Services/TitleRenderer.ts

Parameters

- **rotation:** *number*
- **position:** *ETitlePosition*

Returns *number*

Const getAllFontKeys

getAllFontKeys(webAssemblyContext: *TSciChart* | *TSciChart3D*): *SCRTFontKey*[]

Defined in src/Charting/Visuals/Helpers/NativeObject.ts

Parameters

- **webAssemblyContext:** *TSciChart* | *TSciChart3D*

Returns *SCRTFontKey*[]

getArraysEqual

getArraysEqual<T>(a: *T*[], b: *T*[]): *boolean*

Defined in src/Charting3D/Visuals/Axis/IAxisDescriptor.ts

Type parameters

- **T**

Parameters

- **a:** *T*[]
- **b:** *T*[]

Returns *boolean*

Const getAttributeFromString

`getAttributeFromString(str: string, attributeName: string): number`

Defined in src/utils/svgString.ts

Parameters

- **str:** *string*
- **attributeName:** *string*

Returns *number*

getAxis3dById

`getAxis3dById(axes: ObservableArray<AxisBase3D>, axisId: string): getAxis3dById`

Defined in src/Charting/Visuals/Axis/getAxisById.ts

Parameters

- **axes:** *ObservableArray<AxisBase3D>*
- **axisId:** *string*

Returns *getAxis3dById*

getAxisById

`getAxisById(axes: ObservableArray<AxisBase2D>, axisId: string): getAxisById`

Defined in src/Charting/Visuals/Axis/getAxisById.ts

Parameters

- **axes:** *ObservableArray<AxisBase2D>*
- **axisId:** *string*

Returns *getAxisById*

getAxisGeneric

`getAxisGeneric<T>(axes: ObservableArray<T>, axisId: string): getAxisGeneric`

Defined in src/Charting/Visuals/Axis/getAxisById.ts

Type parameters

- **T**: *AxisCore*

Parameters

- **axes:** *ObservableArray<T>*
- **axisId:** *string*

Returns *getAxisGeneric*

Const getCache

`getCache(wasmContext: TSciChart | TSciChart3D): { keyCache: Map<string, FontKey>; rect: SCRTRectVertex; textBounds: TSRTextBounds; vecColorVertex: VectorColorVertex; vecRects: VectorRectVertex; vector4: TSRVector4; vertex: SCRTColorVertex }`

Defined in src/Charting/Visuals/Helpers/NativeObject.ts

Parameters

- **wasmContext:** *TSciChart* | *TSciChart3D*

Returns { **keyCache**: *Map<string, FontKey>*; **rect**: *SCRTRectVertex*; **textBounds**: *TSRTextBounds*; **vecColorVertex**: *VectorColorVertex*; **vecRects**: *VectorRectVertex*; **vector4**: *TSRVector4*; **vertex**: *SCRTColorVertex* }

- **keyCache**: *Map<string, FontKey>*
- **rect**: *SCRTRectVertex*
- **textBounds**: *TSRTextBounds*
- **vecColorVertex**: *VectorColorVertex*
- **vecRects**: *VectorRectVertex*
- **vector4**: *TSRVector4*
- **vertex**: *SCRTColorVertex*

Const getColor

```
getColor(yIndex: number, xIndex: number, colorPalette: number[], opacity: number,
zValues: number[][], webAssemblyContext: TSciChart, colorMin: number, colorMax: number,
arrayWidth: number, arrayHeight: number, fillValuesOutOfRange: boolean, precision:
number): number
```

Defined in src/Charting/Visuals/RenderableSeries/DrawingProviders/HeatmapHelpers.ts

Parameters

- **yIndex:** *number*
- **xIndex:** *number*
- **colorPalette:** *number*[]
- **opacity:** *number*
- **zValues:** *number*[][]
- **webAssemblyContext:** *TSciChart*
- **colorMin:** *number*
- **colorMax:** *number*
- **arrayWidth:** *number*
- **arrayHeight:** *number*
- **fillValuesOutOfRange:** *boolean*
- **precision:** *number*

Returns *number*

Const getColorDataForTexture

```
getColorDataForTexture(params: IGetColorDataParams, _colorData: UIntVector, precision:
number): UIntVector
```

Defined in src/Charting/Visuals/RenderableSeries/DrawingProviders/HeatmapHelpers.ts

Parameters

- **params:** *IGetColorDataParams*
- **_colorData:** *UIntVector*
- **precision:** *number*

Returns *UIntVector*

Const getCoordinateWithCoordinateMode

getCoordinateWithCoordinateMode(value: *number*, calculator: *CoordinateCalculatorBase*, coordinateMode: *EInnerAxisPlacementCoordinateMode*): *number*

Defined in src/Charting/LayoutManager/AxisLayoutHelpers.ts

Converts a value into a pixel coordinate accordingly to the coordinate mode

Parameters

- **value:** *number*
  the value to convert
- **calculator:** *CoordinateCalculatorBase*
  the CoordinateCalculatorBase which will do the transformation
- **coordinateMode:** *EInnerAxisPlacementCoordinateMode*
  the ECoordinateMode to apply

Returns *number*

the pixel coordinate

Const getDataSeriesDefinition

getDataSeriesDefinition(seriesDefinition: *TSeriesDefinition*): *TDataSeriesDefinition*

Defined in src/Builder/buildSeries.ts

Parameters

- **seriesDefinition:** *TSeriesDefinition*

Returns *TDataSeriesDefinition*

Const getDataSeriesFromRenderableSeriesDefinition

getDataSeriesFromRenderableSeriesDefinition(wasmContext: *TSciChart*, renderableSeriesDefinition: *TSeriesDefinition*, sharedData: *TSharedDataDefinition*): *IDataSeries*

Defined in src/Builder/buildSeries.ts

Parameters

- **wasmContext:** *TSciChart*
- **renderableSeriesDefinition:** *TSeriesDefinition*
- **sharedData:** *TSharedDataDefinition*

Returns *IDataSeries*

getDescriptorsEqual

getDescriptorsEqual(a: *IAxisDescriptor*, b: *IAxisDescriptor*): *boolean*

Defined in src/Charting3D/Visuals/Axis/IAxisDescriptor.ts

Returns true if descriptors are deeply equal, else false

Parameters

- **a:** *IAxisDescriptor*
- **b:** *IAxisDescriptor*

Returns *boolean*

Const getDevCount

getDevCount(): *number*

Defined in src/Charting/Visuals/licenseManager2dState.ts

Returns *number*

Const getFontKey

```
getFontKey(webAssemblyContext: TSciChart | TSciChart3D, labelStyle: TTextStyle,
advanced?: boolean, transformed?: boolean): SCRTFontKey
```

Defined in src/Charting/Visuals/Helpers/NativeObject.ts

get a fontKey required to aquire a native font

Parameters

- **webAssemblyContext:** *TSciChart* | *TSciChart3D*
- **labelStyle:** *TTextStyle*
- Default value **advanced:** *boolean* = false
- Default value **transformed:** *boolean* = false

  set true to get an alternative instance of the font which can be used multiple times while transformations are in effect, without disrupting global font rendering

Returns *SCRTFontKey*

Const getFontString

```
getFontString(fontStyle: string, fontWeight: string, fontSize: number, fontFamily:
string): string
```

Defined in src/utils/font.ts

Creates the font string, which is used to set font on CanvasRenderingContext2D

Parameters

- **fontStyle:** *string*

  The font style
- **fontWeight:** *string*

  The font weight
- **fontSize:** *number*

  The font size in pixels
- **fontFamily:** *string*

  The font family

Returns *string*

Const getFunction

```
getFunction(baseType: EBaseType, type: string): Function
```

Defined in src/Builder/classFactory.ts

Parameters

- **baseType:** *EBaseType*
- **type:** *string*

Returns *Function*

getHorizontalAxisRequiredSize

```
getHorizontalAxisRequiredSize(axisLayoutState: AxisLayoutState): number
```

Defined in src/Charting/LayoutManager/AxisLayoutHelpers.ts

Parameters

- **axisLayoutState:** *AxisLayoutState*

Returns *number*

Const getIncludedAxis

```
getIncludedAxis(allAxes: AxisBase2D[], axisMap: Map<string, boolean>): AxisBase2D[]
```

Defined in src/utils/includedAxis.ts

Parameters

- **allAxes:** *AxisBase2D*[]
- **axisMap:** *Map*<*string*, *boolean*>

Returns *AxisBase2D*[]

Const getInterpolatedColor

```
getInterpolatedColor(htmlColor1: string, htmlColor2: string, coef: number): number
```

Defined in src/Charting/Visuals/RenderableSeries/DrawingProviders/HeatmapHelpers.ts

Parameters

- **htmlColor1:** *string*
- **htmlColor2:** *string*
- **coef:** *number*

Returns *number*

Const getIsDev

```
getIsDev(): boolean
```

Defined in src/Charting/Visuals/licenseManager2dState.ts

Returns *boolean*

Const getIsHorizontal

```
getIsHorizontal(alignment: EAxisAlignment): boolean
```

Defined in src/types/AxisAlignment.ts

Parameters

- **alignment:** *EAxisAlignment*

Returns *boolean*

getIsLicenseDebug

```
getIsLicenseDebug(): boolean
```

Defined in src/Core/storage/localStorageApi.ts

Returns *boolean*

Const getIsVertical

```
getIsVertical(alignment: EAxisAlignment): boolean
```

Defined in src/types/AxisAlignment.ts

Parameters

- **alignment:** *EAxisAlignment*

Returns *boolean*

Const getKey

`getKey(rs: IRenderableSeries, index: number): string`

Defined in src/Charting/ChartModifiers/DataPointSelectionModifier.ts

Parameters

- **rs:** *IRenderableSeries*
- **index:** *number*

Returns *string*

Const getLabel

`getLabel(text: string, styleId: string): LabelInfo`

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelCache.ts

Parameters

- **text:** *string*
- **styleId:** *string*

Returns *LabelInfo*

Const getLabelCoordinates

`getLabelCoordinates(currentAxis: AxisBase2D, labelPlacement: ELabelPlacement, x1Coord: number, x2Coord: number, y1Coord: number, y2Coord: number, textureHeight: number, textureWidth: number, horizontalAlignment?: EHorizontalAlignment, verticalAlignment?: EVerticalAlignment): { xPosition: number; yPosition: number }`

Defined in src/Charting/Visuals/Helpers/drawLabel.ts

Calculates coordinates of the annotation label. The coordinates are defined as an absolute position on the whole SciChartSurface.

Parameters

- **currentAxis:** *AxisBase2D*
- **labelPlacement:** *ELabelPlacement*
- **x1Coord:** *number*
- **x2Coord:** *number*
- **y1Coord:** *number*
- **y2Coord:** *number*
- **textureHeight:** *number*
- **textureWidth:** *number*
- Optional **horizontalAlignment:** *EHorizontalAlignment*
- Optional **verticalAlignment:** *EVerticalAlignment*

Returns { **xPosition**: *number*; **yPosition**: *number* }

- **xPosition**: *number*
- **yPosition**: *number*

Const getLabelValue

`getLabelValue(currentAxis: AxisBase2D, labelCoord: number): string`

Defined in src/Charting/Visuals/Helpers/drawLabel.ts

Parameters

- **currentAxis:** *AxisBase2D*
- **labelCoord:** *number*

Returns *string*

Const getLegendContainerHtml

```
getLegendContainerHtml(placement: ELegendPlacement, textColor: string, backgroundColor: string, margin: Thickness, body: string, isExternal?: boolean): string
```

Defined in src/Charting/Visuals/Legend/SciChartLegendBase.ts

Parameters

- **placement:** *ELegendPlacement*
- **textColor:** *string*
- **backgroundColor:** *string*
- **margin:** *Thickness*
- **body:** *string*
- Default value **isExternal:** *boolean* = false

Returns *string*

Const getLegendItemHtml

```
getLegendItemHtml(orientation: ELegendOrientation, showCheckboxes: boolean, showSeriesMarkers: boolean, item: TLegendItem): string
```

Defined in src/Charting/Visuals/Legend/SciChartLegendBase.ts

Parameters

- **orientation:** *ELegendOrientation*
- **showCheckboxes:** *boolean*
- **showSeriesMarkers:** *boolean*
- **item:** *TLegendItem*

Returns *string*

Const getLicenseType

```
getLicenseType(): LicenseType
```

Defined in src/Charting/Visuals/licenseManager2dState.ts

Returns *LicenseType*

getLicenseWizardMaxPort

```
getLicenseWizardMaxPort(): number
```

Defined in src/Core/storage/localStorageApi.ts

Returns *number*

getLicenseWizardPort

```
getLicenseWizardPort(): number
```

Defined in src/Core/storage/localStorageApi.ts

Returns *number*

Const getLineCoordinates

```
getLineCoordinates(x1Coord: number, y1Coord: number, x2Coord: number, y2Coord: number,
labelHeight: number, labelWidth: number, labelPlacement: ELabelPlacement, currentAxis:
AxisBase2D): { x1LineCoord: number; x2LineCoord: number; y1LineCoord: number;
y2LineCoord: number }
```

Defined in src/Charting/Visuals/Helpers/drawLabel.ts

Calculates annotation line coordinates accordingly to axis alignment and label placement. Returns coordinates relative to seriesViewRect.

Parameters

- **x1Coord:** *number*
- **y1Coord:** *number*
- **x2Coord:** *number*
- **y2Coord:** *number*
- **labelHeight:** *number*
- **labelWidth:** *number*
- **labelPlacement:** *ELabelPlacement*
- **currentAxis:** *AxisBase2D*

Returns { **x1LineCoord**: *number*; **x2LineCoord**: *number*; **y1LineCoord**: *number*; **y2LineCoord**: *number* }

- **x1LineCoord**: *number*
- **x2LineCoord**: *number*
- **y1LineCoord**: *number*
- **y2LineCoord**: *number*

getLineStylesEqual

```
getLineStylesEqual(a: ILineStyle, b: ILineStyle): boolean
```

Defined in src/Charting3D/Visuals/Axis/IAxisDescriptor.ts

Parameters

- **a:** *ILineStyle*
- **b:** *ILineStyle*

Returns *boolean*

Const getMaxSize

```
getMaxSize(): number
```

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelCache.ts

Returns *number*

Const getMinAge

```
getMinAge(): number
```

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelCache.ts

Returns *number*

Const getMonthString

```
getMonthString(month: number): string
```

Defined in src/utils/date.ts

Parameters

- **month:** *number*

Returns *string*

Const getNativeRect

```
getNativeRect(wasmContext: TSciChart, xTopLeft: number, yTopLeft: number, xBottomRight: number, yBottomRight: number): SCRTRectVertex
```

Defined in src/Charting/Visuals/Helpers/NativeObject.ts

Parameters

- **wasmContext:** *TSciChart*
- **xTopLeft:** *number*
- **yTopLeft:** *number*
- **xBottomRight:** *number*
- **yBottomRight:** *number*

Returns *SCRTRectVertex*

Const getNativeTextSize

```
getNativeTextSize(text: string, nativeFont: SCRTFont, textStyle: TNativeTextStyle, webAssemblyContext: TSciChart, rotation?: number): { deltaX: number; deltaY: number; nativeLineSpacing: number; textHeight: number; textWidth: number }
```

Defined in src/utils/text.ts

Parameters

- **text:** *string*
- **nativeFont:** *SCRTFont*
- **textStyle:** *TNativeTextStyle*
- **webAssemblyContext:** *TSciChart*
- Default value **rotation:** *number* = 0

Returns { **deltaX**: *number*; **deltaY**: *number*; **nativeLineSpacing**: *number*; **textHeight**: *number*; **textWidth**: *number* }

- **deltaX**: *number*
- **deltaY**: *number*
- **nativeLineSpacing**: *number*
- **textHeight**: *number*
- **textWidth**: *number*

Const getNearestNonUniformHeatmapPoint

```
getNearestNonUniformHeatmapPoint(xCoordinateCalculator: CoordinateCalculatorBase, yCoordinateCalculator: CoordinateCalculatorBase, heatmapDataSeries: NonUniformHeatmapDataSeries, xHitCoord: number, yHitCoord: number): { xIndex: number; yIndex: number; zValue: number }
```

Defined in src/Charting/Visuals/RenderableSeries/HitTest/hitTestHelpers.ts

Parameters

- **xCoordinateCalculator:** *CoordinateCalculatorBase*

- **yCoordinateCalculator:** *CoordinateCalculatorBase*
- **heatmapDataSeries:** *NonUniformHeatmapDataSeries*
- **xHitCoord:** *number*
- **yHitCoord:** *number*

Returns { **xIndex**: *number*; **yIndex**: *number*; **zValue**: *number* }

- **xIndex**: *number*
- **yIndex**: *number*
- **zValue**: *number*

Const getNearestPoint

```
getNearestPoint(webassemblyContext: TSciChart, xCoordinateCalculator:
CoordinateCalculatorBase, yCoordinateCalculator: CoordinateCalculatorBase, xValues:
SCRTDoubleVector, yValues: SCRTDoubleVector, isSorted: boolean, xHitCoord: number,
yHitCoord: number, hitTestRadius: number): { distance: number; nearestPointIndex:
number }
```

Defined in src/Charting/Visuals/RenderableSeries/HitTest/hitTestHelpers.ts

Parameters

- **webassemblyContext:** *TSciChart*
- **xCoordinateCalculator:** *CoordinateCalculatorBase*
- **yCoordinateCalculator:** *CoordinateCalculatorBase*
- **xValues:** *SCRTDoubleVector*
- **yValues:** *SCRTDoubleVector*
- **isSorted:** *boolean*
- **xHitCoord:** *number*
- **yHitCoord:** *number*
- **hitTestRadius:** *number*

Returns { **distance**: *number*; **nearestPointIndex**: *number* }

- **distance**: *number*
- **nearestPointIndex**: *number*

Const getNearestUniformHeatmapPoint

```
getNearestUniformHeatmapPoint(xCoordinateCalculator: CoordinateCalculatorBase,
yCoordinateCalculator: CoordinateCalculatorBase, heatmapDataSeries:
UniformHeatmapDataSeries, xHitCoord: number, yHitCoord: number): { xIndex: number;
yIndex: number; zValue: number }
```

Defined in src/Charting/Visuals/RenderableSeries/HitTest/hitTestHelpers.ts

Parameters

- **xCoordinateCalculator:** *CoordinateCalculatorBase*
- **yCoordinateCalculator:** *CoordinateCalculatorBase*
- **heatmapDataSeries:** *UniformHeatmapDataSeries*
- **xHitCoord:** *number*
- **yHitCoord:** *number*

Returns { **xIndex**: *number*; **yIndex**: *number*; **zValue**: *number* }

- **xIndex**: *number*
- **yIndex**: *number*
- **zValue**: *number*

Const getNearestXPoint

```
getNearestXPoint(webAssemblyContext: TSciChart, xCoordinateCalculator: CoordinateCalculatorBase, dataSeries: IDataSeries, xHitCoord: number, isSorted: boolean): number
```

Defined in src/Charting/Visuals/RenderableSeries/HitTest/hitTestHelpers.ts

Parameters

- **webAssemblyContext:** *TSciChart*
- **xCoordinateCalculator:** *CoordinateCalculatorBase*
- **dataSeries:** *IDataSeries*
- **xHitCoord:** *number*
- **isSorted:** *boolean*

Returns *number*

Const getNearestXyPoint

```
getNearestXyPoint(webassemblyContext: TSciChart, xCoordinateCalculator: CoordinateCalculatorBase, yCoordinateCalculator: CoordinateCalculatorBase, dataSeries: IDataSeries, xHitCoord: number, yHitCoord: number, hitTestRadius: number): { distance: number; nearestPointIndex: number }
```

Defined in src/Charting/Visuals/RenderableSeries/HitTest/hitTestHelpers.ts

Parameters

- **webassemblyContext:** *TSciChart*
- **xCoordinateCalculator:** *CoordinateCalculatorBase*
- **yCoordinateCalculator:** *CoordinateCalculatorBase*
- **dataSeries:** *IDataSeries*
- **xHitCoord:** *number*
- **yHitCoord:** *number*
- **hitTestRadius:** *number*

Returns { **distance**: *number*; **nearestPointIndex**: *number* }

- **distance**: *number*
- **nearestPointIndex**: *number*

Const getNearestXyyPoint

```
getNearestXyyPoint(webassemblyContext: TSciChart, xCoordinateCalculator: CoordinateCalculatorBase, yCoordinateCalculator: CoordinateCalculatorBase, dataSeries: XyyDataSeries, xHitCoord: number, yHitCoord: number, hitTestRadius: number): { distance: number; nearestPointIndex: number }
```

Defined in src/Charting/Visuals/RenderableSeries/HitTest/hitTestHelpers.ts

Parameters

- **webassemblyContext:** *TSciChart*
- **xCoordinateCalculator:** *CoordinateCalculatorBase*
- **yCoordinateCalculator:** *CoordinateCalculatorBase*
- **dataSeries:** *XyyDataSeries*
- **xHitCoord:** *number*
- **yHitCoord:** *number*
- **hitTestRadius:** *number*

Returns { **distance**: *number*; **nearestPointIndex**: *number* }

- **distance**: *number*

- **nearestPointIndex**: *number*

Const getNextRandomPriceBarFactory

`getNextRandomPriceBarFactory(startDateTimestamp: number, candleIntervalMinutes: number, simulateDateGap: boolean, startPrice: number): (Anonymous function)`

Defined in src/utils/randomPricesDataSource.ts

Parameters

- **startDateTimestamp:** *number*
- **candleIntervalMinutes:** *number*
- **simulateDateGap:** *boolean*
- **startPrice:** *number*

Returns *(Anonymous function)*

Const getNoisySinewave

`getNoisySinewave(pointCount: number, xMax: number, frequency: number, amplitude: number, noiseAmplitude: number): number[][]`

Defined in src/utils/math.ts

Parameters

- **pointCount:** *number*
- **xMax:** *number*
- **frequency:** *number*
- **amplitude:** *number*
- **noiseAmplitude:** *number*

Returns *number*[][]

getOHLCYRange

`getOHLCYRange(indicesRange: NumberRange, openValues: SCRTDoubleVector, highValues: SCRTDoubleVector, lowValues: SCRTDoubleVector, closeValues: SCRTDoubleVector): getOHLCYRange`

Defined in src/Charting/Model/OhlcDataSeries.ts

Parameters

- **indicesRange:** *NumberRange*
- **openValues:** *SCRTDoubleVector*
- **highValues:** *SCRTDoubleVector*
- **lowValues:** *SCRTDoubleVector*
- **closeValues:** *SCRTDoubleVector*

Returns *getOHLCYRange*

Const getOrderId

`getOrderId(): string`

Defined in src/Charting/Visuals/licenseManager2dState.ts

Returns *string*

Const getProductCode

getProductCode(): *string*

Defined in src/Charting/Visuals/licenseManager2dState.ts

Returns *string*

Const getRandomInRange

getRandomInRange(min: *number*, max: *number*, decimalPlaces: *number*): *number*

Defined in src/utils/random.ts

Return random number in range [min, max]

Parameters

- **min:** *number*
- **max:** *number*
- **decimalPlaces:** *number*

Returns *number*

getRubberBandRect

getRubberBandRect(pointFrom: *Point*, pointTo: *Point*, xyDirection: *EXyDirection*, viewportRect: *Rect*): *Rect*

Defined in src/Charting/ChartModifiers/RubberBandXyZoomModifier.ts

Given the starting and end mouse-point, computes a rectangle to perform zoom over. Takes into account the xyDirection

Parameters

- **pointFrom:** *Point*

  the starting point of the mouse
- **pointTo:** *Point*

  the end point of the mouse
- **xyDirection:** *EXyDirection*

  the XyDirection
- **viewportRect:** *Rect*

Returns *Rect*

Const getScrtBrushFromCache

getScrtBrushFromCache(cache: *BrushCache*): *SCRTBrush*

Defined in src/Charting/Drawing/BrushCache.ts

Retrieves a native {@link SCRTBrush} Brush from the provided {@link BrushCache} cache object. The retrieved entity is a wrapper around {@link SCRTPen}

Parameters

- **cache:** *BrushCache*

  The object that stores a brush

Returns *SCRTBrush*

new or existing instance of {@link SCRTBrush}}

Const getScrtPenFromCache

getScrtPenFromCache(penCache: *Pen2DCache*): *SCRTPen*

Defined in src/Charting/Drawing/Pen2DCache.ts

Retrieves a native {@link SCRTPen} Pen from the provided {@link Pen2DCache} cache object

Parameters

- **penCache:** *Pen2DCache*

  The object that stores a pen

Returns *SCRTPen*

the new or existing instance of {@link SCRTPen}}

Const getSharedWasmContext

getSharedWasmContext(): *Promise*<{ CategoryCoordinateCalculatorDouble: {}; DoubleRange: {} & {}; DoubleVector: {}; FlippedCategoryCoordinateCalculatorDouble: {}; FlippedLinearCoordinateCalculatorDouble: {}; FlippedLinearCoordinateCalculatorSingle: {}; FlippedLogarithmicCoordinateCalculator: {}; FloatVector: {}; IntVector: {}; LinearCoordinateCalculatorDouble: {}; LinearCoordinateCalculatorSingle: {}; LogarithmicCoordinateCalculator: {}; NiceDoubleScale: { CalculateTickSpacing: (min: *number*, max: *number*, minorsPerMajor: *number*, maxTicks: *number*) => *DoubleRange*; NiceNum: (range: *number*, round: *boolean*) => *number* }; NiceLogScale: { CalculateLowPrecisionTickSpacing: (min: *number*, max: *number*, logBase: *number*, minorsPerMajor: *number*, maxTicks: *number*) => *DoubleRange*; CalculateTickSpacing: (min: *number*, max: *number*, logBase: *number*, minorsPerMajor: *number*, maxTicks: *number*) => *DoubleRange* }; NumberUtil: { Constrain: (value: *number*, lowerBound: *number*, upperBound: *number*) => *number*; FindIndex: (inputValues: *SCRTDoubleVector*, value: *number*, searchMode: *SCRTFindIndexSearchMode*, dataIsSortedAscending: *boolean*) => *number*; IsDivisibleBy: (value: *number*, divisor: *number*) => *boolean*; IsPowerOf: (value: *number*, power: *number*, logBase: *number*) => *boolean*; LinearInterpolateI: (from: *number*, to: *number*, ratio: *number*) => *number*; Log: (value: *number*, logBase: *number*) => *number*; MinMax: (inputValues: *SCRTDoubleVector*) => *DoubleRange*; MinMaxWithIndex: (inputValues: *SCRTDoubleVector*, startIndex: *number*, count: *number*) => *DoubleRange*; RoundDown: (value: *number*, nearest: *number*) => *number*; RoundDownPower: (value: *number*, power: *number*, logBase: *number*) => *number*; RoundToDigits: (value: *number*, decimals: *number*) => *number*; RoundUp: (value: *number*, nearest: *number*) => *number*; RoundUpPower: (value: *number*, power: *number*, logBase: *number*) => *number* }; ResamplingArgs: {}; ResamplingData: { CategoryData: *ResamplingData*; LinearData: *ResamplingData*; UnevenlySpacedData: *ResamplingData*; UnsortedData: *ResamplingData* }; ResamplingMode: { Auto: *ResamplingMode*; Max: *ResamplingMode*; Mid: *ResamplingMode*; Min: *ResamplingMode*; MinMax: *ResamplingMode*; MinOrMax: *ResamplingMode*; None: *ResamplingMode* }; SCRTAnimationHelperFade: (yValues: *SCRTDoubleVector*, progress: *number*, ysValues: *SCRTDoubleVector*) => *number*; SCRTAnimationHelperScale: (yValues: *SCRTDoubleVector*, zeroLine: *number*, progress: *number*, ysValues: *SCRTDoubleVector*) => *number*; SCRTAnimationHelperSweep: (yValues: *SCRTDoubleVector*, progress: *number*, ysValues: *SCRTDoubleVector*) => *number*; SCRTAnimationHelperWave: (yValues: *SCRTDoubleVector*, durationFraction: *number*, zeroLine: *number*, progress: *number*, ysValues: *SCRTDoubleVector*) => *number*; SCRTBandDrawingParams: {}; SCRTBandSeriesDrawingProvider: {}; SCRTBrush: {}; SCRTBubbleSeriesDrawingProvider: {}; SCRTCandleType: { CandleStick: *SCRTCandleType*; OHLC: *SCRTCandleType* }; SCRTCandlestickSeriesDrawingProvider: {}; SCRTColorVertex: {} & {}; SCRTColumnDrawingParams: {}; SCRTColumnSeriesDrawingProvider: {}; SCRTColumnVertex: {}; SCRTContourParams: {}; SCRTCopyToDestinationInterface: { implement: (wrapper: *SCRTCopyToDestinationInterfaceWrapper*) => *SCRTCopyToDestinationInterface* }; SCRTCreateBitmapTexture: (width: *number*, height: *number*, textureFormat: *eTSRTextureFormat*) => *TSRTexture*; SCRTCreateDahedPen: (color: *number*, thickness: *number*, antialiased: *boolean*, dashPattern: *FloatVector*) => *SCRTPen*; SCRTCreatePalette: (colors: *IntVector*) => *SCRTPalette*; SCRTCredentials: { ApplyLicenseResponse: (response: *string*) => *number*; Dump: () => *string*; GetAllowDebugging: () => *boolean*; GetBuildStamp: () => *string*; GetDeveloperCount: () => *number*; GetEncrypted: (stringToEncrypt: *string*) => *string*; GetEncryptedOrderId: () => *string*; GetExpiryDate: () => *string*; GetLicenseChallenge: () => *string*; GetLicenseDaysRemaining: () => *number*; GetLicenseErrors: () => *string*; GetLicenseType: () => *SCRTLicenseType*; GetOrderId: () => *string*; GetProductCode: () => *string*; HasFeature: (feature: *string*) => *SCRTLicenseType*; Hash256Encode64: (stringToHash: *string*) => *string*; RequiresValidation: () => *boolean*; ResetRuntimeLicense: () => *void*; SetRuntimeLicenseKeyW: (licenseKey: *string*) => *boolean* }; SCRTDoLeakCheck: () => *void*; SCRTDoubleArrayOperations: {}; SCRTDoubleArraysXyResampleOutput: {}; SCRTDoubleResampler: {}; SCRTDoubleResamplerMergeIndicesParams: {}; SCRTDoubleVector: {} & {}; SCRTFifoVector: {}; SCRTFileLoadCallbackInterface: { implement: (wrapper: *SCRTFileLoadCallbackInterfaceWrapper*) => *SCRTFileLoadCallbackInterface* }; SCRTFillTextureAbgr: (texture: *TSRTexture*, width: *number*, height: *number*, pixels: *IntVector*) => *void*; SCRTFillTextureFloat32: (texture: *TSRTexture*, width: *number*, height: *number*, pixels: *SCRTFloatVector*) => *TSRVector4*; SCRTFillVectorSequential: (SCRTFillVectorSequential: *SCRTDoubleVector*, count: *number*) => *void*; SCRTFindIndexSearchMode: { Exact: *SCRTFindIndexSearchMode*; Nearest: *SCRTFindIndexSearchMode*; RoundDown: *SCRTFindIndexSearchMode*; RoundUp: *SCRTFindIndexSearchMode* }; SCRTFloatVector: {}; SCRTFontKey: {}; SCRTFrameRenderer2D: {}; SCRTGetGlobalSampleChartInterface: () => *SCRTSampleChartInterface*; SCRTGetMainRenderContext2D: () => *SCRTRenderContext*; SCRTGetScreenHeight: () => *number*; SCRTGetScreenWidth: () => *number*; SCRTGlowEffect: {}; SCRTHeatmapDrawingParams: {}; SCRTHeatmapSeriesDrawingProvider: {}; SCRTHitTestHelper: { GetNearestXyPoint: (xCoordinateCalculator: *CoordinateCalculator*, yCoordinateCalculator: *CoordinateCalculator*, xValues: *SCRTDoubleVector*, yValues: *SCRTDoubleVector*, isSorted: *boolean*, xHitCoord: *number*, yHitCoord: *number*, hitTestRadius: *number*) => *DoubleRange* }; SCRTInitEngine2D: () => *void*; SCRTLicenseType: { LICENSE_TYPE_COMMUNITY: *SCRTLicenseType*; LICENSE_TYPE_FULL: *SCRTLicenseType*; LICENSE_TYPE_FULL_EXPIRED: *SCRTLicenseType*; LICENSE_TYPE_INVALID_DEVELOPER_LICENSE: *SCRTLicenseType*; LICENSE_TYPE_INVALID_LICENSE: *SCRTLicenseType*; LICENSE_TYPE_NO_LICENSE: *SCRTLicenseType*; LICENSE_TYPE_REQUIRES_VALIDATION: *SCRTLicenseType*; LICENSE_TYPE_SUBSCRIPTION_EXPIRED: *SCRTLicenseType*; LICENSE_TYPE_TRIAL: *SCRTLicenseType*; LICENSE_TYPE_TRIAL_EXPIRED: *SCRTLicenseType* }; SCRTLineDrawingParams: {}; SCRTLineGapMode: { CloseGaps: *SCRTLineGapMode*; Default: *SCRTLineGapMode*; DrawGaps: *SCRTLineGapMode* }; SCRTLineSeriesDrawingProvider: {}; SCRTLineType: { Digital: *SCRTLineType*; List: *SCRTLineType*; Nan: *SCRTLineType*; Strip: *SCRTLineType* };

SCRTMemCopy: (destPtr: *number*, sourcePtr: *number*, count: *number*) => *void*; SCRTMemMove: (destPtr: *number*, sourcePtr: *number*, count: *number*) => *void*; SCRTMountainDrawingParams: {}; SCRTMountainSeriesDrawingProvider: {}; SCRTMultiplyColorVectorOpacity: (originalVector: *IntVector*, resultVector: *IntVector*, factor: *number*) => *void*; SCRTOhlcDrawingParams: {}; SCRTPalette: {} & { GetNoOverrideColorCode: () => *number* }; SCRTPen: {}; SCRTPointDrawingParams: {}; SCRTRectVertex: {} & {}; SCRTRegisterFile: (fileName: *string*, url: *string*, callback: *SCRTFileLoadCallbackInterface*) => *void*; SCRTSampleChartInterface: { implement: (wrapper: *SCRTSampleChartInterfaceWrapper*) => *SCRTSampleChartInterface* }; SCRTScatterSeriesDrawingProvider: {}; SCRTSeriesEffectType: { Glow: *SCRTSeriesEffectType* }; SCRTSetActiveDoubleVector: (SCRTSetActiveDoubleVector: *SCRTDoubleVector*, doubleVector: *number*) => *void*; SCRTSetActiveTexture: (texture: *TSRTexture*) => *void*; SCRTSetClearAlphaParams: (enabled: *boolean*, alpha: *number*) => *void*; SCRTSetGlobalCopyToDestinationInterface: (param0: *SCRTCopyToDestinationInterface*) => *void*; SCRTSetGlobalSampleChartInterface: (param0: *SCRTSampleChartInterface*) => *void*; SCRTSetMainWindowSize: (width: *number*, height: *number*) => *void*; SCRTSetTextureLinearSamplerEnabled: (texture: *TSRTexture*, enabled: *boolean*) => *void*; SCRTSetWaterMarkProperties: (properties: *SCRTWaterMarkProperties*) => *void*; SCRTShadowEffect: {}; SCRTShutdownEngine2D: () => *void*; SCRTSolidBrush: {}; SCRTSplineHelperCubicSpline: (xValues: *SCRTDoubleVector*, yValues: *SCRTDoubleVector*, xsValues: *SCRTDoubleVector*, ysValues: *SCRTDoubleVector*, initialSize: *number*, interpolationPoints: *number*, containsNAN: *boolean*) => *void*; SCRTSpriteType: { FixedSize: *SCRTSpriteType*; Normal: *SCRTSpriteType* }; SCRTStackedColumnDrawingParams: {}; SCRTStackedColumnSeriesDrawingProvider: {}; SCRTSurfaceDestination: { implement: (wrapper: *SCRTSurfaceDestinationWrapper*) => *SCRTSurfaceDestination* }; SCRTTextureBrush: {}; SCRTWaterMarkProperties: {}; SCRTXvaluesProvider: {}; StringVector: {}; TSRCamera: {}; TSRRequestCanvasDraw: (canvasID: *string*) => *void*; TSRRequestDraw: () => *void*; TSRRequestExit: () => *void*; TSRSetDrawRequestsEnabled: (enabled: *boolean*) => *void*; TSRTextBounds: {}; TSRTextLineBounds: {}; TSRVector2: {} & {}; TSRVector3: {} & {}; TSRVector4: {} & {}; UIntVector: {}; VectorColorVertex: {}; VectorColumnVertex: {}; VectorRectVertex: {}; WStringVector: {}; canvas: *HTMLCanvasElement*; canvas2D: *HTMLCanvasElement*; eSCRTBlendMode: { BlendAdditiveAlpha: *eSCRTBlendMode*; BlendAdditiveColor: *eSCRTBlendMode*; BlendAdditiveOneAlpha: *eSCRTBlendMode*; BlendDefault: *eSCRTBlendMode*; BlendDisabled: *eSCRTBlendMode* }; eSCRTBrushMappingMode: { PerPrimitive: *eSCRTBrushMappingMode*; PerScreen: *eSCRTBrushMappingMode* }; eTSRCameraProjectionMode: { CAMERA_PROJECTIONMODE_ORTHOGONAL: *eTSRCameraProjectionMode*; CAMERA_PROJECTIONMODE_PERSPECTIVE: *eTSRCameraProjectionMode* }; eTSRMetaDataType: { BitFlags: *eTSRMetaDataType*; Core: *eTSRMetaDataType*; Defined: *eTSRMetaDataType*; DynamicDefined: *eTSRMetaDataType*; Enum: *eTSRMetaDataType*; Unknown: *eTSRMetaDataType* }; eTSRPlatform: { Android: *eTSRPlatform*; Linux: *eTSRPlatform*; Mac: *eTSRPlatform*; Undefined: *eTSRPlatform*; Web: *eTSRPlatform*; Windows: *eTSRPlatform*; iOS: *eTSRPlatform* }; eTSRRendererType: { TSR_RENDERER_TYPE_D3D11: *eTSRRendererType*; TSR_RENDERER_TYPE_D3D11_LEVEL10: *eTSRRendererType*; TSR_RENDERER_TYPE_D3D12: *eTSRRendererType*; TSR_RENDERER_TYPE_D3D9: *eTSRRendererType*; TSR_RENDERER_TYPE_GL: *eTSRRendererType*; TSR_RENDERER_TYPE_GLES2: *eTSRRendererType*; TSR_RENDERER_TYPE_GLES3: *eTSRRendererType*; TSR_RENDERER_TYPE_METAL: *eTSRRendererType*; TSR_RENDERER_TYPE_UNDEFINED: *eTSRRendererType*; TSR_RENDERER_TYPE_VULKAN: *eTSRRendererType* }; eTSRTextAlignMode: { Center: *eTSRTextAlignMode*; Left: *eTSRTextAlignMode*; Right: *eTSRTextAlignMode* }; eTSRTextureFormat: { TSR_TEXTUREFORMAT_A8B8G8R8: *eTSRTextureFormat*; TSR_TEXTUREFORMAT_R32F: *eTSRTextureFormat* }; eVariableUsage: { Array: *eVariableUsage*; Blob: *eVariableUsage*; DynamicArray: *eVariableUsage*; Normal: *eVariableUsage*; Pointer: *eVariableUsage*; Vector: *eVariableUsage*; VectorOfPointers: *eVariableUsage* } }>

Defined in src/Charting/Visuals/createMaster.ts

Returns *Promise*<{ **CategoryCoordinateCalculatorDouble**: {}; **DoubleRange**: {} & {}; **DoubleVector**: {}; **FlippedCategoryCoordinateCalculatorDouble**: {}; **FlippedLinearCoordinateCalculatorDouble**: {}; **FlippedLinearCoordinateCalculatorSingle**: {}; **FlippedLogarithmicCoordinateCalculator**: {}; **FloatVector**: {}; **IntVector**: {}; **LinearCoordinateCalculatorDouble**: {}; **LinearCoordinateCalculatorSingle**: {}; **LogarithmicCoordinateCalculator**: {}; **NiceDoubleScale**: { **CalculateTickSpacing**: (**min**: *number*, **max**: *number*, **minorsPerMajor**: *number*, **maxTicks**: *number*) => *DoubleRange*; **NiceNum**: (**range**: *number*, **round**: *boolean*) => *number* }; **NiceLogScale**: { **CalculateLowPrecisionTickSpacing**: (**min**: *number*, **max**: *number*, **logBase**: *number*, **minorsPerMajor**: *number*, **maxTicks**: *number*) => *DoubleRange*; **CalculateTickSpacing**: (**min**: *number*, **max**: *number*, **logBase**: *number*, **minorsPerMajor**: *number*, **maxTicks**: *number*) => *DoubleRange* }; **NumberUtil**: { **Constrain**: (**value**: *number*, **lowerBound**: *number*, **upperBound**: *number*) => *number*; **FindIndex**: (**inputValues**: *SCRTDoubleVector*, **value**: *number*, **searchMode**: *SCRTFindIndexSearchMode*, **dataIsSortedAscending**: *boolean*) => *number*; **IsDivisibleBy**: (**value**: *number*, **divisor**: *number*) => *boolean*; **IsPowerOf**: (**value**: *number*, **power**: *number*, **logBase**: *number*) => *boolean*; **LinearInterpolatel**: (**from**: *number*, **to**: *number*, **ratio**: *number*) => *number*; **Log**: (**value**: *number*, **logBase**: *number*) => *number*; **MinMax**: (**inputValues**: *SCRTDoubleVector*) => *DoubleRange*; **MinMaxWithIndex**: (**inputValues**: *SCRTDoubleVector*, **startIndex**: *number*, **count**: *number*) => *DoubleRange*; **RoundDown**: (**value**: *number*, **nearest**: *number*) => *number*; **RoundDownPower**: (**value**: *number*, **power**: *number*, **logBase**: *number*) => *number*; **RoundToDigits**: (**value**: *number*, **decimals**: *number*) => *number*; **RoundUp**: (**value**: *number*, **nearest**: *number*) => *number*; **RoundUpPower**: (**value**: *number*, **power**: *number*, **logBase**: *number*) => *number* }; **ResamplingArgs**: {}; **ResamplingData**: { **CategoryData**: *ResamplingData*; **LinearData**: *ResamplingData*; **UnevenlySpacedData**: *ResamplingData*; **UnsortedData**: *ResamplingData* }; **ResamplingMode**: { **Auto**: *ResamplingMode*; **Max**: *ResamplingMode*; **Mid**: *ResamplingMode*; **Min**: *ResamplingMode*; **MinMax**: *ResamplingMode*; **MinOrMax**: *ResamplingMode*; **None**: *ResamplingMode* }; **SCRTAnimationHelperFade**: (**yValues**: *SCRTDoubleVector*, **progress**: *number*, **ysValues**: *SCRTDoubleVector*) => *number*; **SCRTAnimationHelperScale**: (**yValues**: *SCRTDoubleVector*, **zeroLine**: *number*, **progress**: *number*, **ysValues**: *SCRTDoubleVector*) => *number*; **SCRTAnimationHelperSweep**: (**yValues**: *SCRTDoubleVector*, **progress**: *number*, **ysValues**: *SCRTDoubleVector*) => *number*; **SCRTAnimationHelperWave**: (**yValues**: *SCRTDoubleVector*, **durationFraction**: *number*, **zeroLine**: *number*, **progress**: *number*, **ysValues**: *SCRTDoubleVector*) => *number*;

SCRTBandDrawingParams: {}; **SCRTBandSeriesDrawingProvider**: {}; **SCRTBrush**: {}; **SCRTBubbleSeriesDrawingProvider**: {}; **SCRTCandleType**: { **CandleStick**: *SCRTCandleType*; **OHLC**: *SCRTCandleType* }; **SCRTCandlestickSeriesDrawingProvider**: {}; **SCRTColorVertex**: {} & {}; **SCRTColumnDrawingParams**: {}; **SCRTColumnSeriesDrawingProvider**: {}; **SCRTColumnVertex**: {}; **SCRTContourParams**: {}; **SCRTCopyToDestinationInterface**: { **implement**: (**wrapper**: *SCRTCopyToDestinationInterfaceWrapper*) => *SCRTCopyToDestinationInterface* }; **SCRTCreateBitmapTexture**: (**width**: *number*, **height**: *number*, **textureFormat**: *eTSRTextureFormat*) => *TSRTexture*; **SCRTCreateDahedPen**: (**color**: *number*, **thickness**: *number*, **antialiased**: *boolean*, **dashPattern**: *FloatVector*) => *SCRTPen*; **SCRTCreatePalette**: (**colors**: *IntVector*) => *SCRTPalette*; **SCRTCredentials**: { **ApplyLicenseResponse**: (**response**: *string*) => *number*; **Dump**: () => *string*; **GetAllowDebugging**: () => *boolean*; **GetBuildStamp**: () => *string*; **GetDeveloperCount**: () => *number*; **GetEncrypted**: (**stringToEncrypt**: *string*) => *string*; **GetEncryptedOrderId**: () => *string*; **GetExpiryDate**: () => *string*; **GetLicenseChallenge**: () => *string*; **GetLicenseDaysRemaining**: () => *number*; **GetLicenseErrors**: () => *string*; **GetLicenseType**: () => *SCRTLicenseType*; **GetOrderId**: () => *string*; **GetProductCode**: () => *string*; **HasFeature**: (**feature**: *string*) => *SCRTLicenseType*; **Hash256Encode64**: (**stringToHash**: *string*) => *string*; **RequiresValidation**: () => *boolean*; **ResetRuntimeLicense**: () => *void*; **SetRuntimeLicenseKeyW**: (**licenseKey**: *string*) => *boolean* }; **SCRTDoLeakCheck**: () => *void*; **SCRTDoubleArrayOperations**: {}; **SCRTDoubleArraysXyResampleOutput**: {}; **SCRTDoubleResampler**: {}; **SCRTDoubleResamplerMergeIndicesParams**: {}; **SCRTDoubleVector**: {} & {}; **SCRTFifoVector**: {}; **SCRTFileLoadCallbackInterface**: { **implement**: (**wrapper**: *SCRTFileLoadCallbackInterfaceWrapper*) => *SCRTFileLoadCallbackInterface* }; **SCRTFillTextureAbgr**: (**texture**: *TSRTexture*, **width**: *number*, **height**: *number*, **pixels**: *IntVector*) => *void*; **SCRTFillTextureFloat32**: (**texture**: *TSRTexture*, **width**: *number*, **height**: *number*, **pixels**: *SCRTFloatVector*) => *TSRVector4*; **SCRTFillVectorSequential**: (**SCRTFillVectorSequential**: *SCRTDoubleVector*, **count**: *number*) => *void*; **SCRTFindIndexSearchMode**: { **Exact**: *SCRTFindIndexSearchMode*; **Nearest**: *SCRTFindIndexSearchMode*; **RoundDown**: *SCRTFindIndexSearchMode*; **RoundUp**: *SCRTFindIndexSearchMode* }; **SCRTFloatVector**: {}; **SCRTFontKey**: {}; **SCRTFrameRenderer2D**: {}; **SCRTGetGlobalSampleChartInterface**: () => *SCRTSampleChartInterface*; **SCRTGetMainRenderContext2D**: () => *SCRTRenderContext*; **SCRTGetScreenHeight**: () => *number*; **SCRTGetScreenWidth**: () => *number*; **SCRTGlowEffect**: {}; **SCRTHeatmapDrawingParams**: {}; **SCRTHeatmapSeriesDrawingProvider**: {}; **SCRTHitTestHelper**: { **GetNearestXyPoint**: (**xCoordinateCalculator**: *CoordinateCalculator*, **yCoordinateCalculator**: *CoordinateCalculator*, **xValues**: *SCRTDoubleVector*, **yValues**: *SCRTDoubleVector*, **isSorted**: *boolean*, **xHitCoord**: *number*, **yHitCoord**: *number*, **hitTestRadius**: *number*) => *DoubleRange* }; **SCRTInitEngine2D**: () => *void*; **SCRTLicenseType**: { **LICENSE_TYPE_COMMUNITY**: *SCRTLicenseType*; **LICENSE_TYPE_FULL**: *SCRTLicenseType*; **LICENSE_TYPE_FULL_EXPIRED**: *SCRTLicenseType*; **LICENSE_TYPE_INVALID_DEVELOPER_LICENSE**: *SCRTLicenseType*; **LICENSE_TYPE_INVALID_LICENSE**: *SCRTLicenseType*; **LICENSE_TYPE_NO_LICENSE**: *SCRTLicenseType*; **LICENSE_TYPE_REQUIRES_VALIDATION**: *SCRTLicenseType*; **LICENSE_TYPE_SUBSCRIPTION_EXPIRED**: *SCRTLicenseType*; **LICENSE_TYPE_TRIAL**: *SCRTLicenseType*; **LICENSE_TYPE_TRIAL_EXPIRED**: *SCRTLicenseType* }; **SCRTLineDrawingParams**: {}; **SCRTLineGapMode**: { **CloseGaps**: *SCRTLineGapMode*; **Default**: *SCRTLineGapMode*; **DrawGaps**: *SCRTLineGapMode* }; **SCRTLineSeriesDrawingProvider**: {}; **SCRTLineType**: { **Digital**: *SCRTLineType*; **List**: *SCRTLineType*; **Nan**: *SCRTLineType*; **Strip**: *SCRTLineType* }; **SCRTMemCopy**: (**destPtr**: *number*, **sourcePtr**: *number*, **count**: *number*) => *void*; **SCRTMemMove**: (**destPtr**: *number*, **sourcePtr**: *number*, **count**: *number*) => *void*; **SCRTMountainDrawingParams**: {}; **SCRTMountainSeriesDrawingProvider**: {}; **SCRTMultiplyColorVectorOpacity**: (**originalVector**: *IntVector*, **resultVector**: *IntVector*, **factor**: *number*) => *void*; **SCRTOhlcDrawingParams**: {}; **SCRTPalette**: {} & { **GetNoOverrideColorCode**: () => *number* }; **SCRTPen**: {}; **SCRTPointDrawingParams**: {}; **SCRTRectVertex**: {} & {}; **SCRTRegisterFile**: (**fileName**: *string*, **url**: *string*, **callback**: *SCRTFileLoadCallbackInterface*) => *void*; **SCRTSampleChartInterface**: { **implement**: (**wrapper**: *SCRTSampleChartInterfaceWrapper*) => *SCRTSampleChartInterface* }; **SCRTScatterSeriesDrawingProvider**: {}; **SCRTSeriesEffectType**: { **Glow**: *SCRTSeriesEffectType* }; **SCRTSetActiveDoubleVector**: (**SCRTSetActiveDoubleVector**: *SCRTDoubleVector*, **doubleVector**: *number*) => *void*; **SCRTSetActiveTexture**: (**texture**: *TSRTexture*) => *void*; **SCRTSetClearAlphaParams**: (**enabled**: *boolean*, **alpha**: *number*) => *void*; **SCRTSetGlobalCopyToDestinationInterface**: (**param0**: *SCRTCopyToDestinationInterface*) => *void*; **SCRTSetGlobalSampleChartInterface**: (**param0**: *SCRTSampleChartInterface*) => *void*; **SCRTSetMainWindowSize**: (**width**: *number*, **height**: *number*) => *void*; **SCRTSetTextureLinearSamplerEnabled**: (**texture**: *TSRTexture*, **enabled**: *boolean*) => *void*; **SCRTSetWaterMarkProperties**: (**properties**: *SCRTWaterMarkProperties*) => *void*; **SCRTShadowEffect**: {}; **SCRTShutdownEngine2D**: () => *void*; **SCRTSolidBrush**: {}; **SCRTSplineHelperCubicSpline**: (**xValues**: *SCRTDoubleVector*, **yValues**: *SCRTDoubleVector*, **xsValues**: *SCRTDoubleVector*, **ysValues**: *SCRTDoubleVector*, **initialSize**: *number*, **interpolationPoints**: *number*, **containsNAN**: *boolean*) => *void*; **SCRTSpriteType**: { **FixedSize**: *SCRTSpriteType*; **Normal**: *SCRTSpriteType* }; **SCRTStackedColumnDrawingParams**: {}; **SCRTStackedColumnSeriesDrawingProvider**: {}; **SCRTSurfaceDestination**: { **implement**: (**wrapper**: *SCRTSurfaceDestinationWrapper*) => *SCRTSurfaceDestination* }; **SCRTTextureBrush**: {}; **SCRTWaterMarkProperties**: {}; **SCRTXvaluesProvider**: {}; **StringVector**: {}; **TSRCamera**: {}; **TSRRequestCanvasDraw**: (**canvasID**: *string*) => *void*; **TSRRequestDraw**: () => *void*; **TSRRequestExit**: () => *void*; **TSRSetDrawRequestsEnabled**: (**enabled**: *boolean*) => *void*; **TSRTextBounds**: {}; **TSRTextLineBounds**: {}; **TSRVector2**: {} & {}; **TSRVector3**: {} & {}; **TSRVector4**: {} & {}; **UIntVector**: {}; **VectorColorVertex**: {}; **VectorColumnVertex**: {}; **VectorRectVertex**: {}; **WStringVector**: {}; **canvas**: *HTMLCanvasElement*; **canvas2D**: *HTMLCanvasElement*; **eSCRTBlendMode**: { **BlendAdditiveAlpha**: *eSCRTBlendMode*;

BlendAdditiveColor: *eSCRTBlendMode*; **BlendAdditiveOneAlpha**: *eSCRTBlendMode*; **BlendDefault**: *eSCRTBlendMode*; **BlendDisabled**: *eSCRTBlendMode* }; **eSCRTBrushMappingMode**: { **PerPrimitive**: *eSCRTBrushMappingMode*; **PerScreen**: *eSCRTBrushMappingMode* }; **eTSRCameraProjectionMode**: { **CAMERA_PROJECTIONMODE_ORTHOGONAL**: *eTSRCameraProjectionMode*; **CAMERA_PROJECTIONMODE_PERSPECTIVE**: *eTSRCameraProjectionMode* }; **eTSRMetaDataType**: { **BitFlags**: *eTSRMetaDataType*; **Core**: *eTSRMetaDataType*; **Defined**: *eTSRMetaDataType*; **DynamicDefined**: *eTSRMetaDataType*; **Enum**: *eTSRMetaDataType*; **Unknown**: *eTSRMetaDataType* }; **eTSRPlatform**: { **Android**: *eTSRPlatform*; **Linux**: *eTSRPlatform*; **Mac**: *eTSRPlatform*; **Undefined**: *eTSRPlatform*; **Web**: *eTSRPlatform*; **Windows**: *eTSRPlatform*; **iOS**: *eTSRPlatform* }; **eTSRRendererType**: { **TSR_RENDERER_TYPE_D3D11**: *eTSRRendererType*; **TSR_RENDERER_TYPE_D3D11_LEVEL10**: *eTSRRendererType*; **TSR_RENDERER_TYPE_D3D12**: *eTSRRendererType*; **TSR_RENDERER_TYPE_D3D9**: *eTSRRendererType*; **TSR_RENDERER_TYPE_GL**: *eTSRRendererType*; **TSR_RENDERER_TYPE_GLES2**: *eTSRRendererType*; **TSR_RENDERER_TYPE_GLES3**: *eTSRRendererType*; **TSR_RENDERER_TYPE_METAL**: *eTSRRendererType*; **TSR_RENDERER_TYPE_UNDEFINED**: *eTSRRendererType*; **TSR_RENDERER_TYPE_VULKAN**: *eTSRRendererType* }; **eTSRTextAlignMode**: { **Center**: *eTSRTextAlignMode*; **Left**: *eTSRTextAlignMode*; **Right**: *eTSRTextAlignMode* }; **eTSRTextureFormat**: { **TSR_TEXTUREFORMAT_A8B8G8R8**: *eTSRTextureFormat*; **TSR_TEXTUREFORMAT_R32F**: *eTSRTextureFormat* }; **eVariableUsage**: { **Array**: *eVariableUsage*; **Blob**: *eVariableUsage*; **DynamicArray**: *eVariableUsage*; **Normal**: *eVariableUsage*; **Pointer**: *eVariableUsage*; **Vector**: *eVariableUsage*; **VectorOfPointers**: *eVariableUsage* } }>

Const getSize

`getSize(): number`

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelCache.ts

Returns *number*

Const getStocksDataFactory

`getStocksDataFactory(STEP: number, RANDOM_MIN: number, RANDOM_MAX: number): (Anonymous function)`

Defined in src/utils/randomPricesDataSource.ts

Creates function to generate stock data

Parameters

- **STEP:** *number*
- **RANDOM_MIN:** *number*
- **RANDOM_MAX:** *number*

Returns *(Anonymous function)*

Const getStyleId

`getStyleId(style: TCachedLabelStyle): string`

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelCache.ts

Parameters

- **style:** *TCachedLabelStyle*

Returns *string*

Const getSubTypes

`getSubTypes(baseType: EBaseType): string[]`

Defined in src/Builder/classFactory.ts

Parameters

- **baseType:** *EBaseType*

Returns *string*[]

getTArgbEqual

```
getTArgbEqual(a: TArgb, b: TArgb): boolean
```

Defined in src/Charting3D/Visuals/Axis/IAxisDescriptor.ts

Parameters

- **a:** *TArgb*
- **b:** *TArgb*

Returns *boolean*

Const getTelemetry

```
getTelemetry(): boolean
```

Defined in src/Charting/Visuals/licenseManager2dState.ts

Returns *boolean*

Const getTextBounds

```
getTextBounds(wasmContext: TSciChart | TSciChart3D): TSRTextBounds
```

Defined in src/Charting/Visuals/Helpers/NativeObject.ts

Parameters

- **wasmContext:** *TSciChart* | *TSciChart3D*

Returns *TSRTextBounds*

getTextStylesEqual

```
getTextStylesEqual(a: ITextStyle, b: ITextStyle): boolean
```

Defined in src/Charting3D/Visuals/Axis/IAxisDescriptor.ts

Parameters

- **a:** *ITextStyle*
- **b:** *ITextStyle*

Returns *boolean*

getUniqueValues

```
getUniqueValues(array: string[]): getUniqueValues
```

Defined in src/utils/array.ts

Parameters

- **array:** *string*[]

Returns *getUniqueValues*

Const getUserCookie

```
getUserCookie(): { sessionId: string; sessionStart: number; userId: string }
```

Defined in src/Core/Telemetry.ts

Returns { **sessionId**: *string*; **sessionStart**: *number*; **userId**: *string* }

- **sessionId**: *string*

- **sessionStart**: *number*
- **userId**: *string*

Const getValueWithCoordinateMode

```
getValueWithCoordinateMode(value: number, calculator: CoordinateCalculatorBase, coordinateMode: EInnerAxisPlacementCoordinateMode): number
```

Defined in src/Charting/LayoutManager/AxisLayoutHelpers.ts

Converts a pixel coordinate back to a value

Parameters

- **value:** *number*

  coordinate or dataValue to convert

- **calculator:** *CoordinateCalculatorBase*

  the CoordinateCalculatorBase which will do the transformation

- **coordinateMode:** *EInnerAxisPlacementCoordinateMode*

  the ECoordinateMode to apply

Returns *number*

the data-value or value

Const getVector4

```
getVector4(wasmContext: TSciChart | TSciChart3D, x: number, y: number, z: number, w: number): TSRVector4
```

Defined in src/Charting/Visuals/Helpers/NativeObject.ts

Parameters

- **wasmContext:** *TSciChart* | *TSciChart3D*
- **x:** *number*
- **y:** *number*
- **z:** *number*
- **w:** *number*

Returns *TSRVector4*

Const getVectorColorVertex

```
getVectorColorVertex(wasmContext: TSciChart, maxSize?: number): VectorColorVertex
```

Defined in src/Charting/Visuals/Helpers/NativeObject.ts

Parameters

- **wasmContext:** *TSciChart*
- Default value **maxSize:** *number* = 100

Returns *VectorColorVertex*

Const getVectorRectVertex

```
getVectorRectVertex(wasmContext: TSciChart, maxSize?: number): VectorRectVertex
```

Defined in src/Charting/Visuals/Helpers/NativeObject.ts

Returns an empty vector of Rectangles

Parameters

- **wasmContext:** *TSciChart*

- Default value **maxSize:** *number* = 100

Returns *VectorRectVertex*

Const getVertex

```
getVertex(wasmContext: TSciChart, x: number, y: number, colour?: number): SCRTColorVertex
```

Defined in src/Charting/Visuals/Helpers/NativeObject.ts

Parameters

- **wasmContext:** *TSciChart*
- **x:** *number*
- **y:** *number*
- Optional **colour:** *number*

Returns *SCRTColorVertex*

getVerticalAxisRequiredSize

```
getVerticalAxisRequiredSize(axisLayoutState: AxisLayoutState): number
```

Defined in src/Charting/LayoutManager/AxisLayoutHelpers.ts

Parameters

- **axisLayoutState:** *AxisLayoutState*

Returns *number*

Const getWebGlBrushFromCache

```
getWebGlBrushFromCache(cache: BrushCache): WebGlBrush
```

Defined in src/Charting/Drawing/BrushCache.ts

Retrieves a native WebGlBrush Brush from the provided {@link BrushCache} cache object

Parameters

- **cache:** *BrushCache*

Returns *WebGlBrush*

new or existing instance of WebGlPen}

Const getWebGlPenFromCache

```
getWebGlPenFromCache(penCache: Pen2DCache): WebGlPen
```

Defined in src/Charting/Drawing/Pen2DCache.ts

Retrieves a native WebGlPen Pen from the provided {@link Pen2DCache} cache object. The retrieved entity is a wrapper around {@link SCRTPen}

Parameters

- **penCache:** *Pen2DCache*

  The object that stores a pen

Returns *WebGlPen*

the new or existing instance of WebGlPen}

Const getWindowedYRange

```
getWindowedYRange(webAssemblyContext: TSciChart, xValues: SCRTDoubleVector, yValues:
```

```
SCRTDoubleVector, xRange: NumberRange, getPositiveRange: boolean, isXCategoryAxis:
boolean, isSorted: boolean, minSearchMode?: ESearchMode, maxSearchMode?: ESearchMode):
NumberRange
```

Defined in src/Charting/Model/BaseDataSeries.ts

Parameters

- **webAssemblyContext:** *TSciChart*
- **xValues:** *SCRTDoubleVector*
- **yValues:** *SCRTDoubleVector*
- **xRange:** *NumberRange*
- **getPositiveRange:** *boolean*
- **isXCategoryAxis:** *boolean*
- **isSorted:** *boolean*
- Default value **minSearchMode:** *ESearchMode* = ESearchMode.RoundUp
- Default value **maxSearchMode:** *ESearchMode* = ESearchMode.RoundDown

Returns *NumberRange*

Const getXRange

```
getXRange(range: NumberRange, count: number, widthFraction: number): NumberRange
```

Defined in src/Charting/Visuals/RenderableSeries/FastColumnRenderableSeries.ts

Parameters

- **range:** *NumberRange*
- **count:** *number*
- **widthFraction:** *number*

Returns *NumberRange*

getYyYRange

```
getYyYRange(webAssemblyContext: TSciChart, indicesRange: NumberRange, yValues:
SCRTDoubleVector, y1Values: SCRTDoubleVector): getYyYRange
```

Defined in src/Charting/Model/XyyDataSeries.ts

Parameters

- **webAssemblyContext:** *TSciChart*
- **indicesRange:** *NumberRange*
- **yValues:** *SCRTDoubleVector*
- **y1Values:** *SCRTDoubleVector*

Returns *getYyYRange*

Const guardSameLengthZValuesAndMetadata

```
guardSameLengthZValuesAndMetadata(zValues: number[][], metadata: IPointMetadata[][]):
void
```

Defined in src/Charting/Model/BaseHeatmapDataSeries.ts

Parameters

- **zValues:** *number*[][]
- **metadata:** *IPointMetadata*[][]

Returns *void*

Const handleInvalidAxisAlignment

```
handleInvalidAxisAlignment(alignment: never): never
```

Defined in src/types/AxisAlignment.ts

Parameters

- **alignment:** *never*

Returns *never*

Const hasAllProperties

```
hasAllProperties(obj: any, props: string[]): boolean
```

Defined in src/utils/hasAllProperties.ts

Parameters

- **obj:** *any*
- **props:** *string*[]

Returns *boolean*

hasOwnProperty

```
hasOwnProperty<X, Y>(obj: X, prop: Y): obj is X & Record<Y, unknown>
```

Defined in src/Builder/buildSeries.ts

Type parameters

- **X**: {}
- **Y**: *PropertyKey*

Parameters

- **obj:** *X*
- **prop:** *Y*

Returns *obj* is *X* & *Record*<*Y*, *unknown*>

htmlToElement

```
htmlToElement(html: string): htmlToElement
```

Defined in src/utils/html.ts

Parameters

- **html:** *string*

Returns *htmlToElement*

Const initializeChartEngine2D

```
initializeChartEngine2D(): Promise<{ createChildSurface: (divElementId: string,
canvases: TSciChartSurfaceCanvases, theme: IThemeProvider) => SciChartSurface;
getChildSurfaces: () => SciChartSurface[]; wasmContext: TSciChart }>
```

Defined in src/Charting/Visuals/createMaster.ts

Returns *Promise*<{ **createChildSurface**: (**divElementId**: *string*, **canvases**: *TSciChartSurfaceCanvases*, **theme**: *IThemeProvider*) => *SciChartSurface*; **getChildSurfaces**: () => *SciChartSurface*[]; **wasmContext**: *TSciChart* }>

Const interpolateColor

interpolateColor(from: *number*, to: *number*, progress: *number*): *number*

Defined in src/Charting/Visuals/RenderableSeries/Animations/animationHelpers.ts

Interpolates colors

Parameters

- **from:** *number*
- **to:** *number*
- **progress:** *number*

Returns *number*

Const interpolateLinear

interpolateLinear(x: *number*, x1: *number*, y1: *number*, x2: *number*, y2: *number*): *number*

Defined in src/Charting/Visuals/RenderableSeries/HitTest/hitTestHelpers.ts

Parameters

- **x:** *number*
- **x1:** *number*
- **y1:** *number*
- **x2:** *number*
- **y2:** *number*

Returns *number*

Const interpolateNumber

interpolateNumber(from: *number*, to: *number*, progress: *number*): *number*

Defined in src/Charting/Visuals/RenderableSeries/Animations/animationHelpers.ts

Interpolates numbers

Parameters

- **from:** *number*
- **to:** *number*
- **progress:** *number*

Returns *number*

isArraySorted

isArraySorted(arr: *NumberArray*, ascending?: *boolean*): *boolean*

Defined in src/utils/array.ts

returns true if the array is sorted

Parameters

- **arr:** *NumberArray*
  The array
- Default value **ascending:** *boolean* = true
  If True checks for sorted ascending, if False for descending

Returns *boolean*

Const isBetween

isBetween(value: *number*, start: *number*, end: *number*): *boolean*

Defined in src/utils/pointUtil.ts

Returns whether a number is in the range of 2 other numbers, regardless if ascending or descending

Parameters

- **value:** *number*
- **start:** *number*
- **end:** *number*

Returns *boolean*

Const isConstructor

`isConstructor(someObj: any): boolean`

Defined in src/Core/DeletableEntity.ts

Parameters

- **someObj:** *any*

Returns *boolean*

isNumberArray

`isNumberArray(a: any): isNumberArray`

Defined in src/types/NumberArray.ts

Parameters

- **a:** *any*

Returns *isNumberArray*

isRealNumber

`isRealNumber(number: number): boolean`

Defined in src/utils/isRealNumber.ts

returns true if the number is a real number (not NAN, not Infinite)

Parameters

- **number:** *number*

Returns *boolean*

isTypedArray

`isTypedArray(a: any): isTypedArray`

Defined in src/types/NumberArray.ts

Parameters

- **a:** *any*

Returns *isTypedArray*

layoutAxisParts

`layoutAxisParts(axis: AxisBase2D, layoutFunc: TLayoutAxisPartsWithStrategyFunc): void`

Defined in src/Charting/LayoutManager/AxisLayoutHelpers.ts

Parameters

- **axis:** *AxisBase2D*
- **layoutFunc:** *TLayoutAxisPartsWithStrategyFunc*

Returns *void*

layoutAxisPartsBottomStrategy

```
layoutAxisPartsBottomStrategy(axisRendererWidth: number, axisRendererHeight: number,
axisTitleRendererWidth: number, axisTitleRendererHeight: number, containerBounds:
Rect): TAxisViewRects
```

Defined in src/Charting/LayoutManager/AxisLayoutHelpers.ts

Parameters

- **axisRendererWidth:** *number*
- **axisRendererHeight:** *number*
- **axisTitleRendererWidth:** *number*
- **axisTitleRendererHeight:** *number*
- **containerBounds:** *Rect*

Returns *TAxisViewRects*

layoutAxisPartsLeftStrategy

```
layoutAxisPartsLeftStrategy(axisRendererWidth: number, axisRendererHeight: number,
axisTitleRendererWidth: number, axisTitleRendererHeight: number, containerBounds:
Rect): TAxisViewRects
```

Defined in src/Charting/LayoutManager/AxisLayoutHelpers.ts

Parameters

- **axisRendererWidth:** *number*
- **axisRendererHeight:** *number*
- **axisTitleRendererWidth:** *number*
- **axisTitleRendererHeight:** *number*
- **containerBounds:** *Rect*

Returns *TAxisViewRects*

layoutAxisPartsRightStrategy

```
layoutAxisPartsRightStrategy(axisRendererWidth: number, axisRendererHeight: number,
axisTitleRendererWidth: number, axisTitleRendererHeight: number, containerBounds:
Rect): TAxisViewRects
```

Defined in src/Charting/LayoutManager/AxisLayoutHelpers.ts

Parameters

- **axisRendererWidth:** *number*
- **axisRendererHeight:** *number*
- **axisTitleRendererWidth:** *number*
- **axisTitleRendererHeight:** *number*
- **containerBounds:** *Rect*

Returns *TAxisViewRects*

layoutAxisPartsTopStrategy

```
layoutAxisPartsTopStrategy(axisRendererWidth: number, axisRendererHeight: number,
axisTitleRendererWidth: number, axisTitleRendererHeight: number, containerBounds:
Rect): TAxisViewRects
```

Defined in src/Charting/LayoutManager/AxisLayoutHelpers.ts

Parameters

- **axisRendererWidth:** *number*
- **axisRendererHeight:** *number*
- **axisTitleRendererWidth:** *number*
- **axisTitleRendererHeight:** *number*
- **containerBounds:** *Rect*

Returns *TAxisViewRects*

linearColorMapLerp

`linearColorMapLerp(colorMap: TLinearColorMap, dataValue: number): number`

Defined in src/utils/colorUtil.ts

Linearly interpolates a data-value in a TLinearColorMap, which specifies Gradient Stops, Data Minimum and Maximum and color stepping mode

Parameters

- **colorMap:** *TLinearColorMap*
- **dataValue:** *number*

Returns *number*

Const logDoubleVector

`logDoubleVector(vector: SCRTDoubleVector, name?: string, precision?: number): void`

Defined in src/utils/debug.ts

Parameters

- **vector:** *SCRTDoubleVector*
- Optional **name:** *string*
- Default value **precision:** *number* = 2

Returns *void*

Const logToBase

`logToBase(n: number, base: number): number`

Defined in src/utils/math.ts

Parameters

- **n:** *number*
- **base:** *number*

Returns *number*

Const makeCacheKey

`makeCacheKey(text: string, styleId: string): string`

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelCache.ts

Parameters

- **text:** *string*
- **styleId:** *string*

Returns *string*

makeIncArray

```
makeIncArray(length: number, multiplier?: number, map?: (n: number, index?: number) =>
number): makeIncArray
```

Defined in src/utils/array.ts

Helper method for generating an array of a given length, where the values are the indicies An optional multiplier and map function can be applied.

Parameters

- **length:** *number*
- Optional **multiplier:** *number*
- Optional **map:** (**n**: *number*, **index**?: *number*) => *number*

  ```
  (n: number, index?: number): number
  ```

  Parameters

  - **n:** *number*
  - Optional **index:** *number*

  Returns *number*

Returns *makeIncArray*

Const measureTextHeight

```
measureTextHeight(fontSizePx: number): number
```

Defined in src/Charting/Visuals/TextureManager/TextureManager.ts

Parameters

- **fontSizePx:** *number*

Returns *number*

Const measureTextWidth

```
measureTextWidth(ctx: CanvasRenderingContext2D, text: string): number
```

Defined in src/Charting/Visuals/TextureManager/TextureManager.ts

Parameters

- **ctx:** *CanvasRenderingContext2D*
- **text:** *string*

Returns *number*

memoize

```
memoize<ResultType, ParamsListType>(heavyCalculationFunction: (...params:
ParamsListType) => ResultType, equalityComparisonFunction?: (params: ParamsListType,
prevParams: ParamsListType) => boolean): memoize
```

Defined in src/utils/memoize.ts

Type parameters

- **ResultType**
- **ParamsListType**: *any*[]

Parameters

- **heavyCalculationFunction:** (...**params**: *ParamsListType*) => *ResultType*

(...params: *ParamsListType*): *ResultType*

Parameters

- Rest ...**params:** *ParamsListType*

Returns *ResultType*

- Optional **equalityComparisonFunction:** (**params**: *ParamsListType*, **prevParams**: *ParamsListType*) => *boolean*

  (params: *ParamsListType*, prevParams: *ParamsListType*): *boolean*

  Parameters

  - **params:** *ParamsListType*
  - **prevParams:** *ParamsListType*

  Returns *boolean*

Returns *memoize*

Const numericHashCode

numericHashCode(hash: *number*, value: *number*): *number*

Defined in src/utils/number.ts

Parameters

- **hash:** *number*
- **value:** *number*

Returns *number*

Const parseArgbToHtmlColor

parseArgbToHtmlColor(argbColor: *number*): *string*

Defined in src/utils/parseColor.ts

Converts color from ARGB number format into HTML string format #FFFFFFFF

Parameters

- **argbColor:** *number*
  color in ARGB format

Returns *string*

Const parseCacheKey

parseCacheKey(key: *string*): { styleId: *string*; text: *string* }

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelCache.ts

Parameters

- **key:** *string*

Returns { **styleId**: *string*; **text**: *string* }

- **styleId**: *string*
- **text**: *string*

parseColorToHexStringAbgr

parseColorToHexStringAbgr(input: *string*, opacityOverride?: *number*): *string*

Defined in src/utils/parseColor.ts

Parse colors and convert them to hex string in ABGR format to use in c++ Examples: "#fff", "#ff0000", "rgba(255,255,0,1)", "#11333333"

Parameters

- **input:** *string*
- Optional **opacityOverride:** *number*

Returns *string*

parseColorToHexStringArgb

`parseColorToHexStringArgb(input: string, opacityOverride?: number): string`

Defined in src/utils/parseColor.ts

Parse colors and convert them to hex string to use in c++ Examples: "#fff", "#ff0000", "rgba(255,255,0,1)", "#11333333"

Parameters

- **input:** *string*
- Optional **opacityOverride:** *number*

Returns *string*

parseColorToTArgb

`parseColorToTArgb(input: string): TArgb`

Defined in src/utils/parseColor.ts

Converts an HTML color code to TArgb

Parameters

- **input:** *string*

Returns *TArgb*

parseColorToUIntAbgr

`parseColorToUIntAbgr(input: string, opacity?: number): number`

Defined in src/utils/parseColor.ts

Parameters

- **input:** *string*
- Optional **opacity:** *number*

Returns *number*

parseColorToUIntArgb

`parseColorToUIntArgb(input: string, opacity?: number): number`

Defined in src/utils/parseColor.ts

Parameters

- **input:** *string*
- Optional **opacity:** *number*

Returns *number*

Const parsePc

`parsePc(value: string, containerSize: number): number`

Defined in src/Charting/LayoutManager/BaseAxisLayoutStrategy.ts

Parameters

- **value:** *string*
- **containerSize:** *number*

Returns *number*

Const parseSize

`parseSize(value: number | string, containerSize: number): number`

Defined in src/Charting/LayoutManager/BaseAxisLayoutStrategy.ts

Parameters

- **value:** *number* | *string*
- **containerSize:** *number*

Returns *number*

Const parseTArgbToHtmlColor

`parseTArgbToHtmlColor(targb: TArgb): string`

Defined in src/utils/parseColor.ts

Useful for debugging purposes. Converts TArgb to HTML Color code e.g. '#FF112233'=RGBA

Parameters

- **targb:** *TArgb*

Returns *string*

Const pruneCache

`pruneCache(): void`

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelCache.ts

Returns *void*

Const registerFunction

`registerFunction<T>(baseType: EBaseType, type: string, constructor: T, overWrite?: boolean): void`

Defined in src/Builder/classFactory.ts

Register a pure function

Type parameters

- **T**: *Function*

Parameters

- **baseType:** *EBaseType*

  The base type for, which indicates where the function will be used, Either OptionFunction or OnCreated

- **type:** *string*

  The name of the function

- **constructor:** *T*

  the function to register

- Default value **overWrite:** *boolean* = false

  Whether to override an existing registration for the type name. Default false.

Returns *void*

Const registerType

```
registerType<T>(baseType: EBaseType, type: string, constructor: (options?: any) => T, overWrite?: boolean): void
```

Defined in src/Builder/classFactory.ts

Register a function that returns an object of the specified type.

Type parameters

- **T**: *object*

Parameters

- **baseType:** *EBaseType*

  The EBaseType of the object that is being registered.

- **type:** *string*

  The name of the type, which should be a subtype of the baseType. For custom types, use the actual type name, not the "Custom" value on the subType enum.

- **constructor:** (**options**?: *any*) => *T*

  A function that takes an optional options argument and returns an instance of the desired type.

  ```
  (options?: any): T
  ```

  Parameters

  - Optional **options:** *any*

  Returns *T*

- Default value **overWrite:** *boolean* = false

  Whether to override an existing registration for the type. Default false.

Returns *void*

Const registerWasmType

```
registerWasmType<T>(baseType: EBaseType, type: string, constructor: (wasmContext: TSciChart | TSciChart3D, options?: any) => T, overWrite?: boolean): void
```

Defined in src/Builder/classFactory.ts

Register a function that requires a webAssemblyContext to return an object of the specified type.

Type parameters

- **T**: *object*

Parameters

- **baseType:** *EBaseType*

  The EBaseType of the object that is being registered.

- **type:** *string*

  The name of the type, which should be a subtype of the baseType. For custom types, use the actual type name, not the "Custom" value on the subType enum.

- **constructor:** (**wasmContext**: *TSciChart* | *TSciChart3D*, **options**?: *any*) => *T*

  A function that takes a SciChart 2D WebAssembly Context or SciChart 3D WebAssembly Context and an optional options argument and returns an instance of the desired type.

  ```
  (wasmContext: TSciChart | TSciChart3D, options?: any): T
  ```

  Parameters

  - **wasmContext:** *TSciChart* | *TSciChart3D*
  - Optional **options:** *any*

Returns *T*

- Default value **overWrite:** *boolean* = false

  Whether to override an existing registration for the type. Default false.

Returns *void*

Const resetAxes

```
resetAxes(axes: AxisBase2D[]): void
```

Defined in src/Charting/LayoutManager/LayoutManager.ts

Parameters

- **axes:** *AxisBase2D*[]

Returns *void*

Const resetCache

```
resetCache(): void
```

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelCache.ts

Returns *void*

Const runIfValue

```
runIfValue(value: any, fn: (value: any) => any): any
```

Defined in src/Charting/Model/DataPointSelectionPaletteProvider.ts

Parameters

- **value:** *any*
- **fn:** (**value**: *any*) => *any*

  ```
  (value: any): any
  ```

  Parameters

  - **value:** *any*

  Returns *any*

Returns *any*

Const sendTelemetry

```
sendTelemetry(): void
```

Defined in src/Core/Telemetry.ts

Returns *void*

Const setDevCount

```
setDevCount(value: number): void
```

Defined in src/Charting/Visuals/licenseManager2dState.ts

Parameters

- **value:** *number*

Returns *void*

Const setIsDev

`setIsDev(value: boolean): void`

Defined in src/Charting/Visuals/licenseManager2dState.ts

Parameters

- **value:** *boolean*

Returns *void*

setIsLicenseDebug

`setIsLicenseDebug(value: boolean): setIsLicenseDebug`

Defined in src/Core/storage/localStorageApi.ts

Parameters

- **value:** *boolean*

Returns *setIsLicenseDebug*

Const setLabel

`setLabel(text: string, styleId: string, label: LabelInfo): void`

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelCache.ts

Parameters

- **text:** *string*
- **styleId:** *string*
- **label:** *LabelInfo*

Returns *void*

Const setLicenseType

`setLicenseType(value: LicenseType): void`

Defined in src/Charting/Visuals/licenseManager2dState.ts

Parameters

- **value:** *LicenseType*

Returns *void*

Const setMaxSize

`setMaxSize(size: number): void`

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelCache.ts

Parameters

- **size:** *number*

Returns *void*

Const setMinAge

`setMinAge(ageInMs: number): void`

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelCache.ts

Parameters

- **ageInMs:** *number*

Returns *void*

Const setOrderId

`setOrderId(value: string): void`

Defined in src/Charting/Visuals/licenseManager2dState.ts

Parameters

- **value:** *string*

Returns *void*

Const setProductCode

`setProductCode(value: string): void`

Defined in src/Charting/Visuals/licenseManager2dState.ts

Parameters

- **value:** *string*

Returns *void*

Const setTelemetry

`setTelemetry(value: boolean): void`

Defined in src/Charting/Visuals/licenseManager2dState.ts

Parameters

- **value:** *boolean*

Returns *void*

Const setUserCookie

`setUserCookie(userId: string, sessionId: string, sessionStart: number): void`

Defined in src/Core/Telemetry.ts

Parameters

- **userId:** *string*
- **sessionId:** *string*
- **sessionStart:** *number*

Returns *void*

Const shouldSendTelemetry

`shouldSendTelemetry(): boolean`

Defined in src/Core/Telemetry.ts

Returns *boolean*

storageAvailable

`storageAvailable(): storageAvailable`

Defined in src/Core/storage/localStorageApi.ts

Returns *storageAvailable*

stringOccurrences

```
stringOccurrences(str: string, subStr: string, allowOverlapping?: boolean): stringOccurrences
```

Defined in src/utils/html.ts

Parameters

- **str:** *string*
- **subStr:** *string*
- Default value **allowOverlapping:** *boolean* = false

Returns *stringOccurrences*

Const stripAutoColor

```
stripAutoColor(val: string): string
```

Defined in src/Charting/Themes/IThemeProvider.ts

Parameters

- **val:** *string*

Returns *string*

subArray

```
subArray(array: NumberArray, startIndex?: number, endIndex?: number): subArray
```

Defined in src/types/NumberArray.ts

Parameters

- **array:** *NumberArray*
- Optional **startIndex:** *number*
- Optional **endIndex:** *number*

Returns *subArray*

Const switchData

```
switchData(field: EDataSeriesField, x: any, closey: any, openy1z?: any, high?: any, low?: any): any
```

Defined in src/Charting/Model/Filters/XyFilterBase.ts

Parameters

- **field:** *EDataSeriesField*
- **x:** *any*
- **closey:** *any*
- Optional **openy1z:** *any*
- Optional **high:** *any*
- Optional **low:** *any*

Returns *any*

Const testHasExcluded

```
testHasExcluded(axisMap: Map<string, boolean>): boolean
```

Defined in src/utils/includedAxis.ts

Parameters

- **axisMap:** *Map<string, boolean>*

Returns *boolean*

Const testIsHitForBand

```
testIsHitForBand(isDigitalLine: boolean, xCoordinateCalculator: CoordinateCalculatorBase, yCoordinateCalculator: CoordinateCalculatorBase, xValues: SCRTDoubleVector, getYValue: (index: number) => number, getY1Value: (index: number) => number, pointIndex: number, xHitCoord: number, yHitCoord: number, dataSeries: BaseDataSeries): { isHit: boolean; secondPointIndex: number }
```

Defined in src/Charting/Visuals/RenderableSeries/HitTest/hitTestHelpers.ts

Parameters

- **isDigitalLine:** *boolean*
- **xCoordinateCalculator:** *CoordinateCalculatorBase*
- **yCoordinateCalculator:** *CoordinateCalculatorBase*
- **xValues:** *SCRTDoubleVector*
- **getYValue:** (**index**: *number*) => *number*

  ```
  (index: number): number
  ```

  Parameters

  - **index:** *number*

  Returns *number*

- **getY1Value:** (**index**: *number*) => *number*

  ```
  (index: number): number
  ```

  Parameters

  - **index:** *number*

  Returns *number*

- **pointIndex:** *number*
- **xHitCoord:** *number*
- **yHitCoord:** *number*
- **dataSeries:** *BaseDataSeries*

Returns { **isHit**: *boolean*; **secondPointIndex**: *number* }

- **isHit**: *boolean*
- **secondPointIndex**: *number*

Const testIsHitForColumn

```
testIsHitForColumn(xCoordinateCalculator: CoordinateCalculatorBase, yCoordinateCalculator: CoordinateCalculatorBase, renderableSeries: FastColumnRenderableSeries, xValues: SCRTDoubleVector, yValues: SCRTDoubleVector, pointIndex: number, xHitCoord: number, yHitCoord: number): boolean
```

Defined in src/Charting/Visuals/RenderableSeries/HitTest/hitTestHelpers.ts

Parameters

- **xCoordinateCalculator:** *CoordinateCalculatorBase*
- **yCoordinateCalculator:** *CoordinateCalculatorBase*
- **renderableSeries:** *FastColumnRenderableSeries*
- **xValues:** *SCRTDoubleVector*

- **yValues:** *SCRTDoubleVector*
- **pointIndex:** *number*
- **xHitCoord:** *number*
- **yHitCoord:** *number*

Returns *boolean*

Const testIsHitForErrorBars

```
testIsHitForErrorBars(xCoordinateCalculator: CoordinateCalculatorBase,
yCoordinateCalculator: CoordinateCalculatorBase, renderableSeries:
FastErrorBarsRenderableSeries, xValues: SCRTDoubleVector, yValues: SCRTDoubleVector,
pointIndex: number, xHitCoord: number, yHitCoord: number): { highValue: number; isHit:
boolean; lowValue: number }
```

Defined in src/Charting/Visuals/RenderableSeries/HitTest/hitTestHelpers.ts

Parameters

- **xCoordinateCalculator:** *CoordinateCalculatorBase*
- **yCoordinateCalculator:** *CoordinateCalculatorBase*
- **renderableSeries:** *FastErrorBarsRenderableSeries*
- **xValues:** *SCRTDoubleVector*
- **yValues:** *SCRTDoubleVector*
- **pointIndex:** *number*
- **xHitCoord:** *number*
- **yHitCoord:** *number*

Returns { **highValue**: *number*; **isHit**: *boolean*; **lowValue**: *number* }

- **highValue**: *number*
- **isHit**: *boolean*
- **lowValue**: *number*

Const testIsHitForImpulse

```
testIsHitForImpulse(xCoordinateCalculator: CoordinateCalculatorBase,
yCoordinateCalculator: CoordinateCalculatorBase, renderableSeries:
FastImpulseRenderableSeries, xValues: SCRTDoubleVector, yValues: SCRTDoubleVector,
pointIndex: number, xHitCoord: number, yHitCoord: number, hitTestRadius: number):
boolean
```

Defined in src/Charting/Visuals/RenderableSeries/HitTest/hitTestHelpers.ts

Parameters

- **xCoordinateCalculator:** *CoordinateCalculatorBase*
- **yCoordinateCalculator:** *CoordinateCalculatorBase*
- **renderableSeries:** *FastImpulseRenderableSeries*
- **xValues:** *SCRTDoubleVector*
- **yValues:** *SCRTDoubleVector*
- **pointIndex:** *number*
- **xHitCoord:** *number*
- **yHitCoord:** *number*
- **hitTestRadius:** *number*

Returns *boolean*

Const testIsHitForLine

testIsHitForLine(xCoordinateCalculator: *CoordinateCalculatorBase*, yCoordinateCalculator: *CoordinateCalculatorBase*, xValues: *SCRTDoubleVector*, yValues: *SCRTDoubleVector*, pointIndex: *number*, xHitCoord: *number*, yHitCoord: *number*, hitTestRadius: *number*, dataSeries: *BaseDataSeries*): { isHit: *boolean*; secondPointIndex: *number* }

Defined in src/Charting/Visuals/RenderableSeries/HitTest/hitTestHelpers.ts

Parameters

- **xCoordinateCalculator:** *CoordinateCalculatorBase*
- **yCoordinateCalculator:** *CoordinateCalculatorBase*
- **xValues:** *SCRTDoubleVector*
- **yValues:** *SCRTDoubleVector*
- **pointIndex:** *number*
- **xHitCoord:** *number*

  The X coordinate, isVertical property is already taken into account
- **yHitCoord:** *number*

  The Y coordinate, isVertical property is already taken into account
- **hitTestRadius:** *number*
- **dataSeries:** *BaseDataSeries*

Returns { **isHit**: *boolean*; **secondPointIndex**: *number* }

- **isHit**: *boolean*
- **secondPointIndex**: *number*

Const testIsHitForMountain

testIsHitForMountain(isDigitalLine: *boolean*, xCoordinateCalculator: *CoordinateCalculatorBase*, yCoordinateCalculator: *CoordinateCalculatorBase*, dataSeries: *XyDataSeries*, zeroLineY: *number*, pointIndex: *number*, xHitCoord: *number*, yHitCoord: *number*): { isHit: *boolean*; secondPointIndex: *number* }

Defined in src/Charting/Visuals/RenderableSeries/HitTest/hitTestHelpers.ts

Parameters

- **isDigitalLine:** *boolean*
- **xCoordinateCalculator:** *CoordinateCalculatorBase*
- **yCoordinateCalculator:** *CoordinateCalculatorBase*
- **dataSeries:** *XyDataSeries*
- **zeroLineY:** *number*
- **pointIndex:** *number*
- **xHitCoord:** *number*
- **yHitCoord:** *number*

Returns { **isHit**: *boolean*; **secondPointIndex**: *number* }

- **isHit**: *boolean*
- **secondPointIndex**: *number*

Const testIsHitForOHLC

testIsHitForOHLC(xCoordinateCalculator: *CoordinateCalculatorBase*, yCoordinateCalculator: *CoordinateCalculatorBase*, renderableSeries: *FastColumnRenderableSeries* | *BaseOhlcRenderableSeries*, dataSeries: *OhlcDataSeries*, pointIndex: *number*, xHitCoord: *number*, yHitCoord: *number*, hitTestRadius: *number*): { closeValue: *number*; highValue: *number*; isHit: *boolean*; lowValue: *number*; openValue: *number* }

Defined in src/Charting/Visuals/RenderableSeries/HitTest/hitTestHelpers.ts

Parameters

- **xCoordinateCalculator:** *CoordinateCalculatorBase*
- **yCoordinateCalculator:** *CoordinateCalculatorBase*
- **renderableSeries:** *FastColumnRenderableSeries* | *BaseOhlcRenderableSeries*
- **dataSeries:** *OhlcDataSeries*
- **pointIndex:** *number*
- **xHitCoord:** *number*
- **yHitCoord:** *number*
- **hitTestRadius:** *number*

Returns { **closeValue**: *number*; **highValue**: *number*; **isHit**: *boolean*; **lowValue**: *number*; **openValue**: *number* }

- **closeValue**: *number*
- **highValue**: *number*
- **isHit**: *boolean*
- **lowValue**: *number*
- **openValue**: *number*

Const testIsHitForPoint

```
testIsHitForPoint(xCoordinateCalculator: CoordinateCalculatorBase,
yCoordinateCalculator: CoordinateCalculatorBase, xValues: SCRTDoubleVector, yValues:
SCRTDoubleVector, pointIndex: number, xHitCoord: number, yHitCoord: number,
hitTestRadius: number, dataSeries: BaseDataSeries): boolean
```

Defined in src/Charting/Visuals/RenderableSeries/HitTest/hitTestHelpers.ts

Parameters

- **xCoordinateCalculator:** *CoordinateCalculatorBase*
- **yCoordinateCalculator:** *CoordinateCalculatorBase*
- **xValues:** *SCRTDoubleVector*
- **yValues:** *SCRTDoubleVector*
- **pointIndex:** *number*
- **xHitCoord:** *number*
- **yHitCoord:** *number*
- **hitTestRadius:** *number*
- **dataSeries:** *BaseDataSeries*

Returns *boolean*

Const testIsInBounds

```
testIsInBounds(x: number, y: number, left: number, bottom: number, right: number, top:
number, radius?: number): boolean
```

Defined in src/utils/pointUtil.ts

Tests whether a point is within rectangle bounds

Parameters

- **x:** *number*
- **y:** *number*
- **left:** *number*
- **bottom:** *number*
- **right:** *number*
- **top:** *number*

- Default value **radius:** *number* = 0

Returns *boolean*

Const testIsInInterval

```
testIsInInterval(x: number, intervalStart: number, intervalEnd: number, radius?: number): boolean
```

Defined in src/utils/pointUtil.ts

Tests if X is within radius from the [intervalStart, intervalEnd] interval, intervalStart, intervalEnd values might not be sorted

Parameters

- **x:** *number*
- **intervalStart:** *number*
- **intervalEnd:** *number*
- Default value **radius:** *number* = 0

Returns *boolean*

Const testIsInXBounds

```
testIsInXBounds(xHitTestPoint: number, xDataPointCoord: number, maxDistance: number): boolean
```

Defined in src/utils/pointUtil.ts

Parameters

- **xHitTestPoint:** *number*
- **xDataPointCoord:** *number*
- **maxDistance:** *number*

Returns *boolean*

Const testIsOverAxes

```
testIsOverAxes(xAxisArr: AxisBase2D[], mousePoint: Point): boolean
```

Defined in src/Charting/ChartModifiers/ChartModifierBase2D.ts

Parameters

- **xAxisArr:** *AxisBase2D*[]
- **mousePoint:** *Point*

Returns *boolean*

Const toEngineering

```
toEngineering(value: number, largePrefixes?: string[], smallPrefixes?: string[]): string
```

Defined in src/utils/number.ts

Parameters

- **value:** *number*
- Optional **largePrefixes:** *string*[]
- Optional **smallPrefixes:** *string*[]

Returns *string*

Const toHex

```
toHex(value: number): toHex
```

```
toHex(value: number): string
```

Defined in src/utils/parseColor.ts

Converts a UINT color number to a 8-digit hex code e.g. #FFAABBCC

Parameters

- **value:** *number*

Returns *toHex*

Const toScientific

```
toScientific(value: number, precision: number, logarithmicBase: number): string
```

Defined in src/utils/number.ts

Parameters

- **value:** *number*
- **precision:** *number*
- **logarithmicBase:** *number*

Returns *string*

Const toSuperScript

```
toSuperScript(value: number): string
```

Defined in src/utils/number.ts

Parameters

- **value:** *number*

Returns *string*

Const translateDataValueRectToAbsolute

```
translateDataValueRectToAbsolute(originalRect: Rect, xAxis: AxisBase2D, yAxis: AxisBase2D, seriesViewRect: Rect): Rect
```

Defined in src/utils/translate.ts

Parameters

- **originalRect:** *Rect*
- **xAxis:** *AxisBase2D*
- **yAxis:** *AxisBase2D*
- **seriesViewRect:** *Rect*

Returns *Rect*

Const translateFromCanvasToSeriesViewRect

```
translateFromCanvasToSeriesViewRect(point: Point, seriesViewRect: Rect, allowValuesOutOfBounds?: boolean): Point
```

Defined in src/utils/translate.ts

Translates from canvas to seriesViewRect screen coordinates

Parameters

- **point:** *Point*

- **seriesViewRect:** *Rect*
- Default value **allowValuesOutOfBounds:** *boolean* = false

  will translate even if the point is outside of the seriesViewRect

Returns *Point*

Const translateFromCanvasToSeriesViewRectX

```
translateFromCanvasToSeriesViewRectX(x: number, seriesViewRect: Rect,
allowValuesOutOfBounds?: boolean): number
```

Defined in src/utils/translate.ts

Parameters

- **x:** *number*
- **seriesViewRect:** *Rect*
- Default value **allowValuesOutOfBounds:** *boolean* = false

Returns *number*

Const translateFromCanvasToSeriesViewRectY

```
translateFromCanvasToSeriesViewRectY(y: number, seriesViewRect: Rect,
allowValuesOutOfBounds?: boolean): number
```

Defined in src/utils/translate.ts

Parameters

- **y:** *number*
- **seriesViewRect:** *Rect*
- Default value **allowValuesOutOfBounds:** *boolean* = false

Returns *number*

Const translateFromSeriesViewRectToCanvas

```
translateFromSeriesViewRectToCanvas(point: Point, seriesViewRect: Rect,
allowValuesOutOfBounds?: boolean): Point
```

Defined in src/utils/translate.ts

Translates from seriesViewRect to canvas screen coordinates

Parameters

- **point:** *Point*
- **seriesViewRect:** *Rect*
- Default value **allowValuesOutOfBounds:** *boolean* = false

  will translate even if the point is outside of the seriesViewRect

Returns *Point*

Const translateFromSeriesViewRectToCanvasX

```
translateFromSeriesViewRectToCanvasX(x: number, seriesViewRect: Rect,
allowValuesOutOfBounds?: boolean): number
```

Defined in src/utils/translate.ts

Parameters

- **x:** *number*
- **seriesViewRect:** *Rect*
- Default value **allowValuesOutOfBounds:** *boolean* = false

Returns *number*

Const translateFromSeriesViewRectToCanvasY

```
translateFromSeriesViewRectToCanvasY(y: number, seriesViewRect: Rect,
allowValuesOutOfBounds?: boolean): number
```

Defined in src/utils/translate.ts

Parameters

- **y:** *number*
- **seriesViewRect:** *Rect*
- Default value **allowValuesOutOfBounds:** *boolean* = false

Returns *number*

Const translateToNotScaled

```
translateToNotScaled(value: number): number
```

Defined in src/utils/translate.ts

Parameters

- **value:** *number*

Returns *number*

uintArgbColorIsTransparent

```
uintArgbColorIsTransparent(argbColor: number): boolean
```

Defined in src/utils/colorUtil.ts

Parameters

- **argbColor:** *number*

Returns *boolean*

uintArgbColorLerp

```
uintArgbColorLerp(from: number, to: number, ratio: number): number
```

Defined in src/utils/colorUtil.ts

Linearly interpolates between two colors based on the ratio passed in. E.g. Ratio = 0.0f returns From color, ratio = 1.0f returns To Color. Ratio = 0.5f returns a mix of the two Works for 32 bit colors

Parameters

- **from:** *number*

  the start color, for example 0x0000ffff (format ARGB)

- **to:** *number*

  the end color, for example 0xffffffff

- **ratio:** *number*

  the value between 0 and 1

Returns *number*

uintArgbColorLerp24bit

```
uintArgbColorLerp24bit(from: number, to: number, ratio: number): number
```

Defined in src/utils/colorUtil.ts

Linearly interpolates between two colors based on the ratio passed in. E.g. Ratio = 0.0f returns From color, ratio = 1.0f returns To Color. Ratio = 0.5f returns a mix of the two. Works only for numbers not more than 24 bit

Parameters

- **from:** *number*

  the start color, for example 0xff0000

- **to:** *number*

  the end color, for example 0x00ff00

- **ratio:** *number*

  the value between 0 and 1

Returns *number*

uintArgbColorMultiplyOpacity

`uintArgbColorMultiplyOpacity(argbColor: number, opacity: number): number`

Defined in src/utils/colorUtil.ts

Parameters

- **argbColor:** *number*
- **opacity:** *number*

Returns *number*

uintArgbColorOverrideOpacity

`uintArgbColorOverrideOpacity(argbColor: number, opacity: number): number`

Defined in src/utils/colorUtil.ts

Parameters

- **argbColor:** *number*
- **opacity:** *number*

Returns *number*

uintArgbColorToAbgr

`uintArgbColorToAbgr(argbColor: number): number`

Defined in src/utils/colorUtil.ts

Parameters

- **argbColor:** *number*

Returns *number*

updateAxisLayoutState

`updateAxisLayoutState(axis: AxisBase2D): void`

Defined in src/Charting/LayoutManager/AxisLayoutHelpers.ts

Parameters

- **axis:** *AxisBase2D*

Returns *void*

updateLeftAndRightChartLayoutState

```
updateLeftAndRightChartLayoutState(chartLayoutState: ChartLayoutState,
additionalLeftSize?: number, additionalRightSize?: number): void
```

Defined in src/Charting/LayoutManager/AxisLayoutHelpers.ts

Parameters

- **chartLayoutState:** *ChartLayoutState*
- Default value **additionalLeftSize:** *number* = 0
- Default value **additionalRightSize:** *number* = 0

Returns *void*

updateTopAndBottomChartLayoutState

```
updateTopAndBottomChartLayoutState(chartLayoutState: ChartLayoutState,
additionalTopSize?: number, additionalBottomSize?: number): void
```

Defined in src/Charting/LayoutManager/AxisLayoutHelpers.ts

Parameters

- **chartLayoutState:** *ChartLayoutState*
- Default value **additionalTopSize:** *number* = 0
- Default value **additionalBottomSize:** *number* = 0

Returns *void*

Const updateTsrVector4

```
updateTsrVector4(color: TArgb, tsrColor: TSRVector4): void
```

Defined in src/utils/tsrExtensions.ts

Converts a color in TArgb format to TSRVector4 (RGBA -> x,y,z,w) for use in 3D Engine

Parameters

- **color:** *TArgb*
- **tsrColor:** *TSRVector4*

Returns *void*

Const wrapNativeText

```
wrapNativeText(text: string, maxWidth: number, font: SCRTFont, textBounds:
TSRTextBounds): string
```

Defined in src/utils/text.ts

Wrap a string by adding newline characters, splitting on spaces and wrapping to a maximum size

Parameters

- **text:** *string*
- **maxWidth:** *number*
- **font:** *SCRTFont*
- **textBounds:** *TSRTextBounds*

Returns *string*

Const wrapText

```
wrapText(text: string, maxLength: number): string[]
```

Defined in src/Charting/Visuals/Axis/LabelProvider/TextLabelProvider.ts

Convert a string into an array of lines by splitting on spaces and wrapping to a maximum number of characters

Parameters

- **text:** *string*
- **maxLength:** *number*

Returns *string*[]

Const zeroArray2D

`zeroArray2D(dimensions: number[]): number[][]`

Defined in src/utils/zeroArray2D.ts

description creates a 2-dimensional array filled with zeros

Parameters

- **dimensions:** *number*[]

Returns *number*[][]

Object literals

Const animationHelpers

`animationHelpers: object`

Defined in src/Charting/Visuals/RenderableSeries/Animations/animationHelpers.ts

animationUpdate

`animationUpdate: animationUpdate`

Defined in src/Charting/Visuals/RenderableSeries/Animations/animationHelpers.ts

checkCanDraw

`checkCanDraw: checkCanDraw`

Defined in src/Charting/Visuals/RenderableSeries/Animations/animationHelpers.ts

checkIsAnimationRunning

`checkIsAnimationRunning: checkIsAnimationRunning`

Defined in src/Charting/Visuals/RenderableSeries/Animations/animationHelpers.ts

copyVector

`copyVector: copyVector`

Defined in src/Charting/Visuals/RenderableSeries/Animations/animationHelpers.ts

createPointMarker

`createPointMarker: createPointMarker`

Defined in src/Charting/Visuals/RenderableSeries/Animations/animationHelpers.ts

interpolateColor

● interpolateColor: *interpolateColor*

Defined in src/Charting/Visuals/RenderableSeries/Animations/animationHelpers.ts

interpolateNumber

● interpolateNumber: *interpolateNumber*

Defined in src/Charting/Visuals/RenderableSeries/Animations/animationHelpers.ts

Const annotationHelpers

annotationHelpers: *object*

Defined in src/Charting/Visuals/Annotations/annotationHelpers.ts

calcNewApex

● calcNewApex: *calcNewApex*

Defined in src/Charting/Visuals/Annotations/annotationHelpers.ts

createSvg

● createSvg: *createSvg*

Defined in src/Charting/Visuals/Annotations/annotationHelpers.ts

Const autoReverseEasing

autoReverseEasing: *object*

Defined in src/Core/Animations/EasingFunctions.ts

Reversable functions that go from 0..1..0 used throughout SciChart when animations are used

linear

linear(t: *number*): *number*

Defined in src/Core/Animations/EasingFunctions.ts

No easing acceleration like linear, but it does reverse from 0..1..0

Parameters

- **t:** *number*

Returns *number*

Const chartBuilder

chartBuilder: *object*

Defined in src/Builder/chartBuilder.ts

build2DChart

● build2DChart: *build2DChart*

Defined in src/Builder/chartBuilder.ts

inheritdoc

buildAnnotations

● buildAnnotations: *buildAnnotations*

Defined in src/Builder/chartBuilder.ts

inheritdoc

buildAxes

● buildAxes: *buildAxes*

Defined in src/Builder/chartBuilder.ts

inheritdoc

buildChart

● buildChart: *buildChart*

Defined in src/Builder/chartBuilder.ts

inheritdoc

buildDataSeries

● buildDataSeries: *buildDataSeries*

Defined in src/Builder/chartBuilder.ts

inheritdoc

buildModifiers

● buildModifiers: *buildModifiers*

Defined in src/Builder/chartBuilder.ts

inheritdoc

buildPieChart

● buildPieChart: *buildPieChart*

Defined in src/Builder/chartBuilder.ts

inheritdoc

buildSeries

● buildSeries: *buildSeries*

Defined in src/Builder/chartBuilder.ts

inheritdoc

chartReviver

● chartReviver: *chartReviver*

Defined in src/Builder/chartBuilder.ts

inheritdoc

configureChart

● configureChart: *configureChart*

Defined in src/Builder/chartBuilder.ts

inheritdoc

registerFunction

● registerFunction: *registerFunction*

Defined in src/Builder/chartBuilder.ts

inheritdoc

registerType

● registerType: *registerType*

Defined in src/Builder/chartBuilder.ts

inheritdoc

registerWasmType

● registerWasmType: *registerWasmType*

Defined in src/Builder/chartBuilder.ts

inheritdoc

Const easing

easing: *object*

Defined in src/Core/Animations/EasingFunctions.ts

Easing functions used throughout SciChart when animations are used

cubic

● cubic(t: *number*): *number*

Defined in src/Core/Animations/EasingFunctions.ts

inheritdoc

Parameters

- **t:** *number*

Returns *number*

elastic

● elastic(t: *number*): *number*

Defined in src/Core/Animations/EasingFunctions.ts

inheritdoc

Parameters

- **t:** *number*

Returns *number*

inCirc

`inCirc(t: number): number`

Defined in src/Core/Animations/EasingFunctions.ts

inheritdoc

Parameters

- **t:** *number*

Returns *number*

inCubic

`inCubic(t: number): number`

Defined in src/Core/Animations/EasingFunctions.ts

inheritdoc

Parameters

- **t:** *number*

Returns *number*

inExpo

`inExpo(t: number): number`

Defined in src/Core/Animations/EasingFunctions.ts

inheritdoc

Parameters

- **t:** *number*

Returns *number*

inOutCirc

`inOutCirc(t: number): number`

Defined in src/Core/Animations/EasingFunctions.ts

inheritdoc

Parameters

- **t:** *number*

Returns *number*

inOutCubic

`inOutCubic(t: number): number`

Defined in src/Core/Animations/EasingFunctions.ts

inheritdoc

Parameters

- **t:** *number*

Returns *number*

inOutExpo

`inOutExpo(t: number): number`

Defined in src/Core/Animations/EasingFunctions.ts

inheritdoc

Parameters

- **t:** *number*

Returns *number*

inOutQuad

`inOutQuad(t: number): number`

Defined in src/Core/Animations/EasingFunctions.ts

inheritdoc

Parameters

- **t:** *number*

Returns *number*

inOutQuart

`inOutQuart(t: number): number`

Defined in src/Core/Animations/EasingFunctions.ts

inheritdoc

Parameters

- **t:** *number*

Returns *number*

inOutQuint

`inOutQuint(t: number): number`

Defined in src/Core/Animations/EasingFunctions.ts

inheritdoc

Parameters

- **t:** *number*

Returns *number*

inOutSine

`inOutSine(t: number): number`

Defined in src/Core/Animations/EasingFunctions.ts

inheritdoc

Parameters

- **t:** *number*

Returns *number*

inQuad

`inQuad(t: number): number`

Defined in src/Core/Animations/EasingFunctions.ts

inheritdoc

Parameters

- **t:** *number*

Returns *number*

inQuart

`inQuart(t: number): number`

Defined in src/Core/Animations/EasingFunctions.ts

inheritdoc

Parameters

- **t:** *number*

Returns *number*

inQuint

`inQuint(t: number): number`

Defined in src/Core/Animations/EasingFunctions.ts

inheritdoc

Parameters

- **t:** *number*

Returns *number*

inSine

`inSine(t: number): number`

Defined in src/Core/Animations/EasingFunctions.ts

inheritdoc

Parameters

- **t:** *number*

Returns *number*

linear

linear(t: *number*): *number*

Defined in src/Core/Animations/EasingFunctions.ts

inheritdoc

Parameters

- **t:** *number*

Returns *number*

outCirc

outCirc(t: *number*): *number*

Defined in src/Core/Animations/EasingFunctions.ts

inheritdoc

Parameters

- **t:** *number*

Returns *number*

outCubic

outCubic(t: *number*): *number*

Defined in src/Core/Animations/EasingFunctions.ts

inheritdoc

Parameters

- **t:** *number*

Returns *number*

outExpo

outExpo(t: *number*): *number*

Defined in src/Core/Animations/EasingFunctions.ts

inheritdoc

Parameters

- **t:** *number*

Returns *number*

outQuad

outQuad(t: *number*): *number*

Defined in src/Core/Animations/EasingFunctions.ts

inheritdoc

Parameters

- **t:** *number*

Returns *number*

outQuart

outQuart(t: *number*): *number*

Defined in src/Core/Animations/EasingFunctions.ts

inheritdoc

Parameters

- **t:** *number*

Returns *number*

outQuint

outQuint(t: *number*): *number*

Defined in src/Core/Animations/EasingFunctions.ts

inheritdoc

Parameters

- **t:** *number*

Returns *number*

outSine

outSine(t: *number*): *number*

Defined in src/Core/Animations/EasingFunctions.ts

inheritdoc

Parameters

- **t:** *number*

Returns *number*

quadratic

quadratic(t: *number*): *number*

Defined in src/Core/Animations/EasingFunctions.ts

inheritdoc

Parameters

- **t:** *number*

Returns *number*

Const handler

handler: *object*

Defined in src/Core/DeletableEntity.ts

construct

construct(): *handler*

Defined in src/Core/DeletableEntity.ts

Returns *handler*

Const hashUtils

hashUtils: *object*

Defined in src/utils/hash.ts

generateBooleanHash

generateBooleanHash: *generateBooleanHash*

Defined in src/utils/hash.ts

generateCombinedHash

generateCombinedHash: *generateCombinedHash*

Defined in src/utils/hash.ts

generateHash

generateHash: *generateHash*

Defined in src/utils/hash.ts

generateNumberHash

generateNumberHash: *generateNumberHash*

Defined in src/utils/hash.ts

generateObjectHash

generateObjectHash: *generateObjectHash*

Defined in src/utils/hash.ts

Const hitTestHelpers

hitTestHelpers: *object*

Defined in src/Charting/Visuals/RenderableSeries/HitTest/hitTestHelpers.ts

createHitTestInfo

createHitTestInfo: *createHitTestInfo*

Defined in src/Charting/Visuals/RenderableSeries/HitTest/hitTestHelpers.ts

getNearestNonUniformHeatmapPoint

getNearestNonUniformHeatmapPoint: *getNearestNonUniformHeatmapPoint*

Defined in src/Charting/Visuals/RenderableSeries/HitTest/hitTestHelpers.ts

getNearestPoint

getNearestPoint: *getNearestPoint*

Defined in src/Charting/Visuals/RenderableSeries/HitTest/hitTestHelpers.ts

getNearestUniformHeatmapPoint

- getNearestUniformHeatmapPoint: *getNearestUniformHeatmapPoint*

Defined in src/Charting/Visuals/RenderableSeries/HitTest/hitTestHelpers.ts

getNearestXPoint

- getNearestXPoint: *getNearestXPoint*

Defined in src/Charting/Visuals/RenderableSeries/HitTest/hitTestHelpers.ts

getNearestXyPoint

- getNearestXyPoint: *getNearestXyPoint*

Defined in src/Charting/Visuals/RenderableSeries/HitTest/hitTestHelpers.ts

getNearestXyyPoint

- getNearestXyyPoint: *getNearestXyyPoint*

Defined in src/Charting/Visuals/RenderableSeries/HitTest/hitTestHelpers.ts

testIsHitForBand

- testIsHitForBand: *testIsHitForBand*

Defined in src/Charting/Visuals/RenderableSeries/HitTest/hitTestHelpers.ts

testIsHitForColumn

- testIsHitForColumn: *testIsHitForColumn*

Defined in src/Charting/Visuals/RenderableSeries/HitTest/hitTestHelpers.ts

testIsHitForErrorBars

- testIsHitForErrorBars: *testIsHitForErrorBars*

Defined in src/Charting/Visuals/RenderableSeries/HitTest/hitTestHelpers.ts

testIsHitForImpulse

- testIsHitForImpulse: *testIsHitForImpulse*

Defined in src/Charting/Visuals/RenderableSeries/HitTest/hitTestHelpers.ts

testIsHitForLine

- testIsHitForLine: *testIsHitForLine*

Defined in src/Charting/Visuals/RenderableSeries/HitTest/hitTestHelpers.ts

testIsHitForMountain

- testIsHitForMountain: *testIsHitForMountain*

Defined in src/Charting/Visuals/RenderableSeries/HitTest/hitTestHelpers.ts

testIsHitForOHLC

● testIsHitForOHLC: *testIsHitForOHLC*

Defined in src/Charting/Visuals/RenderableSeries/HitTest/hitTestHelpers.ts

testIsHitForPoint

● testIsHitForPoint: *testIsHitForPoint*

Defined in src/Charting/Visuals/RenderableSeries/HitTest/hitTestHelpers.ts

Const labelCache

labelCache: *object*

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelCache.ts

A global cache for labels, used by all labelProviders, to reduce the amount of time spent creating label textures.

checkStyle

● checkStyle: *checkStyle*

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelCache.ts

Check if a text style matches the one for the given id

freeStyle

● freeStyle: *freeStyle*

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelCache.ts

Notify the cache that a style is no longer used. Linked labels are only deleted when there are no remaining uses, and then only after minAge has passed.

getLabel

● getLabel: *getLabel*

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelCache.ts

Get a label from the cache. Returns undefined if none found.

getMaxSize

● getMaxSize: *getMaxSize*

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelCache.ts

Get the maximum number of labels allowed to be stored in the cache. Used when calling pruneCache

getMinAge

● getMinAge: *getMinAge*

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelCache.ts

Get the minimum age (time since last used) of labels in the cache. This prevents recently used labels from being pruned, or removed when style is freed

getSize

● getSize: *getSize*

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelCache.ts

getStyleId

● getStyleId: *getStyleId*

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelCache.ts

Get an identifier for the given text style. Returns an existing identifier if a matching style exists in the cache

pruneCache

● pruneCache: *pruneCache*

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelCache.ts

Remove old labels from the cache, if there are more than MaxSize.

resetCache

● resetCache: *resetCache*

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelCache.ts

Completely clears and resets the cache. Testing use only

setLabel

● setLabel: *setLabel*

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelCache.ts

Add a label to the cache

setMaxSize

● setMaxSize: *setMaxSize*

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelCache.ts

Set the maximum number of labels allowed to be stored in the cache. Used when calling pruneCache

setMinAge

● setMinAge: *setMinAge*

Defined in src/Charting/Visuals/Axis/LabelProvider/LabelCache.ts

Set the minimum age (time since last used) of labels in the cache. This prevents recently used labels from being pruned, or removed when style is freed

Const licenseManager2dState

licenseManager2dState: *object*

Defined in src/Charting/Visuals/licenseManager2dState.ts

getDevCount

getDevCount: *getDevCount*

Defined in src/Charting/Visuals/licenseManager2dState.ts

getIsDev

getIsDev: *getIsDev*

Defined in src/Charting/Visuals/licenseManager2dState.ts

getLicenseType

getLicenseType: *getLicenseType*

Defined in src/Charting/Visuals/licenseManager2dState.ts

getOrderId

getOrderId: *getOrderId*

Defined in src/Charting/Visuals/licenseManager2dState.ts

getProductCode

getProductCode: *getProductCode*

Defined in src/Charting/Visuals/licenseManager2dState.ts

getTelemetry

getTelemetry: *getTelemetry*

Defined in src/Charting/Visuals/licenseManager2dState.ts

setDevCount

setDevCount: *setDevCount*

Defined in src/Charting/Visuals/licenseManager2dState.ts

setIsDev

setIsDev: *setIsDev*

Defined in src/Charting/Visuals/licenseManager2dState.ts

setLicenseType

setLicenseType: *setLicenseType*

Defined in src/Charting/Visuals/licenseManager2dState.ts

setOrderId

● setOrderId: *setOrderId*

Defined in src/Charting/Visuals/licenseManager2dState.ts

setProductCode

● setProductCode: *setProductCode*

Defined in src/Charting/Visuals/licenseManager2dState.ts

setTelemetry

● setTelemetry: *setTelemetry*

Defined in src/Charting/Visuals/licenseManager2dState.ts

Const localStorageApi

localStorageApi: *object*

Defined in src/Core/storage/localStorageApi.ts

clearLicensingDebug

● clearLicensingDebug: *clearLicensingDebug*

Defined in src/Core/storage/localStorageApi.ts

getIsLicenseDebug

● getIsLicenseDebug: *getIsLicenseDebug*

Defined in src/Core/storage/localStorageApi.ts

getLicenseWizardMaxPort

● getLicenseWizardMaxPort: *getLicenseWizardMaxPort*

Defined in src/Core/storage/localStorageApi.ts

getLicenseWizardPort

● getLicenseWizardPort: *getLicenseWizardPort*

Defined in src/Core/storage/localStorageApi.ts

setIsLicenseDebug

● setIsLicenseDebug: *setIsLicenseDebug*

Defined in src/Core/storage/localStorageApi.ts

storageAvailable

● storageAvailable: *storageAvailable*

Defined in src/Core/storage/localStorageApi.ts

Const testLayoutManager

testLayoutManager: *object*

Defined in src/Charting/LayoutManager/AxisLayoutHelpers.ts

layoutAxisPartsBottomStrategy

layoutAxisPartsBottomStrategy: *layoutAxisPartsBottomStrategy*

Defined in src/Charting/LayoutManager/AxisLayoutHelpers.ts

layoutAxisPartsLeftStrategy

layoutAxisPartsLeftStrategy: *layoutAxisPartsLeftStrategy*

Defined in src/Charting/LayoutManager/AxisLayoutHelpers.ts

layoutAxisPartsRightStrategy

layoutAxisPartsRightStrategy: *layoutAxisPartsRightStrategy*

Defined in src/Charting/LayoutManager/AxisLayoutHelpers.ts

layoutAxisPartsTopStrategy

layoutAxisPartsTopStrategy: *layoutAxisPartsTopStrategy*

Defined in src/Charting/LayoutManager/AxisLayoutHelpers.ts

updateAxisLayoutState

updateAxisLayoutState: *updateAxisLayoutState*

Defined in src/Charting/LayoutManager/AxisLayoutHelpers.ts

Legend

- Constructor
- Property
- Method
- Accessor
- Inherited constructor
- Inherited property
- Inherited method
- Inherited accessor
- Property
- Method
- Protected property
- Protected method
- Static property
- Static method