Start Here - RenderableSeries Overview

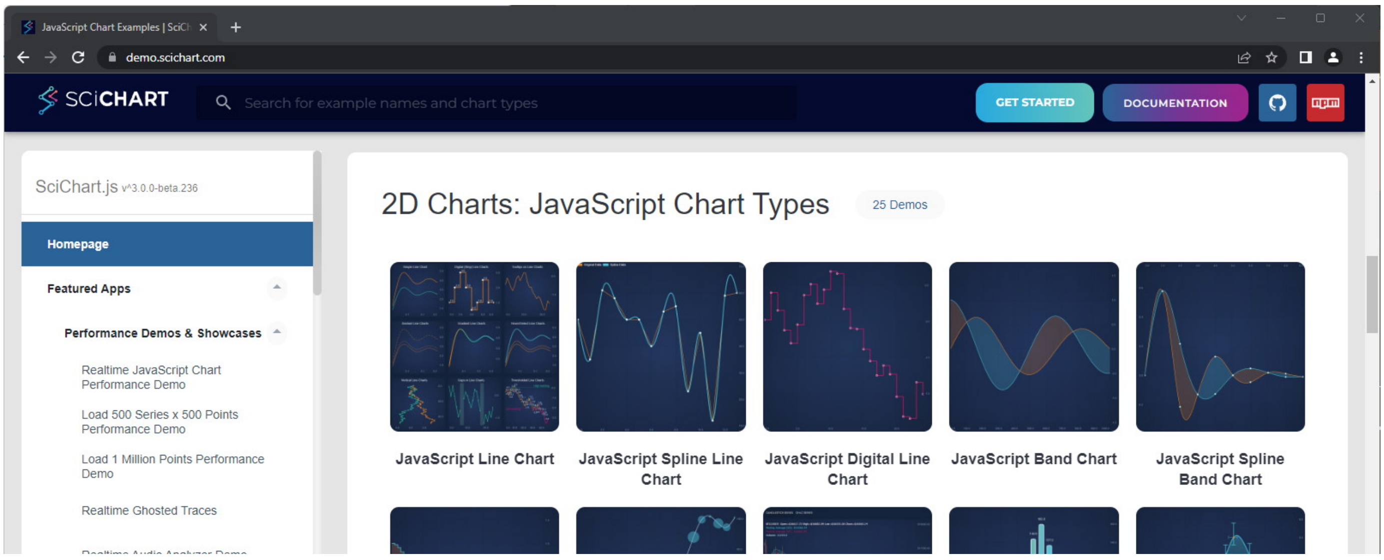
The RenderableSeries in SciChart are visual representations of **X,Y Numeric** or **Date** data. Other libraries call them 'Chart Types'.

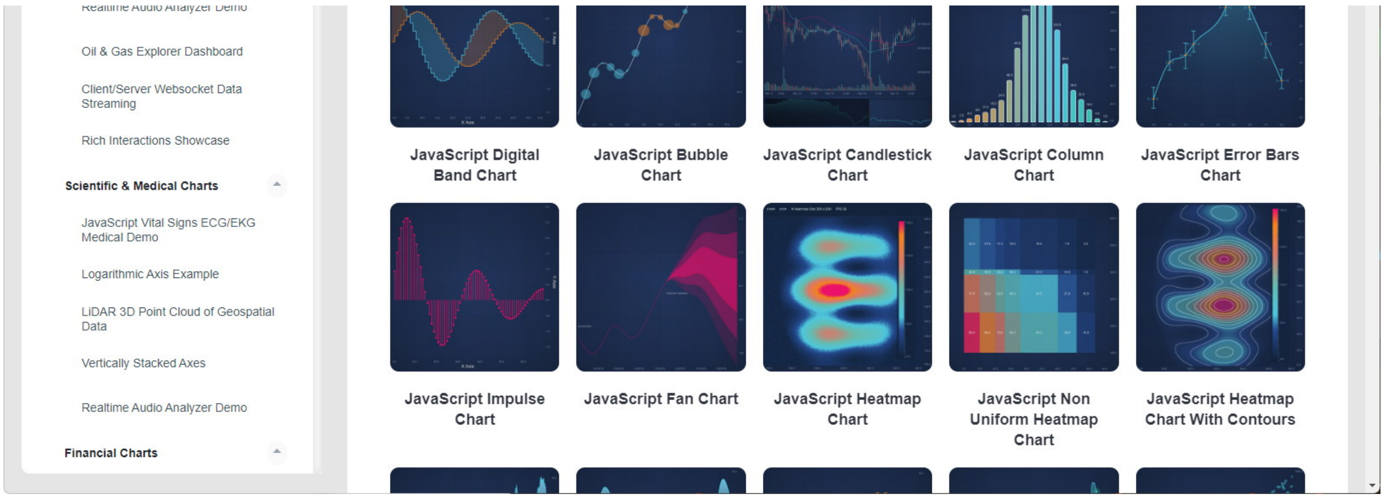
Some RenderableSeries render simple X,Y values (2D points in space), while some render additional information (such as X,Y0,Y1 values, or X,Y,Z values).

In This Topic

1. Chart Types in SciChart.js

SciChart.js supports a number of 2D & 3D Chart types. These include:


JavaScript Chart Examples | SciCh
demo.scichart.com
SCICHART
Search for example names and chart types
GET STARTED
DOCUMENTATION
SciChart.js v^3.0.0-beta.236
Homepage
Featured Apps
Performance Demos & Showcases
Realtime JavaScript Chart Performance Demo
Load 500 Series x 500 Points Performance Demo
Load 1 Million Points Performance Demo
Realtime Ghosted Traces
2D Charts: JavaScript Chart Types
25 Demos
JavaScript Line Chart
JavaScript Spline Line Chart
JavaScript Digital Line Chart
JavaScript Band Chart
JavaScript Spline Band Chart


Realtime Audio Analyzer Demo

Oil & Gas Explorer Dashboard

Client/Server Websocket Data Streaming

Rich Interactions Showcase

Scientific & Medical Charts

JavaScript Vital Signs ECG/EKG Medical Demo

Logarithmic Axis Example

LiDAR 3D Point Cloud of Geospatial Data

Vertically Stacked Axes

Realtime Audio Analyzer Demo

Financial Charts

JavaScript Digital Band Chart | JavaScript Bubble Chart | JavaScript Candlestick Chart | JavaScript Column Chart | JavaScript Error Bars Chart

JavaScript Impulse Chart | JavaScript Fan Chart | JavaScript Heatmap Chart | JavaScript Non Uniform Heatmap Chart | JavaScript Heatmap Chart With Contours

Line Series 🔖	Uniform Heatmap Series 🔖
Digital (Step) Line Series 🔖	Non-Uniform Heatmap Series 🔖
Spline Line Series 🔖	Heatmap Contours Series 🔖
Mountain (Area) Series 🔖	Candlestick Series 🔖
Digital (Step) Mountain Series 🔖	OHLC Series 🔖
Spline Mountain Series 🔖	Lollipop (Impulse or Stem) Series 🔖
Column Series 🔖	Error Bars Series 🔖
Stacked Column Series 🔖	Fan Charts 🔖
Grouped Column Series 🔖	Pie Charts 🔖
100% Stacked Column Series 🔖	Donut Charts 🔖
Stacked Mountain Series 🔖	Band (High-Low Fill) Series 🔖
100% Stacked Mountain Series 🔖	Digital Band Series 🔖
Scatter Series 🔖	Bubble Series 🔖
Text Series 🔖	

💡 Click on the links above to see documentation and live embedded code snippets for each of the chart types in SciChart.js

2. Common 2D Chart Features

As well as just render 2D Data, a number of chart types support additional properties and features.

For example, most 2D Chart types in SciChart.js support:

SciChart.js Feature	Description
Dynamic Data Updates	All SciChart.js chart types support dynamic updates to data out of the box. Insert, append, update, delete - modify data and the chart updates FAST. See the DataSeries API documentation for more details.
Per-point coloring	SciChart.js chart types support per-point coloring based on a rule. Xy values over/under a threshold, values with a property can be colored individually. See the PaletteProvider API for more details.
Data-point markers	SciChart.js chart types support adding a pointmarker (circle, square, triangle, cross marker) at data-points. See the PointMarkers documentation for more details.
Data-point Text Labels	SciChart.js supports fast, WebGL based text labels at datapoints, which can be customised. See the DataLabels API for more details.
Render Data Transforms	
Showing / hiding series	You can show/hide series in SciChart.js as well as be notified when a series is hidden. See the isVisibleChanged documentation for more details.
Select series	Using a ChartModifier (an attached behaviour) you can add Series Selection, Hover callbacks and styling into SciChart.js. See the SeriesSelectionModifier documentation for more details.
Select data-points	Using a ChartModifier (attached behaviour), you can add data-point selection detection, callbacks and styling into SciChart.js See the DataPointSelectionModifier docs for more details.
Tooltips and Cursors	Of course, SciChart.js supports customisable tooltips and cursors using ChartModifiers (attached behaviour). See the CursorModifier or RolloverModifier docs for more details.
Nulls (Gaps) in Data	For line charts, mountain charts and 2D cartesian chart types you can draw nulls (gaps) in data, or even change style in the line using a technique we have. See the Drawing Gaps in Series documentation for more details.

We are continually improving and adding features to SciChart.js. Some of the things on the roadmap are: more chart types, more interactions & even allowing custom drawing.

3. The RenderableSeries Class

All 2D Chart types in SciChart.js are derived from the *BaseRenderableSeries* type. This is a JavaScript class which is added to the *sciChartSurface.renderableSeries* collection and is rendered using our own proprietary WebAssembly / WebGL based rendering engine.

Each RenderableSeries is rendered to the screen, displaying the data from an associated DataSeries .

Renderable Series Properties

The properties common to the *BaseRenderableSeries class* are listed below.

BaseRenderableSeries property	Description
dataLabelProvider	(New to v3.0) The dataLabelprovider allows creation of per **data-point text labels**. Please see the Data Labels API section for a complete walk-through of text labels on chart series.
dataSeries	The DataSeries is the data-source for the RenderableSeries. Please see DataSeries API section for a complete walk-through of the DataSeries API
drawNanAs	How to treat NAN (Not a number) values in the input dataSeries. See *ELineDrawMode* for a list of values.
renderDataTransform	(New to v3.4) A RenderDataTransform may be optionally applied to a RenderableSeries to transform data immediately before drawing. Can be used to change the visual output e.g. create interpolated series, insert points into series and change styles in a series without changing the data.
pointMarker	A *PointMarker* is used to draw an **optional point-marker at each data-point**. Applicable to some series types only.
stroke	A Stroke for lines, outlines and edges of this RenderableSeries. Acceptable values include RGB format e.g. "#FF0000", RGBA format e.g. "#FF000077" and RGBA format e.g. "rgba(255,0,0,0.5)"
strokeThickness	The Stroke Thickness for lines, outlines and edges of this RenderableSeries
strokeDashArray	Some chart types which support lines (e.g. Line series,

	Spline Line series, Mountain Series) support dashed lines. This property accepts an array which defines [dot, dash] length in pixels
opacity	An Opacity factor of the Series that controls its semi-transparency level, where value 1 means the Series is opaque; 0 means transparent.
xAxisId	The XAxisId of the series allows you to attach a series to a specific axis. If you only have single X and Y Axis you can leave this default.
yAxisId	The YAxisId of the series allows you to attach a series to a specific axis. If you only have single X and Y Axis you can leave this default.
isvisible	When true, the series is visible. To hide a series, set IsVisible = false.
isVisibleChanged	An *EventHandler* allowing you to subscribe to isVisible changed callbacks
effect	An optional ShaderEffect for modifying the render output of a RenderableSeries.
paletteProvider	The PaletteProvider API **allows changing the color of a series on a per-point basis**. For more details about the PaletteProvider API see the individual examples for Line Series, Column Series etc...
hitTestProvider	The *HitTestProvider* exposes the Hit-Test API, used to **determine whether a series has been clicked on, hovered** or find the nearest Xy datapoint to a mouse coordinate.

Constructor options on RenderableSeries

Every RenderableSeries has a set of constructor options allowing for fast initialization and setup by passing in a javascript object. Each property reflected on the series has an optional constructor option parameter.

For example this code:

Setting properties of series Copy Code

```
const lineSeries = new FastLineRenderableSeries(wasmContext);
```

```
lineSeries.stroke = "Red";
lineSeries.strokeThickness = 3;
lineSeries.dataSeries = new XyDataSeries(wasmContext, {xValues, yValues});
lineSeries.isVisible = true;
lineSeries.opacity = 0.7;
```

is equivalent to this:

Constructor parameters of series Copy Code

```
const lineSeries = new FastLineRenderableSeries(wasmContext, {
    stroke: "Red",
    strokeThickness: 3,
    dataSeries: new XyDataSeries(wasmContext, {xValues, yValues}),
    isVisible: true,
    opacity: 0.7
});
```

4. Where Next?

To learn about a specific chart type, and to find out how to configure them, visit the following pages:

- The Line Series Type
- The Scatter Series Type
- The Mountain Series Type

etc...

See Also

2D Chart Types

The Line Series Type
The Scatter Series Type
The Column Series Type
The Mountain (Area) Series Type
The Candlestick Series type
The OHLC Series Type
The Uniform Heatmap Chart Type