* [Chart.js](http://docs.google.com/)
* [Getting Started](http://docs.google.com/getting-started/)
  + [Installation](http://docs.google.com/getting-started/installation.html)
  + [Integration](http://docs.google.com/getting-started/integration.html)
  + [Usage](http://docs.google.com/getting-started/usage.html)
* [General](http://docs.google.com/general/)
  + [Responsive](http://docs.google.com/general/responsive.html)
  + [Pixel Ratio](http://docs.google.com/general/device-pixel-ratio.html)
  + [Interactions](http://docs.google.com/general/interactions/)
    - [Events](http://docs.google.com/general/interactions/events.html)
    - [Modes](http://docs.google.com/general/interactions/modes.html)
  + [Options](http://docs.google.com/general/options.html)
  + [Colors](http://docs.google.com/general/colors.html)
  + [Fonts](http://docs.google.com/general/fonts.html)
* [Configuration](http://docs.google.com/configuration/)
  + [Animations](http://docs.google.com/configuration/animations.html)
  + [Layout](http://docs.google.com/configuration/layout.html)
  + [Legend](http://docs.google.com/configuration/legend.html)
  + [Title](http://docs.google.com/configuration/title.html)
  + [Tooltip](http://docs.google.com/configuration/tooltip.html)
  + [Elements](http://docs.google.com/configuration/elements.html)
* [Charts](http://docs.google.com/)
  + [Line](http://docs.google.com/line.html)
  + [Bar](http://docs.google.com/bar.html)
  + [Radar](http://docs.google.com/radar.html)
  + [Doughnut & Pie](http://docs.google.com/doughnut.html)
  + [Polar Area](http://docs.google.com/polar.html)
  + [Bubble](http://docs.google.com/bubble.html)
  + [Scatter](http://docs.google.com/scatter.html)
  + [Area](http://docs.google.com/area.html)
  + [Mixed](http://docs.google.com/mixed.html)
* [Axes](http://docs.google.com/axes/)
  + [Cartesian](http://docs.google.com/axes/cartesian/)
    - [Category](http://docs.google.com/axes/cartesian/category.html)
    - [Linear](http://docs.google.com/axes/cartesian/linear.html)
    - [Logarithmic](http://docs.google.com/axes/cartesian/logarithmic.html)
    - [Time](http://docs.google.com/axes/cartesian/time.html)
  + [Radial](http://docs.google.com/axes/radial/)
    - [Linear](http://docs.google.com/axes/radial/linear.html)
  + [Labelling](http://docs.google.com/axes/labelling.html)
  + [Styling](http://docs.google.com/axes/styling.html)
* [Developers](http://docs.google.com/developers/)
  + [Chart.js API](http://docs.google.com/developers/api.html)
  + [Updating Charts](http://docs.google.com/developers/updates.html)
  + [Plugins](http://docs.google.com/developers/plugins.html)
  + [New Charts](http://docs.google.com/developers/charts.html)
  + [New Axes](http://docs.google.com/developers/axes.html)
  + [Contributing](http://docs.google.com/developers/contributing.html)
* [Additional Notes](http://docs.google.com/notes/)
  + [Comparison Table](http://docs.google.com/notes/comparison.html)
  + [Popular Extensions](http://docs.google.com/notes/extensions.html)
  + [License](http://docs.google.com/notes/license.html)
* [Published with GitBook](https://www.gitbook.com)

[**Line**](http://docs.google.com/)

Line

A line chart is a way of plotting data points on a line. Often, it is used to show trend data, or the comparison of two data sets.

## Example Usage

var myLineChart = new Chart(ctx, {  
 type: 'line',  
 data: data,  
 options: options  
});

## Dataset Properties

The line chart allows a number of properties to be specified for each dataset. These are used to set display properties for a specific dataset. For example, the colour of a line is generally set this way.

All point\* properties can be specified as an array. If these are set to an array value, the first value applies to the first point, the second value to the second point, and so on.

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| label | String | The label for the dataset which appears in the legend and tooltips. |
| xAxisID | String | The ID of the x axis to plot this dataset on. If not specified, this defaults to the ID of the first found x axis |
| yAxisID | String | The ID of the y axis to plot this dataset on. If not specified, this defaults to the ID of the first found y axis. |
| backgroundColor | Color | The fill color under the line. See [Colors](http://docs.google.com/general/colors.html#colors) |
| borderColor | Color | The color of the line. See [Colors](http://docs.google.com/general/colors.html#colors) |
| borderWidth | Number | The width of the line in pixels. |
| borderDash | Number[] | Length and spacing of dashes. See [MDN](https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D/setLineDash) |
| borderDashOffset | Number | Offset for line dashes. See [MDN](https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D/lineDashOffset) |
| borderCapStyle | String | Cap style of the line. See [MDN](https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D/lineCap) |
| borderJoinStyle | String | Line joint style. See [MDN](https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D/lineJoin) |
| cubicInterpolationMode | String | Algorithm used to interpolate a smooth curve from the discrete data points. [more...](#gjdgxs) |
| fill | Boolean/String | How to fill the area under the line. See [area charts](http://docs.google.com/area.html) |
| lineTension | Number | Bezier curve tension of the line. Set to 0 to draw straightlines. This option is ignored if monotone cubic interpolation is used. |
| pointBackgroundColor | Color/Color[] | The fill color for points. |
| pointBorderColor | Color/Color[] | The border color for points. |
| pointBorderWidth | Number/Number[] | The width of the point border in pixels. |
| pointRadius | Number/Number[] | The radius of the point shape. If set to 0, the point is not rendered. |
| pointStyle | String/String[]/Image/Image[] | Style of the point. [more...](http://docs.google.com/configuration/elements#point-styles) |
| pointHitRadius | Number/Number[] | The pixel size of the non-displayed point that reacts to mouse events. |
| pointHoverBackgroundColor | Color/Color[] | Point background color when hovered. |
| pointHoverBorderColor | Color/Color[] | Point border color when hovered. |
| pointHoverBorderWidth | Number/Number[] | Border width of point when hovered. |
| pointHoverRadius | Number/Number[] | The radius of the point when hovered. |
| showLine | Boolean | If false, the line is not drawn for this dataset. |
| spanGaps | Boolean | If true, lines will be drawn between points with no or null data. If false, points with NaN data will create a break in the line |
| steppedLine | Boolean/String | If the line is shown as a stepped line. [more...](#30j0zll) |

### cubicInterpolationMode

The following interpolation modes are supported:

* 'default'
* 'monotone'.

The 'default' algorithm uses a custom weighted cubic interpolation, which produces pleasant curves for all types of datasets.

The 'monotone' algorithm is more suited to y = f(x) datasets : it preserves monotonicity (or piecewise monotonicity) of the dataset being interpolated, and ensures local extremums (if any) stay at input data points.

If left untouched (undefined), the global options.elements.line.cubicInterpolationMode property is used.

### Stepped Line

The following values are supported for steppedLine:

* false: No Step Interpolation (default)
* true: Step-before Interpolation (eq. 'before')
* 'before': Step-before Interpolation
* 'after': Step-after Interpolation

If the steppedLine value is set to anything other than false, lineTension will be ignored.

## Configuration Options

The line chart defines the following configuration options. These options are merged with the global chart configuration options, Chart.defaults.global, to form the options passed to the chart.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Default | Description |
| showLines | Boolean | true | If false, the lines between points are not drawn. |
| spanGaps | Boolean | false | If false, NaN data causes a break in the line. |

## Default Options

It is common to want to apply a configuration setting to all created line charts. The global line chart settings are stored in Chart.defaults.line. Changing the global options only affects charts created after the change. Existing charts are not changed.

For example, to configure all line charts with spanGaps = true you would do:

Chart.defaults.line.spanGaps = true;

## Data Structure

The data property of a dataset for a line chart can be passed in two formats.

### Number[]

data: [20, 10]

When the data array is an array of numbers, the x axis is generally a [category](http://docs.google.com/axes/cartesian/category.html#category-cartesian-axis). The points are placed onto the axis using their position in the array. When a line chart is created with a category axis, the labels property of the data object must be specified.

### Point[]

data: [{  
 x: 10,  
 y: 20  
 }, {  
 x: 15,  
 y: 10  
 }]

This alternate is used for sparse datasets, such as those in [scatter charts](http://docs.google.com/scatter.html#scatter-chart). Each data point is specified using an object containing x and y properties.

# Stacked Area Chart

Line charts can be configured into stacked area charts by changing the settings on the y axis to enable stacking. Stacked area charts can be used to show how one data trend is made up of a number of smaller pieces.

var stackedLine = new Chart(ctx, {  
 type: 'line',  
 data: data,  
 options: {  
 scales: {  
 yAxes: [{  
 stacked: true  
 }]  
 }  
 }  
});

# High Performance Line Charts

When charting a lot of data, the chart render time may start to get quite large. In that case, the following strategies can be used to improve performance.

## Data Decimation

Decimating your data will achieve the best results. When there is a lot of data to display on the graph, it doesn't make sense to show tens of thousands of data points on a graph that is only a few hundred pixels wide.

There are many approaches to data decimation and selection of an algorithm will depend on your data and the results you want to achieve. For instance, [min/max](http://digital.ni.com/public.nsf/allkb/F694FFEEA0ACF282862576020075F784) decimation will preserve peaks in your data but could require up to 4 points for each pixel. This type of decimation would work well for a very noisy signal where you need to see data peaks.

## Disable Bezier Curves

If you are drawing lines on your chart, disabling bezier curves will improve render times since drawing a straight line is more performant than a bezier curve.

To disable bezier curves for an entire chart:

new Chart(ctx, {  
 type: 'line',  
 data: data,  
 options: {  
 elements: {  
 line: {  
 tension: 0, // disables bezier curves  
 }  
 }  
 }  
});

## Disable Line Drawing

If you have a lot of data points, it can be more performant to disable rendering of the line for a dataset and only draw points. Doing this means that there is less to draw on the canvas which will improve render performance.

To disable lines:

new Chart(ctx, {  
 type: 'line',  
 data: {  
 datasets: [{  
 showLine: false, // disable for a single dataset  
 }]  
 },  
 options: {  
 showLines: false, // disable for all datasets  
 }  
});

## Disable Animations

If your charts have long render times, it is a good idea to disable animations. Doing so will mean that the chart needs to only be rendered once during an update instead of multiple times. This will have the effect of reducing CPU usage and improving general page performance.

To disable animations

new Chart(ctx, {  
 type: 'line',  
 data: data,  
 options: {  
 animation: {  
 duration: 0, // general animation time  
 },  
 hover: {  
 animationDuration: 0, // duration of animations when hovering an item  
 },  
 responsiveAnimationDuration: 0, // animation duration after a resize  
 }  
});

results matching ""

No results matching ""