 [Chart.js](http://docs.google.com/docs/3.9.1/)

[Home](http://docs.google.com/docs/3.9.1/)

[API](http://docs.google.com/docs/3.9.1/api/)

[Samples](http://docs.google.com/docs/3.9.1/samples/)

Ecosystem Ecosystem

* [Awesome (opens new window)](https://github.com/chartjs/awesome)
* [Slack (opens new window)](https://chartjs-slack.herokuapp.com/)
* [Stack Overflow (opens new window)](https://stackoverflow.com/questions/tagged/chart.js)

[GitHub (opens new window)](https://github.com/chartjs/Chart.js)

[Home](http://docs.google.com/docs/3.9.1/)

[API](http://docs.google.com/docs/3.9.1/api/)

[Samples](http://docs.google.com/docs/3.9.1/samples/)

Ecosystem Ecosystem

* [Awesome (opens new window)](https://github.com/chartjs/awesome)
* [Slack (opens new window)](https://chartjs-slack.herokuapp.com/)
* [Stack Overflow (opens new window)](https://stackoverflow.com/questions/tagged/chart.js)

[GitHub (opens new window)](https://github.com/chartjs/Chart.js)

* [Information](http://docs.google.com/docs/3.9.1/samples/information.html)
* Bar Charts
* Line Charts
* Other charts
* Area charts
* Scales
* Scale Options
* Legend
* Title
* Subtitle
* Tooltip
* Scriptable Options
* Animations
* Advanced
  + [Data Decimation](http://docs.google.com/docs/3.9.1/samples/advanced/data-decimation.html)
  + [Derived Axis Type](http://docs.google.com/docs/3.9.1/samples/advanced/derived-axis-type.html)
  + [Derived Chart Type](http://docs.google.com/docs/3.9.1/samples/advanced/derived-chart-type.html)
  + [Linear Gradient](http://docs.google.com/docs/3.9.1/samples/advanced/linear-gradient.html)
  + [Programmatic Event Triggers](http://docs.google.com/docs/3.9.1/samples/advanced/programmatic-events.html)
  + [Animation Progress Bar](http://docs.google.com/docs/3.9.1/samples/advanced/progress-bar.html)
  + [Radial Gradient](http://docs.google.com/docs/3.9.1/samples/advanced/radial-gradient.html)
* Plugins
* [Utils](http://docs.google.com/docs/3.9.1/samples/utils.html)

[**#**](#gjdgxs) Linear Gradient

getGradient config setup actions

let width, height, gradient; function getGradient(ctx, chartArea) { const chartWidth = chartArea.right - chartArea.left; const chartHeight = chartArea.bottom - chartArea.top; if (!gradient || width !== chartWidth || height !== chartHeight) { // Create the gradient because this is either the first render // or the size of the chart has changed width = chartWidth; height = chartHeight; gradient = ctx.createLinearGradient(0, chartArea.bottom, 0, chartArea.top); gradient.addColorStop(0, Utils.CHART\_COLORS.blue); gradient.addColorStop(0.5, Utils.CHART\_COLORS.yellow); gradient.addColorStop(1, Utils.CHART\_COLORS.red); } return gradient; }

let width, height, gradient;  
function getGradient(ctx, chartArea) {  
 const chartWidth = chartArea.right - chartArea.left;  
 const chartHeight = chartArea.bottom - chartArea.top;  
 if (!gradient || width !== chartWidth || height !== chartHeight) {  
 // Create the gradient because this is either the first render  
 // or the size of the chart has changed  
 width = chartWidth;  
 height = chartHeight;  
 gradient = ctx.createLinearGradient(0, chartArea.bottom, 0, chartArea.top);  
 gradient.addColorStop(0, Utils.CHART\_COLORS.blue);  
 gradient.addColorStop(0.5, Utils.CHART\_COLORS.yellow);  
 gradient.addColorStop(1, Utils.CHART\_COLORS.red);  
 }  
 return gradient;  
}

const config = { type: 'line', data: data, options: { responsive: true, plugins: { legend: { position: 'top', }, } }, };

const config = {  
 type: 'line',  
 data: data,  
 options: {  
 responsive: true,  
 plugins: {  
 legend: {  
 position: 'top',  
 },  
 }  
 },  
};

const DATA\_COUNT = 7; const NUMBER\_CFG = {count: DATA\_COUNT, min: -100, max: 100}; const labels = Utils.months({count: 7}); const data = { labels: labels, datasets: [ { label: 'Dataset 1', data: Utils.numbers(NUMBER\_CFG), borderColor: function(context) { const chart = context.chart; const {ctx, chartArea} = chart; if (!chartArea) { // This case happens on initial chart load return; } return getGradient(ctx, chartArea); }, }, ] };

const DATA\_COUNT = 7;  
const NUMBER\_CFG = {count: DATA\_COUNT, min: -100, max: 100};  
const labels = Utils.months({count: 7});  
const data = {  
 labels: labels,  
 datasets: [  
 {  
 label: 'Dataset 1',  
 data: Utils.numbers(NUMBER\_CFG),  
 borderColor: function(context) {  
 const chart = context.chart;  
 const {ctx, chartArea} = chart;  
 if (!chartArea) {  
 // This case happens on initial chart load  
 return;  
 }  
 return getGradient(ctx, chartArea);  
 },  
 },  
 ]  
};

const actions = [ { name: 'Randomize', handler(chart) { chart.data.datasets.forEach(dataset => { dataset.data = Utils.numbers({count: chart.data.labels.length, min: -100, max: 100}); }); chart.update(); } }, { name: 'Add Data', handler(chart) { const data = chart.data; if (data.datasets.length > 0) { data.labels = Utils.months({count: data.labels.length + 1}); for (let index = 0; index < data.datasets.length; ++index) { data.datasets[index].data.push(Utils.rand(-100, 100)); } chart.update(); } } }, { name: 'Remove Data', handler(chart) { chart.data.labels.splice(-1, 1); // remove the label first chart.data.datasets.forEach(dataset => { dataset.data.pop(); }); chart.update(); } } ];

const actions = [  
 {  
 name: 'Randomize',  
 handler(chart) {  
 chart.data.datasets.forEach(dataset => {  
 dataset.data = Utils.numbers({count: chart.data.labels.length, min: -100, max: 100});  
 });  
 chart.update();  
 }  
 },  
 {  
 name: 'Add Data',  
 handler(chart) {  
 const data = chart.data;  
 if (data.datasets.length > 0) {  
 data.labels = Utils.months({count: data.labels.length + 1});  
 for (let index = 0; index < data.datasets.length; ++index) {  
 data.datasets[index].data.push(Utils.rand(-100, 100));  
 }  
 chart.update();  
 }  
 }  
 },  
 {  
 name: 'Remove Data',  
 handler(chart) {  
 chart.data.labels.splice(-1, 1); // remove the label first  
 chart.data.datasets.forEach(dataset => {  
 dataset.data.pop();  
 });  
 chart.update();  
 }  
 }  
];

## [**#**](#30j0zll) Docs

* [Colors](http://docs.google.com/docs/3.9.1/general/colors.html)
  + [Patterns and Gradients](http://docs.google.com/docs/3.9.1/general/colors.html#patterns-and-gradients)
* [Data structures (labels)](http://docs.google.com/docs/3.9.1/general/data-structures.html)
* [Options](http://docs.google.com/docs/3.9.1/general/options.html)
  + [Scriptable Options](http://docs.google.com/docs/3.9.1/general/options.html#scriptable-options)
* [Line](http://docs.google.com/docs/3.9.1/charts/line.html)

Last Updated: 8/3/2022, 12:46:38 PM

←  [Derived Chart Type](http://docs.google.com/docs/3.9.1/samples/advanced/derived-chart-type.html)   [Programmatic Event Triggers](http://docs.google.com/docs/3.9.1/samples/advanced/programmatic-events.html)  →