 [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) Radial Gradient

createRadialGradient3 config data setup

function createRadialGradient3(context, c1, c2, c3) { const chartArea = context.chart.chartArea; if (!chartArea) { // This case happens on initial chart load return; } const chartWidth = chartArea.right - chartArea.left; const chartHeight = chartArea.bottom - chartArea.top; if (width !== chartWidth || height !== chartHeight) { cache.clear(); } let gradient = cache.get(c1 + c2 + c3); if (!gradient) { // Create the gradient because this is either the first render // or the size of the chart has changed width = chartWidth; height = chartHeight; const centerX = (chartArea.left + chartArea.right) / 2; const centerY = (chartArea.top + chartArea.bottom) / 2; const r = Math.min( (chartArea.right - chartArea.left) / 2, (chartArea.bottom - chartArea.top) / 2 ); const ctx = context.chart.ctx; gradient = ctx.createRadialGradient(centerX, centerY, 0, centerX, centerY, r); gradient.addColorStop(0, c1); gradient.addColorStop(0.5, c2); gradient.addColorStop(1, c3); cache.set(c1 + c2 + c3, gradient); } return gradient; }

function createRadialGradient3(context, c1, c2, c3) {  
 const chartArea = context.chart.chartArea;  
 if (!chartArea) {  
 // This case happens on initial chart load  
 return;  
 }  
 const chartWidth = chartArea.right - chartArea.left;  
 const chartHeight = chartArea.bottom - chartArea.top;  
 if (width !== chartWidth || height !== chartHeight) {  
 cache.clear();  
 }  
 let gradient = cache.get(c1 + c2 + c3);  
 if (!gradient) {  
 // Create the gradient because this is either the first render  
 // or the size of the chart has changed  
 width = chartWidth;  
 height = chartHeight;  
 const centerX = (chartArea.left + chartArea.right) / 2;  
 const centerY = (chartArea.top + chartArea.bottom) / 2;  
 const r = Math.min(  
 (chartArea.right - chartArea.left) / 2,  
 (chartArea.bottom - chartArea.top) / 2  
 );  
 const ctx = context.chart.ctx;  
 gradient = ctx.createRadialGradient(centerX, centerY, 0, centerX, centerY, r);  
 gradient.addColorStop(0, c1);  
 gradient.addColorStop(0.5, c2);  
 gradient.addColorStop(1, c3);  
 cache.set(c1 + c2 + c3, gradient);  
 }  
 return gradient;  
}

const config = { type: 'polarArea', data: data, options: { plugins: { legend: false, tooltip: false, }, elements: { arc: { backgroundColor: function(context) { let c = colors[context.dataIndex]; if (!c) { return; } if (context.active) { c = helpers.getHoverColor(c); } const mid = helpers.color(c).desaturate(0.2).darken(0.2).rgbString(); const start = helpers.color(c).lighten(0.2).rotate(270).rgbString(); const end = helpers.color(c).lighten(0.1).rgbString(); return createRadialGradient3(context, start, mid, end); }, } } } };

const config = {  
 type: 'polarArea',  
 data: data,  
 options: {  
 plugins: {  
 legend: false,  
 tooltip: false,  
 },  
 elements: {  
 arc: {  
 backgroundColor: function(context) {  
 let c = colors[context.dataIndex];  
 if (!c) {  
 return;  
 }  
 if (context.active) {  
 c = helpers.getHoverColor(c);  
 }  
 const mid = helpers.color(c).desaturate(0.2).darken(0.2).rgbString();  
 const start = helpers.color(c).lighten(0.2).rotate(270).rgbString();  
 const end = helpers.color(c).lighten(0.1).rgbString();  
 return createRadialGradient3(context, start, mid, end);  
 },  
 }  
 }  
 }  
};

function generateData() { return Utils.numbers({ count: DATA\_COUNT, min: 0, max: 100 }); } const data = { labels: Utils.months({count: DATA\_COUNT}), datasets: [{ data: generateData() }] };

function generateData() {  
 return Utils.numbers({  
 count: DATA\_COUNT,  
 min: 0,  
 max: 100  
 });  
}  
const data = {  
 labels: Utils.months({count: DATA\_COUNT}),  
 datasets: [{  
 data: generateData()  
 }]  
};

const DATA\_COUNT = 5; Utils.srand(110); const chartColors = Utils.CHART\_COLORS; const colors = [chartColors.red, chartColors.orange, chartColors.yellow, chartColors.green, chartColors.blue]; const cache = new Map(); let width = null; let height = null; const actions = [ { name: 'Randomize', handler(chart) { chart.data.datasets.forEach(dataset => { dataset.data = generateData(); }); chart.update(); } }, ];

const DATA\_COUNT = 5;  
Utils.srand(110);  
const chartColors = Utils.CHART\_COLORS;  
const colors = [chartColors.red, chartColors.orange, chartColors.yellow, chartColors.green, chartColors.blue];  
const cache = new Map();  
let width = null;  
let height = null;  
const actions = [  
 {  
 name: 'Randomize',  
 handler(chart) {  
 chart.data.datasets.forEach(dataset => {  
 dataset.data = generateData();  
 });  
 chart.update();  
 }  
 },  
];

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

* [Polar Area Chart](http://docs.google.com/docs/3.9.1/charts/polar.html)
  + [Styling](http://docs.google.com/docs/3.9.1/charts/polar.html#styling)
* [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)

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

←  [Animation Progress Bar](http://docs.google.com/docs/3.9.1/samples/advanced/progress-bar.html)   [Chart Area Border](http://docs.google.com/docs/3.9.1/samples/plugins/chart-area-border.html)  →