* [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/)
  + [Responsive](http://docs.google.com/responsive.html)
  + [Pixel Ratio](http://docs.google.com/device-pixel-ratio.html)
  + [Interactions](http://docs.google.com/interactions/)
    - [Events](http://docs.google.com/interactions/events.html)
    - [Modes](http://docs.google.com/interactions/modes.html)
  + [Options](http://docs.google.com/options.html)
  + [Colors](http://docs.google.com/colors.html)
  + [Fonts](http://docs.google.com/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/charts/)
  + [Line](http://docs.google.com/charts/line.html)
  + [Bar](http://docs.google.com/charts/bar.html)
  + [Radar](http://docs.google.com/charts/radar.html)
  + [Doughnut & Pie](http://docs.google.com/charts/doughnut.html)
  + [Polar Area](http://docs.google.com/charts/polar.html)
  + [Bubble](http://docs.google.com/charts/bubble.html)
  + [Scatter](http://docs.google.com/charts/scatter.html)
  + [Area](http://docs.google.com/charts/area.html)
  + [Mixed](http://docs.google.com/charts/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)

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

Colors

When supplying colors to Chart options, you can use a number of formats. You can specify the color as a string in hexadecimal, RGB, or HSL notations. If a color is needed, but not specified, Chart.js will use the global default color. This color is stored at Chart.defaults.global.defaultColor. It is initially set to 'rgba(0, 0, 0, 0.1)'

You can also pass a [CanvasGradient](https://developer.mozilla.org/en-US/docs/Web/API/CanvasGradient) object. You will need to create this before passing to the chart, but using it you can achieve some interesting effects.

## Patterns and Gradients

An alternative option is to pass a [CanvasPattern](https://developer.mozilla.org/en-US/docs/Web/API/CanvasPattern) or [CanvasGradient](https://developer.mozilla.org/en/docs/Web/API/CanvasGradient) object instead of a string colour.

For example, if you wanted to fill a dataset with a pattern from an image you could do the following.

var img = new Image();  
img.src = 'https://example.com/my\_image.png';  
img.onload = function() {  
 var ctx = document.getElementById('canvas').getContext('2d');  
 var fillPattern = ctx.createPattern(img, 'repeat');  
  
 var chart = new Chart(ctx, {  
 data: {  
 labels: ['Item 1', 'Item 2', 'Item 3'],  
 datasets: [{  
 data: [10, 20, 30],  
 backgroundColor: fillPattern  
 }]  
 }  
 })  
}

Using pattern fills for data graphics can help viewers with vision deficiencies (e.g. color-blindness or partial sight) to [more easily understand your data](http://betweentwobrackets.com/data-graphics-and-colour-vision/).

Using the [Patternomaly](https://github.com/ashiguruma/patternomaly) library you can generate patterns to fill datasets.

var chartData = {  
 datasets: [{  
 data: [45, 25, 20, 10],  
 backgroundColor: [  
 pattern.draw('square', '#ff6384'),  
 pattern.draw('circle', '#36a2eb'),  
 pattern.draw('diamond', '#cc65fe'),  
 pattern.draw('triangle', '#ffce56'),  
 ]  
 }],  
 labels: ['Red', 'Blue', 'Purple', 'Yellow']  
};

results matching ""

No results matching ""