 [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)

* [Chart.js](http://docs.google.com/docs/3.9.1/)
* Getting Started
* General
* Configuration
* Chart Types
* Axes
* Developers
  + [Developers](http://docs.google.com/docs/3.9.1/developers/)
  + [API](http://docs.google.com/docs/3.9.1/developers/api.html)
  + [New Axes](http://docs.google.com/docs/3.9.1/developers/axes.html)
  + [New Charts](http://docs.google.com/docs/3.9.1/developers/charts.html)
  + [Contributing](http://docs.google.com/docs/3.9.1/developers/contributing.html)
  + [Plugins](http://docs.google.com/docs/3.9.1/developers/plugins.html)
  + [Publishing an extension](http://docs.google.com/docs/3.9.1/developers/publishing.html)
  + [TypeDoc](http://docs.google.com/docs/3.9.1/api/)
  + [Updating Charts](http://docs.google.com/docs/3.9.1/developers/updates.html)

[**#**](#gjdgxs) Plugins

Plugins are the most efficient way to customize or change the default behavior of a chart. They have been introduced at [version 2.1.0 (opens new window)](https://github.com/chartjs/Chart.js/releases/tag/2.1.0) (global plugins only) and extended at [version 2.5.0 (opens new window)](https://github.com/chartjs/Chart.js/releases/tag/v2.5.0) (per chart plugins and options).

## [**#**](#30j0zll) Using plugins

Plugins can be shared between chart instances:

const plugin = { /\* plugin implementation \*/ };  
// chart1 and chart2 use "plugin"  
const chart1 = new Chart(ctx, {  
 plugins: [plugin]  
});  
const chart2 = new Chart(ctx, {  
 plugins: [plugin]  
});  
// chart3 doesn't use "plugin"  
const chart3 = new Chart(ctx, {});

Plugins can also be defined directly in the chart plugins config (a.k.a. *inline plugins*):

WARNING

*inline* plugins are not registered. Some plugins require registering, i.e. can't be used *inline*.

const chart = new Chart(ctx, {  
 plugins: [{  
 beforeInit: function(chart, args, options) {  
 //..  
 }  
 }]  
});

However, this approach is not ideal when the customization needs to apply to many charts.

## [**#**](#1fob9te) Global plugins

Plugins can be registered globally to be applied on all charts (a.k.a. *global plugins*):

Chart.register({  
 // plugin implementation  
});

WARNING

*inline* plugins can't be registered globally.

## [**#**](#3znysh7) Configuration

### [**#**](#2et92p0) Plugin ID

Plugins must define a unique id in order to be configurable.

This id should follow the [npm package name convention (opens new window)](https://docs.npmjs.com/files/package.json#name):

* can't start with a dot or an underscore
* can't contain any non-URL-safe characters
* can't contain uppercase letters
* should be something short, but also reasonably descriptive

If a plugin is intended to be released publicly, you may want to check the [registry (opens new window)](https://www.npmjs.com/search?q=chartjs-plugin-) to see if there's something by that name already. Note that in this case, the package name should be prefixed by chartjs-plugin- to appear in Chart.js plugin registry.

### [**#**](#tyjcwt) Plugin options

Plugin options are located under the options.plugins config and are scoped by the plugin ID: options.plugins.{plugin-id}.

const chart = new Chart(ctx, {  
 options: {  
 foo: { ... }, // chart 'foo' option  
 plugins: {  
 p1: {  
 foo: { ... }, // p1 plugin 'foo' option  
 bar: { ... }  
 },  
 p2: {  
 foo: { ... }, // p2 plugin 'foo' option  
 bla: { ... }  
 }  
 }  
 }  
});

#### [**#**](#3dy6vkm) Disable plugins

To disable a global plugin for a specific chart instance, the plugin options must be set to false:

Chart.register({  
 id: 'p1',  
 // ...  
});  
const chart = new Chart(ctx, {  
 options: {  
 plugins: {  
 p1: false // disable plugin 'p1' for this instance  
 }  
 }  
});

To disable all plugins for a specific chart instance, set options.plugins to false:

const chart = new Chart(ctx, {  
 options: {  
 plugins: false // all plugins are disabled for this instance  
 }  
});

#### [**#**](#1t3h5sf) Plugin defaults

You can set default values for your plugin options in the defaults entry of your plugin object. In the example below the canvas will always have a lightgreen backgroundColor unless the user overrides this option in options.plugins.custom\_canvas\_background\_color.color.

const plugin = {  
 id: 'custom\_canvas\_background\_color',  
 beforeDraw: (chart, args, options) => {  
 const {ctx} = chart;  
 ctx.save();  
 ctx.globalCompositeOperation = 'destination-over';  
 ctx.fillStyle = options.color;  
 ctx.fillRect(0, 0, chart.width, chart.height);  
 ctx.restore();  
 },  
 defaults: {  
 color: 'lightGreen'  
 }  
}

## [**#**](#4d34og8) Plugin Core API

Read more about the [existing plugin extension hooks](http://docs.google.com/api/interfaces/Plugin).

### [**#**](#2s8eyo1) Chart Initialization

Plugins are notified during the initialization process. These hooks can be used to setup data needed for the plugin to operate.



### [**#**](#17dp8vu) Chart Update

Plugins are notified throughout the update process.



### [**#**](#3rdcrjn) Scale Update

Plugins are notified throughout the scale update process.



### [**#**](#26in1rg) Rendering

Plugins can interact with the chart throughout the render process. The rendering process is documented in the flowchart below. Each of the green processes is a plugin notification. The red lines indicate how cancelling part of the render process can occur when a plugin returns false from a hook. Not all hooks are cancelable, however, in general most before\* hooks can be cancelled.



### [**#**](#lnxbz9) Event Handling

Plugins can interact with the chart during the event handling process. The event handling flow is documented in the flowchart below. Each of the green processes is a plugin notification. If a plugin makes changes that require a re-render, the plugin can set args.changed to true to indicate that a render is needed. The built-in tooltip plugin uses this method to indicate when the tooltip has changed.



### [**#**](#35nkun2) Chart destroy

Plugins are notified during the destroy process. These hooks can be used to destroy things that the plugin made and used during its life. The destroy hook has been deprecated since Chart.js version 3.7.0, use the afterDestroy hook instead.



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

←  [Contributing](http://docs.google.com/docs/3.9.1/developers/contributing.html)   [Publishing an extension](http://docs.google.com/docs/3.9.1/developers/publishing.html)  →