 [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
  + [Configuration](http://docs.google.com/docs/3.9.1/configuration/)
  + [Animations](http://docs.google.com/docs/3.9.1/configuration/animations.html)
  + [Canvas background](http://docs.google.com/docs/3.9.1/configuration/canvas-background.html)
  + [Data Decimation](http://docs.google.com/docs/3.9.1/configuration/decimation.html)
  + [Device Pixel Ratio](http://docs.google.com/docs/3.9.1/configuration/device-pixel-ratio.html)
  + [Elements](http://docs.google.com/docs/3.9.1/configuration/elements.html)
  + [Interactions](http://docs.google.com/docs/3.9.1/configuration/interactions.html)
  + [Layout](http://docs.google.com/docs/3.9.1/configuration/layout.html)
  + [Legend](http://docs.google.com/docs/3.9.1/configuration/legend.html)
  + [Locale](http://docs.google.com/docs/3.9.1/configuration/locale.html)
  + [Responsive Charts](http://docs.google.com/docs/3.9.1/configuration/responsive.html)
  + [Subtitle](http://docs.google.com/docs/3.9.1/configuration/subtitle.html)
  + [Title](http://docs.google.com/docs/3.9.1/configuration/title.html)
  + [Tooltip](http://docs.google.com/docs/3.9.1/configuration/tooltip.html)
* Chart Types
* Axes
* Developers

[**#**](#gjdgxs) Interactions

Namespace: options.interaction, the global interaction configuration is at Chart.defaults.interaction. To configure which events trigger chart interactions, see [events](#30j0zll).

| Name | Type | Default | Description |
| --- | --- | --- | --- |
| mode | string | 'nearest' | Sets which elements appear in the interaction. See [Interaction Modes](#1fob9te) for details. |
| intersect | boolean | true | if true, the interaction mode only applies when the mouse position intersects an item on the chart. |
| axis | string | 'x' | Can be set to 'x', 'y', 'xy' or 'r' to define which directions are used in calculating distances. Defaults to 'x' for 'index' mode and 'xy' in dataset and 'nearest' modes. |
| includeInvisible | boolean | false | if true, the invisible points that are outside of the chart area will also be included when evaluating interactions. |

By default, these options apply to both the hover and tooltip interactions. The same options can be set in the options.hover namespace, in which case they will only affect the hover interaction. Similarly, the options can be set in the options.plugins.tooltip namespace to independently configure the tooltip interactions.

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

The following properties define how the chart interacts with events. Namespace: options

| Name | Type | Default | Description |
| --- | --- | --- | --- |
| events | string[] | ['mousemove', 'mouseout', 'click', 'touchstart', 'touchmove'] | The events option defines the browser events that the chart should listen to for. Each of these events trigger hover and are passed to plugins. [more...](#3znysh7) |
| onHover | function | null | Called when any of the events fire over chartArea. Passed the event, an array of active elements (bars, points, etc), and the chart. |
| onClick | function | null | Called if the event is of type 'mouseup', 'click' or ''contextmenu' over chartArea. Passed the event, an array of active elements, and the chart. |

### [**#**](#3znysh7) Event Option

For example, to have the chart only respond to click events, you could do:

const chart = new Chart(ctx, {  
 type: 'line',  
 data: data,  
 options: {  
 // This chart will not respond to mousemove, etc  
 events: ['click']  
 }  
});

Events for each plugin can be further limited by defining (allowed) events array in plugin options:

const chart = new Chart(ctx, {  
 type: 'line',  
 data: data,  
 options: {  
 // All of these (default) events trigger a hover and are passed to all plugins,  
 // unless limited at plugin options  
 events: ['mousemove', 'mouseout', 'click', 'touchstart', 'touchmove'],  
 plugins: {  
 tooltip: {  
 // Tooltip will only receive click events  
 events: ['click']  
 }  
 }  
 }  
});

Events that do not fire over chartArea, like mouseout, can be captured using a simple plugin:

const chart = new Chart(ctx, {  
 type: 'line',  
 data: data,  
 options: {  
 // these are the default events:  
 // events: ['mousemove', 'mouseout', 'click', 'touchstart', 'touchmove'],  
 },  
 plugins: [{  
 id: 'myEventCatcher',  
 beforeEvent(chart, args, pluginOptions) {  
 const event = args.event;  
 if (event.type === 'mouseout') {  
 // process the event  
 }  
 }  
 }]  
});

For more information about plugins, see [Plugins](http://docs.google.com/docs/3.9.1/developers/plugins.html)

### [**#**](#2et92p0) Converting Events to Data Values

A common occurrence is taking an event, such as a click, and finding the data coordinates on the chart where the event occurred. Chart.js provides helpers that make this a straightforward process.

const chart = new Chart(ctx, {  
 type: 'line',  
 data: data,  
 options: {  
 onClick: (e) => {  
 const canvasPosition = Chart.helpers.getRelativePosition(e, chart);  
 // Substitute the appropriate scale IDs  
 const dataX = chart.scales.x.getValueForPixel(canvasPosition.x);  
 const dataY = chart.scales.y.getValueForPixel(canvasPosition.y);  
 }  
 }  
});

When using a bundler, the helper functions have to be imported seperatly, for a full explanation of this please head over to the [integration](http://docs.google.com/docs/3.9.1/getting-started/integration.html#helper-functions) page

## [**#**](#1fob9te) Modes

When configuring the interaction with the graph via interaction, hover or tooltips, a number of different modes are available.

options.hover and options.plugins.tooltip extend from options.interaction. So if mode, intersect or any other common settings are configured only in options.interaction, both hover and tooltips obey that.

The modes are detailed below and how they behave in conjunction with the intersect setting.

See how different modes work with the tooltip in [tooltip interactions sample](http://docs.google.com/docs/3.9.1/samples/tooltip/interactions.html)

### [**#**](#tyjcwt) point

Finds all of the items that intersect the point.

const chart = new Chart(ctx, {  
 type: 'line',  
 data: data,  
 options: {  
 interaction: {  
 mode: 'point'  
 }  
 }  
});

### [**#**](#3dy6vkm) nearest

Gets the items that are at the nearest distance to the point. The nearest item is determined based on the distance to the center of the chart item (point, bar). You can use the axis setting to define which coordinates are considered in distance calculation. If intersect is true, this is only triggered when the mouse position intersects an item in the graph. This is very useful for combo charts where points are hidden behind bars.

const chart = new Chart(ctx, {  
 type: 'line',  
 data: data,  
 options: {  
 interaction: {  
 mode: 'nearest'  
 }  
 }  
});

### [**#**](#1t3h5sf) index

Finds item at the same index. If the intersect setting is true, the first intersecting item is used to determine the index in the data. If intersect false the nearest item, in the x direction, is used to determine the index.

const chart = new Chart(ctx, {  
 type: 'line',  
 data: data,  
 options: {  
 interaction: {  
 mode: 'index'  
 }  
 }  
});

To use index mode in a chart like the horizontal bar chart, where we search along the y direction, you can use the axis setting introduced in v2.7.0. By setting this value to 'y' on the y direction is used.

const chart = new Chart(ctx, {  
 type: 'bar',  
 data: data,  
 options: {  
 interaction: {  
 mode: 'index',  
 axis: 'y'  
 }  
 }  
});

### [**#**](#4d34og8) dataset

Finds items in the same dataset. If the intersect setting is true, the first intersecting item is used to determine the index in the data. If intersect false the nearest item is used to determine the index.

const chart = new Chart(ctx, {  
 type: 'line',  
 data: data,  
 options: {  
 interaction: {  
 mode: 'dataset'  
 }  
 }  
});

### [**#**](#2s8eyo1) x

Returns all items that would intersect based on the X coordinate of the position only. Would be useful for a vertical cursor implementation. Note that this only applies to cartesian charts.

const chart = new Chart(ctx, {  
 type: 'line',  
 data: data,  
 options: {  
 interaction: {  
 mode: 'x'  
 }  
 }  
});

### [**#**](#17dp8vu) y

Returns all items that would intersect based on the Y coordinate of the position. This would be useful for a horizontal cursor implementation. Note that this only applies to cartesian charts.

const chart = new Chart(ctx, {  
 type: 'line',  
 data: data,  
 options: {  
 interaction: {  
 mode: 'y'  
 }  
 }  
});

## [**#**](#3rdcrjn) Custom Interaction Modes

New modes can be defined by adding functions to the Chart.Interaction.modes map. You can use the Chart.Interaction.evaluateInteractionItems function to help implement these.

Example:

import { Interaction } from 'chart.js';  
import { getRelativePosition } from 'chart.js/helpers';  
/\*\*  
 \* Custom interaction mode  
 \* @function Interaction.modes.myCustomMode  
 \* @param {Chart} chart - the chart we are returning items from  
 \* @param {Event} e - the event we are find things at  
 \* @param {InteractionOptions} options - options to use  
 \* @param {boolean} [useFinalPosition] - use final element position (animation target)  
 \* @return {InteractionItem[]} - items that are found  
 \*/  
Interaction.modes.myCustomMode = function(chart, e, options, useFinalPosition) {  
 const position = getRelativePosition(e, chart);  
 const items = [];  
 Interaction.evaluateInteractionItems(chart, 'x', position, (element, datasetIndex, index) => {  
 if (element.inXRange(position.x, useFinalPosition) && myCustomLogic(element)) {  
 items.push({element, datasetIndex, index});  
 }  
 });  
 return items;  
};  
// Then, to use it...  
new Chart.js(ctx, {  
 type: 'line',  
 data: data,  
 options: {  
 interaction: {  
 mode: 'myCustomMode'  
 }  
 }  
})

If you're using TypeScript, you'll also need to register the new mode:

declare module 'chart.js' {  
 interface InteractionModeMap {  
 myCustomMode: InteractionModeFunction;  
 }  
}

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

←  [Elements](http://docs.google.com/docs/3.9.1/configuration/elements.html)   [Layout](http://docs.google.com/docs/3.9.1/configuration/layout.html)  →