LightningChart JS Migration Guide

From version 3.x to version 4.0.0.

`dispose` and `restore` changes

All `restore()` methods have been removed.

`dispose()` now **permanently** destroys the component. This action is no longer reversable.

To temporarily hide components, use `setVisible(false)`

`Axis.setInterval`

Wrapped parameters of `Axis.setInterval` method in an object.

Also, optional parameter `stopAxisAfter` default value was changed from `false` to `true`.

Migration examples:

`Axis.setInterval(start, end)` -> `Axis.setInterval({ start, end, stopAxisAfter: true })`

`Axis.setInterval(start, end, false)` -> `Axis.setInterval({ start, end, animate: false, stopAxisAfter: true })`

`Axis.setInterval(start, end, false, true)` -> `Axis.setInterval({ start, end, animate: false, stopAxisAfter: true })`

Special note for use of `Axis.setInterval` with progressive scrolling axis; previously this involved code like:

```
// Specify visible axis interval and enable progressive scrolling
Axis.setInterval(0, 1000).setScrollStrategy(AxisScrollStrategies.progressive)
```

After v4.0.0, `setInterval` stops axis from scrolling by default. So, in order to get the expected behavior, `stopAxisAfter: false` has to be added:

```
// Specify visible axis interval and enable progressive scrolling
Axis.setInterval({ start: 0, end: 1000, stopAxisAfter: false }).setScrollStrategy(AxisScrollStrategies.progressive)
```

`Axis.stop` and `Axis.release` changes

Both `Axis.stop` and `Axis.release` methods have been renamed. Usage of `stop` should be replaced with `Axis.setStopped(true)` and `release` with

`Axis.setStopped(false)`

`Axis.onScaleChange`

Renamed to `Axis.onIntervalChange`.

Breaking change in number of parameters. Previously used like:

`Axis.onScaleChange((start, end) => ...)`

Now should be changed to:

`Axis.onIntervalChange((axis, start, end) => ...)`

`ChartXY.addHeatmapSeries`

Old poorly performing `ChartXY.addHeatmapSeries` API has been removed. To create Heatmap Grid series use `ChartXY.addHeatmapGridSeries` or `ChartXY.addHeatmapScrollingGridSeries` instead.

Heatmap Mesh series functionality (`IntensitySeriesTypes.Mesh`) has been removed completely. We have plans to reintroduce this at a later point. If you require this functionality, please contact us (support@lightningchart.com) and let us know your use case.

`Chart3D.addSurfaceSeries`

Old poorly performing `Chart3D.addSurfaceSeries` API has been removed. To create Surface Grid series use `Chart3D.addSurfaceGridSeries` or `Chart3D.addSurfaceScrollingGridSeries` instead.

3D Mesh series functionality has been removed completely. We have plans to reintroduce this at a later point. If you require this functionality, please contact us (support@lightningchart.com) and let us know your use case.

Highlight styles

All styles that are specific only to components when they are highlighted have been removed.

After v4.0.0 all highlighted components are automatically highlighted according to the used theme - in dark themes, color is brightened, whereas in light themes, color is darkened.

If you need to apply different style for hovered or highlighted component, then you can utilize methods like `onMouseEnter` and `onHighlight` :

```
 // Change line series style when mouse is over it.
lineSeries.onMouseEnter(() => {
  lineSeries.setStrokeStyle(new SolidLine({ thickness: 2, fillStyle: new SolidFill({ color: ColorRGBA(255, 0, 0) }) }))
})
lineSeries.onMouseLeave(() => {
  lineSeries.setStrokeStyle(new SolidLine({ thickness: 2, fillStyle: new SolidFill({ color: ColorRGBA(0, 255, 0) }) }))
})
```

Themes

After v4.0.0, the following built-in Themes are available:

- `darkGold` (default)
- `light`
- `lightNature`
- `cyberSpace`
- `turquoiseHexagon`

Themes that utilize file resources

`cyberSpace` and `turquoiseHexagon` utilize file assets included along-side the library. In order to use these, you need to setup resources hosting (see resourcesBaseUrl docs).

Custom themes

Removed built-in custom color theme utilities: `customTheme`, `customSimpleTheme`, `customComplexTheme`

The `Theme` interface has been completely rewritten in v4.0.0. The intention with `Theme` rewrite is to:

1. replace ambiguous properties (not sure what it affects, or affects many things) with non-ambiguous ones (name clearly describes what it should do)
2. make Theme properties as flexible as possible. For example, separate properties for X and Y axis styles rather than having one property for all types of axes.

Unfortunately, this results in very hard migration from previous `Theme` usages and a significantly increased number of properties.

Starting with v4.0.0 the topic of custom color themes is moving to a separate open-source repository @arction/lcjs-themes. At this time, the repository is very small and only includes a minimal example of making a custom color theme in v4.0.0. Going forward, both LCJS users and developers can extend this repository with the intention of making custom color themes easier for us both.

For latest information of custom color themes, refer to FAQ in API documentation and read section "How to use Custom Color Themes?".

`setTitleMarginBottom`, `setTitleMarginTop`

These methods and relative getters have been removed. Their usage is replaced with `setTitleMargin`, which accepts an object that can have properties for `left`, `right`, `top` and `bottom`. The behavior is unchanged.

```
 // Before
chart.setTitleMarginBottom(40)

// After
chart.setTitleMargin({ bottom: 40 })
```

`setMouseInteractions`

Before v4.0.0, having mouse interactions on could have negative impacts on performance, so often the official instruction was to disable them unless required. Furthermore, many components had them disabled by default and some components even didn't support them.

Starting with v4.0.0, all mouse interactions are enabled by default. The performance of having mouse interactions enabled is not a real concern anymore, regardless of the component in question, even massive line series, etc.

One side effect you might see from this is after migration is that series may start highlighting on mouse hover. To disable highlighting on hover use `setHighlightOnHover(false)`

`onHover, offHover`

These methods has been removed, their use cases can be replaced by using methods: `onMouseEnter`, `onMouseLeave` and `solveNearestFromScreen`.

If you copied their usage from a Interactive Example, please refer to the updated code of that example to learn how to use the alternate methods for the same use case.

`onHighlight`

`onHighlight` method was previously defined as:

```
 onHighlight(handler: (isHighlighted: boolean) => void): Token
```

In v.4.0.0, the callback parameter changes from `boolean` to `boolean | number`.

```
 onHighlight(handler: (isHighlighted: boolean | number) => void): Token
```

`Axis.getHeight`

No replacement exists currently.

To force exact pixel alignment of an Axis, configure it using `Axis.setThickness`.

`Dashboard.mapCharts` and `Dashboard.forEachCharts`

Use `Dashboard.getCells()` and filter for `Chart` instances or the instance type you want to interact with.

```
 const cells = dashboard.getCells()
cells.forEach((cell) => {
    if (cell.panel instanceof Chart3D) {
        cell.panel.setBoundingBox({ x: 2, y: 1, z: 1 })
    }
})
```

`LegendBoxOptions.disposeOnClick`

The `LegendBoxAddOptions.disposeOnClick` property has been changed to `toggleVisibilityOnClick`.

Functionality is also changed to use setVisible internally, rather than dispose.

`solveNearestFromSegment`

This method has been removed, use `solveNearestFromScreen` instead.

`disableAnimations`

All `disableAnimations` methods have been removed. It has been replaced with `setAnimationsEnabled(false)`.

`setMaxPointCount`

All `setMaxPointCount` methods have been removed. All previous usage should migrate to using `setDataCleaning` method.

Previous usage of `setMaxPointCount`:

```
series.setMaxPointCount(1000);
```

Usage in 4.0:

```
series.setDataCleaning({
  minDataPointCount: 1000,
});
```

Be extra careful with above migration, since some series types support specifying `maxDataPointCount` but this behaves differently from previous `setMaxPointCount` configuration.

`OHLCSeries` previously behaved in a different manner to other series with regards to `setMaxPointCount` configuration. To replicate exact same behavior as before, `maxDataPointCount` configuration should be supplied:

Previous usage of `OHLCSeries.setMaxPointCount`:

```
ohlcSeries.setMaxPointCount(1000);
```

Usage in 4.0:

```
ohlcSeries.setDataCleaning({
  maxDataPointCount: 1000,
});
```

`setDataCleaningThreshold`

All `setDataCleaningThreshold` methods have been removed. All previous usage should migrate to using `setDataCleaning` method.

Previous usage of `setDataCleaningThreshold`:

```
series.setDataCleaningThreshold(1000);
```

Usage in 4.0:

```
series.setDataCleaning({ progressiveDataCleaningThreshold: 1000 });
```

`setMouseInteractionsWhileScrolling`

This method has been removed. If you were using this method before upgrading to v.4.x, you should use `setAutoCursorEnabledDuringAxisAnimation` method instead.

`setMouseInteractionsWhileZooming`

This method has been removed. If you were using this method before upgrading to v.4.x, you should use `setAutoCursorEnabledDuringAxisAnimation` method instead.

LineStyle.thickness

`LineStyle.thickness` property has been removed. Please use `LineStyle.getThickness()` instead.

Type change in many `setStrokeStyle` methods

In previous library versions, many `setStrokeStyle` (or equivalent) methods only accepted `SolidLine` as argument. For the most part, this only meant that you could not supply `emptyLine` in TypeScript applications.

After v4.0.0, these methods now accept `LineStyle`. This is a more abstract type than `SolidLine`, which means that `emptyLine` can also be used. This should not affect any *loosely typed usages* because `LineStyle` has all same methods as `SolidLine` (`setThickness`, `setFillStyle`).

```
 // Example of loosely typed use of setStrokeStyle
// This is OK
lineSeries.setStrokeStyle((stroke) => stroke.setThickness(1))
```

However, it is possible that if you explicitly defined the expected type of the callback, then you may get type issues after migrating to v4.0.0:

```
 // Example of explicitly defining expected callback type
lineSeries.setStrokeStyle((stroke: SolidLine) => stroke.setThickness(1))

// You might get type errors above, so you should remove the type definition, like so:
lineSeries.setStrokeStyle((stroke) => stroke.setThickness(1))
```

All APIs affected by this:

- `SpiderChart.setAxisStyle`
- `SpiderChart.getAxisStyle`
- `SpiderChart.setNibStyle`
- `SpiderChart.getNibStyle`
- `ConstantLine.setStrokeStyle`
- `ConstantLine.getStrokeStyle`
- `ConstantLine.setStrokeStyleHighlight`
- `AreaSeriesBipolar.setPositiveStrokeStyle`
- `AreaSeriesBipolar.setPositiveStrokeStyleHighlight`
- `AreaSeriesBipolar.setNegativeStrokeStyle`
- `AreaSeriesBipolar.setNegativeStrokeStyleHighlight`
- `AreaSeriesMonopolar.setStrokeStyleHighlight`
- `AreaRangeSeries.setHighStrokeStyle`
- `AreaRangeSeries.setLowStrokeStyle`
- `AreaRangeSeries.setHighStrokeStyleHighlight`
- `AreaRangeSeries.setLowStrokeStyleHighlight`
- `LineSeries.setStrokeStyle`
- `LineSeries.getStrokeStyle`
- `LineSeries.setStrokeStyleHighlight`
- `LineSeries.setStrokeStyleHighlight`
- `PointLineSeries.setStrokeStyle`
- `PointLineSeries.getStrokeStyle`
- `PointLineSeries.setStrokeStyleHighlight`
- `PointLineSeries.setStrokeStyleHighlight`

`setLineStyle`, `getLineStyle`

All occurences of these methods have been renamed as below:

- `setLineStyle` -> `setStrokeStyle`
- `getLineStyle` -> `getStrokeStyle`

`Axis.setTickStyle`

Method removed, use `Axis.setTickStrategy` to style ticks.

`ZoomBandChart.attachedAxis`

Changed from `attachedAxis: Axis` to `attachedAxes: Axis[]`.

`HeatmapGridSeries.setPixelInterpolationMode`

Renamed to `setIntensityInterpolation`.

`onAxisAreaMouseDragStart`, `onAxisAreaMouseDrag`, `onAxisAreaMouseDragStop`, `onAxisInteractionAreaMouseTouchStart`, `onAxisInteractionAreaMouseTouch`, `onAxisInteractionAreaMouseTouchStop`, `offAxisInteractionAreaMouseTouchStart`,

`offAxisInteractionAreaMouseTouchStart`, `offAxisInteractionAreaMouseTouch`, `offAxisInteractionAreaMouseTouchStop`

These methods were not named correctly, and have now been renamed like below:

- `onAxisAreaMouseDragStart` -> `onAxisInteractionAreaMouseDragStart`
- `onAxisAreaMouseDrag` -> `onAxisInteractionAreaMouseDrag`
- `onAxisAreaMouseDragStop` -> `onAxisInteractionAreaMouseDragStop`
- `onAxisInteractionAreaMouseTouchStart` -> `onAxisInteractionAreaTouchStart`
- `onAxisInteractionAreaMouseTouch` -> `onAxisInteractionAreaTouch`
- `onAxisInteractionAreaMouseTouchStop` -> `onAxisInteractionAreaTouchStop`
- `offAxisInteractionAreaMouseTouchStart` -> `offAxisInteractionAreaTouchStart`
- `offAxisInteractionAreaMouseTouch` -> `offAxisInteractionAreaTouch`
- `offAxisInteractionAreaMouseTouchStop` -> `offAxisInteractionAreaTouchStop`

TypeScript support

Minimum TypeScript version support bumped from `2.8` to `4.1`.

ChartXY default AutoCursor behavior

The default AutoCursor behavior in ChartXY has been changed.

Now the AutoCursor is enabled during Axis animations (when the Axis interval change is animated, i.e. during zooming) by default.

To restore previous behavior, use the new `ChartXY.setAutoCursorEnabledDuringAxisAnimation` method.

`UILegendBox`

This type has been removed and replaced with `LegendBox`