LightningChartJS Migration guide

From version 1.3.1 and before to version 2.0.0 and after

LineSet improvements

While this is not a compatibility breaking change, we feel this is worthy enough to highlight here.

We have improved drawing for thick lines in our library, reducing the memory usage by up to ~75%, while simultaneously improving the visual of the lines.

Axis Changes

In this release we have refactored how our axis tick strategies work to allow for more complex axes and a better visual style. As a result, the way tick strategies are used was changed.

- *Axis.setTickStrategy(TickStrategy, (optional)tickStrategyMutator)* was added.
  - This is used to set the *Tick Strategies* for the *Axis*, as well as styling elements of *Tick Strategies*.
  - The mutator is optional, and only used when styling or modifying elements of the *Tick Strategy*
- DateTime TickStrategy Origin is now set through a setter method
  - Use a mutator to change the *DateOrigin* for *DateTime Tick Strategy*
  - *Axis.setTickStrategy(AxisTickStrategies.DateTime, (tickStrategy) => { tickStrategy.setDateOrigin(dateOrigin) })*

Axis ticks have now been separated into three different categories:

- Major ticks
  - These are always shown
  - They represent major thresholds in the shown scale
- Minor ticks
  - Shown by default, can be hidden
  - Are fit between major ticks
  - Labels are shown if they can fit without overlapping other labels
    - Otherwise only some, or none, are shown
    - Tick and grid lines are shown regardless
- Great ticks
  - Used with DateTime Tick Strategy
  - Shown for great thresholds (such as Years in a date)
  - Can be hidden
- Extreme ticks
  - Hidden by default, can be enabled
  - Shown at the ends of the Axis
  - Useful for always showing the current extreme values of an Axis

The different ticks can be styled by using the *Axis.setTickStrategy()* method, via using the optional *tickStrategyMutator:*

- *Axis.setTickStrategy(TickStrategy, (mutator) => { mutator.setMajorTickStyle((tickStyle) => { tickStyle.setGridStrokeStyle(…) }) })*

AxisTickStrategies.NumericWithUnits

NumericWithUnits TickStrategy has been removed. The same functionality can be achieved by using ***Numeric TickStrategy*** and its ***setFormattingFunction()*** method.

Default Axis tick strategy removed from chart / axis creation

Old behavior:

- On Chart creation:

  LightningChart.ChartXY({ defaultAxisXTickStrategy: AxisTickStrategies.DateTime() })

- *On Axis creation:*

  ChartXY.addAxisX(undefined, AxisTickStrategies.DateTime())

New behavior:

- On Chart creation:

  LightningChart.ChartXY().setTickStrategy(AxisTickStrategies.DateTime)

- On Axis creation:

  ChartXY.addAxisX().setTickStrategy(AxisTickStrategies.DateTime)

WebGL Extension Requirements

LightningChart JS now requires the following WebGL extensions to work properly:

- ANGLE_instanced_arrays
- EXT_blend_minmax
- OES_element_index_uint
- OES_standard_derivatives
- OES_vertex_array_object
- WEBGL_lose_context

These extensions are implemented in all modern browsers on both desktop and mobile. If any of these extensions is missing, then a dismissible warning will be shown to notify users of possibly incorrectly working features.

You'll already see the reason why we're doing this – PointSet performance has been improved significantly.

Dashboard Options

Creation of Dashboard was simplified. Chart Options are no longer separated as a separate options object. `columnSpan` and `rowSpan` are now optional options that default to 1 if no value is defined.

- Previously: *Dashboard.createChartXY({ columnIndex: 0, rowIndex: 0, columnSpan: 1, rowSpan: 1, chartXYOptions: { theme: Themes.dark, ... } })*
- Now: *Dashboard.createChartXY({ columnIndex: 0, rowIndex: 0, columnSpan: 1, rowSpan: 1, theme: Themes.dark, … })*
    - *The ColumnSpan and RowSpan are now optional and will default to 1 if no value is given.*

Polyfills for requestAnimationFrame and cancelAnimationFrame

Version 1.3.1 and before, we had included our own polyfills for these functionalities. From version 2.0.0 onwards, developers will need to add their own polyfills for these. A library such as requestAnimationFrame polyfill library can be used for this. This was changed to follow best practices for polyfilling in libraries.

ColorHEX changes

We've changed the order of values in our ColorHEX method (used to be #ARGB, #AARRGGBB) to correspond the CSS form of #RGBA / #RRGGBBAA

Deprecated APIs removed

- *SolidGauge.setDataLabelFormater* removed -> Use *SolidGauge.setDataLabelFormatter*
- *SolidGauge.getDataLabelFormater* removed -> Use *SolidGauge.getDataLabelFormatter*
- *setChartBackgroundStroke* removed -> Use *setChartBackgroundStrokeStyle*
- *getChartBackgroundStroke* removed -> Use *getChartBackgroundStrokeStyle*
- *setMaxPointsCount* removed -> Use *setMaxPointCount*
- *containerId* engine option removed -> Use *container* engine option instead. This allows passing either the containerId as a string like before, or just passing the container (div) itself.