Chart Library API

How to Use Chart Library

Make sure you linked chart library on the web page:

```
<script type="text/javascript" src="chart.min.js"></script>
```

Create chart instance:

```
<div id="chartContainer" style="width: 800px; height: 600px"></
div>
<script type="text/javascript">
    let container = document.getElementById('chartContainer');
    let options = {};
    let myChart = new T4PChart(container, options);
    /* ... your code here ... */
</script>
```

After that you can access chart API via myChart variable:

```
myChart.setSymbol('EURCAD');
myChart.setTitle('EUR / CAD');
myChart.setTimeframe('1H');
```

You will find detailed chart API in the **Chart API** section.

Chart Instance Options

As you can see from previous example you can provide chart instance with options object. Here is the list of some available options:

general.defaultSymbol
    Defines symbol by default
    Default value: 'EURUSD'.

general.defaultTitle
    Defines default symbol display name.
    Default value: 'EUR/USD'.

general.defaultTimeframe

Defines default timeframe.
Default value: '5M'.

general.defaultChartType
Defines default chart type.
Default value: 'candles'.
Valid chart types are: *candles*, *bars*, *line*, *area*.

general.defaultScale
Defines default chart scale for time axis.
Default value: 1.

general.defaultLayout
Defines default multiscreen layout.
Default value: 0 (single screen).
Valid options are: *0 (single screen)*, *1 (horizontal split)*, *2 (vertical split)*, *3 (horizontal split x3)*, *4 (vertical split x3)*, *5 (quarters view)*, *6 (one+five)*, *7 (six charts)*.

general.displayChange
Toggles asset price change (in percent) for the current timeframe in OHLC panel.
Default value: false.

general.displayIndicatorNames
Toggles active indicators short names on the chart.
Default value: true.

general.displayTimestamp
Toggles candle timestamp in OHLC panel.
Default value: false.

general.displayVolume
Toggles candles volume on the chart (and in OHLC panel).
Default value: true.

general.selectionControls
If true then control buttons will be shown if there is a drawing or an indicator selected (usually three buttons: settings, styles and delete).
Default value: true.

general.saveLayout
If true then chart will restore last used layout (including symbols, timeframes and chart types for each chart panel), otherwise *default* options will be used.

Default value: true.

general.saveIndicators
If true then chart will restore all indicators from previous session.
Default value: true.

general.saveDrawings
If true then chart will restore all drawings from previous session.
Default value: false.

general.mobilePointButtonText
Text on the button used in the process of adding drawings when user specifies points.
Default value: 'Next'.

toolbar.timeframes
Defines available timeframes and its' names.
Default value: {
'1T':'1 Tick',
'1M':'1 minute',
'5M':'5 minutes',
'15M':'15 minutes',
'30M':'30 minutes',
'1H':'1 hour',
'4H':'4 hours',
'1D':'1 day',
'1W':'1 week'
}.
Valid timeframes: *1T, 10S, 30S*, *1M*, *5M*, *15M*, *30M*, *1H*, *4H*, *1D*, *1W, 1MO*.

toolbar.elements
Defines active toolbar items.
Default value is: ['symbol', 'timeframe', 'type', 'indicators', 'drawings', 'multiview', 'clear', 'screenshot', 'language'].
Valid toolbar elements: *symbol*, *timeframe*, *type*, *indicators*, *drawings*, *multiview*, *clear*, *screenshot*, *language*.

system.roundedCandles
If *true* then candles will have rounded corners (depending on zoom).
Default value: true.

Custom indicator names:

You can change default indicator short names (which are displayed on the chart if

general.displayIndicatorNames options is enabled):

indicators.Aroon
Default value: 'Aroon'

indicators.AverageDirectionalIndex
Default value: 'ADX'

indicators.BollingerBands
Default value: 'BBands'

indicators.BullsBearsPower
Default value: 'BBPower'

indicators.DonchianChannel
Default value: 'DonChan'

indicators.Envelope
Default value: 'Envelope'

indicators.ExponentialMovingAverage
Default value: 'EMA'

indicators.Fractals
Default value: 'Fractals'

indicators.IchimokuCloud
Default value: 'IchiCloud'

indicators.Momentum
Default value: 'Moment'

indicators.MovingAverageConvergenceDivergence
Default value: 'MACD'

indicators.ParabolicSAR
Default value: 'PSAR'

indicators.PivotPoints
Default value: 'PPoints'

indicators.RelativeStrengthIndex
Default value: 'RSI'

indicators.SimpleMovingAverage
Default value: 'SMA'

indicators.StandardDeviation
Default value: 'StDev'

indicators.StochasticOscillator
Default value: 'Stoch'

indicators.WilliamsAlligator
Default value: 'WillAll'

indicators.ZigZag
Default value: 'ZigZag'

Color options:

You can also create your custom color theme using color options.

colors.background
Defines the chart's background color.
Default value: '#ffffff'

colors.title
Defines the asset title color.
Default value: '#737d89'

colors.frame
Defines the color of delimiters between charts when multiscreen is enabled.
Default value: '#b5c1ca'

colors.frameActive
Defines the color of the active chart frame.
Default value: '#1cb5ee'

colors.frameFullscreen
Defines the color of the chart frame in fullscreen mode when multiscreen is enabled.
Default value: '#f85e29'

colors.grid
Defines the color of the chart's dotted grid lines.
Default value: '#cfd2d9'

colors.axisLine
Defines the color of the axes.
Default value: '#d5dbe6'

colors.axisText
Defines the color of the text labels on the axes.
Default value: '#828a9b'

colors.crossLine
Defines the color of the cross (mouse cursor position guide lines).
Default value: '#1cb5ee'

colors.candleRise
Defines the color of rising candles.
Default value: '#7ac522'

colors.candleRiseBorder
Defines the color of rising candles border.
Default value: '#707070'

colors.candleFall
Defines the color of falling candles.
Default value: '#e64b3b'

colors.candleFallBorder
Defines the color of falling candles border.
Default value: '#707070'

colors.candleShadow
Defines the candle shadows color.
Default value: '#707070'

colors.barRise
Defines the color of rising bars.
Default value: '#7ac522'

colors.barFall
Defines the color of falling bars.
Default value: '#e64b3b'

colors.futureLine
Defines the color of the trading idea time (used when Trading Room widget integrated).

Default value: '#c8d1d7'

colors.futureBackground
Defines the background color of the chart area after trading idea time line (used when Trading Room widget integrated).
Default value: 'rgba(200, 206, 215, 0.5)'

colors.mobilePointButtonBg
Background color of the button used in the process of adding drawings when user specifies points.
Default value: '#1cb5ee'

colors.mobilePointButtonTxt
Text color of the button used in the process of adding drawings when user specifies points.
Default value: '#ffffff'

Another way to specify colors is to call chart API method setColor (see below).

Chart API

Chart API consists of two parts: general API and data API. General API provides access to most commonly used methods such as getting and setting active symbol and timeframe, changing current layout etc. Data API provides full control over historical data displayed on the chart (candles).

General API Methods

chart.setTimezone(timezone)
Specifies current timezone used to display date / time values. Keep in mind that *timezone* value must be valid timezone IANA/Olson TZ string such as: 'Australia/Sydney', 'America/Los_Angeles', 'Europe/Rome' etc.
By default timezone is 'UTC'.

chart.setLayout(scheme)
Sets multiple charts layout. Available *scheme* values are:

- *0* - one single chart;
- *1* - two charts split horizontally;
- *2* - two charts split vertically;
- *3* - three charts split horizontally;

- *4* - three charts split vertically;
- *5* - four charts, table 2x2;
- *6* - six charts, one primary and five small charts;
- *7* - six charts, table 3x2.

chart.getLayout()
Returns current layout scheme (see *chart.setLayout* method).

chart.setSymbol(symbol, <index>)
Sets chart symbol. Parameter *index* is used to specify chart panel in multiscreen mode. By default active chart panel will be affected.

chart.getSymbol(<index>)
Returns chart symbol. Parameter *index* is used to specify chart panel in multiscreen mode. By default active chart panel is used.

chart.setDisplayName(title, <index>)
Sets chart symbol title. Parameter *index* is used to specify chart panel in multiscreen mode. By default active chart panel will be affected.

chart.getDisplayName(<index>)
Returns chart symbol title. Parameter *index* is used to specify chart panel in multiscreen mode. By default active chart panel is used.

chart.setTimeframe(timeframe, <index>)
Sets chart timeframe. Parameter *index* is used to specify chart panel in multiscreen mode. By default active chart panel will be affected.
Available timeframes are:
- *1T* - 1 tick (in fact equals 1 second);
- *1M* - 1 minute;
- *5M* - 5 minutes;
- *15M* - 15 minutes;
- *30M* - 30 minutes;
- *1H* - 1 hour;
- *4H* - 4 hours;
- *1D* - 1 day;
- *1W* - 1 week.

chart.getTimeframe(<index>)
Returns chart timeframe. Parameter *index* is used to specify chart panel in multiscreen mode. By default active chart panel is used.

chart.setChartType(type, <index>)

Sets chart type. Parameter *index* is used to specify chart panel in multiscreen mode. By default active chart panel will be affected.
Available chart type values are: '*candles*', '*bars*', '*line*', '*area*'.

chart.getChartType(<index>)
Returns chart type string. Parameter *index* is used to specify chart panel in multiscreen mode. By default active chart panel is used.

chart.setColor(element, color)
Sets the color of the element.
Available element values are: 'background', 'grid', 'axisText', 'candleBorder'.
Color should be a CSS color string value.

chart.setDecimals(symbol, digits)
Set a price format for a specified symbol using fixed-point notation.
digits - the number of digits to appear after the decimal point (default *null*).

chart.getDecimals(symbol)
Get current number of digits to appear after the decimal point for the specified symbol.

chart.setRange(timestampFrom, timestampTo, <index>)
Sets visible chart area range. Parameter *index* is used to specify chart panel in multiscreen mode. By default active chart panel will be affected.
Parameters *timestampFrom* and *timestampTo* are unix timestamps.

chart.addDeal(symbol, timestamp, entry, <takeProfit>, <stopLoss>)
Adds trade deal on the chart for specified symbol.
Returns [*Deal object]*.

chart.addIndicator(name, parameters, <index>)
Adds indicator to the chart. Parameter *index* is used to specify chart panel in multiscreen mode. By default active chart panel is used.
See detailed API below in section *Indicators API*.

chart.getIndicators(<index>)
Gets previously added indicators. Parameter *index* is used to specify chart panel in multiscreen mode. By default active chart panel is used.
Returns Array of *[Indicator object]*.

chart.clearIndicators(<index>)
Deletes all indicators on the chart. Parameter *index* is used to specify chart panel in multiscreen mode. By default active chart panel is used.

chart.addDrawing(name, parameters, <index>)
Adds drawing to the chart. Parameter *index* is used to specify chart panel in multiscreen mode. By default active chart panel is used.
See detailed API below in section *Drawings API*.

chart.getDrawings(<index>)
Gets previously added drawings. Parameter *index* is used to specify chart panel in multiscreen mode. By default active chart panel is used.
Returns Array of *[Drawing object]*.

chart.clearDrawings(<index>)
Deletes all drawings on the chart. Parameter *index* is used to specify chart panel in multiscreen mode. By default active chart panel is used.

Indicators API:

In order to add indicator via Chart API you need to call chart.addIndicator method and pass indicator name and all parameters it requires:

chart.addIndicator(name, parameters, <index>)
Parameter *index* is used to specify chart panel in multiscreen mode. By default active chart panel is used.

Example:

```
chart.addIndicator('RelativeStrengthIndex', {
    n: 15,
    overbought: 70,
    oversold: 30,
    color: '#65b22e',
    bold: true,
    colorBg: 'rgba(174,208,131,0.2)'
});
```

Important! All color parameters must be valid CSS color string values!

Here is the list of all available indicators at the moment:
*Note that default parameters values listed below are set as default only when you add indicators via chart UI. In case of API call you **must** specify all listed parameters.*

Aroon
name: 'Aroon'

settings:
n - Length, default value: 15
styles:
colorUp - Up line color, default value: '#ef7d00'
boldUp - Up line bold, default value: false
colorDown - Down line color, default value: '#006ab4'
boldDown - Down line bold, default value: false

Average Directional Index
name: 'AverageDirectionalIndex'
settings:
n - Length, default value: 14
styles:
colorADX - ADX line color, default value: '#003f84'
boldADX - ADX line bold, default value: false
colorPDI - Plus DI line color, default value: '#f78d59'
boldPDI - Plus DI line bold, default value: false
colorMDI - Minus DI line color, default value: '#5bc488'
boldMDI - Minus DI line bold, default value: false

Bollinger Bands
name: 'BollingerBands'
settings:
n - Length, default value: 15
k - StdDev, default value: 1
styles:
colorBasis - Bass line color, default value: '#0e73b9'
boldBasis - Bass line bold, default value: false
colorBorder - Borders color, default value: '#00b0ea'
boldBorder - Borders bold, default value: false
colorBg - Background color, default value: 'rgba(0,176,235,0.2)'

Bulls Bears Power
name: 'BullsBearsPower'
settings:
n - Length, default value: 15
showBull - Display Bulls line, default value: true
showBear - Display Bears line, default value: true
showCombined - Display Combined line, default value: true
styles:
colorBull - Bulls line color, default value: '#8bc344'
boldBull - Bulls line bold, default value: false
colorBear - Bears line, default value: '#d55744'
boldBear - Bears line bold, default value: false

Donchian Channel

name: 'DonchianChannel'

settings:

n - Length, default value: 15

styles:

colorBasis - Bass line color, default value: '#ee7d00'

boldBasis - Bass line bold, default value: false

colorBorder - Borders color, default value: '#ffcb00'

boldBorder - Borders bold, default value: false

colorBg - Background color, default value: 'rgba(255,202,0,0.2)'

Envelope

name: 'Envelope'

settings:

n - Length, default value: 15

p - Percentage, default value: 5

styles:

colorMiddle - Middle line color, default value: '#ce2867'

boldMiddle - Middle line bold, default value: false

colorBorder - Borders, default value: '#f194b7'

boldBorder - Borders bold, default value: false

Exponential Moving Average

name: 'ExponentialMovingAverage'

settings:

n - Length, default value: 15

styles:

color - Line color, default value: '#eb609e'

bold - Line bold, default value: false

Fractals

name: 'Fractals'

settings:

n - Periods, default value: 2

fillMax - fill maxima arrows instead of contour, default value: false

fillMin - fill minima arrows instead of contour, default value: false

styles:

colorMax - Maxima arrows color, default value: '#8bc344'

colorMin - Minima arrows, default value: '#d55744'

Ichimoku Cloud

name: 'IchimokuCloud'

settings:

nConv - Conversion, default value: 9
nBase - Base, default value: 26
nSpan - Lagging Span, default value: 52
nDisp - Displacement, default value: 26

styles:

colorConv - Conversion line color, default value: '#e20612'
boldConv - Conversion line bold, default value: false
colorBase - Base line color, default value: '#65b22e'
boldBase - Base line bold, default value: false
colorLag - Lagging line color, default value: '#006ab4'
boldLag - Lagging line bold, default value: false
colorALine - Lead 1 color, default value: '#934b95'
boldALine - Lead 1 bold, default value: false
colorBLine - Lead 2 color, default value: '#ef7d00'
boldBLine - Lead 2 bold, default value: false
colorBg1 - Background 1 color, default value: 'rgba(226,6,18,0.2)'
colorBg2 - Background 2 color, default value: 'rgba(101,178,46,0.3)'

Momentum

name: 'Momentum'

settings:

n - Length, default value: 15

styles:

color - Line color, default value: '#eb609e'
bold - Line bold, default value: false

Moving Average Convergence Divergence

name: 'MovingAverageConvergenceDivergence'

settings:

nFast - Fast, default value: 12
nSlow - Slow, default value: 26
nSignal - Signal, default value: 9

styles:

colorHistogram - Histogram color, default value: '#944b96'
colorMACD - MACD line color, default value: '#65b22e'
boldMACD - MACD line bold, default value: false
colorSignal - Signal line color, default value: '#40c4ef'
boldSignal - Signal line bold, default value: false

Pivot Points

name: 'PivotPoints'

styles:

color - Levels color, default value: '#016ab3'
bold - Levels bold, default value: false

Relative Strength Index

name: 'RelativeStrengthIndex'

settings:

n - Length, default value: 15

overbought - Overbought, default value: 70

oversold - Oversold, default value: 30

styles:

color - Line color, default value: '#65b22e'

bold - Line bold, default value: false

colorBg - Background color, default value: 'rgba(174,208,131,0.2)'

Simple Moving Average

name: 'SimpleMovingAverage'

settings:

n - Length, default value: 15

styles:

color - Line color, default value: '#ee7d00'

bold - Line bold, default value: false

Standard Deviation

name: 'StandardDeviation'

settings:

n - Length, default value: 15

styles:

color - Line color, default value: '#3ab5f9'

bold - Line bold, default value: false

Stochastic Oscillator

name: 'StochasticOscillator'

settings:

n - parameter K, default value: 15

upper - Upper Band, default value: 80

d - parameter D, default value: 3

lower - Lower Band, default value: 20

k - Smooth, default value: 3

styles:

colorK - K line color, default value: '#e20613'

boldK - K line bold, default value: false

colorD - D line color, default value: '#ee7d00'

boldD - D line bold, default value: false

colorBg - Background color, default value: 'rgba(255,202,0,0.2)'

Williams Alligator

name: 'WilliamsAlligator'
settings:
nJaw - Jaw Length, default value: 21
offsetJaw - Jaw Offset, default value: 8
nTeeth - Teeth Length, default value: 13
offsetTeeth - Teeth Offset, default value: 5
nLips - Lips Length, default value: 8
offsetLips - Lips Offset, default value: 3
styles:
colorJaw - Jaw line color, default value: '#aed083'
boldJaw - Jaw line bold, default value: false
colorTeeth - Teeth line color, default value: '#e20613'
boldTeeth - Teeth line bold, default value: false
colorLips - Lips line color, default value: '#006ab4'
boldLips - Lips line bold, default value: false

Zig Zag
name: '*ZigZag*'
settings:
p - Percentage, default value: 5
styles:
color - Line color, default value: '#9552c1'
bold - Line bold, default value: false

Indicator object methods:

indicator.name()
Returns indicator name.

indicator.param(key, <value>)
Sets or returns (if *value* is not specified) specified indicator's parameter.

indicator.params(<items>)
Sets or returns (if *items* is not specified) indicator's parameters.

indicator.drop()
Removes indicator from the chart.

Drawings API:

Adding drawings via Chart API is similar to adding indicators: you need to call chart.addDrawing method and pass drawing name and all parameters it requires. In addition to that however it is required to specify points data.

chart.addDrawing(name, parameters, <index>)

Parameter *index* is used to specify chart panel in multiscreen mode. By default active chart panel is used.

Example:

```
// Let's get actual timestamp and value of current asset
let {timestamp, value} = chart.getLastClose();
// We're going to add GannFan and therefore 2 points required
let points = [
    {
        timestamp: timestamp - 7200, // 7200 seconds = 2 hours
        value: value * 0.9999,
    },
    {
        timestamp: timestamp,
        value: value,
    }
];
chart.addDrawing('GannFan', {
    points: points,
    colorLine11: '#df1120',
    boldLine11: true,
    colorLine12: '#0f6cb1',
    boldLine12: false,
    colorLine13: '#ec7d21',
    boldLine13: false,
    colorLine14: '#934e95',
    boldLine14: false,
    colorLine18: '#67b138',
    boldLine18: true,
    colorBg: 'rgba(130,138,155,0.2)'
});
```

Important! All color parameters must be valid CSS color string values!

Here is the list of all available drawings at the moment:
*Note that default parameters values listed below are set as default only when you add drawings via chart UI. In case of API call you **must** specify all listed parameters and points data.*

ABCD Pattern

name: 'ABCDPattern'
points required: 4
styles:

color - Lines color, default value: '#df1120'
bold - Lines bold, default value: false
colorBg - Background color, default value: 'rgba(130,138,155,0.2)'

Arrow
name: 'Arrow'
points required: 2
styles:
colorBg - Background color, default value: 'rgba(221,23,41,0.2)'

Circle
name: 'Circle'
points required: 2
styles:
color - Stroke color, default value: '#df1120'
bold - Stroke bold, default value: false
colorBg - Background color, default value: 'rgba(130,138,155,0.2)'

Fibonacci Arcs
name: 'FibonacciArcs'
points required: 2
styles:
colorLevel0236 - Level 0.236 color, default value: '#eb202d'
boldLevel0236 - Level 0.236 bold, default value: false
colorLevel0382 - Level 0.382 color, default value: '#64b12d'
boldLevel0382 - Level 0.382 bold, default value: false
colorLevel05 - Level 0.5 color, default value: '#eb5f9e'
boldLevel05 - Level 0.5 bold, default value: false
colorLevel0618 - Level 0.618 color, default value: '#ffca00'
boldLevel0618 - Level 0.618 bold, default value: false
colorLevel0786 - Level 0.786 color, default value: '#00afea'
boldLevel0786 - Level 0.786 bold, default value: false
colorLevel1 - Level 1 color, default value: '#eb202d'
boldLevel1 - Level 1 bold, default value: false
colorLevel1618 - Level 1.618 color, default value: '#006ab4'
boldLevel1618 - Level 1.618 bold, default value: false
colorLevel2618 - Level 2.618 color, default value: '#ed7c00'
boldLevel2618 - Level 2.618 bold, default value: false
colorBg - Background color, default value: 'rgba(130,138,155,0.2)'

Fibonacci Retracement
name: 'FibonacciRetracement'
points required: 2
styles:

colorLevel0 - Level 0 color, default value: '#df1120'
boldLevel0 - Level 0 bold, default value: false
colorLevel0236 - Level 0.236 color, default value: '#eb202d'
boldLevel0236 - Level 0.236 bold, default value: false
colorLevel0382 - Level 0.382 color, default value: '#64b12d'
boldLevel0382 - Level 0.382 bold, default value: false
colorLevel05 - Level 0.5 color, default value: '#eb5f9e'
boldLevel05 - Level 0.5 bold, default value: false
colorLevel0618 - Level 0.618 color, default value: '#ffca00'
boldLevel0618 - Level 0.618 bold, default value: false
colorLevel0786 - Level 0.786 color, default value: '#00afea'
boldLevel0786 - Level 0.786 bold, default value: false
colorLevel1 - Level 1 color, default value: '#eb202d'
boldLevel1 - Level 1 bold, default value: false
colorLevel1618 - Level 1.618 color, default value: '#006ab4'
boldLevel1618 - Level 1.618 bold, default value: false
colorLevel2618 - Level 2.618 color, default value: '#ed7c00'
boldLevel2618 - Level 2.618 bold, default value: false
colorBg - Background color, default value: 'rgba(130,138,155,0.2)'

Forecast

name: 'Forecast'
points required: 2
styles:
colorSuccess - Success color, default value: '#64b12d'
colorFailure - Failure color, default value: '#eb202d'
opacity - Background opacity, default value: 0.2

Gann Box

name: 'GannBox'
points required: 2
styles:
colorLevelH0 - Horizontal Level 0 color, default value: '#df1120'
boldLevelH0 - Horizontal Level 0 bold, default value: false
colorLevelV0 - Vertical Level 0 color, default value: '#df1120'
boldLevelV0 - Vertical Level 0 bold, default value: false
colorLevelH025 - Horizontal Level 0.25 color, default value: '#0f6cb1'
boldLevelH025 - Horizontal Level 0.25 bold, default value: false
colorLevelV025 - Vertical Level 0.25 color, default value: '#0f6cb1'
boldLevelV025 - Vertical Level 0.25 bold, default value: false
colorLevelH0382 - Horizontal Level 0.382 color, default value: '#ec7d21'
boldLevelH0382 - Horizontal Level 0.382 bold, default value: false
colorLevelV0382 - Vertical Level 0.382 color, default value: '#ec7d21'
boldLevelV0382 - Vertical Level 0.382 bold, default value: false

colorLevelH05 - Horizontal Level 0.5 color, default value: '#67b138'
boldLevelH05 - Horizontal Level 0.5 bold, default value: false
colorLevelV05 - Vertical Level 0.5 color, default value: '#67b138'
boldLevelV05 - Vertical Level 0.5 bold, default value: false
colorLevelH0618 - Horizontal Level 0.618 color, default value: '#1db1e8'
boldLevelH0618 - Horizontal Level 0.618 bold, default value: false
colorLevelV0618 - Vertical Level 0.618 color, default value: '#1db1e8'
boldLevelV0618 - Vertical Level 0.618 bold, default value: false
colorLevelH075 - Horizontal Level 0.75 color, default value: '#934e95'
boldLevelH075 - Horizontal Level 0.75 bold, default value: false
colorLevelV075 - Vertical Level 0.75 color, default value: '#934e95'
boldLevelV075 - Vertical Level 0.75 bold, default value: false
colorLevelH1 - Horizontal Level 1 color, default value: '#de1020'
boldLevelH1 - Horizontal Level 1 bold, default value: false
colorLevelV1 - Vertical Level 1 color, default value: '#de1020'
boldLevelV1 - Vertical Level 1 bold, default value: false
colorBg - Background color, default value: 'rgba(130,138,155,0.2)'

Gann Fan
name: 'GannFan'
points required: 2
styles:
colorLine11 - Line 1/1 color, default value: '#df1120'
boldLine11 - Line 1/1 bold, default value: false
colorLine12 - Line 1/2 color, default value: '#0f6cb1'
boldLine12 - Line 1/2 bold, default value: false
colorLine13 - Line 1/3 color, default value: '#ec7d21'
boldLine13 - Line 1/3 bold, default value: false
colorLine14 - Line 1/4 color, default value: '#934e95'
boldLine14 - Line 1/4 bold, default value: false
colorLine18 - Line 1/8 color, default value: '#67b138'
boldLine18 - Line 1/8 bold, default value: false
colorBg - Background color, default value: 'rgba(130,138,155,0.2)'

Head and Shoulders
name: 'HeadAndShoulders'
points required: 7
styles:
color - Lines color, default value: '#df1120'
bold - Lines bold, default value: false
colorBg - Background color, default value: 'rgba(130,138,155,0.2)'

Horizontal Line
name: 'HorizontalLine'

points required: 1
settings:
level - Price level, default value: calculated after point placement
styles:
color - Line color, default value: '#df1120'
bold - Line bold, default value: false

Line
name: 'Line'
points required: 2
styles:
color - Line color, default value: '#df1120'
bold - Line bold, default value: false

Parallel Channel
name: 'ParallelChannel'
points required: 3
styles:
color - Lines color, default value: '#df1120'
bold - Lines bold, default value: false
colorBg - Background color, default value: 'rgba(130,138,155,0.2)'

Pitchfork
name: 'Pitchfork'
points required: 3
styles:
color - Base line color, default value: '#df1120'
bold - Base line bold, default value: false
colorMiddle - Middle lines color, default value: '#fa7f54'
boldMiddle - Middle lines bold, default value: false
colorBorder - Border lines color, default value: '#ffd35b'
boldBorder - Border lines bold, default value: false
colorBg - Background color, default value: 'rgba(130,138,155,0.2)'

Text
name: 'Text'
points required: 2
settings:
text - Text to display, default value: ''
styles:
color - Text color, default value: '#df1120'
wrap - Wrap text using boundaries, default value: true
bold - Bold text, default value: false
fontSize - Font size, default value: 12

colorBg - Background color, default value: 'rgba(130,138,155,0)'
align - Text alignment, default value: 'left', supported values: 'left', 'center', 'right'

Trade
name: 'Trade'
points required: 3
styles:
colorEntry - Entry point color, default value: '#1cb5ee'
boldEntry - Entry point bold, default value: false
colorProfit - Take profit color, default value: '#65b230'
boldProfit - Take profit bold, default value: false
colorLoss - Stop loss color, default value: '#e20613'
boldLoss - Stop loss bold, default value: false

Triangle
name: 'Triangle'
points required: 3
styles:
color - Stroke color, default value: '#df1120'
bold - Stroke bold, default value: false
colorBg - Background color, default value: 'rgba(130,138,155,0.2)'

Vertical Line
name: 'VerticalLine'
points required: 1
styles:
color - Line color, default value: '#df1120'
bold - Line bold, default value: false

Drawing object methods:

drawing.name()
Returns drawing name.

drawing.points(<points>)
Sets or returns (if *points* is not specified) drawing's points.
Each point is represented by Object with *timestamp* and *value* keys:
{
timestamp: [int],
value: [double]
}
Where *timestamp* is unix timestamp.

drawing.param(key, <value>)
Sets or returns (if *value* is not specified) specified drawing's parameter.

drawing.params(<items>)
Sets or returns (if *items* is not specified) drawing's parameters.

drawing.drop()
Removes drawing from the chart.

Deal object API:

After deal object created with method *chart.addDeal* you can manipulate it via Deal API.

General methods:

deal.symbol(<symbol>)
Sets or returns (if *symbol* is not specified) deal symbol.

deal.timestamp(<timestamp>)
Sets or returns (if *timestamp* is not specified) deal timestamp.

deal.type(<type>)
Sets or returns (if *type* is not specified) deal type.
Valid type values: *'buy'*, *'sell'*.

deal.entry(<entry>)
Sets or returns (if *entry* is not specified) deal entry point.
If *takeProfit* or *stopLoss* is already set then changing entry point will affect TP/SL values as well:
delta = previousEntry - entry
newSL = previousSL + delta
newTP = previousTP + delta

deal.timestamp(<timestamp>)
Sets or returns (if *timestamp* is not specified) deal timestamp.

deal.takeProfit(<takeProfit>)
Sets or returns (if *takeProfit* is not specified) deal *takeProfit* value. Limited with entry point value depending on deal type.
Clears *takeProfit* value if *takeProfit* is null.

deal.stopLoss(<stopLoss>)

Sets or returns (if *stopLoss* is not specified) deal *stopLoss* value. Limited with entry point value depending on deal type.
Clears *stopLoss* value if *stopLoss* is null.

deal.drop()

Removes deal from the chart.

Informational purpose methods:

Next methods affect only data on display which is not bound with actual deal parameters.

deal.title(<title>)

Sets or returns (if *title* is not specified) deal title (appears on the left side on deal selection).
Default value is ''.

deal.net(<net>)

Sets or returns (if *net* is not specified) deal net value title. Also affects deal *color* depending on sign "-" in net value string.
Default value is ''.

deal.profit(<profit>)

Sets or returns (if *profit* is not specified) deal profit title (appears on the right side on deal selection).
Default value is *null*.

deal.loss(<loss>)

Sets or returns (if *loss* is not specified) deal loss title (appears on the right side on deal selection).
Default value is *null*.

deal.close(timestamp)

Marks deal as closed (additional elements will be added on profit and loss lines).
Returns Deal object.

deal.profitLabel(<text>)

Sets or returns (if *text* is not specified) take profit label text. If *text* is *null* then take profit line and label will be hidden.
Default value is 'PROFIT'.

deal.lossLabel(<text>)

Sets or returns (if *text* is not specified) stop loss label text. If *text* is *null* then take profit line and label will be hidden.
Default value is 'LOSS'.

deal.color(<color>)

Sets or returns (if *color* is not specified) deal color. This parameter is also affected by *deal.net* method.
Default value is '#00000'.

deal.colorProfit(<color>)

Sets or returns (if *color* is not specified) take profit line color. If *color* is *null* then color set by deal.color() is used.
Default value is *null*.

deal.colorLoss(<color>)

Sets or returns (if *color* is not specified) stop loss line color. If *color* is *null* then color set by deal.color() is used.
Default value is *null*.

Deal events:

There are few deal events that can be handled with special method *deal.on*:

deal.on(event, handler)

Sets event handler for specified *event*. Available values for *event* are:

- *closeTakeProfit* - fired on X-button click near PROFIT label (which appears if event handler is set)
- *closeStopLoss* - fired on X-button click near LOSS label (which appears if event handler is set)
- *closeDeal* - fired on X-button click near deal label
- *changeTakeProfit* - fired on moving PROFIT label via mouse or touch
- *setTakeProfit* - fired on end of moving PROFIT label via mouse or touch
- *changeStopLoss* - fired on moving LOSS label via mouse or touch
- *setStopLoss* - fired on end of moving LOSS label via mouse or touch
- *expand* - fired on deal selection
- *collapse* - fired on deal deselection

Operator *this* inside handler function will point to the deal.

Examples:

```
deal.on('closeTakeProfit', function () {
    this.takeProfit(null);
});
```

Remove deal take profit on X-button click near PROFIT label.

```
deal.on('closeStopLoss', function () {
    this.stopLoss(null);
});
```

Remove deal stop loss on X-button click near LOSS label.

```
deal.on('closeDeal', function () {
    this.drop();
});
```

Remove deal on X-button click near deal title.

```
deal.on('changeTakeProfit', function () {
    this.profit(this.takeProfit() * 10);
});
```

Change profit label to x10 value of takeProfit on takeProfit change.

```
deal.on('changeStopLoss', function () {
    this.loss(this.stopLoss() * 10);
});
```

Change loss label to x10 value of stopLoss on stopLoss change.

Data Management API

Data management API is accessible through the *data* property of chart instance object.

```
myChart.data
```

Specifying Available Symbols:

Chart allows you to search and select active symbol from the list in toolbar. In order to make it possible the first thing you have to do after you've initialized the chart instance is to specify available symbols. You can do that by calling method

setSymbols:

data.setSymbols(symbols)
Where *symbols* is an array of strings.

Example:

```
let symbols = ['EURUSD', 'EURCAD', 'GBPUSD', 'GBPCAD'];
myChart.data.setSymbols(symbols);
```

Specifying Trading Hours:

Trading hours are used by chart for two things:

- timestamps interpolation outside loaded data;
- proper handling new rates when using *addRate* | *addRates* methods.

In most cases it is essential to specify trading hours for symbol before any data will be loaded. It is recommended to specify trading hours for all assets right after chart initialization.

data.setSchedule(symbol, schedule, offset)
Set trading hours for specified symbol.
Where *schedule* is an array of time intervals (in minutes). And *offset* is amount of minutes starting from 12:00 AM used to align candles. For example if you have market opening at 9:00 PM GMT and you need every daily candle to be opened at this time then set *offset* to 1260 (1260 minutes equals 9:00 PM).

Each time interval is defined by an object: {
start: <Integer>,
end: <Integer>
}
Where *start* and *end* values are amount of minutes starting from the beginning of the week (Sunday 12:00 AM).

Example:

```
/*
Let's create schedule for some asset with daily session from
12:00 PM to 19:00 PM every day except Saturday and Sunday
*/
let intervals = [];
let dayStart = 720; // equals 12:00 PM
let dayEnd = 1140; // equals 19:00 PM
// Days of week: 0 - Sun, 1 - Mon, 2 - Tue, 3 - Wed, 4 - Thu,
5 - Fri, 6 - Sat
```

```
for (let dayOfWeek = 1; dayOfWeek <= 5; dayOfWeek++) {
    intervals.push(
        start: dayStart + dayOfWeek * 1440, // 1440 - minutes
        in day
        end: dayEnd + dayOfWeek * 1440
    );
}
myChart.data.setSchedule('SYMBOL', intervals, dayStart);
```

data.setSchedules(schedules)

Set trading hours for multiple symbols.
Where *schedules* is an array of objects: [{
symbols: <String>[],
intervals: <Object>[],
offset: <Integer>
}]

Note that you specify multiple symbols for each schedule when using *setSchedules* method.

Default trading hours are set as single interval from 0 to 1440 * 7 (end of week) with offset = 0.

Setting Candles Data:

There are two methods allowing to directly set candles on the chart: *setCandle* and *setCandles*. Keep in mind that when multi-chart layout is enabled it is possible that the same symbol / timeframe settings will be applied to different charts. That is why you do not set chart panel explicitly but specifying symbol and timeframe instead. Chart will automatically add candles for all charts with the same symbol and timeframe.

Note: If there is no chart with specified symbol and timeframe then request will be ignored.

data.setCandle(symbol, timeframe, timestamp, open, high, low, close, [volume])

Set candle on the chart(s).
Where *timestamp* is unix timestamp and *volume* is optional.

Example:

```
// Get unix timestamp and align it with five-minute intervals
let current5M = Math.floor(Date.now() / 1000 % 300);
myChart.data.setCandle('EURUSD', '5M', current5M, 1.1, 1.105,
1.091, 1.102, 217.853);
```

data.setCandles(candles)

Set multiple candles.

Where *candles* is an array of objects: [{

symbol: <String>,

timeframe: <String>,

timestamp: <Integer>,

open: <Double>,

high: <Double>,

low: <Double>,

close: <Double>,

volume: <Double> *// optional*

}]

Adding Rates:

Usually you don't want to recalculate candles manually every time you get new rates from server. For that purpose you have two methods allowing you to pass rates directly: *addRate* and *addRates*. According to trading hours specified by *setSchedules* method chart will automatically calculate what candles are affected by new rates and change them or build new ones if necessary.

Note: Adding rates will work properly only in case if you correctly specified trading hours.

data.addRate(symbol, timestamp, rate, [volume])

Update chart candles data with new rate accordingly to trading hours.

Where *timestamp* is unix timestamp and *volume* is optional.

```
let now = Math.floor(Date.now() / 1000);
myChart.data.addRate('EURUSD', now, 1.105, 10);
// This will update candles which specified timestamp belongs
to
```

data.addRates(rates)

Update chart candles data with new rates accordingly to trading hours.

Where *candles* is an array of objects: [{

symbol: <String>,

timestamp: <Integer>,

rate: <Double>,

volume: <Double> *// optional*

}]

Handling Candles Request:

Due to performance optimizations it is not recommended to load all historical data at once. Chart can automatically calculate how many candles it needs to load and display and fire corresponding event. This event will be fired both on symbol / timeframe change or initial set and on chart panning and zooming.

In order to handle this event you need to use *addEventHandler* method of chart API:

```
myChart.addEventHandler('onCandlesRequest', function (symbol,
timeframe, timestampFrom, timestampTo, count, callback) {
    /*
    When timestamps are not specified - that means this is
    initial request and there are no candles loaded on the chart
    yet. You're supposed to load <count> latest candles.
    Otherwise you need to load candles according either to
    <timestampFrom> and <timestampTo> parameters (if timestampTo
    is set) or <timestampFrom> and <count> parameters.
    */
    // Here is your custom load-candles function which (in this
    particular example) returns Promise
    loadCandlesFromServer(symbol, timeframe, timestamp,
    count).then(function (loadedCandles) {
        callback(loadedCandles); // callback function will add
        loaded candles
    });
});
```

Note: If you're handling this event then you do not need to load initial historical data explicitly since this event will be triggered both on chart load and symbol / timeframe change.