

---

# 目錄

序言	1.1
更新日志	1.2
1、Charting Library是什么	1.3

## 2、入门指南

2-1、图表库内容	2.1
2-2、运行图表库	2.2

## 3、数据绑定

3-1、如何连接我的数据	3.1
3-2、JS Api	3.2
3-3、UDF	3.3
3-4、Symbology	3.4
3-5、交易时段	3.5
3-6、报价	3.6

## 4、图表定制

4-1、定制概述	4.1
4-2、Widget构造器	4.2
4-3、Widget方法	4.3
4-4、图表方法	4.4
4-5、功能集	4.5
4-6、服务端定制	4.6
4-7、定制的使用案例	4.7

## 5、交易终端

---

5-1、交易终端简介	5.1
5-2、交易控制器	5.2
5-3、经纪商API	5.3
5-4、交易主机	5.4
5-5、账户管理器	5.5
5-6、交易对象和常量	5.6
6、储存和载入图表	6.1
7、创建自定义研究	6.2
7、最佳做法	6.3
9、经常被问到的问题	6.4
10、版本变更点	6.5

## 附录

分辨率	7.1
时间范围	7.2
本地化	7.3
覆盖	7.4
绘图覆盖	7.5
研究覆盖	7.6
形状与覆盖	7.7
订阅	7.8
交易元语	7.9
在K线上做标记	7.10
委托	7.11
WatchedValue	7.12

# TradingView 中文开发文档 V1.1

本书翻译自官方wiki：[https://github.com/tradingview/charting\\_library/wiki](https://github.com/tradingview/charting_library/wiki)

未申请license的同学，点击上面官方wiki会报404错误：)

TradingView为优秀的交易技术分析金融图表，拥有丰富的技术指标库，并拥有直接交易的交易终端插件。

## 本项目地址

- 仓库：<https://github.com/zlq4863947/tradingViewWikiCn>
- 在线阅读：<https://zlq4863947.gitbooks.io/tradingview/>

## 开发交流QQ群

- tradingview开发：313839516

# 更新日志

## 1.1 -- 20171223

- 图片文字汉化
- 链接与内容修正
- 官方wiki同步更新

## 1.0 -- 20170910

- 初版做成

## 0.1 ~ 0.9 -- 20???????

- 膜拜大神

膜拜



# Charting Library是什么


具有开放数据API并可下载的图表控件。这是一个独立的解决方案，您可以下载，托管在您的服务器上，连接您自己的数据，并在您的网站/应用程序中免费使用。您所要做的是:

步骤	您会得到什么
1. <code>git clone</code> 下载Charting Library并且运行它	在主机上运行的图表的例子
2. 使用我们的API将您的数据插入图表库。您可以参考这些例子	部署并运行图表库并且加载您的数据

如果您想定制您的图表，那么您可以更进一步。

- 查询图表库 [定制概述](#) 和 [定制用例](#)
- [创建](#)自定义研究

## 拥有交易终端？

 交易终端是为强大图表配备的随时可用的产品，使图表具备交易功能。[阅读更多](#)

## 最佳实践

阅读这篇[文章](#)会让您避免常见的错误，节约您的时间。

## 图表库内容

Charting Library 图表库包可在GitHub上获得（必须获得授权才能访问GitHub上的这个私有资源）。您可以检出最新的稳定版本( `master branch`)或最新的开发版本( `unstable branch`)。如果想要访问此资源请联系我们。

您可以通过在浏览器控制台中执行 `TradingView.version()` 来查看图表库版本。

### 图表库内容

```
+ /charting_library
  - charting_library.min.js
+ /datafeed
+ /udf
  - datafeed.js
+ /static
- index.html
```

- `/charting_library` 包含所有的库文件。
- `/charting_library/charting_library.min.js` 包含Charting Library widget 接口。不建议修改该文件。
- `/charting_library/datafeeds/udf/datafeed.js` 包含UDF-compatible 的datafeed包装器（用于实现JS API以连接Charting Library和UDF连接datafeed）。例子中的datafeed包装器实现了脉冲实时仿真数据。您可以自由编辑此文件。
- `/charting_library/static` 文件夹中存储图表库内部资源，不适用于其他目的。
- `/index.html` 为使用Charting Library widget 的html例子。
- `/test.html` 为不同的图表库自定义功能使用的示例。
- `/mobile*.html` 也是Widget自定义的示例。

从版本1.1开始，所有内部库的JS和CSS代码都被内联和缩小，以减少页面加载时间。您要编辑的文件不会被缩小。

## 运行图表库

---

图表库是开箱即用的，但您必须设置HTTP服务器才能将库的文件夹绑定到某个域名。

即，您可以设置Apache监听任何主机的空闲端口，然后复制图表库的文件夹到服务器目录。

您可以在[这里](#)看到图表库的工作示例。此示例使用datafeed处理服务器运行在[这里](#)。

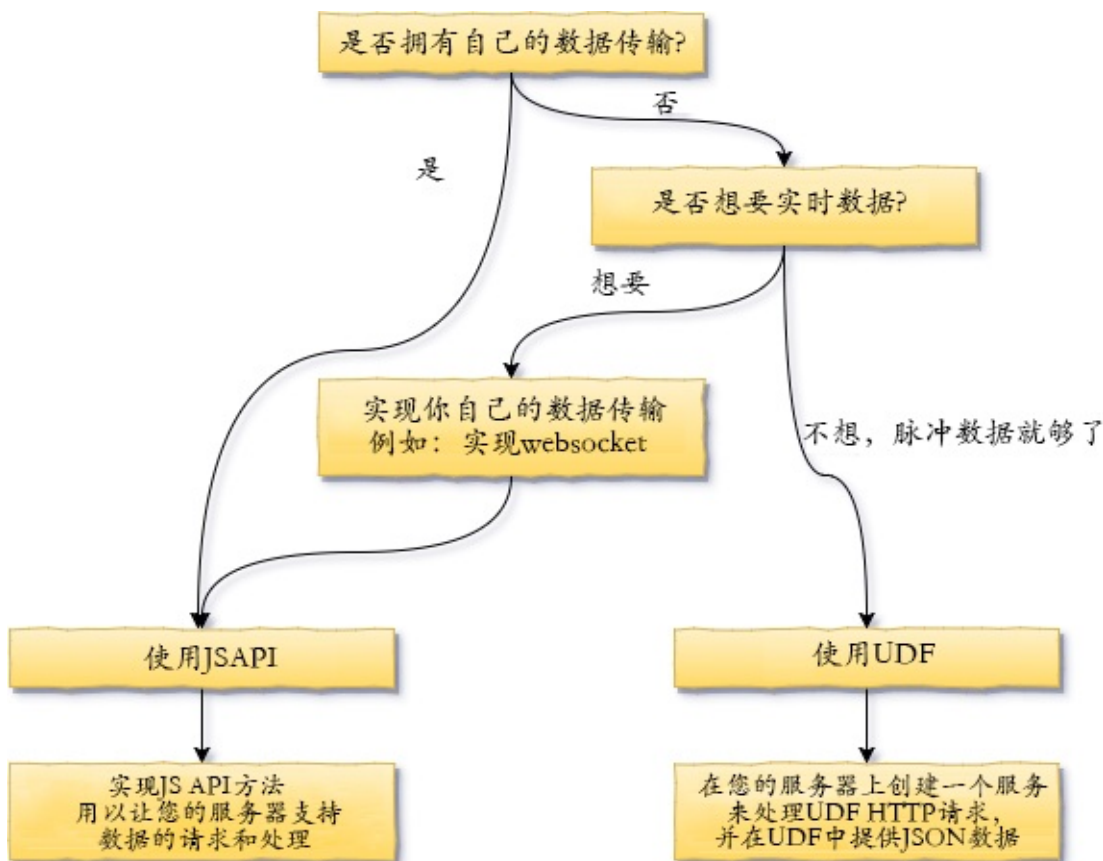
重要: 这个datafeed只是个示例。它只包含了十几个股票数据（从雅虎获得），仅提供DWM。然而，它支持报价。请使用它仅用于测试目的。

## 如何连接我的数据

图表库并不包含市场数据，你必须提供所需格式的数据。示例数据引擎集成了雅虎金融API的历史数据。图表可以用两种方式接收数据：

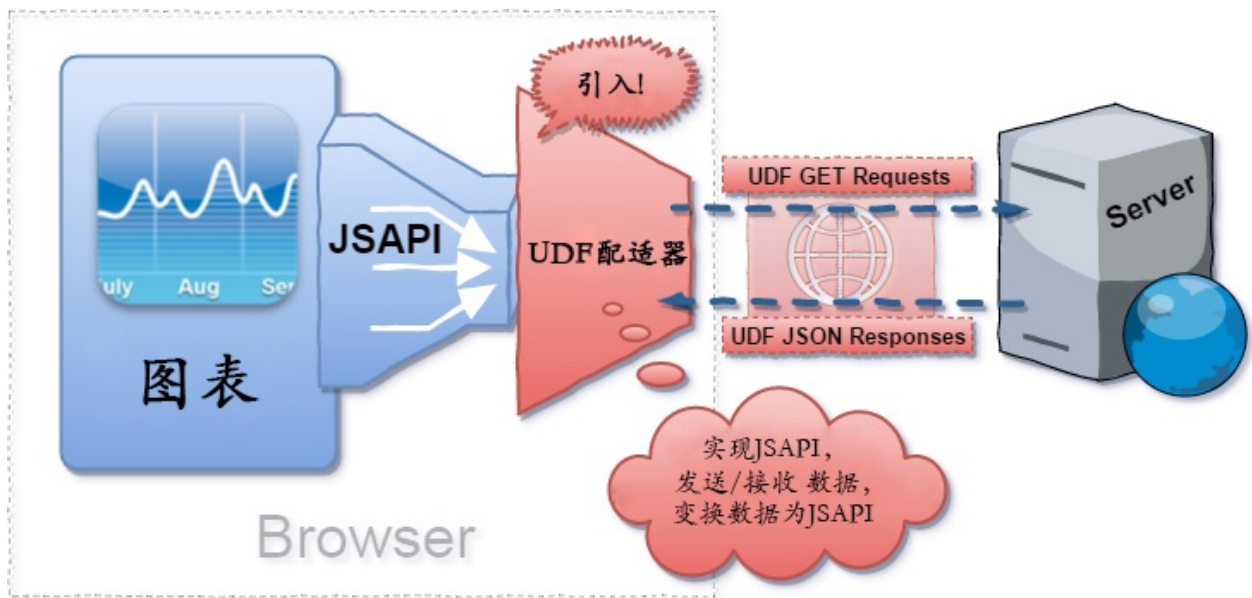
1. 使用服务器推送技术实时更新，例如通过WebSocket。这样你的图表将会自动更新价格。为了达到这个目的，你必须使用JavaScript API并且准备好自己的传输方法。
2. 以PULL /脉冲(pulse)/刷新为基础进行更新（如今天的大多数基于Web的图表），其中图表数据每X秒更新一次（图表客户端将要求服务器模拟PUSH更新），或者被用户手动重新加载。为此，请使用UDF协议并编写自己的datafeed包装器。

### JavaScript API 或者 UDF?

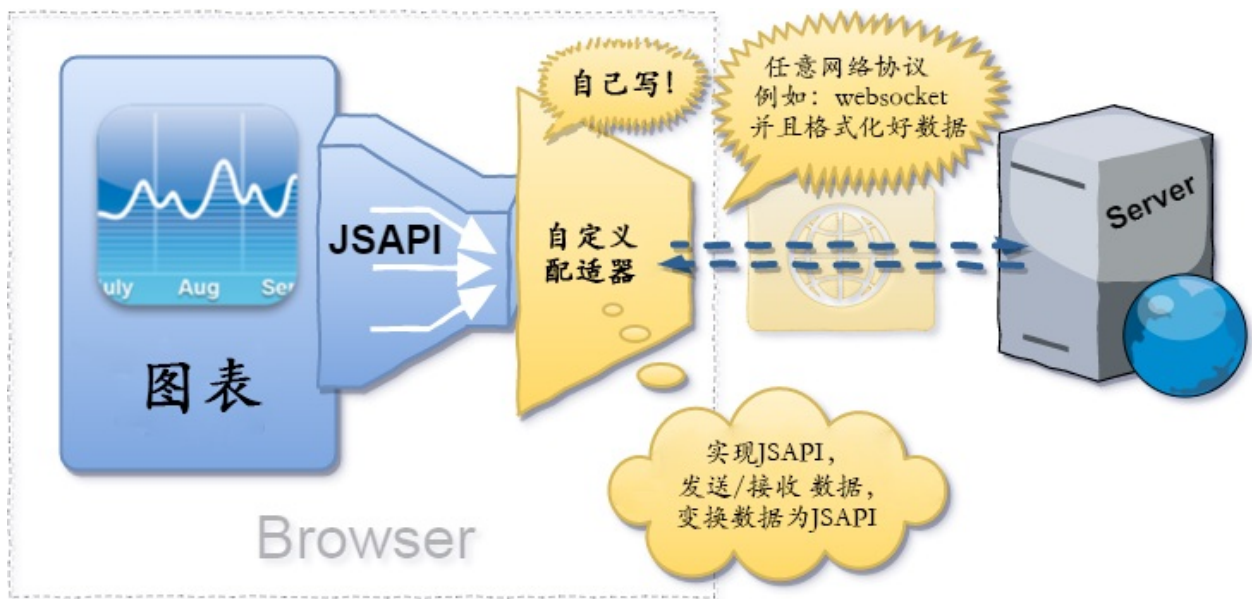


### UDF scheme





## JSAPI scheme



上图显示了UDF和JSAPI之间的区别。图表库必须的部分为蓝色。红色的部分（默认数据传输）包含在默认包中（具有非最小化的源代码），并可以被替换。您可能会看到默认数据传输实现JS API与图表交互。也可以使用默认传输实现UDF协议与服务器通信。

1. 如果您已经准备好了数据传输（websocket流式传输，轮询或任何其他传输），或者如果您不需要流式传输数据 - 请使用我们的JavaScript API，这是非常紧凑和简单的实现。您必须创建一个小型客户端数据适配器来传输数据到我们的图表。
2. 如果您没有任何传输，并且不需要数据流传输数据（例如，您所需要的数据脉冲），那么您将必须创建（或使用）至少一个服务器端的数据feed包装器。您可以使用任何语言和技术来实现这一目的：您的包装器只需要支持我们的数据交换协议（我们称之为

为UDF)，以便能够为您的图表提供数据。您将不得不使用您最喜欢的语言在您的后端和我们的图表之间创建一个小型服务端数据适配器。

## 想要例子？

一个例子实现UDF-compatible(case #2 described above) 服务器端包装器可以在[这个](#)连接中获得。它使用Yahoo!数据。

一个例子实现**JS API**(和UDF客户端时间一致) 为Charting Library 包的一部分 (查看datafeed.js文件).

# JS Api

这是啥？一套JS方法（特定的公共接口）。



我该怎么使用它？：您应该创建一个JS对象，它将以某种方式接收数据，并响应Charting Library请求。

数据缓存（历史和股票信息）在图表库中实现。当你创建一个实现描述接口的对象时，只需通过 `datafeed` 参数将其传递给图表库Widget的构造函数。

## Methods

1. `onReady`
2. `searchSymbolsByName`
3. `resolveSymbol`
4. `getBars`
5. `subscribeBars`
6. `unsubscribeBars`
7. `calculateHistoryDepth`
8. `getMarks`
9. `getTimescaleMarks`
10. `getServerTime`

 交易终端专属：

1. `getQuotes`
2. `subscribeQuotes`
3. `unsubscribeQuotes`
4.  `subscribeDepth`
5.  `unsubscribeDepth`

## onReady(callback)

```
1.callback: function(configurationData)
  i.configurationData: object (见下文)
```

此方法旨在提供填充配置数据的对象。这些数据会影响图表的行为，所以它被称为服务端定制。

Charting Library 要求您使用回调函数来传递datafeed的configurationData参数。

configurationData是一个对象，现在支持以下属性：

### exchanges

一个交易所数组。Exchange是一个对象 {value, name, desc} 。

value将被作为exchange参数传递给 searchSymbolsByName (见下文)。

exchanges= [] 会导致商品查询列表中看不到交易所过滤器。使用 value= "" 来创建通配符筛选器（所有的交易所）。

### symbols\_types

一个商品类型过滤器数组。该商品类型过滤器是个对象 {name, value} 。

value 将被作为 symbolType 参数传递给searchSymbolsByName 。

symbolsTypes = [] 会导致商品查询列表中看不到商品类型过滤器。使用 value = ""来创建通配符筛选器（所有的商品类型）。

### supported\_resolutions

一个表示服务器支持的分辨率数组，分辨率可以是数字或字符串。如果分辨率是一个数字，它被视为分钟数。字符串可以是“\*D”，“\*W”，“\_M”（\_的意思是任何数字）。格式化详细参照: [文章](#)。

'resolutions'=undefined 或 [] 时，分辨率拥有默认内容（见：<http://tradingview.com/e/>）。

例：[1, 15, 240, "D", "6M"] 您将在分辨率中得到 "1 分钟, 15 分钟, 4 小时, 1 天, 6 个月" 。

### supports\_marks

布尔值来标识您的 datafeed 是否支持在K线上显示标记。

### supports\_timescale\_marks

布尔值来标识您的 datafeed 是否支持时间刻度标记。

### supports\_time

将此设置为 true 假如您的datafeed提供服务器时间（unix时间）。它用于调整时间刻度上的价格比例。

## searchSymbolsByName(userInput, exchange, symbolType, onResultReadyCallback)

1. `userInput` :string，用户在商品搜索中输入的文字。
2. `exchange` :string，请求的交易所（由用户选择）。空值表示没有指定。
3. `symbolType` :string，请求的商品类型：指数、股票、外汇等等（由用户选择）。空值表示没有指定。
4. `onResultReadyCallback` :function(result)
  - i. `result` : 数组 (见下文)

方法介绍：提供一个匹配用户搜索的商品列表。`result` 为预期的商品，像下面这样：

```
[
  {
    "symbol": <商品缩写名>,
    "full_name": <商品全称 -- 例: BTCE:BTCUSD>,
    "description": <商品描述>,
    "exchange": <交易所名>,
    "ticker": <商品代码, 可选>,
    "type": "stock" | "futures" | "bitcoin" | "forex" | "index"
  }, {
    // .....
  }
]
```

如果没有找到商品，则应该使用空数组来调用回调。查看更多关于 `ticker` 值的细节 [在这里](#)

## resolveSymbol(symbolName, onSymbolResolvedCallback, onResolveErrorCallback)

1. `symbolName` :string 类型，商品名称 或 `ticker` if provided.
2. `onSymbolResolvedCallback` :function(SymbolInfo)
3. `onResolveErrorCallback` :function(reason)

方法介绍：通过商品名称解析商品信息(SymbolInfo)。

## getBars(symbolInfo, resolution, from, to, onHistoryCallback, onErrorCallback, firstDataRequest)

1. `symbolInfo` :SymbolInfo 商品信息对象
2. `resolution` :string （分辨率）
3. `from` :unix 时间戳, 最左边必须的K线时间
4. `to` :unix 时间戳, 最右边必须的K线时间
5. `onHistoryCallback` :function(数组 bars , meta ={ noData = false })
  - i. `bars` :Bar对象数组 {time, close, open, high, low, volume}[]
  - ii. `meta` :object {noData = true | false, nextTime - unix time}
6. `onErrorCallback` :function(reason : 错误原因)

7. `firstDataRequest` : 布尔值, 以标识是否第一次调用此商品/分辨率的历史记录。当设置为 `true` 时你可以忽略 `to` 参数 (这取决于浏览器的 `Date.now()`) 并返回K线数组直到当前K线 (包括它)。

方法介绍: 通过日期范围获取历史K线数据。图表库希望通过 `onHistoryCallback` 仅一次调用, 接收所有的请求历史。而不被多次调用。

发生不断自动刷新图表问题时, 请检查 `from` 与 `onHistoryCallback` 方法返回的 `bars` 时间是否一致, 没有数据时请返回 `noData = true`

`nextTime` 历史中下一个K线柱的时间。只有在请求的时间段内没有数据时, 才应该被设置。

`noData` 只有在请求的时间段内没有数据时, 才应该被设置。

**Remark:** `bar.time` 为以毫秒开始的Unix时间戳 (UTC标准时区)。

**Remark:** `bar.time` 对于日K线预期一个交易日 (未开始交易时) 以 00:00 UTC为起点。图表库会根据商品的交易 ([Session](#)) 时间进行匹配。

**Remark:** `bar.time` 对于月K线为这个月的第一个交易日, 除去时间的部分。

## subscribeBars(symbolInfo, resolution, onRealtimeCallback, subscriberUID, onResetCacheNeededCallback)

1. `symbolInfo` : object [SymbolInfo](#)
2. `resolution` : string 分辨率
3. `onRealtimeCallback` : function(bar)
  - i. `bar` : object {time, close, open, high, low, volume}
4. `subscriberUID` : object
5. `onResetCacheNeededCallback` (从1.7开始): function()将在bars数据发生变化时执行

方法介绍: 订阅K线数据。图表库将调用 `onRealtimeCallback` 方法以更新实时数据。

**Remark:** 当您调用 `onRealtimeCallback` 且K线时间等于最近一条K线时间时, 那么这条最近的K线将被您传入的K线所替换。例:

1. 最近一条K线为 {1419411578413, 10, 12, 9, 11}
2. 您的调用 `onRealtimeCallback({1419411578413, 10, 14, 9, 14})`
3. 图表库通过时间找出K线 1419411578413 已经存在, 并且是最近的那一个
4. 图表库替换K线, 因此现在最近一条K线为 {1419411578413, 10, 14, 9, 14}

**Remark 2:** 是否可以更新最近的K线或追加一条新的, 取决于 `onRealtimeCallback`。如果您调用此功能尝试更新历史记录中的一个K线, 则会收到错误消息。

**Remark 3:** 现在, 在图表接收到数据后, 没有办法改变历史上的K线。

## unsubscribeBars(subscriberUID)

1. `subscriberUID` :object

方法介绍：取消订阅K线数据。在调用 `subscribeBars` 方法时,图表库将跳过与 `subscriberUID` 相同的对象。

## calculateHistoryDepth(resolution, resolutionBack, intervalBack)

1. `resolution` :请求商品的分辨率
2. `resolutionBack` :期望历史周期刻度。支持的值: D | M
3. `intervalBack` :数量

方法介绍：算出历史数据周期刻度，使您能够重写所需的历史深度。

通过一些参数，让您知道要获得的是什么样数据。 以下是几个例子:

- `calculateHistoryDepth("D", "M", 12)` 调用: 图表库请求 12 个月的日线数据
- `calculateHistoryDepth(60, "D", 15)` 调用: 图表库请求15天的60分钟数据

如果你不想重写处理方法，这个函数应该返回 `undefined` 。如果你想要重写，它应该返回一个对象 `{resolutionBack, intervalBack}` 。

例子:

假设实现为

```
Datafeed.prototype.calculateHistoryDepth = function(resolution, resolutionBack, intervalBack) {
  if (period == "1D") {
    return {
      resolutionBack: 'M',
      intervalBack: 6
    };
  }
}
```

以上代码为当图表库将要求分辨率为 `1D` ，历史为6个月的深度。 在其他情况下，历史深度将具有其他默认值。

## getMarks(symbolInfo, startDate, endDate, onDataCallback, resolution)

1. `symbolInfo` :`SymbolInfo` 商品信息对象
2. `startDate` : unix 时间戳, 最左边必须的K线时间



3. `endDate` : unix 时间戳, 最右边必须的K线时间
4. `onDataCallback` : function(标记数字 `marks` )
5. `resolution` : string

方法介绍：获取可见K线范围的标记。图表预期每调用一次 `getMarks` 就会调用一次 `onDataCallback`。

`mark` 为具有以下属性的对象：

- **id**: 唯一标识id。当用户点击标记时，将传递给相应的回调: [respective callback](#)
- **time**: unix time, UTC
- **color**: `red | green | blue | yellow | { border: '#ff0000', background: '#00ff00' }`
- **text**: 标记弹出式文字。支持HTML
- **label**: 印在标记上的文字。单字符
- **labelFontColor**: label的文字颜色
- **minSize**: 标记的最小尺寸 (diameter, pixels)

每个K线允许几个标记（现在最多为10个）。不允许标记脱离K线。

**Remark:** 只有当您声明您的后端是支持标记时才会调用这个函数。 [supporting marks](#).

## getTimescaleMarks(symbolInfo, startDate, endDate, onDataCallback, resolution)

1. `symbolInfo` : [SymbolInfo](#) object
2. `startDate` : unix时间戳 (UTC). Leftmost visible bar's time.
3. `endDate` : unix时间戳 (UTC). Rightmost visible bar's time.
4. `onDataCallback` : function(array of `mark` s)
5. `resolution` : string

图表库调用此函数获取可见K线范围的时间刻度标记。图表预期您每个调用 `getTimescaleMarks` 会调用一次 `onDataCallback`。

`mark` 为具有以下属性的对象：

- **id**: 唯一标识id。当用户点击标记时，将传递给相应的回调: [respective callback](#)
- **time**: unix time, UTC
- **color**: `red | green | blue | yellow | ... | #000000`
- **label**: 印在标记上的文字。单字符
- **tooltip**: 字符串数组。数组的每个元素都是工具提示的单独行内容。

每个K线只允许一个标记。不允许标记脱离K线。

**Remark:** 只有当您声明您的后端是支持标记时才会调用这个函数。 [upporting marks](#).



## getServerTime(callback)

1. `callback` : `function(unixTime)`

当图表需要知道服务器时间时，如果配置标志 `supports_time` 设置为 `true`，则调用此函数。图表库预期只调用一次回调。所提供的时间没有毫秒。例子：`1445324591`。它是用来显示倒数的价格范围。

## 交易终端专属

## getQuotes(symbols, onDataCallback, onErrorCallback)

1. `symbols` : 商品名称数组
2. `onDataCallback` : `function(array of data )`
  - i. `data` : 商品报价数据
3. `onErrorCallback` : `function(reason)`

当图表需要报价数据时，将调用此函数。图表库预期在收到所有请求数据时调用 `onDataCallback`。

## subscribeQuotes(symbols, fastSymbols, onRealtimeCallback, listenerGUID)

1. `symbols` : 很少更新的商品数组（建议频率为每分钟一次）。这些商品在观察列表中，但它们目前不可见。
2. `fastSymbols` : 频繁更新的商品数组（一次在10秒或更快）
3. `onRealtimeCallback` : `function(array of data )`
  - i. `data` : 商品报价数据
4. `listenerGUID` : 监听的唯一标识符

交易终端当需要接收商品的实时报价时调用此功能。图表预期您每次要更新报价时都会调用 `onRealtimeCallback`。

## unsubscribeQuotes(listenerGUID)

1. `listenerGUID` : 监听的唯一标识符

交易终端当不需要再接收商品的实时报价时调用此函数。当图表库遇到 `listenerGUID` 相同的对象会跳过 `subscribeQuotes` 方法。

## subscribeDepth(symbolInfo, callback): String

1. `symbolInfo` : `SymbolInfo` object
2. `callback` : `function(depth)`
  - i. `depth` : object {`snapshot`, `asks`, `bids`}
    - i. `snapshot` : `Boolean` - 如果 `true` 时 `asks` 和 `bids` 具有全部深度，否则只包含更新的级别。
    - ii. `asks` : 买盘数组 {`price`, `volume`}
    - iii. `bids` : 卖盘数组 {`price`, `volume`}

交易终端当要接收商品的实时level 2 信息（DOM）时，调用此函数。图表预期您每次要更新深度数据时都会调用回调。

此方法应返回唯一标识（`subscriberUID`），用于取消订阅数据。

## `unsubscribeDepth(subscriberUID)`

1. `subscriberUID` : `String`

交易终端当不希望接收此监听时调用此函数。

# UDF

这是啥?: Universal Data Feed 图表库提供数据。  
我该怎么使用它?: 您应该创建小型的HTTP服务，让它从您的数据库中获取数据并响应图表库请求。

## 表式响应概念

Datafeed 响应通常可以被视为表。例如，关于交易所的商品列表的响应，可以被视为每个商品代表一行，并有存在一些列（minimal\_price\_movement，description，has\_intraday等）。每个列可以是一个数组（因此，它将为每个表的行提供单独的值）。但是当所有表的行具有相同的列值时，可能会出现这种情况：列的值可以是单独的JSON响应。

例如:

让我们假设我们已经请求了名称为NYSE(纽约证券交易所)的商品列表。响应（伪格式）可能像这样

```
{
  symbols: ["MSFT", "AAPL", "FB", "GOOG"],
  min_price_move: 0.1,
  description: ["Microsoft corp.", "Apple Inc", "Facebook", "Google"]
}
```

如果我们试图将这个回应设想成一张表，那么它就像这样

Symbol（商品代码）	min_price_move（最小价格变动）	描述
MSFT	0.1	Microsoft corp.
AAPL	0.1	Apple Inc
FB	0.1	Facebook
GOOG	0.1	Google

## API调用

### Datafeed 配置数据

Request: GET /config

Response: 图表库期望接收与JS API调用[setup\(\)](#)相同结构的JSON数据。此外，还应该有两个附加属性：

- **supports\_search**: 设置这一选项为 `true` 如果你的datafed 支持商品查询和人商品解析逻辑。
- **supports\_group\_request**: 设置这一选项为 `true` 如果您的datafeed只提供所有商品集合的完整信息，并且无法进行商品搜索或单个商品解析。

`supports_search` 和 `supports_group_request` 两者之中有只有一个可以为 `true` 。

**Remark:** 如果你的datafeed 没有实现这个调用(根本不响应或发送404)，将使用默认配置。这样：

```
{
  supports_search: false,
  supports_group_request: true,
  supported_resolutions: ["1", "5", "15", "30", "60", "1D", "1W", "1M"],
  supports_marks: false,
  supports_time: true
}
```

## 商品集合信息

Request: GET /symbol\_info?group=<group\_name>

1. `group_name` : string

Example: GET /symbol\_info?group=NYSE

Response: 预期响应是具有以下列出的属性的对象。每个属性都被视为表的一列，如上所述（请参见[表式响应](#)）。响应结构与[SymbolInfo](#)类似（但不等于），因此有关所有字段的详细信息，请参见其描述。

- **symbol**
- **description**
- **exchange-listed/exchange-traded**
- **minmovement/minmov**（注意：minmov已被弃用，并将在未来的版本中被删除）
- **minmovement2/minmov2**（注意：minmov2已被弃用，并将在未来的版本中被删除）
- **fractional**
- **pricescale**
- **has-intraday**
- **has-no-volume**
- **type**

- **ticker**
- **timezone**
- **session-regular**(mapped to `SymbolInfo.session` )
- **supported-resolutions**
- **force-session-rebuild**
- **has-daily**
- **intraday-multipliers**
- **has-fractional-volume**(obsolete)
- **volume\_precision**
- **has-weekly-and-monthly**
- **has-empty-bars**

示例：以下是对datafeed的响应示例 `GET /symbol_info?group=NYSE` (数据为手工制造):

```
{
  symbol: ["AAPL", "MSFT", "SPX"],
  description: ["Apple Inc", "Microsoft corp", "S&P 500 index"],
  exchange-listed: "NYSE",
  exchange-traded: "NYSE",
  minmov: 1,
  minmov2: 0,
  pricescale: [1, 1, 100],
  has-dwm: true,
  has-intraday: true,
  has-no-volume: [false, false, true]
  type: ["stock", "stock", "index"],
  ticker: ["AAPL~0", "MSFT~0", "$SPX500"],
  timezone: "America/New_York",
  session-regular: "0900-1600",
}
```

**Remark 1:** 如果您的datafeed配置`supports_group_request` : `true`或根本没有响应配置请求，则将使用此调用。

**Remark 2:** 如果您的datafeed 被请求不支持的集合（如果您对请求#1（支持的集合）的响应是正确的），则会发生404错误）。

**Remark 3:** 使用此模式（获取大量的商品数据）在浏览器中存储一些用户不需要的数据。因此，如果您的商品列表有多个项目，请考虑支持商品搜索/单个商品解析。

## 商品解析

Request: `GET /symbols?symbol=<symbol>`

1. `symbol` : string. 商品名称或者代码.

例: `GET /symbols?symbol=AAL` , `GET /symbols?symbol=NYSE:MSFT`

Response: JSON包含的对象与SymbolInfo完全一样

**Remark:** 如果您的datafeed配置supports\_group\_request : false 和 supports\_search : true , 则将执行此调用。

## 商品检索

Request: GET /search?query=<query>&type=<type>&exchange=<exchange>&limit=<limit>

1. query : string. 用户在商品搜索编辑框中输入的文本
2. type : string. 您的后台支持的类型之一
3. exchange : string. 您的后台支持的交易所之一
4. limit : integer. 响应最大项目数

例: GET /search?query=AA&type=stock&exchange=NYSE&limit=15

Response: 响应将是调用JS API后返回的一个数组类型的商品记录

**Remark:** 如果您的datafeed配置supports\_group\_request : false 和 supports\_search : true , 则将执行此调用。

## K线柱

Request: GET /history?symbol=<ticker\_name>&from=<unix\_timestamp>&to=<unix\_timestamp>&resolution=<resolution>

1. symbol : 商品名称或者代码
2. from : unix timestamp (UTC) or leftmost required bar
3. to : unix timestamp (UTC) or rightmost required bar
4. resolution : string

例: GET /history?symbol=BEAM-0&resolution=D&from=1386493512&to=1395133512

Response: 响应的预期是一个对象，下面列出了一些属性。每个属性都被视为表的列，如上所述。

- **s:** 状态码。预期值: ok | error | no\_data
- **errmsg:** 错误消息。只在 s = 'error'时出现
- **t:** K线时间. unix时间戳 (UTC)
- **c:** 收盘价
- **o:** 开盘价 (可选)
- **h:** 最高价 (可选)
- **l:** 最低价 (可选)
- **v:** 成交量 (可选)
- **nextTime:** 下一个K线柱的时间 如果在请求期间无数据 (状态码为 no\_data ) (可选)

**Remark:** bar time 对于日K线柱预期为 一个交易日 (not session start day) 以 00:00 UTC为起点。 Charting Library 会根据SymbolInfo的Session时间进行匹配。

**Remark:** K线时间对于月K线柱为这个月的第一个交易日，除去时间的部分。

**Remark:** 价格应作为数字传递，而不是使用字符串。

例:

```
{
  s: "ok",
  t: [1386493512, 1386493572, 1386493632, 1386493692],
  c: [42.1, 43.4, 44.3, 42.8]
}
```

```
{
  s: "no_data",
  nextTime: 1386493512
}
```

```
{
  s: "ok",
  t: [1386493512, 1386493572, 1386493632, 1386493692],
  c: [42.1, 43.4, 44.3, 42.8],
  o: [41.0, 42.9, 43.7, 44.5],
  h: [43.0, 44.1, 44.8, 44.5],
  l: [40.4, 42.1, 42.8, 42.3],
  v: [12000, 18500, 24000, 45000]
}
```

**nextTime** 是怎么工作的

假设您以分辨率= 1观看图表，并且Library要求您以[2015年4月3日 16:00 UTC + 0，2015年4月3日 19:00 UTC + 0]为范围，向纽约证券交易所请求股票数据。4月3日为受难节，交易所休假。Library假定你会作出如下响应:

```
{
  s: "no_data",
  nextTime: 1428001140000 //2015年4月2日 18:59:00 GMT+0
}
```

因此 **nextTime** 是一个从Library的原始请求边界的左侧（在假想时间线上）的K线柱时间。

所有省略的价格将被视为等于收盘价。

## 标识

**Request:** GET /marks?symbol=<ticker\_name>&from=<unix\_timestamp>&to=<unix\_timestamp>&resolution=<resolution>

1. `symbol` : symbol name or ticker.
2. `from` : unix timestamp (UTC) or leftmost visible bar
3. `to` : unix timestamp (UTC) or rightmost visible bar
4. `resolution` : string

**Response:** 响应预期是一个对象，下面列出了一些属性。此对象与JS API中的[respective response](#)相似，但每个属性都被视为表的列，如上所述。

```
{
  id: [array of ids],
  time: [array of times],
  color: [array of colors],
  text: [array of texts],
  label: [array of labels],
  labelFontColor: [array of label font colors],
  minSize: [array of minSizes],
}
```

**Remark:** 备注：如果您的datafeed在传输的配置数据中发送了`supports_marks : true`，则会调用此方法。

## 时间刻度标记

**Request:** GET /timescale\_marks?symbol=<ticker\_name>&from=<unix\_timestamp>&to=<unix\_timestamp>&resolution=<resolution>

1. `symbol` : symbol name or ticker.
2. `from` : unix timestamp (UTC) or leftmost visible bar
3. `to` : unix timestamp (UTC) or rightmost visible bar
4. `resolution` : string

**Response:** 响应预期为一个具有下列属性的数组对象。

1. `id` : unique identifier of a mark
2. `color` : rgba color
3. `label` : 显示在圆圈中的文字
4. `time` : unix time
5. `tooltip` : tooltip text

**Remark:** This call will be requested if your datafeed sent `supports_timescale_marks: true` in configuration data.

## 服务器时间



Request: `GET /time`

Response: Numeric unix time without milliseconds. Example: 1445324591

## 报价

Request: `GET /quotes?symbols=<ticker_name_1>,<ticker_name_2>,...,<ticker_name_n>`

Example: `GET /quotes?symbols=NYSE%3AAA%2CNYSE%3AF%2CNasdaqNM%3AAAPL`

Response: Response is an object.

- **s**: status code for request. Expected values: `ok` | `error`
- **errmsg**: error message for client
- **d:symbols data** array

Example:

```

{
  "s": "ok",
  "d": [{
    "s": "ok",
    "n": "NYSE:AA",
    "v": {
      "ch": "+0.16",
      "chp": "0.98",
      "short_name": "AA",
      "exchange": "NYSE",
      "description": "Alcoa Inc. Common",
      "lp": "16.57",
      "ask": "16.58",
      "bid": "16.57",
      "open_price": "16.25",
      "high_price": "16.60",
      "low_price": "16.25",
      "prev_close_price": "16.41",
      "volume": "4029041"
    }
  ]}, {
  "s": "ok",
  "n": "NYSE:F",
  "v": {
    "ch": "+0.15",
    "chp": "0.89",
    "short_name": "F",
    "exchange": "NYSE",
    "description": "Ford Motor Compan",
    "lp": "17.02",
    "ask": "17.03",
    "bid": "17.02",
    "open_price": "16.74",
    "high_price": "17.08",
    "low_price": "16.74",
    "prev_close_price": "16.87",
    "volume": "7713782"
  }
}]
}

```

## 构造函数

```
Datafeeds.UDFCompatibleDatafeed = function(datafeedURL, updateFrequency, protocolVersion)
```

### datafeedURL

这是一个数据服务器的URL，它将得到请求和返回数据。

### updateFrequency（更新频率）

这是一个有间隔的实时数据请求，**datafeed**将以毫秒为单位发送到服务器。默认值为10000（10秒）。

## **protocolVersion**（协议版本）

1 - 过时的协议，每当图表滚动时，都会重新从服务器请求所有数据。

2（默认） - 数据请求是增量的。当商品或分辨率未改变时（除了最后2个条），不会再次请求相同数据。

# Symbology (商品代码体系)

图表需要您自己提供数据,所以商品代码体系100%您来决定。仅返回商品信息(图表库定义格式)和使用任意的符号商品。实际上,商品名称可以为任意字符串。

但有些细节你应该知道:

1. 我们自己定义的商品名称必须为此格式: `EXCHANGE:SYMBOL`。图表默认使用此格式
2. 已有其他商品代码体系或只有一个?这里有ticker特殊术语提供给您。Ticker为商品唯一标识符,只用于图表的内部,您的用户将不会看到它。

## 商品信息结构

这一节非常重要。图表库用户遇到的72.2%的问题,都是由于错误的/格式错误的SymbolInfo数据引起的。

SymbolInfo是一个包含商品metadata的对象。该对象是解析商品的结果。SymbolInfo有以下字段:

### name

商品名称。您的用户将看到它(作为一个字符串)。此外,如果您不使用 `tickers`,它将用于数据请求。

### ticker

它是您的商品代码体系中此商品的唯一标识符。如果您指定此属性,则其值将用于所有数据请求, `ticker` 如果未明确指定,则被视为等于 `symbol`。(译者注:请一定指定 `ticker`,如果没有 `ticker` 可以将 `symbol` 赋值给 `ticker`,未指定 `ticker` 时会发生错误。)

### description

商品说明。这个商品说明将被打印在图表的标题栏中。

### session

商品交易时间。请参阅交易日细节了解更多详情。[交易时段](#)

### exchange, listed\_exchange

现在，这两个字段都为某个交易所的略称。将被显示在图表标题栏中。目前此字段不用于其他目的。

### timezone

这个商品的交易所时区。我们希望以olsondb格式获取时区的名称。支持的时区为:

```
UTC
America/New_York
America/Los_Angeles
America/Chicago
America/Phoenix
America/Toronto
America/Vancouver
America/Argentina/Buenos_Aires
America/El_Salvador
America/Sao_Paulo
America/Bogota
Europe/Moscow
Europe/Athens
Europe/Berlin
Europe/London
Europe/Madrid
Europe/Paris
Europe/Warsaw
Australia/Sydney
Australia/Brisbane
Australia/Adelaide
Australia/ACT
Asia/Almaty
Asia/Ashkhabad
Asia/Tokyo
Asia/Taipei
Asia/Singapore
Asia/Shanghai
Asia/Seoul
Asia/Tehran
Asia/Dubai
Asia/Kolkata
Asia/Hong_Kong
Asia/Bangkok
Pacific/Auckland
Pacific/Chatham
Pacific/Fakaofu
Pacific/Honolulu
America/Mexico_City
Africa/Johannesburg
Asia/Kathmandu
US/Mountain
```

### minmov(最小波动), pricescale(价格精度), minmove2, fractional(分数)

1. 最小的价格变化是由这些值决定的。

2. **PriceScale** 参数确定了图表价格量表上的价格线之间的间隔。

这三个键有不同意义时，使用通常价格和分数价格。

### 通常价格

```
MinimalPossiblePriceChange (最小可能价格变动) = minmov / pricescale
```

**minmov** 数字型单位组成一个tick。例如，美国股票价格和tick有小数，并可以上下浮动+/- 0.01。

### 分数价格

分数显示价格, 1 - xx'yy (例如, 133'21) 或 2 - xx'yy'zz (例如, 133'21'5)。

### minmove2<0>

这是一个神奇的数字来格式化复杂情况下的价格。这里有一些例子:

```
典型的股票以0.01价格增量: minmov = 1, pricecale = 100, minmove2 = 0
ZBM2014 (国债), 1/32: minmov = 1, pricecale = 32, minmove2 = 0
ZCM2014 (玉米), 2/8: minmov = 2, pricecale = 8, minmove2 = 0
ZFM2014 (5年期国债), 1/32的1/4: minmov = 1, pricecale = 128, minmove2 = 4
```

### has\_intraday

布尔值显示商品是否具有日内（分钟）历史数据。如果它为 **false**，则当图表中的该商品处于活动状态时，日内分辨率的所有按钮将被禁用。如果设置为 **true**，则由datafeed直接提供的所有分辨率必须在intraday\_multipliers数组中设定。

### supported\_resolutions

在这个商品的分辨率选择器中启用一个分辨率数组。数组的每个项目都是字符串。

被datafeed支持（见datafeed配置数据）但不受当前商品支持的分辨率, 将在分辨率选择器部件中禁用。如果更改商品，新商品不支持选定的分辨率，则分辨率将切换到支持的分辨率列表中的第一项。分辨率可用性逻辑（伪代码）：

```
resolutionAvailable =
  resolution.isIntraday ?
    symbol.has_intraday && symbol.supports_resoluition(resolution) :
    symbol.supports_resoluition(resolution);
```

如果在商品信息中没有supported\_resolutionsin，则所有DWM(daily, weekly, monthly)分辨率都可用。如果has\_intraday为true，则日内分辨率可用。

支持的分辨率也会影响可用的时间范围。如果使用不支持的分辨率，则时间范围将不可用。

### **intraday\_multipliers <[]>**

这是一个包含日内分辨率(分钟单位)的数组，**datafeed**将会自行构建它。

举例来说：如果**datafeed**报告说它支持 `["1", "5", "15"]`，但事实上股票X只有1分钟的数据，股票X将设定 `intraday_multipliers = [1]`，那么Charting Library将自行构建5分钟和15分钟的分辨率。

### **has\_seconds**

布尔值显示商品是否具有以秒为单位的历史数据。如果它为 `false`，那么在图表中此商品处于活动状态时，所有秒的分辨率的按钮将被禁用。如果它为 `true`，则由**datafeed**直接提供的所有分辨率必须在 `seconds_multipliers` 数组中设定。

### **seconds\_multipliers <[]>**

这是一个包含秒分辨率(以秒为单位，无小数)，**datafeed**将会自行构建它。

举例来说：如果**datafeed**报告说它支持 `["1S", "5S", "15S"]`，但事实上股票X只有1秒钟的数据，股票X将设定 `seconds_multipliers = [1]`，那么Charting Library将自行构建5S和15S的分辨率。

### **has\_daily**

布尔值显示商品是否具有以日为单位的历史数据。如果它为`false`，则Charting Library将自行构建日单位的分辨率。如果没有，则会向**datafeed**请求这些数据。

### **has\_weekly\_and\_monthly**

布尔值显示商品是否具有以W和M为单位的历史数据。如果它为`false`，则Charting Library将通过日单位的分辨率自行构建。如果没有，则会向**datafeed**请求这些数据。

### **has\_empty\_bars**

布尔值显示在交易过程中，当**datafeed**没有数据返回时，**library**是否会生成空的K柱。

即，如果您的交易时间为0900-1600，而您的实际数据在11:00和12:00之间没有交易，而您的**has\_empty\_bars**为`true`，那么Library会在此段时间贴上退化的K柱。

### **force\_session\_rebuild**

布尔值显示**library**是否会随着当前交易而过滤K柱。如果为`false`，则当**library**从其他分辨率构建数据或将**has\_empty\_bars**设置为`true`时，K柱将被过滤。如果为`true`，Library将会删除那些不是交易K柱的数据。

### has\_no\_volume

布尔表示商品是否拥有成交量数据。

### has\_fractional\_volume | 已过时(1.1 - 1.5), 用法与volume\_precision相反

如果has\_fractional\_volume = true，成交量指标值将不会舍入为整数值。

### volume\_precision <0>

整数显示此商品的成交量数字的小数位。0表示只显示整数。1表示保留小数位的1个数字字符，等等。

### data\_status

数据状态码。状态显示在图表的右上角。支持的值:

- streaming(实时)
- endofday(已收盘)
- pulsed(脉冲)
- delayed\_streaming(延迟流动中)

### expired

期满，布尔值显示此商品是否为到期的期货合约。

### expiration\_date

到期日(Unix时间戳)。如果expired = true，则必须设置此值。图表库将从该时间点而不是实际时刻请求该商品的数据。

### sector

板块，将在股票信息中显示。

### industry

行业，将在股票信息中显示。

### currency\_code

货币代码，将在商品信息中显示。



# 交易时段

图表库期望在商品信息中获取交易时段。交易时段是可交易的时间范围。每个交易时段都应该有左右边界。在图表库中，交易时段的格式为“HHMM-HHMM”。例如，交易时段从上午9:30到下午16:00应该表示为 0930-1600 。

有一个特殊情况的商品交易7\*24小时(例如：比特币或其它数字货币)。交易时段的字符串应该为 24x7 。交易时段将会发生在交易时区。

如果交易时段左边界大于右边边界（例 1700-0900 ），则此交易时段被视为隔夜。隔夜交易始终在前一天开始：例如，如果商品在星期一至星期五 1700-0900 交易，则这周（#i）第一交易时段 = 前一周（#i-1）星期日17:00开始,到本周（#i）星期一09:00结束。

每个交易日可能会有多个交易时段。如果有多个交易时段，您应该将整个交易时段分为以逗号分隔的多个交易时段。例如，假设当天的交易时间为9:30至14:00，然后为14:30至17:00，交易时段应为 0930-1400,1430-1700 。

此外，交易时间可能会有所差异。这是您可以使用：特殊的说明符。例如，如果商品全部时间都为0900-1630，但星期一比较特殊（交易时段为0900-1400），这交易时段应为 0900-1630|0900-1400:2 。让我们看看这个字符串的细节。

片段	含义
0900-1630	交易时段为0900-1630。默认情况下，此会话将分配给所有非周末日，因为它后面没有：说明符。
\	交易时段分隔符。负责分隔不同的交易时段。
0900-1400	交易时段为0900-1400。这是一天的交易时段（见下文）。
:	日期说明符。该字符在少时说明符后，后跟日期号码。
2	上述交易时段的日期号码 (0900-1400)

日期号码：星期日为1，星期六为7（2--星期一，3--星期二，等等）。

可以覆盖一个或多个日期。例如，在 0900-1630 | 0900-1400:23 中，0900-1400交易时段将被分配到第2天和第3天（星期一，星期二）。

## version: 1.1:

可以使用分号指定一周的第一个交易日。例：

```
"1;0900-1630|0900-1400:2" : 每周的第一天是星期日  
"0900-1630|0900-1400:2;6" : 每周的第一天是星期五  
"0900-1630|0900-1400:2"   : 每周的第一天是星期一（默认值）
```

**Remark:** 默认情况下，所有非24x7商品在星期六和星期日被视为不可交易。所以如果您的商品在周末交易，您应该明确指定交易日。例如：某个商品 1000-1600 星期日到星期五交易，则交易时段应写为 1000-1600:123456

使用此解析器检查交易时段字符串：<http://tradingview.github.io/checksession.html>

## 报价

报价是简要描述交易的数据集。图表库支持观察列表（在交易终端配置中），并使用报价来显示相关的商品信息。

图表库中不管是JS API还是UDF都使用相同的数据结构来进行报价。

以下是响应对象的描述：

### 商品报价数据

- `s`：商品的状态码。预期的值为：`ok` | `error`
- `n`：商品名称。此值必须与请求中完全相同
- `v`：`object`, 商品报价对象
  - `ch`：价格变动（通常从当天的开盘价计算）
  - `chp`：价格变动百分比
  - `short_name`：商品略称
  - `exchange`：交易所名称
  - `description`：商品的简短描述
  - `lp`：最后的成交价格
  - `ask`：买盘价
  - `bid`：卖盘价
  - `spread`：费率
  - `open_price`：当天开盘价
  - `high_price`：当天最高价
  - `low_price`：当天最低价
  - `prev_close_price`：昨天收盘价
  - `volume`：当天成交量

## 定制概述

定制是一个相当模糊的概念，所以写几篇文章对它进行说明。

## 通过数据流定制

他们最关心的是数据相关的东西。定制的配置可以通过datafeed响应来实现。响应配置实例：

```
{
  supports_search: true,
  supports_group_request: false,
  supports_marks: true,
  exchanges: [
    {value: "", name: "All Exchanges", desc: ""},
    {value: "XETRA", name: "XETRA", desc: "XETRA"},
    {value: "NSE", name: "NSE", desc: "NSE"}
  ],
  symbolsTypes: [
    {name: "All types", value: ""},
    {name: "Stock", value: "stock"},
    {name: "Index", value: "index"}
  ],
  supportedResolutions: [ "1", "15", "30", "60", "D", "2D", "3D", "W", "3W", "M", '6M' ]
};
```

在JS API可以找到更详细的说明。

## 在客户端进行定制

允许您最大化的定制UI/UX。这些定制通过定义图表控件中的构造函数的参数完成。

图表控件构造函数调用的示例：

```
var widget = new TradingView.widget({
  fullscreen: true,
  symbol: 'AA',
  interval: 'D',
  toolbar_bg: '#f4f7f9',
  allow_symbol_change: true,
  container_id: "tv_chart_container",
  datafeed: new Datafeeds.UDFCompatibleDatafeed("http://demo-feed.tradingview.com"),
  library_path: "charting_library/",
  locale: "en",
  drawings_access: { type: 'black', tools: [ { name: "Regression Trend" } ] },
  disabled_features: ["use_localstorage_for_settings", "volume_force_overlay"],
  enabled_features: ["move_logo_to_main_pane"],
  overrides: {
    "mainSeriesProperties.style": 0,
    "symbolWatermarkProperties.color" : "#944",
    "volumePaneSize": "tiny"
  },
  studies_overrides: {
    "bollinger bands.median.color": "#33FF88",
    "bollinger bands.upper.linewidth": 7
  },
  debug: true,
  time_frames: [
    { text: "50y", resolution: "6M" },
    { text: "1d", resolution: "5" },
  ],
  charts_storage_url: 'http://saveload.tradingview.com',
  client_id: 'tradingview.com',
  user_id: 'public_user',
  favorites: {
    intervals: ["1D", "3D", "3W", "W", "M"],
    chartTypes: ["Area", "Line"]
  }
});
```

详情参考：[Widget构造器](#)

## 也可以看看


- [Widget方法](#)
- [定制的使用案例](#)

# Widget构造器

当调用构造函数时，您可以定义图表库widget的参数。例：

```
new TradingView.widget({
  symbol: 'A',
  interval: 'D',
  timezone: "America/New_York",
  container_id: "tv_chart_container",
  locale: "ru",
  datafeed: new Datafeeds.UDFCompatibleDatafeed("https://demo_feed.tradingview.com")
});
```

查看下列完整支持的参数列表。请记住，在图表初始化后在更改这些参数是不起作用的。如果要在初始化图表之后更改图表的状态，请使用[widget方法](#)。

属性标记为的只在交易终端可用。

## symbol, interval [mandatory]

您的图表的初始商品和间隔。间隔的格式在另一篇[文章](#)中说明。

## timeframe

设置图表的初始时间范围。时间范围是加载并显示在屏幕上的K线范围。有效的时间范围是一个数字加一个字母，D为数天，M为数月。

## container\_id [mandatory]

`id` 属性为指定要包含widget的DOM元素id。

## datafeed [mandatory]

JavaScript对象的实现接口 [JS API](#) 以反馈图表及数据。

## timezone

图表的初始时区。时间刻度上的数字取决于这个时区。请参阅[支持的时区列表](#)。设置为交易所时区。覆盖默认值，您应该使用[覆盖章节](#)。

## debug

将此属性设置为true时，可使图表将详细的API日志写入控制台。与功能集的 `charting_library_debug_mode` 用法相同。

## library\_path

`static` 文件夹的路径

## width, height

widget的尺寸，请确保widget拥有足够的空间。

**Remark:** 如果您想让图表占据所有可用的空间，请不要使用 '100%' 这样的字段。使用 `fullscreen` 参数来代替（见下文）。这是因为DOM节点在不同浏览器中有调整大小的问题。

## fullscreen

布尔值显示图表是否占用窗口中所有可用的空间。

## autosize

显示图表是否应占据容器中所有可用空间并在窗口调节大小时自动进行调整。该参数介绍在1.3版本中被引入。

## symbol\_search\_request\_delay

在商品搜索按下键后请求之前，以毫秒为单位延迟。

## auto\_save\_delay

延迟秒数等待 `onAutoSaveNeeded` 可以被再次调用。该参数介绍在1.5版本中。

## toolbar\_bg

工具栏背景颜色

## study\_count\_limit

自1.5版本起。

图表或多功能图布局的最大研究数量。最小值为2。

## studies\_access

版本：1.1具有以下结构的对象：

```
{
  type: "black" | "white",
  tools: [
    {
      name: "<study name>",
      grayed: true
    },
    <... >
  ]
}
```

- `type` 是列表类型。支持的值: `black` (所有列出的项目会被禁用), `white` (只有列出的项目会被启用)。
- `tools` 对象数组。每个对象可以具有以下属性：
  - `name` (强制的) 研究的名称。使用相同的名称，你可以看到他们在指标控件。
  - `grayed` 布尔值，表明这项研究将可见，但看起来像是被禁用的。如果研究为 `grayed`，当用户点击它时，会调用 `onGrayedObjectClicked` 回调方法。

## drawings\_access

版本：1.1 该属性与上述的 `studies_access` 具有相同的结构。使用与您在UI中看到的名称相同的名称。

**Remark:** 基于字体的绘图有一个特殊情况。使用 `Font Icons` 的名字时，这个组是一个特例，它的绘图不能被启用或禁用 - 可以启用或禁用整个组。

## saved\_data

JS对象包含保存的图表内容（JSON，请参阅下面的保存/加载调用）。如果在创建图表时已经有图表的JSON，请使用此参数。如果要将图表内容加载到已初始化的图表中，请使用 `loadData()` 控件方法。

## locale

图表库的本地化处理。详情：[本地化](#)

## numeric\_formatting



该对象包含数字的格式化选项。目前唯一可能的选择是 `decimal_sign`。

例: `numeric_formatting: { decimal_sign: "," }`

## customFormatters

它是一个包含以下字段的对象：

1. `timeFormatter`
2. `dateFormatter`

您可以使用这些格式化方法自定义显示日期和时间的值。这两个值都是具有方法 `format` 和 `formatLocal` 的对象：

```
function format(date)
function formatLocal(date)
```

这些函数返回表示date或time的文本。`formatLocal` 将日期和时间转换为本地时区。

例:

```
customFormatters: {
  timeFormatter: {
    format: function(date) { var _format_str = '%h:%m'; return _format_str.replace('%h',
, date.getUTCHours(), 2). replace('%m', date.getUTCMinutes(), 2). replace('%s', date.g
etUTCSeconds(), 2); }
  },
  dateFormatter: {
    format: function(date) { return date.getUTCFullYear() + '/' + date.getUTCMonth() +
 '/' + date.getUTCDate(); }
  }
}
```

## overrides

对Widget对象的默认属性进行覆盖。覆盖属性意味着为其分配默认值。您可以覆盖大部分图表的属性（也可以由用户通过UI编辑）使用 `overrides` 参数构造控件。`overrides` 应该是一个具有范围的对象。每个字段名是重写属性的名称，字段值是这些属性的期望值。例子:

```
overrides: {
  "symbolWatermarkProperties.color": "rgba(0, 0, 0, 0)"
}
```

这个 `override` 将使水印100%不透明（不可见）。所有可定制的属性都列在[单独的文章](#)中。从1.5开始，您可以使用绘图覆盖。[绘图覆盖](#)。

## disabled\_features, enabled\_features

包含功能在默认情况下启用/禁用名称的数组。功能表示图表功能的一部分（更是UI/UX的一部分）。[这里](#)。此处列出了支持的功能。例：

```
TradingView.onready(function()
{
    var widget = new TradingView.widget({
        /* .... */
        disabled_features: ["header_widget", "left_toolbar"],
    });
});
```

## snapshot\_url

当用户按快照按钮时,使用base64编码将当前图表快照保存并返回URL。该服务返回完整的保存图像URL。

## indicators\_file\_name

包含您编写的指标的文件路径。查看[更多细节](#)。

## preset

preset 是一组预定义窗口小部件设置的名称。预设中使用的所有设置也可以直接在窗口小部件的构造函数中使用。现在只支持 `mobile` 预设。此预设的示例可在线获取。

## studies\_overrides

使用此选项自定义默认指标的样式及输入值。您还可以使用此参数自定义 `compare` 系列的样式和输入值。[查看更多](#)

## time\_frames

在图表底部的时间范围选择器中可以看见这个时间范围列表。例：

```
time_frames: [
    { text: "50y", resolution: "6M", description: "50 Years" },
    { text: "3y", resolution: "W", description: "3 Years", title: "3yr" },
    { text: "8m", resolution: "D", description: "8 Month" },
    { text: "3d", resolution: "5", description: "3 Days" },
    { text: "1000y", resolution: "W", description: "All", title: "All" },
]
```

时间范围是一个包含 `text` 和 `resolution` 属性的对象。文本必须具有以下格式：`<integer><y|m|d>` (`\d+(y|m|d)` 为正则表达式)。分辨率是具有通用分辨率格式的字符串。请参阅[本主题](#)了解有关时间范围的更多信息。在1.7中添加了描述属性，并显示在弹出菜单中。此参数是可选的（如果时间范围描述符不包含此属性：`title`（如果指定）或使用）。`title`属性在1.9中添加，此值将覆盖从`text`属性生成的默认标题。此参数是可选的。

## charts\_storage\_url, client\_id, user\_id

这些参数是有关于高阶图表的保存/加载。查看[更多细节](#)。

## charts\_storage\_api\_version

您的后台版本。支持的值: `"1.0"` | `"1.1"`。指标模板从 `1.1` 开始得到支持。

## load\_last\_chart

将此参数设置为 `true` 如果您希望库加载用户的最后一张图表（您也应该具有 `[save/load|Saving-and-Loading-Charts]`）。

## custom\_css\_url (since 1.4)

将您的自定义CSS添加到图表中。`url`应该是到`'static'`文件夹的绝对或相对路径。

## favorites

默认支持该项目。此选项需要禁用`localStorage`的使用（请参阅[功能集](#)以了解更多）。`favorites` property 为一个对象，拥有以下属性：

- **intervals**(间隔): 收藏的间隔数组。例：`["D", "2D"]`
- **chartTypes**(图表类型): 收藏的图表类型数组。图表类型名称与图表的UI中的英文版本相同。例：`["Area", "Candles"]`

## save\_load\_adapter (since 1.12)

包含保存/加载功能的对象。如果设置了，应有以下方法：

### Chart layouts

1. `getAllCharts(): Promise<ChartMetaInfo[]>`

获取所有保存的图表。

`ChartMetaInfo` 具有以下字段的对象：

- `id` - 图表id
- `name` - 图表名
- `symbol` - 图表的商品
- `resolution` - 分辨率
- `timestamp` - 最后修改日期（从01/01/2015 UTC午夜开始的毫秒数）。

## 2. `removeChart(chartId): Promise<void>`

删除图表。 `chartId` 是图表的唯一ID（参见上面的 `getAllCharts`）。

## 3. `saveChart(chartData: ChartData): Promise<ChartId>`

存储图表。

`ChartData` 具有以下字段的对象:

- `id` - 图表的唯一标识（如果未保存则可能是 `undefined`）。
- `name` - 图表名
- `symbol` - 图表的商品
- `resolution` - 分辨率
- `content` - 图表的内容

`ChartId` - 图表唯一id (string)

## 4. `getChartContent(chartId): Promise<ChartContent>`

通过服务器加载图表

`ChartContent` 带有图表内容的字符串（参见 `saveChart` 函数中的 `ChartData :: content` 字段）。

## Study Templates

### 1. `getAllStudyTemplates(): Promise<StudyTemplateMetaInfo[]>`

获取所有保存的研究模板。

`StudyTemplateMetaInfo` 具有以下字段的对象:

- `name` - 研究模板名称

### 2. `removeStudyTemplate(studyTemplateInfo: StudyTemplateMetaInfo): Promise<void>`

删除研究模板

### 3. `saveStudyTemplate(studyTemplateData: StudyTemplateData): Promise<void>`

存储研究模板

`StudyTemplateData` 具有以下字段的对象:

- `name` - 研究模板名称

- `content` - 研究模板的内容

4. `getStudyTemplateContent(studyTemplateInfo: StudyTemplateMetaInfo): Promise<StudyTemplateContent>`

通过服务器加载研究模板

`StudyTemplateContent` - 研究模板的内容 (string)

如果同时设置了 `charts_storage_url` 和 `save_load_adapter` 将被设置 - `save_load_adapter`

重要：所有函数应该返回 `Promise`（或 `Promise` 类对象）。

## settings\_adapter (since 1.11)

包含设置/删除的方法。使用它将图表设置保存到您的首选存储，包括服务器端。如果设置了，应该有以下方法：

1. `initialSettings: Object` 初始化设置
2. `setValue(key: string, value: string): void` 存储键/值对
3. `removeValue(key: string): void` 删除键

## 交易终端专属

### 组件工具栏

该对象包含图表右侧窗口小部件栏的设置。右侧窗口小部件栏中的数据窗口，观察列表和详细信息选项卡可以使用Widget构造函数中的`widgetbar` 开启此功能：

```
widgetbar: {
  details: true,
  watchlist: true,
  watchlist_settings: {
    default_symbols: ["NYSE:AA", "NYSE:AAL", "NASDAQ:AAPL"],
    readonly: false
  }
}
```

- **details**: 启用右侧窗口小部件栏中的详细信息窗口小部件。
- **watchlist**: 启用右侧小部件栏中的观察列表小部件。
- **watchlist\_settings.default\_symbols <[]>**: 给观察列表设置商品数组。
- **\*\*watchlist\_settings.readonly**: 给观察列表开启只读模式。

## rss\_news\_feed

使用此属性更改RSS新闻。您可以为每个商品类型设置不同的rss，或为每个商品使用一个rss。对象将拥有 `default` 属性，其他属性是可选的；它们的名字为商品的类型。每个属性都拥有一个对象(或对象数组)并且具有以下属性：

1. `url` 请求的URL。它可以包含以下花括号中的标签（将会被终端所更改）：`{SYMBOL}`，`{TYPE}`，`{EXCHANGE}`。
2. `name` 在每一个新闻的底部显示一个反馈。

例：

```
{
  "default": [ {
    url: "https://articlefeeds.nasdaq.com/nasdaq/symbols?symbol={SYMBOL}",
    name: "NASDAQ"
  }, {
    url: "http://feeds.finance.yahoo.com/rss/2.0/headline?s={SYMBOL}&region=US&lang=en-US",
    name: "Yahoo Finance"
  } ]
}
```

另一个例子：

```
{
  "default": {
    url: "https://articlefeeds.nasdaq.com/nasdaq/symbols?symbol={SYMBOL}",
    name: "NASDAQ"
  }
}
```

更多例子：

```
{
  "default": {
    url: "https://articlefeeds.nasdaq.com/nasdaq/symbols?symbol={SYMBOL}",
    name: "NASDAQ"
  },
  "stock": {
    url: "http://feeds.finance.yahoo.com/rss/2.0/headline?s={SYMBOL}&region=US&lang=en-US",
    name: "Yahoo Finance"
  }
}
```

## news\_provider



交易控制器可以让您在线交易。[阅读更多](#)

```
new TradingView.widget({  
    /* ... */  
    trading_controller: new MyTradingController()  
});
```

### brokerFactory

使用这个字段来传递构造[经纪商API](#)的实现函数。这是一个接收[交易主机](#)并返回[经纪商API](#)的函数。

### brokerConfig

`brokerConfig: { configFlags: {...} }` 使用此字段设置交易终端的配置标志。[了解更多](#)

## 也可以看看

- [定制概述](#)
- [Widget方法](#)
- [功能集](#)
- [存储于加载图表](#)
- [覆盖默认研究参数](#)
- [覆盖默认图表参数](#)



# Widget方法

以下是widget支持的方法列表。您可以使用widget构造函数返回给您的widget对象来调用它们。



**Remark:** 请注意，只有在onChartReady回调触发后才可以调用这些方法。所以常见的做法就是这样：

```
widget.onChartReady(function() {  
    // 现在可以调用其他widget的方法了  
});
```

## Methods

在1.5之前[Chart Methods](#) 归属于 **Widget**. 请参阅完整的操作列表[here](#)

- 订阅图表事件
  - [onChartReady\(callback\)](#)
  - [onSymbolChange\(callback\)](#)[已过时]
  - [onIntervalChange\(callback\)](#)[已过时]
  - [onAutoSaveNeeded\(callback\)](#)[已过时]
  - [onBarMarkClicked\(callback\)](#)[已过时]
  - [onTimescaleMarkClicked\(callback\)](#)[已过时]
  - [onGrayedObjectClicked\(callback\)](#)
  - [onScreenshotReady\(callback\)](#)[已过时]
  - [onTick\(callback\)](#)[已过时]
  - [onShortcut\(shortcut, callback\)](#)
  - [subscribe\(event, callback\)](#)
- 图表动作
  - [chart\(\)](#)
  - [setLanguage\(locale\)](#)
  - [setSymbol\(symbol, interval, callback\)](#)
  - [remove\(\)](#)
  - [closePopupsAndDialogs\(\)](#)
  - [selectLineTool\(drawingId\)](#)
  - [selectedLineTool\(\)](#)
- 保存/加载图表

- [save\(callback\)](#)
  - [load\(state\)](#)
  - [getSavedCharts\(callback\)](#)
  - [loadChartFromServer\(chartRecord\)](#)
  - [saveChartToServer\(onCompleteCallback, onFailCallback, saveAsSnapshot, options\)](#)
  - [removeChartFromServer\(chartId, onCompleteCallback\)](#)
- 自定义UI控件
  - [onContextMenu\(callback\)](#)
  - [createButton\(options\)](#)
- 对话框
  - [showNoticeDialog\(params\)](#)
  - [showConfirmDialog\(params\)](#)
  - [showLoadChartDialog\(\)](#)
  - [showSaveAsChartDialog\(\)](#)
- Getters
  - [symbolInterval\(callback\)](#)
  - [mainSeriesPriceFormatter\(\)](#)
  - [getIntervals\(\)](#)
  - [getStudiesList\(\)](#)
- 定制
  - [addCustomCSSFile\(url\)](#)
  - [applyOverrides\(overrides\)](#)
  - [applyStudiesOverrides\(overrides\)](#)
-  交易终端特制
  - [showSampleOrderDialog\(order\)](#)
  - [watchList\(\)](#)
-  多图表布局
  - [chart\(index\)](#)
  - [activeChart\(\)](#)
  - [chartsCount\(\)](#)
  - [layout\(\)](#)
  - [setLayout\(layout\)](#)

## 订阅图表事件

### [onChartReady\(callback\)](#)

1. `callback : function()`

当图表初始化并准备就绪时，图表库将调用提供的回调。你可以从这一刻安全地调用所有其他方法。

## onGrayedObjectClicked(callback)

1. `callback : function(subject)`
  - i. `subject : object {type, name}`
    - i. `type : drawing | study`
    - ii. `name : string`, 被点击的主题名称

每次用户点击灰色的对象时，图表库都会调用此回调函数。例：

```
new TradingView.widget({
  drawings_access: {
    type: "black",
    tools: [
      { name: "Regression Trend" },
      { name: "Trend Angle", grayed: true },
    ]
  },
  studies_access: {
    type: "black",
    tools: [
      { name: "Aroon" },
      { name: "Balance of Power", grayed: true },
    ]
  },
  <...> // other widget settings
});

widget.onChartReady(function() {
  widget.onGrayedObjectClicked(function(data) {
    // 当您尝试创建力量平衡研究或趋势图形时
    // 此方法将被调用

    alert(data.name + " is grayed out!");
  })
});
```

## onShortcut(shortcut, callback)

1. `shortcut`
2. `callback : function(data)`

每当按下快捷键时，图书馆将会调用此回调。

例：

```
widget.onShortcut("alt+s", function() {
    widget.executeActionById("symbolSearch");
});
```

## subscribe(event, callback)

1. `event` : can be

Event name	Library Version	Description
toggle_sidebar		绘图工具栏 显示/隐藏
indicators_dialog		显示指标对话框
toggle_header		图表头 显示/隐藏
edit_object_dialog		显示图/研究属性对话框
chart_load_requested		即将载入的新图表
chart_loaded		
mouse_down		
mouse_up		
drawing	1.7	在图表上增加绘图。参数包含具有 <code>value</code> 字段的对象，该字段为绘图的名
study	1.7	在图表上增加指标。参数包含具有 <code>value</code> 字段的对象，该字段为指标的名
undo	1.7	
redo	1.7	
reset_scales	1.7	复位比例按钮被点击后
compare_add	1.7	显示比较对话框
add_compare	1.7	添加比较工具
load_study template	1.7	研究模板被载入后
onTick		回调将被调用每当最近的K线更新时
onAutoSaveNeeded		每当用户更改图表时，库将调用该回调。 <code>Chart change</code> 意味着可以撤消用户的任何操作。回调不会在五秒内多次调用。参见 <a href="#">auto_save_delay</a>
onScreenshotReady		每当用户创建屏幕截图和服务端返回创建的图像名称时。

onMarkClick		每次当用户点击K线上的标记时。 <a href="#">mark on bar</a> . 标记ID将作为参数传递。
onTimescaleMarkClick		每当用户单击时间刻度标记时，将调用回调。 标记ID将作为参数传递
onSelectedLineToolChanged		每次当选择的线工具更改时，将调用回调。
 layout_about_to_be_changed		要更改图表的数量或位置时
 layout_changed		已经更改图表的数量或位置时
 activeChartChanged		活动图表被更改

2. `callback` : `function(arguments)`

当GUI事件发生时，库将调用回调。 每个事件都可以有不同的参数。

## 图表功能

### chart()

返回图表对象，可用于调用[Chart-Methods](#)

### setLanguage(locale)

1. `locale` : [language code](#)

设置Widget的语言。 目前此调用将重新加载图表。请避免使用它。

### setSymbol(symbol, interval, callback)

- 1. `symbol` : `string`
- 2. `interval` : `string`
- 3. `callback` : `function()`

使图表更改其商品和分辨率。 新商品的数据到达后调用回调。

### remove()

从您的页面中删除widget。

### closePopupsAndDialogs()

调用此方法会关闭上下文菜单或对话框（如果已显示）。

## selectLineTool(drawingId)

1. `drawingId` : 可以为一个标识符 或
  - i. `cursor`
  - ii. `dot`
  - iii. `arrow_cursor`
  - iv. `eraser`
  - v. `measure`
  - vi. `zoom`
  - vii. `brush`

选择与绘图按钮上的单击相同的图形或光标。

## selectedLineTool()

返回所选图形或光标的标识符（见上文）。

# Saving/Loading Charts

## save(callback)

1. `callback` : function(object)

将图表状态保存到JS对象。图表库将调用您的回调函数并将状态对象作为参数传递。这个调用是一部分低级别的[save/load API](#).

## load(state)

1. `state` : object

从状态对象加载图表。这个调用是一部分低级别的[save/load API](#).

## getSavedCharts(callback)

1. `callback` : function(objects)

`objects` is an array of:

1. `id`
2. `name`

3. `image_url`
4. `modified_iso`
5. `short_symbol`
6. `interval`

返回当前用户在服务器上保存的图表描述列表。

## loadChartFromServer(chartRecord)

1. `chartRecord`是您使用 `getSavedCharts(callback)` 的对象

从服务器加载并显示图表。

## saveChartToServer(onCompleteCallback, onFailCallback, saveAsSnapshot, options)

1. `onCompleteCallback` : `function()`
2. `onFailCallback` : `function()`
3. `saveAsSnapshot` : should be always `false`
4. `options` : `object { chartName }`
  - i. `chartName` : 图表名称。应指定新图表并重命名。
  - ii. `defaultChartName` : 图表的默认名称。如果当前图表没有名称，它将被使用。

将当前图表保存到服务器。

## removeChartFromServer(chartId, onCompleteCallback)

1. `chartId` : 调用 `getSavedCharts(callback)`后获得的 `id`
2. `onCompleteCallback` : `function()`

从服务器移除图表。

# Custom UI Controls

## onContextMenu(callback)

1. `callback` : `function(unixtime, price)`. 此回调将返回一个值（见下文）。

当用户打开图表上的菜单时，库将调用回调函数。UNIX时间和菜单点的价格将作为参数提供。要自定义菜单项，您必须返回项目描述数组。项目描述对象具有以下结构：

```
{
  position: 'top' | 'bottom',
  text: 'Menu item text',
  click: <onItemClicked callback>
}
```

- `position` : 项目在菜单中的位置
- `text` : 菜单项文本
- `click` : 当用户选择您的菜单项时将被调用

添加分隔符使用减号。例: { text: "-", position: "top" } .

要从菜单中删除现有项目，请在项目文本前面使用减号。例: { text: "-Objects Tree..." }

例:

```
widget.onChartReady(function() {
  widget.onContextMenu(function(unixtime, price) {
    return [{
      position: "top",
      text: "First top menu item, time: " + unixtime + ", price: " + price,
      click: function() { alert("First clicked."); }
    },
    { text: "-", position: "top" },
    { text: "-Objects Tree..." },
    {
      position: "top",
      text: "Second top menu item 2",
      click: function() { alert("Second clicked."); }
    },
    {
      position: "bottom",
      text: "Bottom menu item",
      click: function() { alert("Third clicked."); }
    }
  ]
});
```

## createButton(options)

1. `options : object { align: "left" }`
  - i. `align : "right" | "left". default: "left"`

在图表顶部工具栏中创建一个新的DOM元素，并返回此按钮的jQuery对象。您可以使用它直接在图表上附加自定义控件。例：



```
widget.onChartReady(function() {  
    widget.createButton()  
        .attr('title', "My custom button tooltip")  
        .on('click', function (e) { alert("My custom button pressed!"); })  
        .append($('<span>My custom button caption</span>'));  
});
```

## 对话框

Since 1.6 version

### showNoticeDialog(params)

1. `params` : 对象:
  - i. `title` : 标题
  - ii. `body` : 正文
  - iii. `callback` : 当按下ok按钮时调用的函数。

此方法显示一个对话框，其中包含自定义标题和文本以及“确定”按钮。

### showConfirmDialog(params)

1. `params` : 对象:
  - i. `title` : 标题
  - ii. `body` : 正文
  - iii. `callback(result)` : 当按下ok按钮时调用的函数。 `result` 点击ok时为 `true` , 否则为 `false` 。

此方法显示一个带有自定义标题和文本以及“确定”、“取消”按钮的对话框。

### showLoadChartDialog()

显示加载图表对话框。

### showSaveAsChartDialog()

显示另存为...图表对话框。

## Getters

## symbolInterval(callback)

1. `callback : function(result)`
  - i. `result : object {symbol, interval}`

由于1.4开始方法会立即返回结果。回调是为了保证兼容性。

图表库将调用回调函数，参数对象包含图表商品和时间间隔。

## mainSeriesPriceFormatter()

返回一个带有 `format` 方法的对象，用来批量格式化价格。被引入在1.5.

## getIntervals()

返回支持的分辨率数组。被引入在1.7.

## getStudiesList()

返回全部技术指标数组，您可以通过它们创建技术指标指示器。

# 定制

## addCustomCSSFile(url)

1. `url` 绝对或相对路径的 `static` 文件夹

该方法在版本 1.3 中引入。从1.4开始，使用`custom_css_url`替代。

## applyOverrides(overrides)

该方法在版本 1.5 中引入

1. `overrides` 为一个对象，和`overrides`相同。

此方法在不重新加载图表的情况下将覆盖应用属性。

## applyStudiesOverrides(overrides)

该方法在版本 1.9 中引入

1. `overrides` 为一个对象，和`studies_overrides`相同。

此方法将重写研究的指标样式或输入参数，而无需重新加载图表。

## 交易终端特制

以下方法只在[交易终端](#)可用。

### showSampleOrderDialog(order)

1. `order` : object

显示样品订单对话框。这个对话框看起来像Trading View Paper一样。通常您不需要使用样品对话框。这种方法用于交易样本。

### watchList()

该方法在版本 `1.9` 中引入

返回一个对象来操作观察列表。该对象具有以下方法：

1. `getList()` - 允许您获取当前的商品列表。
2. `setList(symbols)` - 允许您将商品列表设置到观察列表中。它将替换整个列表。
3. `onListChanged()` - 您可以通过订阅[Subscription](#)对象返回此回调函数，通知当观察列表发生变化并退订事件，如果没有观察列表将返回null。

## 多图表布局

### chart(index)

1. `index` : 从0开始的图表索引，默认为0。

返回chart对象，用于调用[Chart-Methods](#)

### activeChart()

返回当前chart对象，用于调用[Chart-Methods](#)

### chartsCount()

返回当前布局的图表数目。

## layout()

返回当前布局模式。可能的值：`4`，`6`，`8`，`s`，`2h`，`2-1`，`2v`，`3h`，`3v`，`3s`。

## setLayout(layout)

1. `layout`：Possible values: `4`，`6`，`8`，`s`，`2h`，`2-1`，`2v`，`3h`，`3v`，`3s`。

变更当前图表布局。

## 也可以看看

- [图表方法](#)
- [定制概述](#)
- [Widget构造函数](#)
- [存储与加载图表](#)
- [覆盖默认研究参数](#)
- [覆盖默认图表参数](#)

## 图表方法

---

以下为图表的方法列表。

在**1.4**版本之前 您可以使用Widget的构造函数返回给您的widget对象来调用这些方法。

从**1.5**版本之后 您可以使用Widget的方法返回给您的图表对象来调用这些方法 [chart\(index\)](#) 或 [activeChart\(\)](#)。

## 方法

- 订阅图表事件
  - [onDataLoaded\(\)](#)
  - [onSymbolChanged\(\)](#)
  - [onIntervalChanged\(\)](#)
  - [dataReady\(callback\)](#)
  - [crossHairMoved\(callback\)](#)
- 图表动作
  - [setVisibleRange\(range, callback\)](#)
  - [setSymbol\(symbol, callback\)](#)
  - [setResolution\(resolution, callback\)](#)
  - [resetData\(\)](#)
  - [executeAction\(action\)](#)
  - [executeActionById\(action\)](#)
  - [getCheckableActionState\(action\)](#)
  - [refreshMarks\(\)](#)
  - [clearMarks\(\)](#)
  - [setChartType\(type\)](#)
- 研究与图形
  - [getAllShapes\(\)](#)
  - [getAllStudies\(\)](#)
  - [setEntityVisibility\(id, isVisible\)](#)
  - [createStudy\(name, forceOverlay, lock, inputs, callback, overrides, options\)](#)
  - [createShape\(point, options, callback\)](#)
  - [createMultipointShape\(points, options, callback\)](#)
  - [removeEntity\(entityId\)](#)
  - [createVerticalLine\(point, options\)](#)

- `removeAllShapes()`
  - `removeAllStudies()`
- 指标模板
  - `createStudyTemplate(options, callback)`
  - `applyStudyTemplate(template)`
- Trading Primitives
  - `createOrderLine()`
  - `createPositionLine()`
  - `createExecutionShape()`
- Getters
  - `symbol()`
  - `symbolExt()`
  - `resolution()`
  - `getVisibleRange()`
  - `getVisiblePriceRange()`
  - `priceFormatter()`
  - `chartType()`

## 订阅图表事件

### onDataLoaded()

您可以使用此方法返回的订阅`Subscription`对象进行订阅，以便在加载新历史K线时通知并取消订阅事件。

### onSymbolChanged()

您可以使用此方法返回的`Subscription`对象进行订阅，以便在更改商品时通知并取消订阅该事件。

### onIntervalChanged()

您可以使用此方法返回的`Subscription`对象进行订阅，以便在更改时间间隔时通知并取消订阅该事件。当事件被触发时，它将提供以下参数：

1. `interval`：新间隔
2. `timeframeParameters`：此对象只有一个字段 `timeframe`。用户改变时间间隔时，它包含一个时间间隔。

否则 `timeframe` 为 `undefined` 然后你可以修改它以显示某一时段的K线。有效的时间间隔是一个数字，字母'D'为天，'M'为月。

例如:

```
widget.chart().onIntervalChanged().subscribe(null, function(interval, obj) {  
    obj.timeframe = "12M";  
})
```

## dataReady(callback)

1. `callback : function(interval)`

如果K线数据已被加载或被接收时，图表库将立即调用此回调。

返回 `true` 为已经加载，否则为 `false`。

## crossHairMoved(callback)

在1.5版本之前

1. `callback : function({time, price})`

每当十字线位置改变时，图表库将会调用回调函数。

## 图表动作

## setVisibleRange(range, callback)

1. `range : object, {from to}`
  - i. `from`, `to` : unix timestamps, UTC
2. `callback : function()` . 图表库会调用回调在viewport(视口)设置完成时。

强制图表调整其参数 (`scroll`, `scale`) 使选定的时间段适合视口。

今后将必须设置 `from` 或 `to`。此方法也引入在 1.2 版本。

## setSymbol(symbol, callback)

1. `symbol : string`
2. `callback : function()`

使图表更改商品。新商品的数据到达后调用回调。

## setResolution(resolution, callback)

1. `resolution : string`. 格式化详细参照:[文章](#)。

2. `callback : function()`

使图表更改分辨率。新分辨率的数据到达后调用回调。

## **resetData()**

使图表重新请求datafeed中的数据。通常你需要在图表数据发生变化时调用它。在调用这个之前，你应该调用[onResetCacheNeededCallback](#)。

## **executeActionById(actionId)**

***since version 1.3***

1. `actionId : string`

通过它的id执行一个动作。

显示对话框

```
chartProperties
compareOrAdd
scalesProperties
tmzProperties
paneObjectTree
insertIndicator
symbolSearch
changeInterval
```

其他动作



```

timeScaleReset
chartReset
seriesHide
studyHide
lineToggleLock
lineHide
showLeftAxis
showRightAxis
scaleSeriesOnly
drawingToolbarAction
magnetAction
stayInDrawingModeAction
lockDrawingsAction
hideAllDrawingsAction
hideAllMarks
showCountdown
showSeriesLastValue
showSymbolLabelsAction
showStudyLastValue
showStudyPlotNamesAction
undo
redo
takeScreenshot
paneRemoveAllStudiesDrawingTools

```

例如:

```

// < ... >
widget.chart().executeActionById("undo");
// < ... >
widget.chart().executeActionById("drawingToolbarAction"); // hides or shows the drawing toolbar
// < ... >

```

## getCheckableActionState(actionId)

从1.7版本之后

1. `actionId` : string

获取可选的操作 (例. `lockDrawingsAction` , `stayInDrawingModeAction` , `magnetAction` ) 状态通过它们的id (请参阅上述动作的ID)

## refreshMarks()

再次请求可见标记。

## clearMarks()

删除所有可见标记。

## setChartType(type)

1. `type` : number

设置主数据列的状态。

```
STYLE_BARS = 0;
STYLE_CANDLES = 1;
STYLE_LINE = 2;
STYLE_AREA = 3;
STYLE_HEIKEN_ASHI = 8;
STYLE_HOLLOW_CANDLES = 9;

STYLE_RENKO* = 4;
STYLE_KAGI* = 5;
STYLE_PNF* = 6;
STYLE_PB* = 7;
```

\*- :chart: available in Trading Terminal

## closePopupsAndDialogs()

调用此方法关闭一个上下文菜单或对话框,假设其已经显示。

# 研究与图形

## getAllShapes()

返回所有已创建的图形对象的数组。每个对象都有以下字段：

- `id` : id of a shape
- `name` : name of a shape

## getAllStudies()

返回所有已创建的图形对象的数组。每个对象都有以下字段：

- `id` : id of a study
- `name` : name of a study

## setEntityVisibility(id, isVisible)

通过id设置实体能见度

## createStudy(name, forceOverlay, lock, inputs, callback, overrides, options)

1. `name` : string, 技术指标指示器，你可以在 技术指标 工具栏中看到。
2. `forceOverlay` : 强制图表库将创建的指标放在主窗格中
3. `lock` : boolean, 是否锁定指标
4. `inputs` : (从版本 1.2 之后) 指标数组参数, 这个数组只包含与指标属性页面打印相同顺序的输入值。
5. `callback` : function( entityId )
6. `overrides` : (从版本 1.2 之后) 一个对象 [containing properties](#) 。注意：您不应指定指标名称：应以具有绘图名称的属性路径为起始。
7. `options` : 这个对象只支持关键字 `checkLimit` 。如果为 `true` 时，超出限制，将显示学习限制对话框。

从1.12开始，函数立即返回结果。回调为保持兼容性

创建一个关于主商品的研究。例子:

- `createStudy('MACD', false, false, [14, 30, "close", 9])`
- `createStudy('Moving Average Exponential', false, false, [26])`
- `createStudy('Stochastic', false, false, [26], null, {"%d.color" : "#FF0000"})`
- `chart.createStudy('Moving Average', false, false, [26], null, {'Plot.linewidth': 10})`

**Remark:** `Compare` 研究有2个参数: `[dataSource, symbol]` . 支持 `dataSource values`: `["close", "high", "low", "open"]` .

**Remark 2:** 当您选择在图表上添加数据列时，您实际使用了 `overlay` 指标，这个指标只有一个参数 -- `symbol` . 以下是添加商品的示例：

```
widget.chart().createStudy('Overlay', false, false, ['AAPL']);
```

**Remark 3:** 当您选择比较数据列时，您实际上使用了 `Compare` 指标。它有2个参数 -- `source` 和 `symbol` . 下面是一个添加比较数据列的例子:

```
widget.chart().createStudy('Compare', false, false, ["open", 'AAPL']);
```

## getStudyById(entityId)

1. `entityId` : object. 通过API创建研究时返回的值。

使用以下方法返回一个对象与研究交互：

1. `isUserEditEnabled()` - 如果用户能够删除/更改/隐藏您的形状，则返回 `true`
2. `setUserEditEnabled(enabled)` - 启用或禁用删除/更改/隐藏 用户的研究
3. `getInputsInfo()` - 返回有关所有输入的信息。返回值是具有以下字段的对象数组：
  - `id` - 研究ID
  - `name` - 名称
  - `type` - 类型
  - `localizedName` - 输入翻译成当前语言的名称
4. `getInputValues()` - 返回研究输入的值。返回值是一个对象数组（`StudyInputValue`），它包含以下字段：
  - `id` - 研究ID
  - `value` - 值
5. `setInputValues(inputs)` - 将输入值分配给研究。 `inputs` 应该是一个包含 `StudyInputValue` 对象的数组（见上文）。它可能只包含一些你想改变的输入。
6. `mergeUp()` - 向上合并（如果可以）
7. `mergeDown()` - 向下合并（如果可以）
8. `unmergeUp()` - 向上分解（如果可以）
9. `unmergeDown()` - 向下分解（如果可以）

## createShape(point, options, callback)

1. `point : object {time, [price], [channel]}`
  - i. `time` : `unix time`. 唯一的强制性参数。
  - ii. `price` : 如果您指定 `price`，如果您指定“`price`”，则您的图标将被放置在其水平之上。如果没有指定，则图标会在相应的时间粘贴到K线上。
  - iii. `channel` : 要保持价格水平线，要使用 `channel` 参数（`open`，`high`，`low`，`close`）。如果未指定则以'`open`'为默认值。
2. `options : object {shape, [text], [lock], [overrides]}`
  - i. `shape` 可能的值为['`arrow_up`', '`arrow_down`', '`flag`', '`vertical_line`', '`horizontal_line`']，'`flag`'为默认值。
  - ii. `text` 图形的内容
  - iii. `lock` 是否锁定图形
  - iv. `disableSelection` (since 1.3) 禁用选择
  - v. `disableSave` (since 1.3) 禁用保存
  - vi. `disableUndo` (since 1.4) 禁用撤销
  - vii. `overrides` (since 1.2). 它是一个对象，包含为新图形设置的属性。
  - viii. `zOrder` (since 1.3) 可能的值为[ `top`，`bottom` ]。 `top` 将线工具放在所有其他资源之上，`bottom` 将线工具放在所有其他资源之下，'`top`'为默认值。
  - ix. `showInObjectsTree` : `true` 为默认值。在“对象树”对话框中显示图形。

### 3. `callback : function( entityId )`

从**1.4**开始，函数立即返回结果。回调函数为兼容性。

此调用会在主数据列上指定点位创建一个形状。

## **createMultipointShape(points, options, callback)**

1. `point : object {time, [price], [channel]}`
  - i. `time : unix time`. 唯一的强制性参数。
  - ii. `price` : 如果您指定 `price` , 如果您指定“`price`”, 则您的图标将被放置在其水平之上。如果没有指定, 则图标会在相应的时间粘贴到K线上。
  - iii. `channel` : 要保持价格水平线, 要使用 `channel` 参数 ( `open` , `high` , `low` , `close` )。如果未指定则以'`open`'为默认值。
2. `options : object {shape, [text], [lock], [overrides]}`
  - i. `shape` 可能的值为['`arrow_up`', '`arrow_down`', '`flag`', '`vertical_line`', '`horizontal_line`'], '`flag`'为默认值。
  - ii. `text` 图形的内容
  - iii. `lock` 是否锁定图形
  - iv. `disableSelection` (since 1.3 ) 禁用选择
  - v. `disableSave` (since 1.3 ) 禁用保存
  - vi. `disableUndo` (since 1.4 ) 禁用撤销
  - vii. `overrides` (since 1.2 ). 它是一个对象, 包含为新图形设置的属性。
  - viii. `zOrder` (since 1.3 ) 可能的值为[ `top` , `bottom` ]. `top` 将线工具放在所有其他资源之上, `bottom` 将线工具放在所有其他资源之下, '`top`'为默认值。
  - ix. `showInObjectsTree` : `true` 为默认值。在“对象树”对话框中显示图形。
3. `callback : function( entityId )`

从**1.4**开始，函数立即返回结果。回调函数为兼容性。

查看[形状与覆盖](#)了解更多信息。

此调用会在主数据列上指定几个点位创建一个图形。

## **removeEntity(entityId)**

1. `entityId : object`. 值为创建实体 (图形的研究) 后通过回调传递的值。

删除指定实体。

## **createVerticalLine(point, options)**

1. `point : object {time}`

## 2. `options : object {lock}`

此方法为 `createShape` 的同义词，并且 `shape = 'vertical_line'`. 它被视为 已过时.

## `removeAllShapes()`

删除全部图形(绘图)。

## `removeAllStudies()`

删除全部指标。

# 指标模板

## `createStudyTemplate(options, callback)`

1. `options : object {saveInterval}`
  - i. `saveInterval : boolean`
2. `callback : function(data)`

从**1.4**开始，函数立即返回结果。回调函数为持兼容性。

将指标模板保存到JS对象。图表库将调用您的回调函数并将状态对象作为参数传递。该调用为低级的一部分 [存储与加载图表](#)。

## `applyStudyTemplate(template)`

1. `template : object`

从状态对象加载指标模板。这个调用是低级的一部分。 [存储与加载图表](#)。

# 交易元语（Trading Primitives）

## `createOrderLine(options)`

在图表上创建一个新订单，并返回一个API对象，您可以使用它来控制订单属性和行为。强烈推荐阅读[交易元语](#)在调用此方法之前。

参数 (自1.4):

`options` 是一个具有：`disableUndo` 的对象, 这可以是 `true` 或 `false` . 出于兼容性原因，默认值为 `false` 。

API对象方法：

- `remove()`：从图表中移除位置。调用此方法后不能再使用API对象。
- `onModify(callback)` / `onModify(data, callback)`
- `onMove(callback)` / `onMove(data, callback)`

API对象具有以下列出的一组属性。每个属性都可以通过各自的访问器调用。即，如果要使用 `Extend Left` 属性，请使用 `getExtendLeft()` 或 `setExtendLeft()` 方法。

一般属性:

Property	Type	Supported Values	Default Value
Price	Double	Double	0.0
Text	String	String	""
Tooltip	String	String	""
Quantity	String	String	""
Editable	Boolean	Boolean	true

趋势线属性:

Property	Type	Supported Values	Default Value
Extend Left	Boolean	"inherit" or Boolean	True
Line Length	Integer	"inherit" or 0 .. 100	0
Line Style	Integer	"inherit" or 0 .. 2	2
Line Width	Integer	"inherit" or 1 .. 4	1

字体:

Property	Type	Default Value
Body Font	String	"bold 7pt Verdana"
Quantity Font	String	"bold 7pt Verdana"

颜色:

Property	Type	Default Value
Line Color	String	"rgb(255, 0, 0)"
Body Border Color	String	"rgb(255, 0, 0)"
Body Background Color	String	"rgba(255, 255, 255, 0.75)"
Body Text Color	String	"rgb(255, 0, 0)"
Quantity Border Color	String	"rgb(255, 0, 0)"
Quantity Background Color	String	"rgba(255, 0, 0, 0.75)"
Quantity Text Color	String	"rgb(255, 255, 255)"
Cancel Button Border Color	String	"rgb(255, 0, 0)"
Cancel Button Background Color	String	"rgba(255, 255, 255, 0.75)"
Cancel Button Icon Color	String	"rgb(255, 0, 0)"

例子:

```

widget.chart().createOrderLine()
  .onMove(function() {
    this.setText("onMove called");
  })
  .onModify("onModify called", function(text) {
    this.setText(text);
  })
  .onCancel("onCancel called", function(text) {
    this.setText(text);
  })
  .setText("STOP: 73.5 (5,64%)")
  .setQuantity("2");

```

## createPositionLine(options)

在图表上创建一个新的位置并返回一个API对象，您可以使用它来控制位置属性和行为。强烈推荐阅读[交易元语](#)在调用此方法之前。

参数 (自 1.4):

`options` 是一个具有：`disableUndo` 的对象，这可以是 `true` 或 `false`。出于兼容性原因，默认值为 `false`。

API对象方法：

- `remove()`：从图表中移除位置。调用此方法后不能再使用API对象。
- `onModify(callback)` / `onModify(data, callback)`
- `onMove(callback)` / `onMove(data, callback)`



API对象具有以下列出的一组属性。每个属性都可以通过各自的访问器调用。即，如果要使用 `Extend Left` 属性，请使用 `getExtendLeft()` 或 `setExtendLeft()` 方法。

一般属性:

Property	Type	Supported Values	Default Value
Price	Double	Double	0.0
Text	String	String	""
Tooltip	String	String	""
Quantity	String	String	""

趋势线属性:

Property	Type	Supported Values	Default Value
Extend Left	Boolean	"inherit" or Boolean	True
Line Length	Integer	"inherit" or 0 .. 100	0
Line Style	Integer	"inherit" or 0 .. 2	2
Line Width	Integer	"inherit" or 1 .. 4	1

字体:

Property	Type	Default Value
Body Font	String	"bold 7pt Verdana"
Quantity Font	String	"bold 7pt Verdana"

颜色:

Property	Type	Default Value
Line Color	String	"rgb(0, 113, 224)"
Body Border Color	String	"rgb(0, 113, 224)"
Body Background Color	String	"rgba(255, 255, 255, 0.75)"
Body Text Color	String	"rgb(0, 113, 224)"
Quantity Border Color	String	"rgb(0, 113, 224)"
Quantity Background Color	String	"rgba(0, 113, 224, 0.75)"
Quantity Text Color	String	"rgb(255, 255, 255)"
Reverse Button Border Color	String	"rgb(0, 113, 224)"
Reverse Button Background Color	String	"rgba(255, 255, 255, 0.75)"
Reverse Button Icon Color	String	"rgb(0, 113, 224)"
Close Button Border Color	String	"rgb(0, 113, 224)"
Close Button Background Color	String	"rgba(255, 255, 255, 0.75)"
Close Button Icon Color	String	"rgb(0, 113, 224)"

例子:

```
widget.chart().createPositionLine()  
  .onModify(function() {  
    this.setText("onModify called");  
  })  
  .onReverse("onReverse called", function(text) {  
    this.setText(text);  
  })  
  .onClose("onClose called", function(text) {  
    this.setText(text);  
  })  
  .setText("PROFIT: 71.1 (3.31%)")  
  .setQuantity("8.235")  
  .setPrice(15.5)  
  .setExtendLeft(false)  
  .setLineStyle(0)  
  .setLineLength(25);
```

## createExecutionShape(options)

在图表上创建一个新的执行，并返回一个可以用来控制执行属性的API对象。 在使用此呼叫之前，强烈建议您阅读[交易元语](#)。

参数 (自 1.4):

`options` 是一个具有: `disableUndo` 的对象, 这可以是 `true` 或 `false` . 出于兼容性原因, 默认值为 `false` 。

API对象具有以下列出的一组属性。 每个属性都可以通过各自的访问器调用。 即, 如果要使用 `Extend Left` 属性, 请使用 `getExtendLeft()` 或 `setExtendLeft()` 方法。

API对象方法:

- `remove()`: 从图表中删除执行信号图形。 此调用后不能使用API对象。

一般属性:

Property	Type	Supported Values	Default Value
Price	Double	Double	0.0
Time	Integer	Unix time	0
Direction	String	"buy" or "sell"	"buy"
Text	String	String	"execution"
Tooltip	String	String	""
Arrow Height	Integer	Integer	8
Arrow Spacing	Integer	Integer	1

字体:

Property	Type	Default Value
Font	String	"8pt Verdana"

颜色:

Property	Type	Default Value
Text Color	String	"rgb(0, 0, 0)""
Arrow Color	String	"rgba(0, 0, 255)"

例子:

```
widget.chart().createExecutionShape()  
    .setText("@1,320.75 Limit Buy 1")  
    .setTooltip("@1,320.75 Limit Buy 1")  
    .setTextColor("rgba(0,255,0,0.5)")  
    .setArrowColor("#0F0")  
    .setDirection("buy")  
    .setTime(1413559061758)  
    .setPrice(15.5);
```

# Getters

## symbol()

返回图表商品。

## symbolExt()

返回图表的商品信息对象。该对象具有以下字段：

- `symbol` : the same as `symbol()` method result
- `full_name` : 商品全称
- `exchange` : 商品交易所
- `description` : 商品描述
- `type` : 商品类型

## resolution()

返回图表的分辨率。格式在这个[文章](#)中描述。

## getVisibleRange()

返回对象 `{from, to}`。 `from` 和 `to` 是图表时区的单位时间戳

## getVisiblePriceRange()

Since 1.7

返回对象 `{from, to}`。 `from` 和 `to` 主数据列的可见数据内容。

## priceFormatter()

返回 `format` 方法，用以格式化价格。在1.5中引入。

## chartType()

返回图表类型。

也可以看看

- [Widget方法](#)
- [定制概述](#)
- [Widgetg构造函数](#)
- [存储于加载图表](#)
- [覆盖默认研究参数](#)
- [覆盖默认图表参数](#)

# 功能集

功能 或 功能集 是图表功能的一部分。有简单（原子）和复杂（复合）特征。复杂特征由简单特征组成。禁用复合功能会使得其所有简单的部件都被禁用。支持的功能如下所示。

请注意，下表中的 - 字符不是功能集名称的一部分。

## 控件和其他视觉元素的可见性

### 互动地图的功能

ID	默认状态	库版本	描述
header_widget	on		
- header_widget_dom_node	on		禁用此功能会隐藏头部小部件DOM元素
- header_symbol_search	on		
- symbol_search_hot_key	on	1.9	商品搜索热键
- header_resolutions	on		
- - header_interval_dialog_button	on		
- - - show_interval_dialog_on_key_press	on		
- header_chart_type	on		
- header_settings	on		涉及图表属性按钮
- header_indicators	on		
- header_compare	on		
- header_undo_redo	on		
- header_screenshot	on		
- header_fullscreen_button	on		
compare_symbol	on	1.5	您可以使用此功能集从上下文菜单中删除“比较/覆盖”对话框
border_around_the_chart	on		

header_saveload	on		它不是 header_widget 的一部分
left_toolbar	on		
control_bar	on		涉及底部的导航按钮
timeframes_toolbar	on		
<b>edit_buttons_in_legend</b>	on		
- show_hide_button_in_legend	on	1.7	
- format_button_in_legend	on	1.7	
- study_buttons_in_legend	on	1.7	
- delete_button_in_legend	on	1.7	
<b>context_menus</b>	on		
- pane_context_menu	on		
- scales_context_menu	on		
- legend_context_menu	on		
main_series_scale_menu	on	1.7	显示图表右下角的设置按钮
display_market_status	on		
remove_library_container_border	on		
chart_property_page_style	on		
property_pages	on	1.11	禁用所有属性页
show_chart_property_page	on	1.6	关闭禁用属性
chart_property_page_scales	on		
chart_property_page_background	on		
chart_property_page_timezone_sessions	on		
chart_property_page_trading	on		它只适用于交易终端
countdown	on	1.4	倒计时标签价格规模
caption_buttons_text_if_possible	off	1.4	在可能的情况下，在标题中的“指标”和“比较”按钮上显示文字而不是图标
dont_show_boolean_study_arguments	off	1.4	是否隐藏指标参数
hide_last_na_study_output	off	1.4	隐藏最后一次指标输出
symbol_info	on	1.5	商品信息对话框
timezone_menu	on	1.5	商品信息对话框禁用时区上下文菜单

snapshot_trading_drawings	off	1.6	包含屏幕截图中的订单/位置/执行信号
source_selection_markers	on	1.11	禁用数据列和指标指示器的选择标记
keep_left_toolbar_visible_on_small_screens	off	1.11	防止左侧工具栏在小屏幕上消失

## 元素放置

ID	默认状态	库版本	描述
move_logo_to_main_pane	off		将标志放在主数据列窗格上，而不是底部窗格
header_saveload_to_the_right	off		向右移动保存并加载按钮

## 行为

ID	默认状态	库版本	描述
<b>use_localstorage_for_settings</b>	on		允许将用户设置保存到 localStorage
- items_favoriting	on		禁用此功能会隐藏所有“收藏此项目”按钮
- save_chart_properties_to_local_storage	on		禁用此功能可防止将图表属性（颜色，样式，字体）保存到本地存储，但仍保存最喜欢的项目
create_volume_indicator_by_default	on		
create_volume_indicator_by_default_once	on		
volume_force_overlay	on		在与主数据列相同的窗格中放置成交量指示器
right_bar_stays_on_scroll	on		确定缩放行为：禁用光标下的K线
constraint_dialogs_movement	on		阻止从图表中移动对话框
charting_library_debug_mode	off		启用日志
show_dialog_on_snapshot_ready	on		禁用此功能允许您以静默方式进行快照



study_market_minimized	on		涉及“指标”对话框，确定它是否紧凑或包含搜索栏和类别
study_dialog_search_control	on	1.6	在指标对话框中显示搜索对话框
side_toolbar_in_fullscreen_mode	off		使用此功能，您可以在全屏模式下启用绘图工具栏
same_data_requery	off		允许您使用相同的商品调用 setSymbol 来刷新数据
disable_resolution_rebuild	off		显示的时间与DataFeed提供的时间完全一致，而不进行对齐。如果您希望图表构建一些分辨率，则不建议使用此方法。
chart_scroll	on	1.10	允许滚动图表
chart_zoom	on	1.10	允许缩放图表
high_density_bars	off	1.11	允许缩小在1个屏幕上显示超过60000条K线
cl_feed_return_all_data	off	1.11	允许您从Feed中返回更多的条，并立即在图表上显示

## 大石块(Big Rocks)

ID	默认状态	库版本	描述
	study_templates	off	
datasource_copypaste	on		允许复制图纸和研究
seconds_resolution	off	1.4	支持秒分辨率

## 交易终端

ID	默认状态	终端版本	描述
support_multicharts	on		启用与多图表布局相关的上下文菜单操作（克隆，同步）
header_layouttoggle	on		显示标题中的“选择布局”按钮
show_logo_on_all_charts	off		在多功能布局的每个图表上显示徽标
chart_crosshair_menu	on	1.7	在价格范围内启用"加号"按钮进行快速交易
add_to_watchlist	on	1.9	在菜单中启用“添加商品到观察列表”项
footer_screenshot	on	1.11	显示页脚中的截图按钮（客户经理）
open_account_manager	on	1.11	默认情况下保留客户经理的打开

## 定制的使用案例

图表库允许您自定义外观、数据显示的方式、默认属性等等。

客户端和服务端的定制，其中一些是通过构造函数，其他的可以使用widget或图表方法实现。

这里是唯一可以找到最常用的定制链接和描述的地方。

### 默认仪表和分辨率

更改默认商品（仪表）和分辨率（间隔）。

最小支持的分辨率为1秒。

[文档说明](#)

### 默认可见范围 (时间范围)

更改默认分辨率K线的时间范围

[文档说明](#)

### 分辨率的默认可见范围

当用户更改分辨率时，更改K线的时间范围。看这里的样本：

[文档说明](#)

### 初始时区

您可以设置默认使用的时区。用户也可以在菜单中更改。

[文档说明](#)

### 图表大小

您可以将图表作为元素放置在网页上或使用全屏模式。

[宽度和高度](#)

[全屏模式](#)

## 自动尺寸

### 图表颜色

自定义图表的颜色，使其完美适合您的网站。

1. 工具栏颜色 - [文档说明](#)
2. 图表颜色 - [文档说明](#)

### 指标

1. 限制1个图表布局的指标量 - [文档说明](#)
2. 限制显示和可以添加的指标 - [文档说明](#)
3. 在服务器上添加您自己的指标 - [文档说明](#)
4. 更改指标的默认属性 - [文档说明](#)
5. 更改默认属性（立即生效） - [文档说明](#)

### 绘图

1. 限制哪些绘图可以被显示或被添加 - [文档说明](#)
2. 更改绘图的默认属性 - [文档说明](#)
3. 更改默认属性（立即生效） - [文档说明](#)

### 语言

选择图表图书馆20多个翻译中的一个。[文档说明](#)

注意：语言是在创建图表时设置的。如果没有重新创建图表，就无法更改。

### 数字和日期的格式化

1. 更改十进制数字 - [文档说明](#)
2. 为数据和时间设置自定义格式化方法 - [文档说明](#)
3. 价格根据商品信息进行格式化 - [文档说明](#)

### 图表的默认属性

您可以更改属性对话框中显示的任何属性。

1. 初始化 - [文档说明](#)
2. 立即生效 - [文档说明](#)

## 服务器的快照

TradingView允许您在其服务器上保存快照，但如果您希望您也可以更改它。

[文档说明](#)

## 显示/隐藏图表的元素

如果您不需要图表的某些元素 (工具栏、按钮或其他控件), 您可以隐藏它们。

1. 大多数图表元素可以通过[功能集](#)使用shown/hidden
2. 您可以添加自己的CSS - [文档说明](#)

## 图表底部的时间范围

时间范围是K线的时间段和优先显示时段的分辨率。 您可以自定义列表。

[文档说明](#)

## 收藏的间隔/图表样式的初始化列表

默认情况下，您可以选择在顶部工具栏上显示什么间隔和图表样式。如果在 [功能集](#) 中设置 `items_favoriting` 为 `enabled`，则允许用户改变他们。

[文档说明](#)

## 菜单中显示分辨率

1. 在datafeed的配置对象中提供了完整的分辨率列表 - [文档说明](#)
2. 根据商品信息在列表中启用或禁用分辨率 - [文档说明](#)
3. 可以设置喜欢的分辨率的初始列表 - [文档说明](#)

## 成交量指标

尽管有其他指标，如果仪表支持，则默认添加成交量指标 [文档说明](#)。

您可以禁用此行为 [功能集](#)

## 上下文菜单

您可以向上下文菜单添加新元素或隐藏现有项目。

[文档说明](#)

## 定制工具栏上的按钮

您可以将自己的按钮添加到图表的顶部工具栏上。

[文档说明](#)

### 观察列表

可以为观察列表选择默认商品，并根据需要设置只读状态。


[文档说明](#)

### 新闻资讯


您可以附加任何RSS订阅，甚至可以根据金融工具类型选择feed。

[文档说明](#)

## 交易终端

 此页面上的所有内容仅与交易终端相关。

交易终端是一个即用型产品，为那些想要有一个全功能图表解决方案，以及从图表交易的能力。This 产品基于图表库，并包含其所有功能，而且还包含一大堆新功能。交易终端资源[链接](#)。

该产品正在开发中，因此一些功能还没有。它们被标记为：

## 交易终端功能

### 交易功能

您可以通过图表进行交易，您需要做的所有工作就是实现您的[经纪商API](#)并将其接入到图表 widget。



### 高级订单对话框

完全可定制的订单对话框允许设置市价/限价/止损/限价止损 订单，输入止损和获利价格，选择到期日和计算风险。

FX:USDJPY, US Dollar vs Japanese Yen

Bid

103.298

SELL

0.5

Ask

103.303

BUY

MarketLimitStop

Price

PricePips103.309=1.1 Pips

☒ Stop Loss

PricePips25.0=103.559

☒ Take Profit

PricePips75.0=102.559

Amount

Manual% Risk100000=0.26 %

Risk 0.26% (\$242.03)  
Risk/reward ratio: 3.00  
Reward 0.77% (\$726.10)

Sell

账户管理器

在底部的交互式表格中显示订单/仓位和帐户信息，也可以在其中嵌入任何其他小部件。

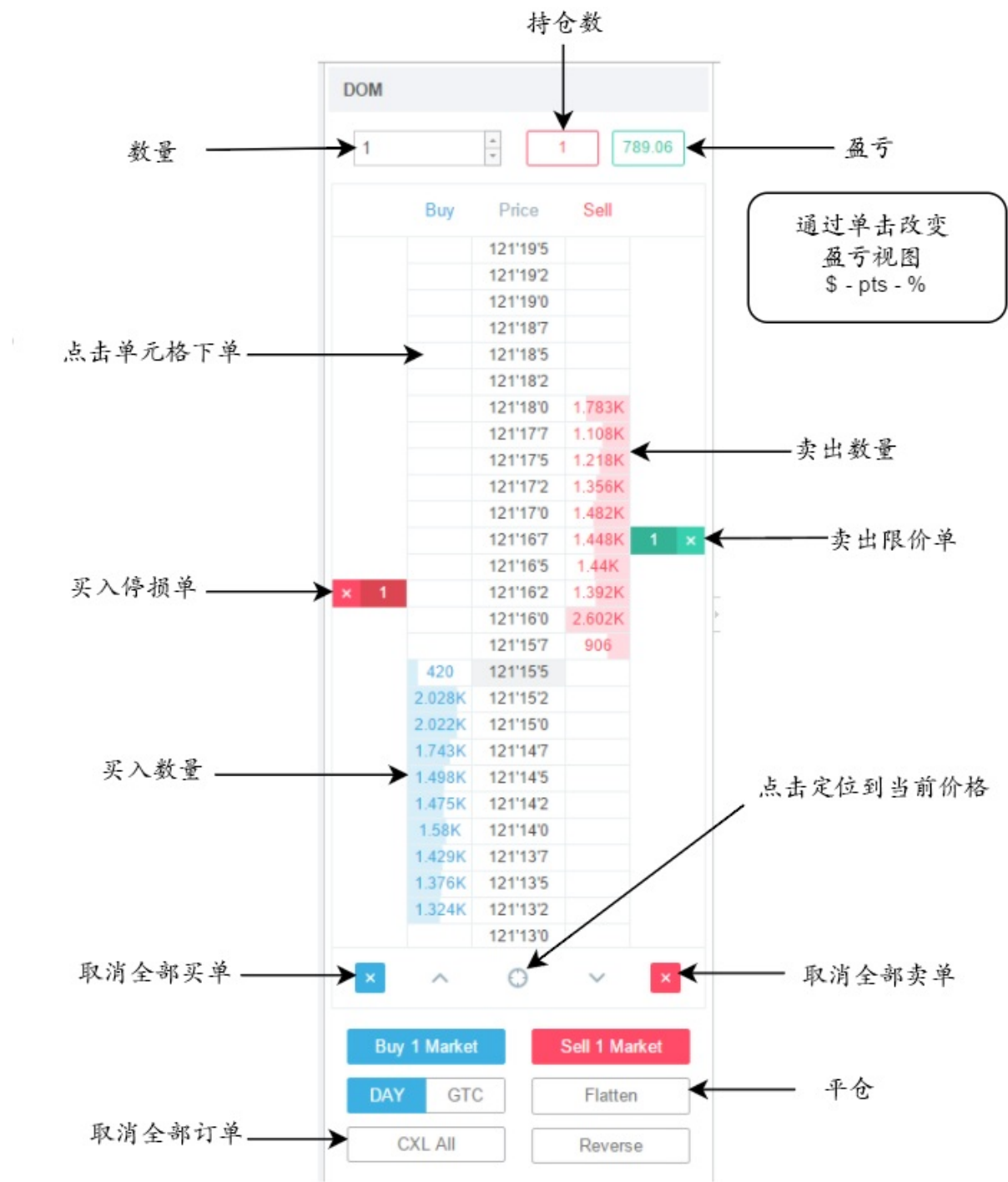
详细了解此功能：

- [如何启用账户管理器](#)

DOM小部件

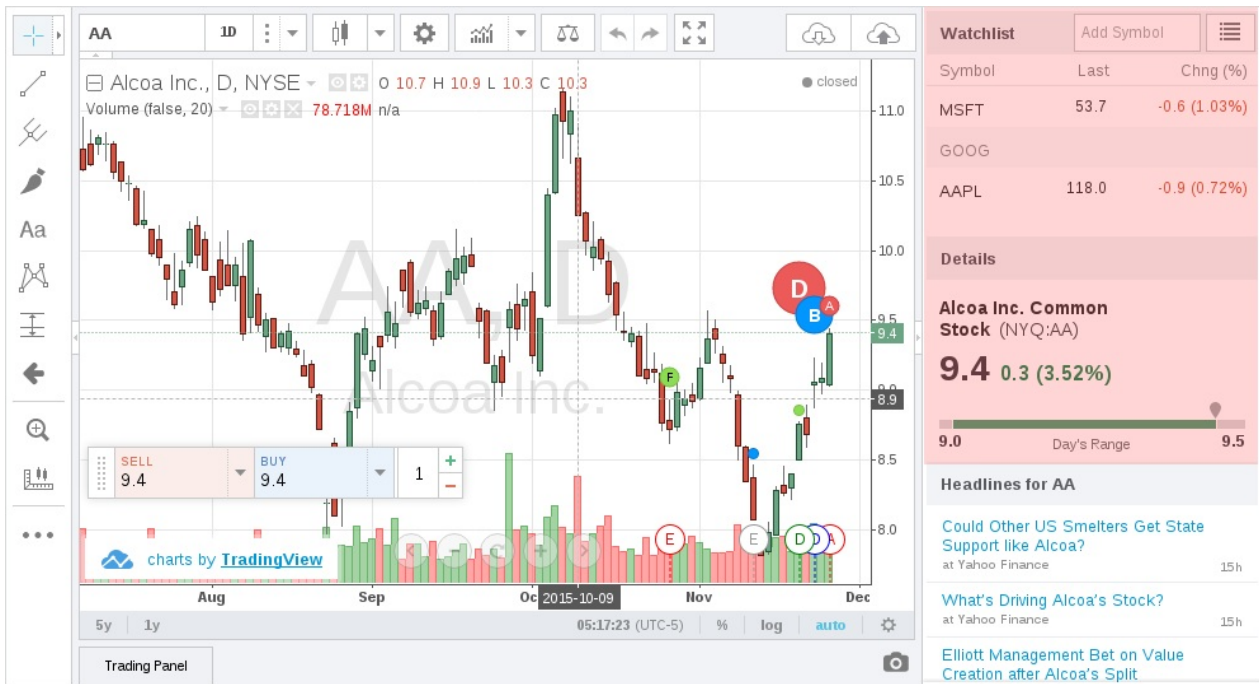
您可以在交互式DOM中显示订单/持仓以及Level-2数据。





### 侧边栏报价（商品详情和观察列表）

在交易终端中，您可以拥有观察列表和商品详情窗口小部件（请参阅下面的快照）功能。

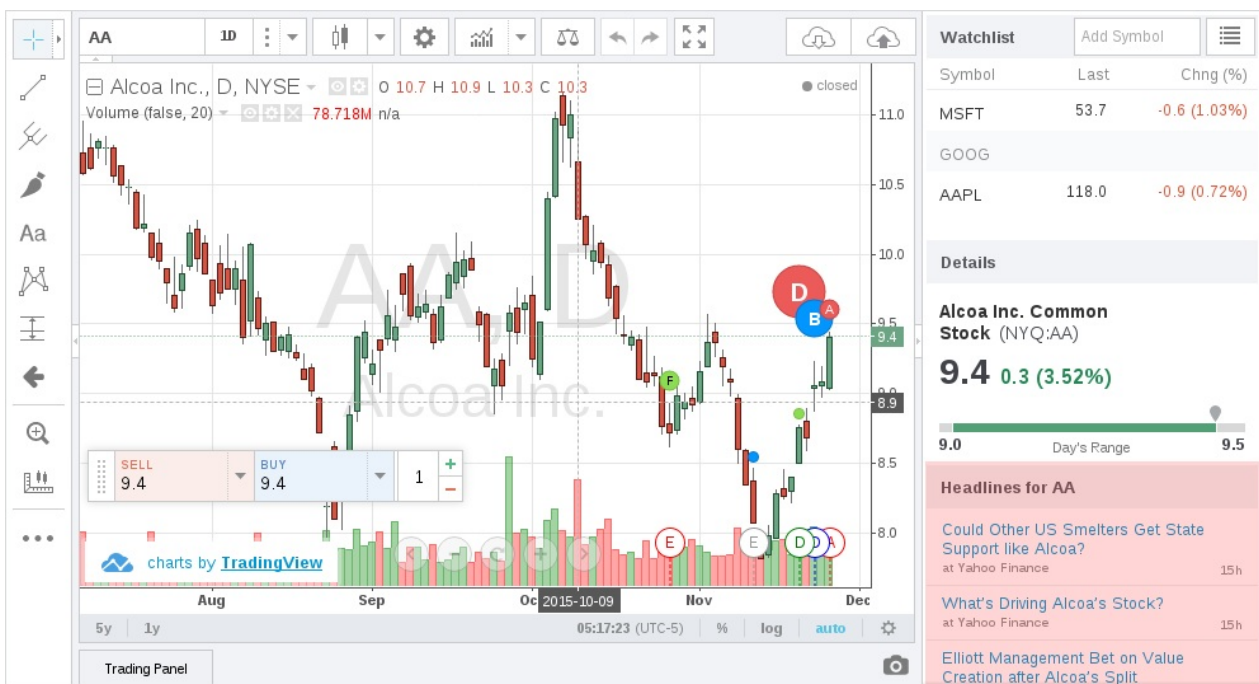


详细了解此功能：

- 如何启用侧边栏报价
- 如何提供报价数据：取决于您使用什么样的数据集成 --JS API 或 UDF

## 侧边栏市场新闻Feed

您可以在图表的侧栏中直接显示新闻提要。我们对新闻Feed的支持是灵活的：例如，您可以为不同类型的商品提供不同的Feed，等等。

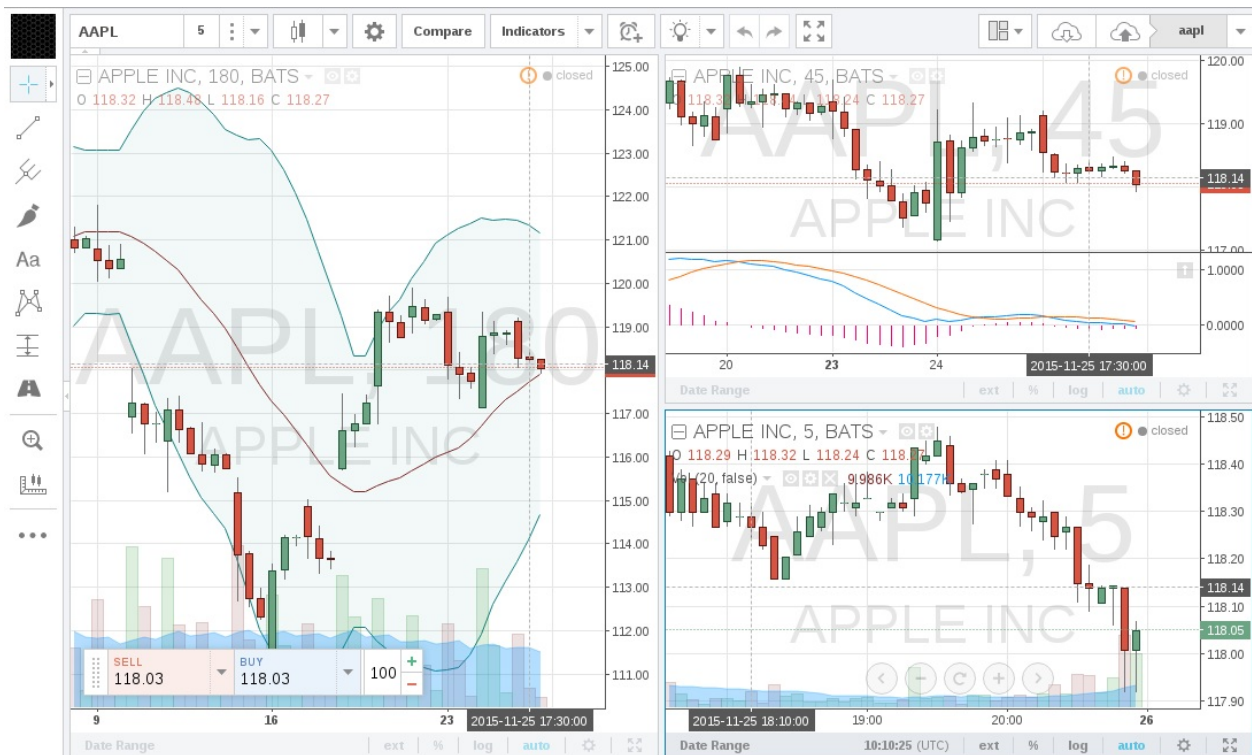


详细了解此功能：

- 如何启用侧边栏新闻
- 如何设置要使用的Feed

## 多图表布局

在同一个小程序中，您可以同时显示多个图表。这使您的用户能够使用更广泛的策略，以及更广泛的市场观点的能力。您不必做任何事情来启用或调整它：它可以开箱即用。



## 日本图表类型：卡吉图、砖形图、OX（点数）图、新价线

这些类型的图表将可以开箱即用，就像Heikin Ashi（平均K线图）在图表库中可用一样。


## 量能分布图：🕒

这项研究将需要一些服务器端的支持。准备好后，我们会提供更多的细节。

## 绘图工具模板：🕒

此功能将需要您的后端支持。我们将更新我们的开源数据后端以支持此功能，因此请考虑使用它来最大限度地减少您的工作。


## 如何使用文档

由于交易终端基于图表库，我们决定将文档合并到单个维基中。所以所有的文档都在一个地方。您唯一应该记住的是，特定于交易终端的功能标有这个可爱的绿色标记：。

## 也可以看看

- [如何将您的交易接口连接到图表](#)
- [交易终端专用的Widget方法](#)
- [用于交易终端的Widget构造函数参数](#)

# 交易控制器

 此页面上的所有内容仅适用于[[交易终端]]。

交易控制器是一种能使您在线交易的东西。其主要目的是将我们的图表与您的交易逻辑连接起来。就 JS 而言,它是一个 `object` 以便公开特定的接口。下面是一个终端控制器的方法。

## 必需的方法

### setHost(host)

这种方法在初始化时通过[[交易主机]]传递给控制器。

### configFlags()

实现这个方法提供配置标志对象。结果是具有以下键的对象，即可以 `true / false`：

- `supportReversePosition`

经纪商支持反向交易。

如果经纪商不支持，则图表将具有相反的按钮，但会按相反的顺序排列。

- `supportClosePosition`

经纪商支持平仓。

如果经纪人不支持，图表将会有关闭按钮，但会按顺序排列。

- `supportReducePosition`

经纪商支持在没有订单的情况下更换仓位。

- `supportPLUpdate`

经纪人提供PL的头寸。如果经纪商自己计算利润/损失，它应该调用 `[[plUpdate|Trading-Host#plupdatepositionid-pl]]` 一旦PL改变了。否则图表将计算PL作为当前交易与头寸平均价格之间的差额。

- `supportBrackets`

经纪支持括号单（获利和止损单）。如果此标志为 `true`，则图表将显示仓位的编辑按钮，并将 `Edit position...` 添加到仓位的上下文菜单。

- **supportMultiposition**

支持多重仓位可防止为反向交易创建默认实现。

- **supportCustomBottomWidget**

此标志可用于使用自定义更改默认的帐户管理器小部件。

- **showQuantityInsteadOfAmount**

该标志可用于在订单对话框中将“金额”更改为“数量”

- **supportDOM**

此标志用来启用DOM小部件。如果 `supportLevel2Data` 为 `false` 只有 `last price` 并且订单将被显示。

- **supportLevel2Data**

Level2数据用于DOM小部件。 `subscribeDepth` 和 `unsubscribeDepth` 应该被实现。

- **supportStopLimitOrders**

此标志向订单对话框添加止损订单类型。

- **supportMarketBrackets**

使用此标志可以为市场订单禁用括号单。默认情况下启用。

- **supportEditAmount**

此标志可帮助您在编辑现有订单时禁用金额控制。

## **durations(): array of objects**

订单过期期权清单。这是可选的。如果您不希望在订单中显示持续时间，请勿实现。

对象有两个key： `{ name, value }`。

例：

```
return [{ name: 'DAY', value: 'DAY' }, { name: 'GTC', value: 'GTC' }];
```

头寸：**Deferred**

订单：**Deferred**

执行信号(symbol)：**Deferred**

图表调用这些方法来请求位置/订单/执行并在图表上显示它们。

您应该返回相应的列表[[positions|Trading-Objects-and-Constants#position]],  
[[orders|Trading-Objects-and-Constants#order]] or [[executions|Trading-Objects-and-Constants#execution]]

## supportFloatingPanel()

此方法将返回 `true` ,以显示浮动交易面板。

## supportBottomWidget()

此方法将返回 `true` ,以显示底层交易面板。

## chartContextMenuItems(e)

图表可以在上下文菜单中有子菜单 `交易` 。返回子菜单项的列表。格式是一样的 `buttonDropdownItems` 。 `e` 是浏览器传递的上下文对象

## bottomContextMenuItems()

底部交易面板可以有一个上下文菜单。返回此菜单的项目列表。格式与 `buttonDropdownItems` 相同。

## isTradable(symbol)

此方法是浮动交易面板所必需的。是否可以访问交易面板取决于这个函数的结果：`true` 或 `false` 。如果所有符号都可以交易，则不需要使用此方法。

## createBottomWidget(container)

当创建底部交易面板时需要调用这个函数。您应该创建DOM对象并将其附加到 `container` 中。容器在需要时显示一个垂直滚动条。

## accountManagerInfo()

当`supportCustomBottomWidget`为 `false` 时，调用此方法。返回用于构建帐户管理器的信息。有关详细信息，请参阅\[\[帐户管理器]]。

## showOrderDialog([[order|Trading-Objects-and-Constants#order]])

当用户请求创建或修改订单时，图表会调用此函数。

所以我们给您一个自己的对话框，它100%由您管理。

## **placeOrder([[order|Trading-Objects-and-Constants#order]], silently)**

当用户想要下订单时调用方法。订单预先填写了部分或全部信息。如果 `silently` 为 `true`，则不显示任何对话框。

## **modifyOrder([[order|Trading-Objects-and-Constants#order]], silently, focus)**

`order` 是要修改的订单对象 2. `silently` - 如果是 `true`，则不显示任何对话框 3. `focus` - `\[Focus constant \ | Trading-Objects-and-Constants \ #focusoptions ]]`。它可以由图表初始化。

1. `order` 是要修改的订单对象
2. `silently` - 为 `true` 时，不显示任何对话框
3. `focus` - `[[Focus constant|Trading-Objects-and-Constants#focusoptions]]`. 它可以由图表初始化。

当用户要修改现有的订单时调用方法。

## **cancelOrder(orderId, silently)**

这个方法被调用来取消指定 `id` 的订单。如果 `silently` 为 `true`，则不显示任何对话框。

## **cancelOrders(symbol, side, ordersIds, silently)**

1. `symbol` - 商品字符串
2. `side` : "sell" 或 "buy"
3. `ordersIds` - 已经被 `symbol` 和 `side` 收集的订单id列表 如果 `silently` 为 `true`，则不显示任何对话框。

此方法被调用以取消 `symbol` 和 `side` 的多个订单。

## **editPositionBrackets(positionId, focus)**

如果 `supportBrackets` 配置标志开启显示用于编辑利润和止损的对话框，则调用此方法。

1. `positionId` 是要修改的现有位置的ID
2. `focus` - `[[Focus constant|Trading-Objects-and-Constants#focusoptions]]`.



## closePosition(positionId, silently)

如果 `supportBrackets` 配置标志开启显示用于平仓的对话框，则调用此方法。如果 `silently` 为 `true`，则不显示任何对话框。

## reversePosition(positionId, silently)

如果“supportReversePosition”配置标志打开，则通过id来反转头寸，将调用此方法。如果 `silently` 为 `true`，则不显示任何对话框。

## symbolInfo(symbol) : Deferred (or Promise)

1. `symbol` - 商品字符串

该方法由内部Order Dialog，DOM面板和浮动交易面板调用以获取商品信息。结果是具有以下数据的对象：对象具有 `min`，`max` and `step` 指定数量字段步骤和边界的字段“min”，“max”和“step”。

- `qty` - object with fields `min`，`max` and `step` 它指定数量字段的步骤和边界。
- `pipSize` - 大小为1个点（例如，欧元兑美元0.0001）
- `pipValue` - 帐户货币为1点的价值\（例如，美元兑换EURUSD的1美元）
- `minTick` - 最低价格变动\（例如，EURUSD\0.00001）。它用于价格领域。
- `description` - 要在对话框中显示的说明
- `type` - 仪器类型，只有“forex”事项 - 它可以在订单对话框中进行负点检查

## accountInfo() : Deferred (or Promise)

内部订单对话框调用此方法获取帐户信息。

现在应该只返回一个字段：1。`currencysign`：字符串 - 这是一个帐户币值的符号

## subscribePL(positionId)

如果 `supportPLUpdate` 配置标志为 `true`，则应该执行方法。由于该方法被经纪人调用，通过 `\[ [plUpdate \ | Trading-Host \ #plupdatepositionid-pl ]]`方法提供损益。

## unsubscribePL(positionId)

如果 `supportPLUpdate` 配置标志为 `true`，则应该执行方法。由于这种方法被经纪人调用，来停止提供损益。

## subscribeEquity()

如果您使用标准订单对话框并支持止损，则应执行方法。这种方法被经纪人调用，  
[[equityUpdate|Trading-Host#equityupdateequity]]

## unsubscribeEquity()


如果您使用标准订单对话框并支持止损，则应执行方法。由于这种方法被经纪人导游，停止提供股权更新。

这就是它！

## See Also

- [[How to connect|Widget-Constructor#chart-trading\_controller]] your trading controller to the chart
- [[Trading Host]]

# 经纪商API

 此页面上的所有内容仅与交易终端相关。

经纪商API是一种能让你在线交易的接口。它的主要目的是将我们的图表与您的交易逻辑连接起来。就 JS 而言，它是一个预期暴露特定接口的 对象 。下面是一个API的\*方法，终端将有这样一个列表。

## 必要的方法

### constructor(host)

经纪商API的构造函数通常需要交易主机。

### positions : Promise

交易终端调用这种方法来请求仓位。你应该返回仓位数组。

### orders : Promise

交易终端调用这种方法来请求订单。你应该返回订单数组。

### executions(symbol) : Promise

交易终端调用这种方法来请求执行。你应该返回执行数组。

### trades : Promise

交易终端调用这种方法来请求交易（个别仓位）。你应该返回交易对象数组。

### chartContextMenuActions(e)

图表可以在菜单中拥有一个子菜单 交易 。返回子菜单的项目列表。格式与 buttonDropdownItems 相同。

e 是浏览器传递的上下文对象

### connectionStatus()

通常你不需要返回 1 以外的值，因为当你创建widget时经纪商已经连接。如果要在加载数据时在底部面板中显示一个spinner，则可以使用它。可能的返回值是：

```
ConnectionStatus.Connected = 1
ConnectionStatus.Connecting = 2
ConnectionStatus.Disconnected = 3
ConnectionStatus.Error = 4
```

## isTradable(symbol)

该功能是浮动交易面板所必需的。通过面板进行交易的能力取决于这个函数的结果：`true` 还是 `false`。如果所有商品都可以交易，则不需要实现这个方法。

## accountManagerInfo()

此功能用于返回账户管理器的信息。请参阅[账户管理器](#)了解更多信息。

## showOrderDialog(order)

当用户请求创建或修改订单时，此功能由图表调用。

所以我们给您使用您自己的对话框的能力。

## placeOrder(order, silently)

方法在用户想要下订单时调用。订单预先填写了部分或全部信息。如果 `silently` 是 `true`，则不显示任何对话框。

## modifyOrder(order, silently, focus)

1. `order` 是要修改的订单对象
2. `silently` - 如果是 `true`，则不应显示任何订单对话框
3. `focus` - [OrderTicketFocusControl constants](#)。它可以通过图表初始化。

方法在用户想要修改现有订单时被调用。

## cancelOrder(orderId, silently)

这个方法被调用来取消给定 `id` 的单个订单。如果 `silently` 是 `true`，则不显示任何对话框。

## cancelOrders(symbol, side, orderIds, silently)

1. `symbol` - symbol string
2. `side` : `Side` or `undefined`
3. `ordersIds` - `symbol` 和 `side` 收集的ids。如果 `silently` 是 `true`，则不显示任何对话框。

这个方法被调用来取消 `symbol` 和 `side` 的多个订单。

## editPositionBrackets(positionId, focus)

如果 `supportPositionBrackets` 配置标志打开，将显示一个用于编辑止盈和止损的对话框，此方法将被调用。1. `positionId` 要修改的现有仓位ID 2. `focus` - `Focus constant`。

## closePosition(positionId, silently)

如果 `supportClosePosition` 配置标志打开，通过仓位id平仓时，此方法将被调用。如果 `silently` 是 `true`，则不显示任何对话框。

## reversePosition(positionId, silently)

如果 `supportReversePosition` 配置标志打开，通过仓位id反转时，此方法将被调用。如果 `silently` 是 `true`，则不显示任何对话框。

## editTradeBrackets(tradeId, focus)

如果 `supportTradeBrackets` 配置标志打开，将显示一个用于编辑止盈和止损的对话框。1. `tradeId` 是要修改的交易id 2. `focus` - `Focus constant`。

## closeTrade(tradeId, silently)

如果 `supportCloseTrade` 配置标志打开以通过交易id关闭交易，则调用此方法。如果 `silently` 是 `true`，则不显示任何对话框。

## symbolInfo(symbol) : Deferred (or Promise)

1. `symbol` - symbol string

这个方法由内部的Order Dialog，DOM面板和浮动交易面板调用，以获取商品信息。结果是具有以下数据的对象：

- `qty` - 对象拥有这3个属性：`min`、`max`、`step`，用于指定数量的 `step` 和边界。
- `pipSize` - 点数的大小（例如，EURUSD为0.0001）
- `pipValue` - 账户币种的点数值（例如，对于以美元为账户的EURUSD为1）

- `minTick` - 最低的价格变动（例如，EURUSD为0.00001）。它用于价格字段。
- `description` - 要在对话框中显示的描述
- `type` - 商品类型, 只要 `forex` 比较特殊 - 它允许在订单对话框中检查负的点数。
- `domVolumePrecision` - DOM 卖出/买入量的小数位数（可选，默认为0）

### **accountInfo() : Deferred (or Promise)**

这个方法被内部的Order Dialog调用来获取帐户信息。现在应该只返回一个字段：

1. `currencySign`：字符串 - 这是一个计数货币的标志

由于这种方法被调用，经纪商应该停止提供利润/损失。

### **subscribeEquity()**

如果您使用标准订单对话框并支持止损，则应实现这个方法。由于这种方法被调用，经纪商应该通过[equityUpdate](#)方法提供净资产更新。

### **unsubscribeEquity()**


如果您使用标准订单对话框并支持止损，则应实现这个方法。由于这种方法被调用，经纪商应该停止提供净资产更新。

这就是它！

## 也可以看看

- [交易主机](#)

# 交易主机

 此页面上的所有内容仅适用于[[交易终端]]。

交易主机是[[交易控制器]]与图表交易子系统之间的交互API。其主要目的是用您的交易逻辑与图表之间交换信息。就 JS 而言，它是一个具有一组函数的 `object`。以下是主机的方法列表。

## 命令

### **showOrderDialog(order, handler, focus) : Deferred**

1. `order` 被放置或修改
2. `handler` 是一个处理买/卖/修改的功能。它应该返回Deferred
3. `focus` - [[Focus constant|Trading-Objects-and-Constants#focusoptions]].

显示标准订单对话框以创建或修改订单，并执行处理程序（如果以下达买/卖/修改指令）。

### **showCancelOrderDialog(orderId, handler) : Deferred**

1. `orderId` 待取消订单的编号
2. `handler` 处理取消的功能。它返回 Deferred

显示一个确认对话框，并按下YES/OK，执行处理程序。

### **showCancelMultipleOrdersDialog(symbol, side, qty, handler) : Deferred**

1. `symbol` 取消订单商品
2. `side` - 取消订单方向
3. `qty` - 取消订单数量
4. `handler` 处理取消的功能。它返回 Deferred

显示一个确认对话框，并按下YES/OK，执行处理程序。

### **showClosePositionDialog([[positionId|Trading-Objects-and-Constants#position]], handler) : Deferred**

1. `positionId` 要平仓的头寸id
2. `handler` 处理平仓的功能。它返回 `Deferred`

显示一个确认对话框，并按下YES/OK，执行处理程序。

## **showReversePositionDialog([[position|Trading-Objects-and-Constants#position]], handler) : `Deferred`**

1. `position` 进行反转
2. `handler` 处理头寸反转的功能。它返回 `Deferred`

显示一个确认对话框，并按下YES/OK，执行处理程序。

## **showPositionBracketsDialog([[position|Trading-Objects-and-Constants#position]], [[brackets|Trading-Objects-and-Constants#brackets]], focus, handler) : `Deferred`**

1. `position` 进行修改
2. `brackets` (可选) new [brackets|Trading-Objects-and-Constants#Brackets]
3. `focus` - [[Focus constant|Trading-Objects-and-Constants#focusoptions]].
4. `handler` is a function to process modification of brackets. It should return `Deferred`

显示默认的编辑括号对话框，并在按下修改时执行处理程序。

## **activateBottomWidget : `Deferred`**

打开底部面板和开关选项卡到交易。

## **showTradingProperties()**

显示属性对话框，切换当前标签到交易。

## **showNotification(title, text, type)**

显示通知。类型可以是 `1` - 成功 或 `0` - 错误。

## **triggerShowActiveOrders()**

触发器显示活动订单。

## **numericFormatter(decimalPlaces)**



以指定的小数位数返回一个[[Formatter|Trading-Objects-and-Constants#focustoptions]]。

## defaultFormatter(symbol)

Returns a default [[Formatter|Trading-Objects-and-Constants#focustoptions]] formatter for the specified instrument. This formatter is created based on [[SymbolInfo|Symbology#symbolinfo-structure]]. 返回指定仪器的默认值[[Formatter|Trading-Objects-and-Constants#focustoptions]]以格式化程序。这个格式化器是基于[[SymbolInfo|Symbology#symbolinfo-structure]]创建的。

## factory

`factory` 是一个对象属性。其成员如下所述。

## factory.createDelegate

创建一个[[Delegate]]对象。

## factory.createWatchedValue

创建一个[[WatchedValue]]对象。

## symbolSnapshot(symbol) : Promise

返回商品报价。

# Getters and Setters

## floatingTradingPanelVisibility: [[WatchedValue]]

返回floatingTradingPanel是否可见。

## showPricesWithZeroVolume: [[WatchedValue]]

返回是否折叠0成交量（最小和最大成交量级别）的级别。

## suggestedQty() : Object

返回的对象属性：

1. value - 获取当前值。它返回Promise.

2. `setValue` - 设置新值
3. `changed` : `[[Subscription]]` 它是在交易浮动面板和对话框中同步数量。

## **setButtonDropdownActions(actions)**

底部交易面板有一个带有下拉列表项目的按钮。此方法可用于替换现有项目。

1. `actions` `[[ActionMetaInfo|Trading-Objects-and-Constants#actionmetainfo]]` 的数组, 每个对象都代表一个下拉项。

## **defaultContextMenuActions()**

提供默认的买/卖, 显示属性操作作为默认值由 `chartContextMenuItems` 返回。

## **defaultDropdownMenuActions(options)**

提供操作的默认下拉列表。您可以在 `[setButtonDropdownActions]` 中使用默认操作。您可以使用 `options` 从结果中添加/删除默认操作:

1. `showFloatingToolbar` : `boolean`;
2. `tradingProperties` : `boolean`;
3. `selectAnotherBroker` : `boolean`;
4. `disconnect` : `boolean`;

## 数据更新

需要使用这些方法来通知图表需要更新信息。

## **orderUpdate([[order|Trading-Objects-and-Constants#order]])**

在添加或更改订单时调用此方法。

## **orderPartialUpdate([[order|Trading-Objects-and-Constants#order]])**

当订单未更改时调用此方法, 但您添加到在账户管理器中显示的订单对象的字段已更改。仅当您要在 `[[Account Manager]]` 中显示自定义字段时才应使用它。

## **positionUpdate ([[position|Trading-Objects-and-Constants#position]])**

在添加或更改头寸时调用此方法。

## **positionPartialUpdate ([[position|Trading-Objects-and-Constants#position]])**

当头寸未更改时调用此方法，但您添加到在账户管理器中显示的头寸对象的字段已更改。仅当您要 [[Account Manager]] 中显示自定义字段时才应使用它。

## **executionUpdate ([[execution|Trading-Objects-and-Constants#execution]])**

添加执行时调用此方法。

## **fullUpdate()**

当数据无效时调用此方法。例如，用户帐户已被更改。


## **plUpdate(positionId, pl)**

当代理商接收到PL更新时调用此方法。当 `configFlags` 中设置 `supportPLUpdate` 标志时，应使用此方法。

## **equityUpdate(equity)**

当经纪商接收到资产净值时调用此方法。标准订单对话框需要此方法。

## 账户管理器

 此页面上的所有内容仅适用于[[交易终端]]。

帐户管理器是一个显示交易信息的交互式表格。通常它有3页：订单/头寸和帐户信息。

要创建帐户管理器，您需要描述每个页面的列并提供数据。

备注 1. `[[supportCustomBottomWidget|Trading-Controller#configFlags]]` 标志应被禁用，以显示客户经理。

备注 2. `[[Trading Controller]]` 应实现 `[[accountManagerInfo|Trading-Controller#accountmanagerinfo]]`

## 帐户管理器Mete信息

此信息将返回 `[[accountManagerInfo|Trading-Controller#accountManagerInfo]]`.

### 帐户管理器头信息

帐户管理器头信息由经纪商的标题和帐户名或帐户列表组成。

**accountTitle: String**

**accountsList: AccountInfo** 数组

**account: `[[WatchedValue]]` of AccountInfo**

`AccountInfo` 是一个只有 `name` 为必须键和对应值的对象。

### 订单页

**orderColumns: array of `[[Column|Account-Manager#columnndescription]]`**

要在订单页面中显示的列的说明。您可以显示`[[order|Trading-Objects-and-Constants#order]]`的任何字段，也可以将自己的字段添加到订单对象中并显示它们。

**possibleOrderStatuses: array of `[[OrderStatus|Trading-Objects-and-Constants#orderstatus]]`**

在订单过滤器中使用的可选状态列表。如果未设置，则使用默认列表。

## hasHistory

如果是 `true`，将显示历史页面。历史上的所有订单将显示在历史记录中。

## 头寸页

**positionColumns:** array of `[[Column|Account-Manager#columnndescription]]`

您可以显示`[[position|Trading-Objects-and-Constants#position]]`的任何字段，或者将您自己的字段添加到位置对象并显示它们。

## 附加页面（例如帐户摘要）

**pages:** array of `[[Page|Account-Manager#page]]`

使用 `pages` 您可以向账户管理器添加新的tab页。每个选项卡都是一组列表。

## Page

`Page` 是额外的账户管理器tab页说明。它是一个包含以下字段的对象：

1. `id` : String

页面的唯一标识

1. `title` : String

页面标题。显示在选项卡上。

1. `tables` : Array of `[[Table|Account-Manager#table]]`.

可以在此选项卡中显示一个或多个表。

## Table

您可以向`[[Page|Account-Manager#page]]`添加一个或多个表。帐户摘要表`metainfo`是一个包含以下字段的对象：

1. `id` : String

唯一标识

1. `title` : String

表的可选标题。

1. `columns` : array of `[[Column|Account-Manager#columnndescription]]`

## 2. `getData : Promise`

此方法用于请求表数据。它返回`promise`（或`Deferred`）并接收它返回的数据数组。每一行都是一个对象。此对象的键是具有相应值的列名称。有一个预定义的字段 `isTotalRow` 可以用来标记一个表的底部的一行。

## 1. `changeDelegate : [[Delegate]]`

用于观察数据更改并更新表。通过 `fire` 方法将新的账户管理器数据传递给`delegate`。

注意：如果表中有多行，并且想使用 `changeDelegate` 更新一行，请确保每行中都有 `id` 字段来标识它。如果表中只有一行，则不是必须的。

# Formatters

**customFormatters:** 一组列格式的描述

可选数组定义自定义格式化。每个描述都是一个包含以下字段的对象：

`name`：唯一标识 `format(options)`：用于格式化单元格值的方法。 `options` 是一个具有以下键的对象：

1. `value` - 要格式化的值
2. `priceFormatter` - 价格标准格式。您可以使用 `format(price)` 方法来设置价格的值。
3. `prevValue` - 可选字段。它是一个以前的值，所以你可以相应地进行比较和格式化。如果当前列具有 `highlightDiff: true` `key`.
4. `row` - 具有当前行中所有键/值对的对象

## 列描述

帐户管理器描述中最有价值的部分是其列的描述。

### label

列标题。它将显示在表的标题行中。

### className

可选的 `className` 被添加到每个值单元格的`html`标签。您可以使用它来自定义表的样式。

以下是预定义类的列表：

class名	描述
tv-data-table__cell--symbol-cell	商品字段的特殊格式化器
tv-data-table__cell--right-align	它将单元格值右对齐
tv-data-table__cell--buttons-cell	单元格按钮

**formatter**

用于格式化数据的格式化器。如果没有设置 `formatter`，则按照原样显示该值。格式化器可以是默认的或者是定制的

以下是默认格式化程序列表：

名称	描述
symbol	它用于商品字段。它显示 <code>brokersymbol</code> ，但是当您单击符号时，将 <code>symbol</code> 字段设置为图表。 <code>property</code> 键被忽略。
side	它用于显示方向：卖或买。
type	用于显示类型：限价/停损/限价停损/市价。
formatPrice	格式化价格
formatPriceForexSup	与 <code>formatPrice</code> 一样，但它使得价格的最后一个字符被上标。只有当仪器的类型为 <code>forex</code> 时，它才起作用。
status	格式化 <code>status</code>
date	显示日期或时间
localDate	显示的日期或时间在本地时区。
fixed	显示一个小数点后2位的数字。
pips	显示一个小数点后1位的数字。
profit	显示利润。它还添加了 <code>+</code> ，分隔成千位，并设置红色或绿色的单元文本颜色。

**property**

`property` 是用于获取显示数据对象的关键字。

**sortProp**

可选的 `sortProp` 是用于数据排序的数据对象的键。

**modificationProperty**

可选的 `modifyProperty` 是数据对象的一个关键字，它被用于修改。

## notSortable

可选的 `notSortable` 可以设置为防止列的排序。

## help

`help` 是列的提示字符串。

## highlightDiff

`highlightDiff` 可以使用 `formatPrice` 和 `formatPriceForexSup` 格式化器来设置字段的更改。

## fixedWidth

如果为 `true`，则当数字减少时，列宽不会减小。

## 上下文菜单

## contextMenuActions(e, activePageItems)


`e`：浏览器传递的上下文对象

`activePageItems`：当前页面的 `ActionMetaInfo` 项目数组

可选方法以创建一个自定义上下文菜单。它返回用 `ActionMetaInfo` 数组解析的 `Promise`。



# 交易对象和常量

 此页面上的所有内容仅适用于[[交易终端]]。

注意：如果您使用TypeScript，您可以从 `broker-api.d.ts` 文件中导入本文的常量/接口/类型。

## 经纪商配置

### configFlags: object

这是一个应该在交易终端的构造函数中传递给`brokerConfig`的对象。每个字段应该有一个布尔值（`true` / `false`）：

- `supportReversePosition`

经纪商支持反转头寸。如果经纪商不支持，图表将有反转按钮，但是它会发出反转订单。

- `supportClosePosition`

经纪商支持平仓。如果经纪商不支持，图表将有平仓按钮，但它将发出平仓订单。

- `supportReducePosition`

经纪商支持在没有订单的情况下更改头寸。

- `supportPLUpdate`

经纪商支持持仓损益(PL)。如果经纪商本身计算利润/损失，则应在PL更改后立即调用`PLUpdate`。否则，图表将计算PL作为当前交易与仓位平均价格之差。

- `supportOrderBrackets`

经纪商支持订单的包围单（止盈和止损）。如果此标志为 `true`，则图表将在图表和账户管理器中的订单编号和修改按钮中显示附加字段。

- `supportPositionBrackets`

经纪商支持仓位的包围单（止盈和止损）。如果此标志为 `true`，则图表将显示位置的编辑按钮，并将 `编辑头寸...` 添加到仓位的上下文菜单中。

- `supportTradeBrackets`

经纪商支持单一交易的包围单（止盈和止损订单）。如果此标志为 `true`，则图表将显示用于交易（单个头寸）的编辑按钮，并将 `编辑头寸...` 添加到交易的上下文菜单中。

- **supportTrades**

经纪商支持单个头寸（交易）。如果设置为 `true`，帐户管理器中将有两个选项卡：单个头寸和净头寸。

- **requiresFIFOCloseTrades**

交易账户需要以先进先出顺序结算交易。

- **supportCloseTrade**

单个头寸（交易）可以关闭。

- **supportMultiposition**

支持多头寸防止创建反转头寸的默认实现。

- **showQuantityInsteadOfAmount**

此标志可用于在订单对话框中将"Amount"更改为"Quantity"

- **supportLevel2Data**

Level2数据用于DOM小部件。应该执行 `subscribeDepth` 和 `unsubscribeDepth`。

- **supportStopLimitOrders**

此标志将止损限价订单类型添加到订单对话框。

- **supportMarketBrackets** 使用这个标志你可以禁止市价单的包围单。默认情况下启用。

- **supportModifyDuration** 使用这个标志你可以修改现有订单的持续时间。默认情况下它被禁用。

## **durations: array of objects**

订单到期选项列表。这是可选的。如果您不希望持续时间显示在订单故障单中，请不要设置它。对象有两个键: `{ name, value }`。

例子:

```
durations: [{ name: 'DAY', value: 'DAY' }, { name: 'GTC', value: 'GTC' }]
```

## **customNotificationFields: array of strings**

可选字段。如果您在显示通知时考虑到订单或仓位中的自定义字段，则可以使用它。

例如，如果在订单中有字段 `additionalType`，并且希望图表在更改时显示通知，则应该设置：

```
customNotificationFields: ['additionalType']
```

## Order

描述一个订单。

- `id` : String
- `symbol` : String
- `brokerSymbol` : String. 如果经纪商商品代码与TV商品代码相同，则可以为空。
- `type` : [OrderType](#)
- `side` : [Side](#)
- `qty` : Double
- `status` : [OrderStatusSide](#)
- `limitPrice` : double
- `stopPrice` : double
- `avg_price` : double
- `filledQty` : double
- `parentId` : String. 如果订单是一组 `parentId` 应该包含基本订单/仓位ID。
- `parentType` : [ParentType](#)

## Position

描述一个头寸。

- `id` : String. 通常id应等于 `brokerSymbol`
- `symbol` : String
- `brokerSymbol` : String. 如果经纪商商品代码与TV商品代码相同，则可以为空。
- `qty` : Double positive
- `side` : [\[\[Side|Trading-Objects-and-Constants#side\]\]](#)
- `avg_price` : Double

## Execution

Describes a single execution.

- symbol : String
- brokerSymbol : String. 如果经纪商商品代码与TV商品代码相同，则可以为空。
- price : double
- time: time\_t
- side : [[Side|Trading-Objects-and-Constants#side]]
- qty : double

## ActionMetaInfo

描述将其放入下拉菜单或上下文菜单中的单个操作。它是一个结构。

- text : String
- checkable : Boolean. 如果需要复选框，将其设置为true。
- checked : Boolean
- 复选框的值。
- enabled: Boolean
- action: function. 当用户单击该项目时执行操作。它有一个参数 - 复选框的值（如果存在）。

## OrderType

用于描述订单状态的字符串常量。

- market
- limit
- stop
- stoplimit

## Side

用于描述订单/交易执行的字符串常量。

- buy
- Sell

## ParentType

用于描述包围单所有者的字符串常量。

- ORDER\_PARENT
- POSITION\_PARENT

# OrderStatus

用于描述订单状态的字符串常量。

状态	描述
pending	订单还未在经纪商一方创建
inactive	bracket order is created but waiting for a base order to be filled
working	订单被创建还未执行成功
rejected	订单因某些原因被拒绝
filled	订单已成交
canceled	订单被取消

# DOMEObject

描述单个DOME响应的对象。

- snapshot : Boolean 正值意味着以前的数据应该被清理
- asks : 根据价格升序排列的DOM价格水平数组
- bids : 根据价格升序排列的DOM等级数组

# DOMELevel

单个DOME价格水平对象。

- price : double
- volume : double

# FocusOptions

打开标准订单对话框或头寸对话框时设置焦点的字符串常量。

- STOP\_PRICE\_FIELD focus stop price for StopLimit orders
- TAKE\_PROFIT\_FIELD focus take profit control

- STOP\_LOSS\_FIELD focus stop loss control

## Brackets

**stopLoss : double**

**takeProfit : double**

## Formatter

具有 `format` 方法的对象可用于将数字格式化为字符串。

## 储存和载入图表

图表库支持保存/加载图表和研究模板（研究模板在 `unstable` 中提供）在2级抽象上：

1. 低级别：保存/加载功能通过 `widget` 的 `save()` / `load()` 方法和 `createStudyTemplate()` / `applyStudyTemplate()` 方法呈现。使用它们的人应该自己处理物理存储。因此，您可以将这些JSON保存到您的位置，例如，您可以将它们嵌入到已保存的页面或用户的工作区域等等。
2. 高级别：图表库可以从您指向的存储中保存/加载图表和研究模板。我们使用Python和PostgreSQL创建了一个小型存储示例，并将其放在我们的[GitHub](#)上。您可以获取它并运行在您自己的服务器上，以便您可以控制所有用户的保存数据。

## 使用高级别保存/加载

下面是想拥有自己图表存储功能所需的步骤：

1. 克隆我们的资源到您的主机上
2. 运行数据服务或使用我们的演示服务。对于不熟悉Django的人来说，这是一个简短的待办事项列表。
  - i. Install Python 3.x and Pip.
  - ii. Install PostgreSQL or some other Django-friendly database engine.
  - iii. 转到图表存储文件夹并运行 `pip install -r requirements.txt`
  - iv. 转到 `charting_library_charts` 文件夹，并在 `settings.py` 中设置数据库连接（参见 `DATABASES` @ line #12）。请记住在PostgreSQL中创建适当的数据库。
  - v. 运行 `python manage.py migrate`。这将创建没有任何数据的数据库 `schema`。
  - vi. 运行 `python manage.py runserver` 运行您的数据库的TEST实例。不要在生产环境中使用上面的命令。使用一些其他的東西（例如Gunicorn）
3. 设置您的图表库页面: 设置 `charts_storage_url = url-of-your-charts-storage`，并在 `widget` 的 `ctor` 中设置 `client_id` 和 `user_id`（见下文）。
4. 请享用！

备注：手动填充/编辑数据库并不是理想做饭。请避免这个，因为你可能会伤害Django。

## 开发自己的后端

图表库将HTTP/HTTPS命令发送到 `charts_storage_url / charts_storage_api_version / charts?client = client_id&user = user_id` 。

`charts_storage_url` , `charts_storage_api_version` , `client_id` 和 `user_id` 是 [widget构造函数](#) 的参数。 您应该执行4个请求的处理：保存图表/加载图表/删除图表/列出图表。

### 列出图表

GET REQUEST: `charts_storage_url/charts_storage_api_version/charts?client=client_id&user=user_id`

响应：JSON对象

1. `status`: "ok" or "error"
2. `data`: 对象数组
  - i. `"timestamp"`: 保存图表时的UNIX时间（例如，1449084321）
  - ii. `"symbol"`: 图表的商品（例如，“AA”）
  - iii. `"resolution"`: 分辨率（例如，“D”）
  - iv. `"id"`: 图表的唯一整数标识符（例如，9163）
  - v. `"name"`: 图表名称（例如，“测试”）

### 存储图表

POST REQUEST: `charts_storage_url/charts_storage_api_version/charts?client=client_id&user=user_id`

1. `"name"`: 图表名称
2. `"content"`: 图表内容
3. `"symbol"`: 图表商品(例如, "AA")
4. `"resolution"`: 图表分辨率 (例如, "D")

响应：JSON对象

1. `"status"`: "ok" or "error"
2. `"id"`: 图表的唯一整数标识符（例如，9163）

### 存储为图表

POST REQUEST: `charts_storage_url/charts_storage_api_version/charts?client=client_id&user=user_id&chart=chart_id`

1. `"name"`: 图表名称
2. `"content"`: 图表内容
3. `"symbol"`: 图表商品(例如, "AA")



4. "resolution": 图表分辨率 (例如, "D")

响应：JSON对象

1. "status": "ok" or "error"

## 加载图表

GET REQUEST: charts\_storage\_url/charts\_storage\_api\_version/charts?  
client=client\_id&user=user\_id&chart=chart\_id

响应：JSON对象

1. status: "ok" or "error"
2. data: 对象数组
  - i. "timestamp": 保存图表时的UNIX时间 (例如, 1449084321)
  - ii. "symbol": 图表的商品 (例如, "AA")
  - iii. "resolution": 分辨率 (例如, "D")
  - iv. "id": 图表的唯一整数标识符 (例如, 9163)
  - v. "name": 图表名称 (例如, "测试")

## 删除图表

DELETE REQUEST: charts\_storage\_url/charts\_storage\_api\_version/charts?  
client=client\_id&user=user\_id&chart=chart\_id

响应：JSON对象

1. "status": "ok" or "error"

## 使用演示图和研究模板存储

我们正在运行演示图存储服务,让您尽可能快地保存/加载新的库的构建。此存储网址为<http://saveload.tradingview.com>。这只是一个演示,所以它是按原样提供的。我们不保证其稳定性。此外,我们一次又一次地从这个存储中删除所有的数据。

## 管理保存的图表访问

您应该关心用户将能够查看和加载哪些图表。基本上,用户可以看到/加载与用户具有相同的 `client_id` 和 `user_id` 的图表。`client_id` 是用户组的标识符。当您使用相同的图表存储时,您的用户群体(即,当您有少量站点)时,可以覆盖这种情况。所以常见的做法是设

置 `client_id = your-site's-URL` 。然而，这取决于你。

`user_id` 在您的 `client_id` 组的上下文中将是用户的id。您可以单独设置每个用户（使每个用户拥有自己的私人图表存储），也可以将所有用户或任何用户组设置为相同，以创建一种公共存储。以下是几个例子：

client_id	user_id	作用
您的网站网址或其他任何内容	唯一用户 ID	每个用户都有他的私人图表存储其他用户看不到。
您的网站网址或其他任何内容	所有用户的相同值	每个用户都可以看到并加载每个保存的图表。
您的网站网址或其他任何信息	注册用户的唯一用户 ID 和所有匿名用户的常量	每个注册用户都有他的私有图表存储其他用户看不到。所有匿名用户都有一个共享存储。

# 创建自定义研究

## 如何显示您的数据作为一个指标

如果您想要在图表上显示一些数据，例如指标，则此处为食用说明。

请遵循以下几个步骤：

1. 为您的数据创建一个新的**ticker**，并设置您的服务器返回此**ticker**有效的**SymbolInfo**。
2. 设置服务器以返回此**ticker**的有效历史数据。
3. 使用以下指标模板并填写所有占位符(**placeholder**)的值：名称，说明和代码。如果需要，还可以修改绘图的默认样式。

```
{
  // 将<study name>替换为您的研究名称
  // 它将由图表库内部使用
  name: "<study name>",
  metaInfo: {
    "_metaInfoVersion": 40,
    "id": "<study name>@tv-basicstudies-1",
    "scriptIdPart": "",
    "name": "<study name>",
    // 此说明将显示在指标窗口中
    // 当调用createStudy方法时，它也被用作"name"参数
    "description": "<study description>",
    // 该描述将显示在图表上
    "shortDescription": "<short study description>",

    "is_hidden_study": true,
    "is_price_study": true,
    "isCustomIndicator": true,

    "plots": [{"id": "plot_0", "type": "line"}],
    "defaults": {
      "styles": {
        "plot_0": {
          "linestyle": 0,
          "visible": true,

          // 绘图线宽度
          "linewidth": 2,

          // 绘制类型：
          // 1 - 直方图
          // 2 - 线形图
          // 3 - 十字指针
          // 4 - 山形图

```

```

        //      5 - 柱状图
        //      6 - 圆圈图
        //      7 - 中断线
        //      8 - 中断区块
        "plottype": 2,

        // 显示价格线?
        "trackPrice": false,

        // 绘制透明度，百分比。
        "transparency": 40,

        // 以#RRGGBB格式绘制颜色
        "color": "#0000FF"
    }
},

// 研究输出值的精度
// (小数点后的位数)。
"precision": 2,

"inputs": {}
},
"styles": {
    "plot_0": {
        // 输出的名字将在样式窗口显示
        "title": "-- output name --",
        "histogramBase": 0,
    }
},
"inputs": [],
},

constructor: function() {
    this.init = function(context, inputCallback) {
        this._context = context;
        this._input = inputCallback;

        // 定义要绘制的商品。
        // 商品应该是一个字符串。
        // 您可以使用PineJS.Std.ticker(this._context)获取所选商品的代码。
        // 例,
        //     var symbol = "AAPL";
        //     var symbol = "#EQUITY";
        //     var symbol = PineJS.Std.ticker(this._context) + "#TEST";
        var symbol = "<TICKER>";
        this._context.new_sym(symbol, PineJS.Std.period(this._context), PineJS.Std.
period(this._context));
    };

    this.main = function(context, inputCallback) {
        this._context = context;
        this._input = inputCallback;

        this._context.select_sym(1);
    };
}

```

```
// 您可以在PineJS.Std对象中使用以下内置函数：
//      open, high, low, close
//      hl2, hlc3, ohlc4
var v = PineJS.Std.close(this._context);
return [v];
}
}
}
```

1. 将指标保存到具有以下结构的自定义指标文件中：

```
__customIndicators = [
    *** 您的指标对象，由模板创建 ***
];
```

请注意，该指标文件是一个JavaScript源文件，它定义了一个指标对象数组。因此，您可以在其中放置多个指标，或者将它们与我们为您编译的指标组合起来。

1. 使用 `indicators_file_name` Widget构造函数的选项来从指标文件加载自定义指标。
2. 图表准备好后，更新您的Widget初始化代码以创建此指标。

例子

假设您希望在图表上显示用户的权益曲线。你必须做以下事情：

- 为新的代码创建一个名称。假设它为 `#EQUITY` 代码。您可以使用您想像到的任何名字。
- 更改服务器的代码以将此代码作为有效商品。为此返回最小的有效SymbolInfo。
- 使服务器返回有效的历史记录。即，服务器可以询问您的数据库的股权历史记录，并返回此数据，就像返回普通商品的历史记录一样（例如 `AAPL`）。
- 采用上述指标模板，创建指标文件（或向现有指标文件添加新指标）。例如：

```
__customIndicators = [
{
    name: "Equity",
    metaInfo: {
        "_metaInfoVersion": 40,
        "id": "Equity@tv-basicstudies-1",
        "scriptIdPart": "",
        "name": "Equity",
        "description": "Equity",
        "shortDescription": "Equity",

        "is_hidden_study": true,
        "is_price_study": true,
        "isCustomIndicator": true,

        "plots": [{"id": "plot_0", "type": "line"}],
        "defaults": {
```

```

        "styles": {
            "plot_0": {
                "linestyle": 0,
                "visible": true,

                // 使线条变细
                "linewidth": 1,

                // 绘制类型为线性图
                "plottype": 2,

                // 显示价格线
                "trackPrice": true,

                "transparency": 40,

                // 为图线设置深红色。
                "color": "#880000"
            }
        },

        // 精度是一位数，如777.7
        "precision": 1,

        "inputs": {}
    },
    "styles": {
        "plot_0": {
            // 输出名字在样式窗口显示
            "title": "Equity value",
            "histogramBase": 0,
        }
    },
    "inputs": [],
},

constructor: function() {
    this.init = function(context, inputCallback) {
        this._context = context;
        this._input = inputCallback;

        var symbol = "#EQUITY";
        this._context.new_sym(symbol, PineJS.Std.period(this._context), PineJS.Std.
period(this._context));
    };

    this.main = function(context, inputCallback) {
        this._context = context;
        this._input = inputCallback;

        this._context.select_sym(1);

        var v = PineJS.Std.close(this._context);
        return [v];
    }
}

```

```
}  
];
```

- 使用 `indicators_file_name` 选项将指标插入图表。
- 更改 `Widget` 的初始化代码。添加如下内容：

```
widget = new TradingView.Widget(/* ... */);  
  
widget.onChartReady(function() {  
    widget.chart().createStudy('Equity', false, true);  
});
```

# 最佳做法

---

## 创造最好的用户体验

我们喜欢我们的图表。我们希望他们是最好的：在整个HTML5世界中最美丽，最灵敏和最强大的图表。我们正在努力实现这些目标。

我们了解我们的图表，以及如何为他们创造最好的用户体验，我们很乐意与您分享我们的知识。本文档介绍了将图表库集成到您的网站/应用程序中的几种最佳做法。主要的一点是经常考虑您的用户和他们的经验。

### 1. 了解图表库是什么，不是什么

图表库是一个能够显示价格的图表组件，图表和技术分析工具。图标库是一个奇幻的图表，只限于此。如果您想要一些额外的功能（如聊天，特殊商品列表，最热门的交易，广告等），最好的方法是在图表之外实现它们。但是，如果要将外部功能与库连接，您可以使用库的API来连接它们。

### 2. 返回与库请求完全一样多的K线

库会询问您的后端数据，并提供每个请求所需的数据范围。尊重这些界限，并尽可能地填充此范围的数据。不要返回更多的K线。不要返回范围外的K线。如果要扩展库请求的默认数据范围，请使用我们的JS API（请参阅`calculateHistoryDepth`）。

### 3. 返回与库请求完全一样多的标记

与上述K线相同。只发送符合要求范围的标记。

### 4. 不要覆盖`calculateHistoryDepth()`以获取超过2个屏幕的数据

图表库避免加载用户没有要求的内容。在图表中加载更多的K线，意味着需要更多的CPU和内存。这意味着的响应效率会降低。

### 5. 不要让你的图表看起来像一团糟

用户喜欢美丽的图表。像我们一样 请记住，在定制尺寸或风格时，请保持您的图表看起来不错。避免嵌入与整个图表风格不同的自定义控件。



## 6. 避免制作非常小的图表

图书馆支持的最小尺寸是600x600像素。避免使图表更小，因为它看起来像一团糟。使用 `mobile` 预设，或者如果您需要比上述更小的图表，可以隐藏一些工具栏。

## 7. 使用适当的语言

图表库已翻译成数十种语言。使用符合用户需求的语言。

## 8. 如果您遇到问题

我们总是渴望帮助你。但是，不幸的是，我们真的很忙，所以我们没有太多时间。请帮助我们有效地度过时间，并始终将您的图书馆的版本更新为最新的 `unstable` 版本，以检查问题是否仍然发生。如果有，请与我们联系。

另外，检查您传递给图表库的数据，并确保它符合我们的要求，如文档中所述。要特别注意 `SymbolInfo` 的内容，因为它是最常见的发生错误的地方（根据我们的统计）。

您可以看我们的输出 [demo data service](#) 并将其与您的对比，以确保您的后端行为是正确的。

在开发过程中始终在 `Widget` 构造函数选项中使用 `debug: true`，并在生产环境中将其删除，以使代码更快地工作。

## 9. 阅读文档

我们花了很多时间为您创建这些文档，使您的生活更轻松。请试一试。

## 10. 为您的解决方案选择适当的数据传输

注意JS API和UDF之间的差异，并选择最符合您需求的API。如果您需要真正快速的数据更新或数据流传输，请勿使用UDF。如果您的后端有十几个符号，请勿使用UDF进行数据分组（请参阅 `supports_group_request`）。

## 11. 不要尝试嗅探我们的代码并使用未记录的功能

我们的文档中没有提到的所有功能都是变化的主题，没有任何警告和向后兼容性。您签署的法律协议也严格禁止修改源代码。

## 12. 不要在您的生产网站上使用我们的演示数据源

这个数据源只是一个演示，不适合实际使用。它可能不稳定，不能承受显著的负载。

### 13. 使用**API**进行自定义。避免编辑**CSS**。

我们不保证**CSS**选择器的向后兼容性。

### 14. 发送到客户端时，将服务器设置为**gzip**文件

这是静态**HTML**内容的常见最佳做法。加载图标库的**HTML**文件会减少用户的等待时间。

## 经常被问到的问题

---

### 数据问题

- ▶ 1. 如何连接我的数据？如何添加新的商品代码？
- ▶ 2. 是否有JS API实现的例子？
- ▶ 3. 是否有WebSocket数据传输的例子？
- ▶ 4. 是否有ASP.NET，Python，PHP等后端数据源的例子。？
- ▶ 5. 如何显示存储在TXT/CSV/Excel文件中的数据？
- ▶ 6. 为什么我的数据没有显示/显示不正确/从服务器提取错误？
- ▶ 7. 图表库不断要求数据。如何判断数据是否完成？
- ▶ 8. 如何在图表上更改小数位数？
- ▶ 9. 如果每个时间戳都有一个单一的价格怎么办？？

### GUI问题

- ▶ 1. 如何订阅图表事件？？
- ▶ 2. 如何将默认K线风格从蜡烛更改为另一种？
- ▶ 3. 如何更改图表上的分辨率列表（时间间隔），使其变灰？
- ▶ 4. 如何隐藏GUI元素(商品、间隔、按钮等)？

### 其他问题

- ▶ 1. [\[\[Widget|http://tradingview.com/widget/\]\]](http://tradingview.com/widget/), [\[\[Charting Library|https://www.tradingview.com/HTML5-stock-forex-bitcoin-charting-library/\]\]](https://www.tradingview.com/HTML5-stock-forex-bitcoin-charting-library/) 和 [\[\[Trading Terminal|https://www.tradingview.com/trading-terminal/\]\]](https://www.tradingview.com/trading-terminal/) 有什么区别？
- ▶ 2. 如何添加自定义指标？

## 版本变更点

我们尽最大努力使每个新版本与前一版本完全兼容，但有时需要大的更改需要在升级到新版本时对代码进行微小的更改。

注意：您可以通过在浏览器控制台中执行 `TradingView.version()` 来检查图表库版本。

以下是变更列表：

**Version 1.11** 交易终端中仍然支持以下内容，但在将来的版本中将不再使用以下内容：

- `supportDOME` 重命名为 `supportDOM`
- 更改了签名 `showClosePositionDialog`
- `showEditBracketsDialog` 重命名为 `showPositionBracketsDialog`，更改了签名

### Version 1.10

- Default behavior of Volume indicator is changed.

Previous behavior: Volume indicator is added/removed when an instrument or a resolution is switched depending on volume support by the instrument. You can get back to this behavior by disabling `create_volume_indicator_by_default_once` featureset.

New behavior: Volume indicator is added on first load of an empty chart if it is supported by an active instrument.

### Version 1.9

- We don't compile Pine scripts anymore. You can still use scripts compiled before.

### Version 1.8 of Trading Terminal

- Chart has no option to show only actual orders anymore. Appropriate methods have been removed.
- `showOrderDialog` receives an object instead of arguments list
- `showSampleOrderDialog` removed, use `[[showOrderDialog|Trading-Host#showorderdialogorder-handler]]` instead
- `showOrderDialog` removed from `[[Trading Controller|Trading-Controller]]`, use `placeOrder` and `modifyOrder` receive `silently` argument instead
- `reversePosition`, `closePosition`, `cancelOrder` have an additional argument `silently`. They should should appropriate dialogs by themselves from now.

### Version 1.7

- Since this version it is not enough to call `setSymbol` with the same symbol. You should call `onResetCacheNeededCallback` from `subscribeBars` first. Then you can use `setSymbol` or new `resetData` method of the chart.
- JSAPI protocol version 1 is not supported any more. `nextTime` and `noData` must be provided.

### Version 1.5

- Added `source` argument to MACD. You should change MACD creation code to pass `source` also. `chartWidget.chart().createStudy('MACD', false, false, [12, 26, "close", 9])`

### Version 1.4

- Override `transparency` is not supported anymore. We added transparency support to every color property. Use `rgba` form to define a color with transparency. Example:  
`"symbolWatermarkProperties.color" : "rgba(60, 70, 80, 0.05)"`

### Version 1.3

- Override `paneProperties.gridProperties.*` is not supported anymore. Please use `paneProperties.vertGridProperties.*` and `paneProperties.horzGridProperties.*`
- Override `mainSeriesProperties.candleStyle.wickColor` is not supported anymore. Use `mainSeriesProperties.candleStyle.wickUpColor` and `mainSeriesProperties.candleStyle.wickDownColor`

## 分辨率

分辨率或时间间隔是K线的时间段。图表库支持日内分辨率(seconds, minutes, hours) and DWM 分辨率 (daily, weekly, monthly)。图表库API有很多方法用以接收和返回分辨率。

### 日内

#### 秒

格式: `xS`, 条件 `x` 为数字类型的秒数。例如: 1S - 1秒, 2S - 2秒, 100S - 100秒。

#### 分钟

格式: `x`, 条件 `x` 为数字类型的分钟数。例如: 1 - 1分钟, 2 - 2分钟, 100 - 100分钟。

#### 小时

重要: 用户界面允许用户输入几个小时, 格式为 `xh` 或 `xH`, 它永远不会传递给API。小时必须使用图表库API中的分钟数来设置。

例如: 60 - 1小时, 120 - 2小时, 240 - 4小时。

## DWM

### 天 (Days)

格式: `xD`, 条件 `x` 为数字类型的天数。例如: 1D - 1天, 2D - 2天, 100D - 100天。

### 周 (Weeks)

格式: `xW`, 条件 `x` 为数字类型的周数。例如: 1W - 1周, 2W - 2周, 100W - 100周。

### 月 (Months)

格式: `xM`, 条件 `x` 为数字类型的月数。例如: 1M - 1个月, 2M - 2个月, 100M - 100个月。

### 年

年是使用月数设置的。例如: 12M - 1年, 24M - 2年, 48 - 4年。

## 也可以看看

[如何设置图表上可用分辨率的列表](#)

[如何设置产品支持的分辨率列表](#)

[在图表上设置初始分辨率](#)

[获取当前图表分辨率](#)

[更改图表的分辨率](#)

# 时间范围

可以看到图表底部的工具栏。左侧的每个按钮都会切换图表的时间范围。切换时间范围意味着：

- 1. 切换图表分辨率
- 2. 强制图表K线水平缩放以使整个请求范围可见

即，单击 1Y 将使图表将分辨率切换到 1D，并按比例缩放以显示1年前的所有K线。每个时间范围都有自己的图表分辨率。这里是默认时间范围的列表：

Time Frame	Chart Resolution
5Y	W
1Y	D
6M	120
3M	60
1M	30
5D	5
1D	1

可以使用相应的widget自定义默认时间范围列表。

备注：要求分辨率不适用于当前图表商品的时间范围将被隐藏。



# 本地化

图表库支持本地化，并已被翻译成多种语言。您所要做的仅仅是在创建图表时指定 `locale` 参数。

我们的翻译为众包的，所以每个人都可以参与。访问[我们的WebtranslateIt页面](#)来参与。

支持的语言列表

语言	<code>locale</code> 参数值
中国语（台湾）	zh_TW
中国语	zh
荷兰语（荷兰）	nl_NL
英语	en
法语	fr
德语 (德国)	de_DE
希腊语	el
意大利语	it
日语	ja
韩语	ko
波斯语(伊朗)	fa_IR
葡萄牙语	pt
俄语	ru
西班牙语	es
泰语(泰国)	th_TH
土耳其语	tr
越南语	vi

## 覆盖

本主题包含图表属性说明。这些属性被处理作为可定制的。其他属性自定义则不被支持。

文件的格式:

```
<property_path>: <default Charting Library value>
```

```
// 支持的值: large, medium, small, tiny
volumePaneSize: "large"

// 在文本编辑器中可用的字体（即，在"文本"绘图工具属性对话框中）
editorFontsList: ['Verdana', 'Courier New', 'Times New Roman', 'Arial']

paneProperties.background: "#ffffff"
paneProperties.vertGridProperties.color: "#E6E6E6"
paneProperties.vertGridProperties.style: LINESTYLE_SOLID
paneProperties.horzGridProperties.color: "#E6E6E6"
paneProperties.horzGridProperties.style: LINESTYLE_SOLID
paneProperties.crossHairProperties.color: "#989898"

// 边际（百分比）。用于自动缩放。
paneProperties.topMargin: 5
paneProperties.bottomMargin: 5

// leftAxisProperties & rightAxisProperties
paneProperties.leftAxisProperties.autoScale:true (see #749)
paneProperties.leftAxisProperties.autoScaleDisabled:false (see #749)
paneProperties.leftAxisProperties.percentage:false
paneProperties.leftAxisProperties.percentageDisabled:false
paneProperties.leftAxisProperties.log:false
paneProperties.leftAxisProperties.logDisabled:false
paneProperties.leftAxisProperties.alignLabels:true

paneProperties.legendProperties.showStudyArguments: true
paneProperties.legendProperties.showStudyTitles: true
paneProperties.legendProperties.showStudyValues: true
paneProperties.legendProperties.showSeriesTitle: true
paneProperties.legendProperties.showSeriesOHLC: true

scalesProperties.showLeftScale : false
scalesProperties.showRightScale : true
scalesProperties.backgroundColor : "#ffffff"
scalesProperties.lineColor : "#555"
scalesProperties.textColor : "#555"
scalesProperties.scaleSeriesOnly : false

timeScale.rightOffset: 5

timezone: "Etc/UTC" # 查看支持的时区列表（在Symbology#时区页面）查看可用值
```

```
// 系列风格。请参阅下面的支持的值
//      Bars = 0
//      Candles = 1
//      Line = 2
//      Area = 3
//      Heiken Ashi = 8
//      Hollow Candles = 9
//      Renko = 4
//      Kagi = 5
//      Point&Figure = 6
//      Line Break = 7
mainSeriesProperties.style: 1

mainSeriesProperties.showCountdown: true
mainSeriesProperties.showLastValue:true
mainSeriesProperties.visible:true
mainSeriesProperties.showPriceLine: true
mainSeriesProperties.priceLineWidth: 1
mainSeriesProperties.lockScale: false
mainSeriesProperties.minTick: "default"
mainSeriesProperties.extendedHours: false

mainSeriesProperties.priceAxisProperties.autoScale:true           (see #749)
mainSeriesProperties.priceAxisProperties.autoScaleDisabled:false (see #749)
mainSeriesProperties.priceAxisProperties.percentage:false
mainSeriesProperties.priceAxisProperties.percentageDisabled:false
mainSeriesProperties.priceAxisProperties.log:false
mainSeriesProperties.priceAxisProperties.logDisabled:false

symbolWatermarkProperties.color : "rgba(0, 0, 0, 0.00)"

// 不同的图表类型默认

// 蜡烛样式
mainSeriesProperties.candleStyle.upColor: "#6ba583"
mainSeriesProperties.candleStyle.downColor: "#d75442"
mainSeriesProperties.candleStyle.drawWick: true
mainSeriesProperties.candleStyle.drawBorder: true
mainSeriesProperties.candleStyle.borderColor: "#378658"
mainSeriesProperties.candleStyle.borderUpColor: "#225437"
mainSeriesProperties.candleStyle.borderDownColor: "#5b1a13"
mainSeriesProperties.candleStyle.wickUpColor: 'rgba( 115, 115, 117, 1)'
mainSeriesProperties.candleStyle.wickDownColor: 'rgba( 115, 115, 117, 1)'
mainSeriesProperties.candleStyle.barColorsOnPrevClose: false

// Hollow Candles styles
mainSeriesProperties.hollowCandleStyle.upColor: "#6ba583"
mainSeriesProperties.hollowCandleStyle.downColor: "#d75442"
mainSeriesProperties.hollowCandleStyle.drawWick: true
mainSeriesProperties.hollowCandleStyle.drawBorder: true
mainSeriesProperties.hollowCandleStyle.borderColor: "#378658"
mainSeriesProperties.hollowCandleStyle.borderUpColor: "#225437"
mainSeriesProperties.hollowCandleStyle.borderDownColor: "#5b1a13"
mainSeriesProperties.hollowCandleStyle.wickColor: "#737375"
```

```
//      Heiken Ashi styles
mainSeriesProperties.haStyle.upColor: "#6ba583"
mainSeriesProperties.haStyle.downColor: "#d75442"
mainSeriesProperties.haStyle.drawWick: true
mainSeriesProperties.haStyle.drawBorder: true
mainSeriesProperties.haStyle.borderColor: "#378658"
mainSeriesProperties.haStyle.borderUpColor: "#225437"
mainSeriesProperties.haStyle.borderDownColor: "#5b1a13"
mainSeriesProperties.haStyle.wickColor: "#737375"
mainSeriesProperties.haStyle.barColorsOnPrevClose: false

//      Bars styles
mainSeriesProperties.barStyle.upColor: "#6ba583"
mainSeriesProperties.barStyle.downColor: "#d75442"
mainSeriesProperties.barStyle.barColorsOnPrevClose: false
mainSeriesProperties.barStyle.dontDrawOpen: false

//      Line styles
mainSeriesProperties.lineStyle.color: "#0303F7"
mainSeriesProperties.lineStyle.linestyle: LINESTYLE_SOLID
mainSeriesProperties.lineStyle.linewidth: 1
mainSeriesProperties.lineStyle.priceSource: "close"

//      Area styles
mainSeriesProperties.areaStyle.color1: "#606090"
mainSeriesProperties.areaStyle.color2: "#01F6F5"
mainSeriesProperties.areaStyle.linecolor: "#0094FF"
mainSeriesProperties.areaStyle.linestyle: LINESTYLE_SOLID
mainSeriesProperties.areaStyle.linewidth: 1
mainSeriesProperties.areaStyle.priceSource: "close"
```

## LineStyles

LINESTYLE\_SOLID = 0

LINESTYLE\_DOTTED = 1

LINESTYLE\_DASHED = 2

LINESTYLE\_LARGE\_DASHED = 3

## 自定义覆盖商品

```
study_Overlay@tv-basicstudies.style: (bars = 0, candles = 1, line = 2, area = 3, heiken  
n ashi = 8, hollow candles = 9)  
study_Overlay@tv-basicstudies.showPriceLine: boolean  
  
study_Overlay@tv-basicstudies.candleStyle.upColor: color  
study_Overlay@tv-basicstudies.candleStyle.downColor: color  
study_Overlay@tv-basicstudies.candleStyle.drawWick: boolean  
study_Overlay@tv-basicstudies.candleStyle.drawBorder: boolean  
study_Overlay@tv-basicstudies.candleStyle.borderColor: color  
study_Overlay@tv-basicstudies.candleStyle.borderUpColor: color  
study_Overlay@tv-basicstudies.candleStyle.borderDownColor: color  
study_Overlay@tv-basicstudies.candleStyle.wickColor: color  
study_Overlay@tv-basicstudies.candleStyle.barColorsOnPrevClose: boolean  
  
study_Overlay@tv-basicstudies.hollowCandleStyle.upColor: color  
study_Overlay@tv-basicstudies.hollowCandleStyle.downColor: color  
study_Overlay@tv-basicstudies.hollowCandleStyle.drawWick: boolean  
study_Overlay@tv-basicstudies.hollowCandleStyle.drawBorder: boolean  
study_Overlay@tv-basicstudies.hollowCandleStyle.borderColor: color  
study_Overlay@tv-basicstudies.hollowCandleStyle.borderUpColor: color  
study_Overlay@tv-basicstudies.hollowCandleStyle.borderDownColor: color  
study_Overlay@tv-basicstudies.hollowCandleStyle.wickColor: color  
study_Overlay@tv-basicstudies.hollowCandleStyle.barColorsOnPrevClose: boolean  
  
study_Overlay@tv-basicstudies.barStyle.upColor: color  
study_Overlay@tv-basicstudies.barStyle.downColor: color  
study_Overlay@tv-basicstudies.barStyle.barColorsOnPrevClose: boolean  
study_Overlay@tv-basicstudies.barStyle.dontDrawOpen: boolean  
  
study_Overlay@tv-basicstudies.lineStyle.color: color  
study_Overlay@tv-basicstudies.lineStyle.linestyle: (solid = 0; dotted = 1; dashed = 2;  
large dashed = 3)  
study_Overlay@tv-basicstudies.lineStyle.linewidth: integer  
study_Overlay@tv-basicstudies.lineStyle.priceSource: open/high/low/close  
study_Overlay@tv-basicstudies.lineStyle.styleType: (bars = 0, candles = 1, line = 2, a  
rea = 3, heiken ashi = 8, hollow candles = 9)  
  
study_Overlay@tv-basicstudies.areaStyle.color1: color  
study_Overlay@tv-basicstudies.areaStyle.color2: color  
study_Overlay@tv-basicstudies.areaStyle.linecolor: color  
study_Overlay@tv-basicstudies.areaStyle.linestyle: (solid = 0; dotted = 1; dashed = 2;  
large dashed = 3)  
study_Overlay@tv-basicstudies.areaStyle.linewidth: integer  
study_Overlay@tv-basicstudies.areaStyle.priceSource: open/high/low/close
```

## 绘图覆盖

以下是绘图覆盖的全部列表及默认值。您可以使用[Widget构造函数](#)的 `overrides` 参数来更改默认值。在列表的底部，您可以找到值中使用的常量和缩写的列表。

```
linetoolicon: {
  singleChartOnly: true,
  color: 'rgba( 61, 133, 198, 1)',
  snapTo45Degrees:true,
  size: 40,
  icon: 0x263A,
  angle: Math.PI * 0.5,
  scale: 1.0
},
linetoolbezierquadro: {
  linecolor: 'rgba( 21, 153, 128, 1)',
  linewidth: 1.0,
  fillBackground: false,
  backgroundColor: 'rgba( 21, 56, 153, 0.5)',
  transparency: 50,
  linestyle: LINESTYLE_SOLID,
  extendLeft: false,
  extendRight: false,
  leftEnd: LINEEND_NORMAL,
  rightEnd: LINEEND_NORMAL
},
linetoolbeziercubic: {
  linecolor: 'rgba( 21, 153, 128, 1)',
  linewidth: 1.0,
  fillBackground: false,
  backgroundColor: 'rgba( 21, 56, 153, 0.5)',
  transparency: 50,
  linestyle: LINESTYLE_SOLID,
  extendLeft: false,
  extendRight: false,
  leftEnd: LINEEND_NORMAL,
  rightEnd: LINEEND_NORMAL
},
linetooltrendline: {
  linecolor: 'rgba( 21, 153, 128, 1)',
  linewidth: 1.0,
  linestyle: LINESTYLE_SOLID,
  extendLeft: false,
  extendRight: false,
  leftEnd: LINEEND_NORMAL,
  rightEnd: LINEEND_NORMAL,
  font: 'Verdana',
  textcolor: 'rgba( 21, 119, 96, 1)',
  fontsize: 12,
  bold:false,
  italic:false,
```

```
    snapTo45Degrees:true,
    alwaysShowStats: false,
    showPriceRange: false,
    showBarsRange: false,
    showDateTimeRange: false,
    showDistance:false,
    showAngle: false
  },
  linetooltimecycles: {
    linecolor: 'rgba(21, 153, 128, 1)',
    linewidth: 1.0,
    fillBackground: true,
    backgroundColor: 'rgba(106, 168, 79, 0.5)',
    transparency: 50,
    linestyle: LINESTYLE_SOLID
  },
  linetoolsineline: {
    linecolor: 'rgba( 21, 153, 128, 1)',
    linewidth: 1.0,
    linestyle: LINESTYLE_SOLID
  },
  linetooltrendangle: {
    singleChartOnly: true,
    linecolor: 'rgba( 21, 153, 128, 1)',
    linewidth: 1.0,
    linestyle: LINESTYLE_SOLID,
    snapTo45Degrees:true,
    font: 'Verdana',
    textcolor: 'rgba( 21, 119, 96, 1)',
    fontsize: 12,
    bold:true,
    italic:false,
    alwaysShowStats: false,
    showPriceRange: false,
    showBarsRange: false,
    extendRight: false,
    extendLeft: false
  },
  linetooldisjointangle: {
    linecolor: 'rgba( 18, 159, 92, 1)',
    linewidth: 2.0,
    linestyle: LINESTYLE_SOLID,
    fillBackground: true,
    backgroundColor: 'rgba( 106, 168, 79, 0.5)',
    transparency: 50,
    extendLeft: false,
    extendRight: false,
    leftEnd: LINEEND_NORMAL,
    rightEnd: LINEEND_NORMAL,
    font: 'Verdana',
    textcolor: 'rgba( 18, 159, 92, 1)',
    fontsize: 12,
    bold:false,
    italic:false,
    showPrices: false,
    showPriceRange: false,
```

```

        showDateTimeRange: false,
        showBarsRange: false
    },
    linetoolflatbottom: {
        linecolor: 'rgba( 73, 133, 231, 1)',
        linewidth: 2.0,
        linestyle: LINESTYLE_SOLID,
        fillBackground: true,
        backgroundColor: 'rgba( 21, 56, 153, 0.5)',
        transparency: 50,
        extendLeft: false,
        extendRight: false,
        leftEnd: LINEEND_NORMAL,
        rightEnd: LINEEND_NORMAL,
        font: 'Verdana',
        textcolor: 'rgba( 73, 133, 231, 1)',
        fontsize: 12,
        bold:false,
        italic:false,
        showPrices: false,
        showPriceRange: false,
        showDateTimeRange: false,
        showBarsRange: false
    },
    linetoolfibspiral: {
        linecolor: 'rgba( 21, 153, 128, 1)',
        linewidth: 1.0,
        linestyle: LINESTYLE_SOLID
    },
    linetooldaterange: {
        linecolor: 'rgba( 88, 88, 88, 1)',
        linewidth: 1.0,
        font: 'Verdana',
        textcolor: 'rgba( 255, 255, 255, 1)',
        fontsize: 12,
        fillLabelBackground: true,
        labelBackgroundColor: 'rgba( 91, 133, 191, 0.9)',
        labelBackgroundTransparency: 30,
        fillBackground: true,
        backgroundColor: 'rgba( 186, 218, 255, 0.4)',
        backgroundTransparency: 60,
        drawBorder: false,
        borderColor: 'rgba( 102, 123, 139, 1)'
    },
    linetoolpricerange: {
        linecolor: 'rgba( 88, 88, 88, 1)',
        linewidth: 1.0,
        font: 'Verdana',
        textcolor: 'rgba( 255, 255, 255, 1)',
        fontsize: 12,
        fillLabelBackground: true,
        labelBackgroundColor: 'rgba( 91, 133, 191, 0.9)',
        labelBackgroundTransparency: 30,
        fillBackground: true,
        backgroundColor: 'rgba( 186, 218, 255, 0.4)',
        backgroundTransparency: 60,

```



```

        drawBorder: false,
        borderColor: 'rgba( 102, 123, 139, 1)')
    },
    linetooldateandpricerange: {
        linecolor: 'rgba( 88, 88, 88, 1)',
        linewidth: 1.0,
        font: 'Verdana',
        textcolor: 'rgba( 255, 255, 255, 1)',
        fontsize: 12,
        fillLabelBackground: true,
        labelBackgroundColor: 'rgba( 91, 133, 191, 0.9)',
        labelBackgroundTransparency: 30,
        fillBackground: true,
        backgroundColor: 'rgba( 186, 218, 255, 0.4)',
        backgroundTransparency: 60,
        drawBorder: false,
        borderColor: 'rgba( 102, 123, 139, 1)')
    },
    linetoolriskrewardshort: {
        isShort:true,
        linecolor: 'rgba( 88, 88, 88, 1)',
        linewidth: 1.0,
        font: 'Verdana',
        textcolor: 'rgba(255, 255, 255, 1)',
        fontsize: 12,
        fillLabelBackground: true,
        labelBackgroundColor: 'rgba( 88, 88, 88, 1)',
        labelBackgroundTransparency: 0,
        fillBackground: true,
        stopBackground: 'rgba( 255, 0, 0, 0.2)',
        profitBackground: 'rgba( 0, 160, 0, 0.2)',
        stopBackgroundTransparency: 80,
        profitBackgroundTransparency: 80,
        drawBorder: false,
        borderColor: 'rgba( 102, 123, 139, 1)')
    },
    linetoolriskrewardlong: {
        isShort:false,
        linecolor: 'rgba( 88, 88, 88, 1)',
        linewidth: 1.0,
        font: 'Verdana',
        textcolor: 'rgba(255, 255, 255, 1)',
        fontsize: 12,
        fillLabelBackground: true,
        labelBackgroundColor: 'rgba( 88, 88, 88, 1)',
        labelBackgroundTransparency: 0,
        fillBackground: true,
        stopBackground: 'rgba( 255, 0, 0, 0.2)',
        profitBackground: 'rgba( 0, 160, 0, 0.2)',
        stopBackgroundTransparency: 80,
        profitBackgroundTransparency: 80,
        drawBorder: false,
        borderColor: 'rgba( 102, 123, 139, 1)')
    },
    linetoolarrow: {
        linecolor: 'rgba( 111, 136, 198, 1)',

```

```
    linewidth: 2.0,
    linestyle: LINESTYLE_SOLID,
    extendLeft: false,
    extendRight: false,
    leftEnd: LINEEND_NORMAL,
    rightEnd: LINEEND_ARROW,
    font: 'Verdana',
    textcolor: 'rgba( 21, 119, 96, 1)',
    fontsize: 12,
    bold:false,
    italic:false,
    alwaysShowStats: false,
    showPriceRange: false,
    showBarsRange: false,
    showDateTimeRange: false,
    showDistance:false,
    showAngle: false
},
linetoolray: {
    linecolor: 'rgba( 21, 153, 128, 1)',
    linewidth: 1.0,
    linestyle: LINESTYLE_SOLID,
    extendLeft: false,
    extendRight: true,
    leftEnd: LINEEND_NORMAL,
    rightEnd: LINEEND_NORMAL,
    font: 'Verdana',
    textcolor: 'rgba( 21, 119, 96, 1)',
    fontsize: 12,
    bold:false,
    italic:false,
    alwaysShowStats: false,
    showPriceRange: false,
    showBarsRange: false,
    showDateTimeRange: false,
    showDistance:false,
    showAngle: false
},
linetoolextended: {
    linecolor: 'rgba( 21, 153, 128, 1)',
    linewidth: 1.0,
    linestyle: LINESTYLE_SOLID,
    extendLeft: true,
    extendRight: true,
    leftEnd: LINEEND_NORMAL,
    rightEnd: LINEEND_NORMAL,
    font: 'Verdana',
    textcolor: 'rgba( 21, 119, 96, 1)',
    fontsize: 12,
    bold:false,
    italic:false,
    alwaysShowStats: false,
    showPriceRange: false,
    showBarsRange: false,
    showDateTimeRange: false,
    showDistance:false,
```

```
    showAngle: false
  },
  linetoolhorzline: {
    linecolor: 'rgba( 128, 204, 219, 1)',
    linewidth: 1.0,
    linestyle: LINESTYLE_SOLID,
    showPrice: true,
    showLabel: false,
    text: '',
    font: 'Verdana',
    textcolor: 'rgba( 21, 119, 96, 1)',
    fontsize: 12,
    bold:false,
    italic:false,
    horzLabelsAlign: 'center',
    vertLabelsAlign: 'top'
  },
  linetoolhorzray: {
    linecolor: 'rgba( 128, 204, 219, 1)',
    linewidth: 1.0,
    linestyle: LINESTYLE_SOLID,
    showPrice: true,
    showLabel: false,
    text: '',
    font: 'Verdana',
    textcolor: 'rgba( 21, 119, 96, 1)',
    fontsize: 12,
    bold:false,
    italic:false,
    horzLabelsAlign: 'center',
    vertLabelsAlign: 'top'
  },
  linetoolvertline: {
    linecolor: 'rgba( 128, 204, 219, 1)',
    linewidth: 1.0,
    linestyle: LINESTYLE_SOLID,
    showTime: true
  },
  linetoolcirclelines: {
    trendline: {
      visible: true,
      color: 'rgba( 128, 128, 128, 1)',
      linewidth: 1.0,
      linestyle: LINESTYLE_DASHED
    },
    linecolor: 'rgba( 128, 204, 219, 1)',
    linewidth: 1.0,
    linestyle: LINESTYLE_SOLID
  },
  linetoolfibtimezone: {
    horzLabelsAlign: 'right',
    vertLabelsAlign: 'bottom',
    baselinecolor: 'rgba( 128, 128, 128, 1)',
    linecolor: 'rgba( 0, 85, 219, 1)',
    linewidth: 1.0,
    linestyle: LINESTYLE_SOLID,
```

```
    showLabels: true,
    font: 'Verdana',
    fillBackground:false,
    transparency:80,
    trendline: {
        visible: true,
        color: 'rgba( 128, 128, 128, 1)',
        linewidth: 1.0,
        linestyle: LINESTYLE_DASHED
    },
    level1-11: LEVELS_TYPE_C
},
linetooltext: {
    color: 'rgba( 102, 123, 139, 1)',
    text: $.t('Text'),
    font: 'Verdana',
    fontsize: 20,
    fillBackground: false,
    backgroundColor: 'rgba( 91, 133, 191, 0.9)',
    backgroundTransparency: 70,
    drawBorder: false,
    borderColor: 'rgba( 102, 123, 139, 1)',
    bold:false,
    italic:false,
    locked: false,
    fixedSize: true,
    wordWrap: false,
    wordWrapWidth: 400
},
linetooltextabsolute: {
    singleChartOnly: true,
    color: 'rgba( 102, 123, 139, 1)',
    text: $.t('Text'),
    font: 'Verdana',
    fontsize: 20,
    fillBackground: false,
    backgroundColor: 'rgba( 155, 190, 213, 0.3)',
    backgroundTransparency: 70,
    drawBorder: false,
    borderColor: 'rgba( 102, 123, 139, 1)',
    bold: false,
    italic: false,
    locked: true,
    wordWrap: false,
    wordWrapWidth: 400
},
linetoolballoon: {
    color: 'rgba( 102, 123, 139, 1)',
    backgroundColor: 'rgba( 255, 254, 206, 0.7)',
    borderColor: 'rgba( 140, 140, 140, 1)',
    fontWeight: 'bold',
    fontsize: 12,
    font: 'Arial',
    transparency: 30,
    text: $.t('Comment')
},
```

```
linetoolbrush: {
  linecolor: 'rgba( 53, 53, 53, 1)',
  linewidth: 2.0,
  smooth: 5,
  fillBackground: false,
  backgroundColor: 'rgba( 21, 56, 153, 0.5)',
  transparency: 50,
  leftEnd: LINEEND_NORMAL,
  rightEnd: LINEEND_NORMAL
},
linetoolpolyline: {
  linecolor: 'rgba( 53, 53, 53, 1)',
  linewidth: 2.0,
  linestyle: LINESTYLE_SOLID,
  fillBackground: true,
  backgroundColor: 'rgba( 21, 56, 153, 0.5)',
  transparency: 50,
  filled: false
},
linetoolarrowmark: {
  color: 'rgba( 120, 120, 120, 1)',
  text: '',
  fontsize: 20,
  font: 'Verdana'
},
linetoolarrowmarkleft: {
  color: 'rgba( 120, 120, 120, 1)',
  text: '',
  fontsize: 20,
  font: 'Verdana'
},
linetoolarrowmarkup: {
  color: 'rgba( 120, 120, 120, 1)',
  text: '',
  fontsize: 20,
  font: 'Verdana'
},
linetoolarrowmarkright: {
  color: 'rgba( 120, 120, 120, 1)',
  text: '',
  fontsize: 20,
  font: 'Verdana'
},
linetoolarrowmarkdown: {
  color: 'rgba( 120, 120, 120, 1)',
  text: '',
  fontsize: 20,
  font: 'Verdana'
},
linetoolflagmark: {
  color: 'rgba( 255, 0, 0, 1)'
},
linetoolnote: {
  markerColor: 'rgba( 46, 102, 255, 1)',
  textColor: 'rgba( 0, 0, 0, 1)',
```

```
        backgroundColor: 'rgba( 255, 255, 255, 1)',
        backgroundTransparency: 0,
        text: 'Text',
        font: 'Arial',
        fontSize: 12,
        bold: false,
        italic: false,
        locked: false,
        fixedSize: true
    },

    linetoolnoteabsolute: {
        singleChartOnly: true,
        markerColor: 'rgba( 46, 102, 255, 1)',
        textColor: 'rgba( 0, 0, 0, 1)',
        backgroundColor: 'rgba( 255, 255, 255, 1)',
        backgroundTransparency: 0,
        text: 'Text',
        font: 'Arial',
        fontSize: 12,
        bold: false,
        italic: false,
        locked: true,
        fixedSize: true
    },

    linetoolthumbup: {
        color: 'rgba( 0, 128, 0, 1)'
    },
    linetoolthumbdown: {
        color: 'rgba( 255, 0, 0, 1)'
    },
    linetoolpricelabel: {
        color: 'rgba( 102, 123, 139, 1)',
        backgroundColor: 'rgba( 255, 255, 255, 0.7)',
        borderColor: 'rgba( 140, 140, 140, 1)',
        fontWeight: 'bold',
        fontsize: 11,
        font: 'Arial',
        transparency: 30
    },
    linetoolrectangle: {
        color: 'rgba( 21, 56, 153, 1)',
        fillBackground: true,
        backgroundColor: 'rgba( 21, 56, 153, 0.5)',
        linewidth: 1.0,
        snapTo45Degrees:true,
        transparency: 50
    },
    linetoolrotatedrectangle: {
        color: 'rgba( 152, 0, 255, 1)',
        fillBackground: true,
        backgroundColor: 'rgba( 142, 124, 195, 0.5)',
        transparency: 50,
        linewidth: 1.0,
        snapTo45Degrees:true
    }
```

```
},
linetoolellipse: {
  color: 'rgba( 153, 153, 21, 1)',
  fillBackground: true,
  backgroundColor: 'rgba( 153, 153, 21, 0.5)',
  transparency: 50,
  linewidth: 1.0
},
linetoolarc: {
  color: 'rgba( 153, 153, 21, 1)',
  fillBackground: true,
  backgroundColor: 'rgba( 153, 153, 21, 0.5)',
  transparency: 50,
  linewidth: 1.0
},
linetoolprediction: {
  singleChartOnly: true,
  linecolor: 'rgba( 28, 115, 219, 1)',
  linewidth: 2.0,

  sourceBackColor: 'rgba( 241, 241, 241, 1)',
  sourceTextColor: 'rgba( 110, 110, 110, 1)',
  sourceStrokeColor: 'rgba( 110, 110, 110, 1)',

  targetStrokeColor: 'rgba( 47, 168, 255, 1)',
  targetBackColor: 'rgba( 11, 111, 222, 1)',
  targetTextColor: 'rgba( 255, 255, 255, 1)',

  successBackground: 'rgba( 54, 160, 42, 0.9)',
  successTextColor: 'rgba( 255, 255, 255, 1)',

  failureBackground: 'rgba( 231, 69, 69, 0.5)',
  failureTextColor: 'rgba( 255, 255, 255, 1)',

  intermediateBackColor: 'rgba( 234, 210, 137, 1)',
  intermediateTextColor: 'rgba( 109, 77, 34, 1)',

  transparency: 10,
  centersColor: 'rgba( 32, 32, 32, 1)'
},
linetooltriangle: {
  color: 'rgba( 153, 21, 21, 1)',
  fillBackground: true,
  backgroundColor: 'rgba( 153, 21, 21, 0.5)',
  transparency: 50,
  linewidth: 1.0
},
linetoolcallout: {
  color: 'rgba( 255, 255, 255, 1)',
  backgroundColor: 'rgba( 153, 21, 21, 0.5)',
  transparency: 50,
  linewidth: 2.0,
  fontsize: 12,
  font: 'Verdana',
  text: 'Text',
  bordercolor: 'rgba( 153, 21, 21, 1)',
```

```
    bold: false,
    italic: false,
    wordWrap: false,
    wordWrapWidth: 400
  },
  linetoolparallelchannel: {
    linecolor: 'rgba( 119, 52, 153, 1)',
    linewidth: 1.0,
    linestyle: LINESTYLE_SOLID,
    extendLeft: false,
    extendRight: false,
    fillBackground: true,
    backgroundColor: 'rgba( 180, 167, 214, 0.5)',
    transparency: 50,
    showMidline: false,
    midlinecolor: 'rgba( 119, 52, 153, 1)',
    midlinewidth: 1.0,
    midlinestyle: LINESTYLE_DASHED
  },
  linetoolelliottimpulse: {
    degree: 7,
    showWave:true,
    color: 'rgba( 61, 133, 198, 1)',
    linewidth: 1
  },
  linetoolelliotttriangle: {
    degree: 7,
    showWave:true,
    color: 'rgba( 255, 152, 0, 1)',
    linewidth: 1
  },
  linetoolelliotttriplecombo: {
    degree: 7,
    showWave:true,
    color: 'rgba( 106, 168, 79, 1)',
    linewidth: 1
  },
  linetoolelliottcorrection: {
    degree: 7,
    showWave:true,
    color: 'rgba( 61, 133, 198, 1)',
    linewidth: 1
  },
  linetoolelliottdoublecombo: {
    degree: 7,
    showWave:true,
    color: 'rgba( 106, 168, 79, 1)',
    linewidth: 1
  },
  linetoolbarspattern: {
    singleChartOnly: true,
    color:'rgba( 80, 145, 204, 1)',
    mode:BARS_MODE,
    mirrored:false,
    flipped:false
  },
  },
```



```

linetoolghostfeed: {
    singleChartOnly: true,
    averageHL: 20,
    variance: 50,
    candleStyle: {
        upColor: '#6ba583',
        downColor: '#d75442',
        drawWick: true,
        drawBorder: true,
        borderColor: '#378658',
        borderUpColor: '#225437',
        borderDownColor: '#5b1a13',
        wickColor: '#737375'
    },
    transparency: 50
},
linetoolpitchfork: {
    fillBackground:true,
    transparency:80,
    style:PITCHFORK_STYLE_ORIGINAL,
    median: {
        visible: true,
        color: 'rgba( 165, 0, 0, 1)',
        linewidth: 1.0,
        linestyle: LINESTYLE_SOLID
    },
    level0-8: LEVELS_TYPE_C
},
linetoolpitchfan: {
    fillBackground:true,
    transparency:80,
    median: {
        visible: true,
        color: 'rgba( 165, 0, 0, 1)',
        linewidth: 1.0,
        linestyle: LINESTYLE_SOLID
    },
    level0-8: LEVELS_TYPE_C
},
linetoolgannfan: {
    showLabels: true,
    font: 'Verdana',
    fillBackground:true,
    transparency:80,
    level1-9: LEVELS_TYPE_F
},
linetoolganncomplex: {
    fillBackground:false,
    arcsBackground: {
        fillBackground: true,
        transparency: 80
    },
    levels: [/* 6 LEVELS_TYPE_D */],
    fanlines: [/* 11 LEVELS_TYPE_E */],
    arcs: [/* 11 LEVELS_TYPE_E */]
},

```

```

linetoolgannsquare: {
  color: 'rgba( 21, 56, 153, 0.8)',
  linewidth: 1.0,
  linestyle: LINESTYLE_SOLID,
  font: 'Verdana',
  showTopLabels:true,
  showBottomLabels:true,
  showLeftLabels:true,
  showRightLabels:true,
  fillHorzBackground:true,
  horzTransparency:80,
  fillVertBackground:true,
  vertTransparency:80,
  hlevel1-7: LEVELS_TYPE_B,
  vlevel1-7: LEVELS_TYPE_B
},
linetoolfibsresistancefan: {
  fillBackground:true,
  transparency:80,
  grid: {
    color: 'rgba( 128, 128, 128, 1)',
    linewidth: 1.0,
    linestyle: LINESTYLE_SOLID,
    visible:true
  },
  linewidth: 1.0,
  linestyle: LINESTYLE_SOLID,
  font: 'Verdana',
  showTopLabels:true,
  showBottomLabels:true,
  showLeftLabels:true,
  showRightLabels:true,
  snapTo45Degrees:true,
  hlevel1-7: LEVELS_TYPE_B,
  vlevel1-7: LEVELS_TYPE_B
},
linetoolfibretracement: {
  showCoeffs: true,
  showPrices: true,
  font: 'Verdana',
  fillBackground:true,
  transparency:80,
  extendLines:false,
  horzLabelsAlign: 'left',
  vertLabelsAlign: 'middle',
  reverse:false,
  coeffsAsPercents:false,
  trendline: {
    visible: true,
    color: 'rgba( 128, 128, 128, 1)',
    linewidth: 1.0,
    linestyle: LINESTYLE_DASHED
  },
  levelsStyle: {
    linewidth: 1.0,
    linestyle: LINESTYLE_SOLID
  }
}

```

```

    },
    level11-24: LEVELS_TYPE_B
  },
  linetoolfibchannel: {
    showCoeffs: true,
    showPrices: true,
    font: 'Verdana',
    fillBackground:true,
    transparency:80,
    extendLeft:false,
    extendRight:false,
    horzLabelsAlign: 'left',
    vertLabelsAlign: 'middle',
    coeffsAsPercents:false,
    levelsStyle: {
      linewidth: 1.0,
      linestyle: LINESTYLE_SOLID
    },
    level11-24: LEVELS_TYPE_B
  },
  linetoolprojection: {
    showCoeffs: true,
    font: 'Verdana',
    fillBackground:true,
    transparency:80,
    color1: 'rgba( 0, 128, 0, 0.2)',
    color2: 'rgba( 255, 0, 0, 0.2)',
    linewidth: 1.0,
    trendline: {
      visible: true,
      color: 'rgba( 128, 128, 128, 1)',
      linestyle: LINESTYLE_SOLID
    },
    level1: LEVELS_TYPE_C
  },
  linetool5pointspattern: {
    color: 'rgba( 204, 40, 149, 1)',
    textcolor: 'rgba( 255, 255, 255, 1)',
    fillBackground: true,
    backgroundColor: 'rgba( 204, 40, 149, 0.5)',
    font: 'Verdana',
    fontsize:12,
    bold:false,
    italic:false,
    transparency: 50,
    linewidth: 1.0
  },
  linetoolcypherpattern: {
    color: '#CC2895',
    textcolor: '#FFFFFF',
    fillBackground: true,
    backgroundColor: '#CC2895',
    font: 'Verdana',
    fontsize:12,
    bold:false,
    italic:false,

```

```
    transparency: 50,
    linewidth: 1.0
},
linetooltrianglepattern: {
    color: 'rgba( 149, 40, 255, 1)',
    textcolor: 'rgba( 255, 255, 255, 1)',
    fillBackground: true,
    backgroundColor: 'rgba( 149, 40, 204, 0.5)',
    font: 'Verdana',
    fontsize:12,
    bold:false,
    italic:false,
    transparency: 50,
    linewidth: 1.0
},
linetoolabcd: {
    color: 'rgba( 0, 155, 0, 1)',
    textcolor: 'rgba( 255, 255, 255, 1)',
    font: 'Verdana',
    fontsize:12,
    bold:false,
    italic:false,
    linewidth: 2.0
},
linetoolthreedrivers: {
    color: 'rgba( 149, 40, 255, 1)',
    textcolor: 'rgba( 255, 255, 255, 1)',
    fillBackground: true,
    backgroundColor: 'rgba( 149, 40, 204, 0.5)',
    font: 'Verdana',
    fontsize:12,
    bold:false,
    italic:false,
    transparency: 50,
    linewidth: 2.0
},
linetoolheadandshoulders: {
    color: 'rgba( 69, 104, 47, 1)',
    textcolor: 'rgba( 255, 255, 255, 1)',
    fillBackground: true,
    backgroundColor: 'rgba( 69, 168, 47, 0.5)',
    font: 'Verdana',
    fontsize:12,
    bold:false,
    italic:false,
    transparency: 50,
    linewidth: 2.0
},
linetoolfibwedge: {
    singleChartOnly: true,
    showCoeffs: true,
    font: 'Verdana',
    fillBackground:true,
    transparency:80,
    trendline: {
        visible: true,
```

```

        color: 'rgba( 128, 128, 128, 1)',
        linewidth: 1.0,
        linestyle: LINESTYLE_SOLID
    },
    level1-11: LEVELS_TYPE_C
},
linetoolfibcircles: {
    showCoeffs: true,
    font: 'Verdana',
    fillBackground:true,
    transparency:80,
    snapTo45Degrees:true,
    coeffsAsPercents:false,
    trendline: {
        visible: true,
        color: 'rgba( 128, 128, 128, 1)',
        linewidth: 1.0,
        linestyle: LINESTYLE_DASHED
    },
    level1-11: LEVELS_TYPE_C
},
linetoolfibspeeresistancearcs: {
    showCoeffs: true,
    font: 'Verdana',
    fillBackground:true,
    transparency:80,
    fullCircles: false,
    trendline: {
        visible: true,
        color: 'rgba( 128, 128, 128, 1)',
        linewidth: 1.0,
        linestyle: LINESTYLE_DASHED
    },
    level1-11: LEVELS_TYPE_C
},
linetooltrendbasedfibextension: {
    showCoeffs: true,
    showPrices: true,
    font: 'Verdana',
    fillBackground:true,
    transparency:80,
    extendLines:false,
    horzLabelsAlign: 'left',
    vertLabelsAlign: 'middle',
    reverse:false,
    coeffsAsPercents:false,
    trendline: {
        visible: true,
        color: 'rgba( 128, 128, 128, 1)',
        linewidth: 1.0,
        linestyle: LINESTYLE_DASHED
    },
    levelsStyle: {
        linewidth: 1.0,
        linestyle: LINESTYLE_SOLID
    },
},

```

```

    level11-24: LEVELS_TYPE_B
},
linetooltrendbasedfibtime: {
    showCoeffs: true,
    font: 'Verdana',
    fillBackground:true,
    transparency:80,
    horzLabelsAlign: 'right',
    vertLabelsAlign: 'bottom',
    trendline: {
        visible: true,
        color: 'rgba( 128, 128, 128, 1)',
        linewidth: 1.0,
        linestyle: LINESTYLE_DASHED
    },
    level11-11: LEVELS_TYPE_C
},
linetoolschiffpitchfork: {
    fillBackground:true,
    transparency:80,
    style:PITCHFORK_STYLE_SCHIFF,
    median: {
        visible: true,
        color: 'rgba( 165, 0, 0, 1)',
        linewidth: 1.0,
        linestyle: LINESTYLE_SOLID
    },
    level0-8: LEVELS_TYPE_C
},
linetoolschiffpitchfork2: {
    fillBackground:true,
    transparency:80,
    style:PITCHFORK_STYLE_SCHIFF2,
    median: {
        visible: true,
        color: 'rgba( 165, 0, 0, 1)',
        linewidth: 1.0,
        linestyle: LINESTYLE_SOLID
    },
    level0-8: LEVELS_TYPE_C
},
linetoolinsidepitchfork: {
    fillBackground:true,
    transparency:80,
    style:PITCHFORK_STYLE_INSIDE,
    median: {
        visible: true,
        color: 'rgba( 165, 0, 0, 1)',
        linewidth: 1.0,
        linestyle: LINESTYLE_SOLID
    },
    level0-8: LEVELS_TYPE_C
},
linetoolvisibilities: {
    intervalsVisibilities: {
        seconds:true,

```

```
        secondsFrom:1,
        secondsTo:59,
        minutes:true,
        minutesFrom:1,
        minutesTo:59,
        hours:true,
        hoursFrom:1,
        hoursTo:24,
        days:true,
        daysFrom:1,
        daysTo:366,
        weeks:true,
        months:true
    }
}
```

## 常量

这些常量用于绘图的默认值。你不能直接使用他们的名字，而是使用这些值。

```
LINESTYLE_SOLID = 0;
LINESTYLE_DOTTED = 1;
LINESTYLE_DASHED = 2;
LINESTYLE_LARGE_DASHED = 3;

LINEEND_NORMAL = 0;
LINEEND_ARROW = 1;
LINEEND_CIRCLE = 2;

BARS_MODE = 0;
LINE_MODE = 1;
OPENCLOSE_MODE = 2;
LINEOPEN_MODE = 3;
LINEHIGH_MODE = 4;
LINELOW_MODE = 5;
LINEHL2_MODE = 6;

PITCHFORK_STYLE_ORIGINAL = 0;
PITCHFORK_STYLE_SCHIFF = 1;
PITCHFORK_STYLE_SCHIFF2 = 2;
PITCHFORK_STYLE_INSIDE = 3;

LEVELS_TYPE_A = {
    color: color,
    visible: visible
};

LEVELS_TYPE_B = {
    coeff: coeff,
    color: color,
    visible: visible
};
```

```
LEVELS_TYPE_C = {
    coeff: coeff,
    color: color,
    visible: visible,
    linestyle: linestyle,
    linewidth: linewidth
};

LEVELS_TYPE_D = {
    color: color,
    width: width,
    visible: visible
};

LEVELS_TYPE_E = {
    color: color,
    visible: visible,
    width: width,
    x: x,
    y: y
};

LEVELS_TYPE_F = {
    coeff1: coeff1,
    coeff2: coeff2,
    color: color,
    visible: visible,
    linestyle: linestyle,
    linewidth: linewidth
};
```



## 研究覆盖

可以使用 `studies_overrides` 参数为新创建的指标设置默认样式和输入值。它的值应该是一个对象，其中`key`是一个属性被改变的路径，`value`是它的新值。例：

```
studies_overrides: {
  "volume.volume.color.0": "#00FFFF",
  "volume.volume.color.1": "#0000FF",
  "volume.volume.transparency": 70,
  "volume.volume.ma.color": "#FF0000",
  "volume.volume.ma.transparency": 30,
  "volume.volume.ma.linewidth": 5,
  "volume.show.ma": true,
  "volume.options.showStudyArguments": false,
  "bollinger_bands.median.color": "#33FF88",
  "bollinger_bands.upper.linewidth": 7
}
```

在上面的例子中，所有创建的布林带将具有上边线宽度 = 7（除非您通过API创建并为此线指定了另一个值）。

## 如何设置研究名称

您应该在新建研究对话框中使用研究名称，但采用小写形式。所以，如果你想覆盖默认的EMA长度，尝试使用 `moving average exponential.length`。同样的原则适用于输入名称：使用名称，您可以在“研究属性”对话框中看到它们（也使用小写字母）。例

如：`stochastic.smooth d`。

## 比较

您可以通过 `比较` 自定义新的系列。使用 `compare.plot` 自定义行和 `compare.source` 来更改价格来源：

```
"compare.plot.color": "#000000",
"compare.source": "high"
```

## 语法

属性路径是用点（.）分割的一组小写标识符。路径格式如下所述。

备注：如果一个plot/band/area/input名称是相同的，则您会得到一个错误。在这种情况下，您可以通过在路径中添加 `:plot`，`:band`，`:area` 或 `:input` 来指定一个确切的目的地。（例如 `short:plot.color`）

## Study input

格式: `indicator_name.input_name`

- **indicator\_name**: 使用名称，您可以在“指标器”对话框中看到它。
- **input\_name**: 使用名称，你可以在指标的属性对话框中看到它（例如 `show ma`）

例如: `volume.show ma`，`bollinger bands.length`

## 绘图属性

Format: `indicator_name.plot_name.property_name`

- **indicator\_name**: `< ... >`
- **plot\_name**: 你可以在指标的属性对话框中看到它（例如 `Volume` 或 `Plot`）
- **property\_name**: 下列之一:
  - **transparency**
  - **linewidth**
  - **plotype**. 支持的绘图类型有:
    - `line`（线形图）
    - `histogram`（直方图）
    - `cross`（十字指针）
    - `area`（山形图）
    - `columns`（柱状图）
    - `circles`（圆圈图）
    - `line_with_breaks`（中断线）
    - `area_with_breaks`（中断区块）

例子: `volume.volume.transparency`，`bollinger bands.median.linewidth`

## 绘图颜色

格式: `indicator_name.plot_name.color<.color_index>`

- **indicator\_name**: `< ... >`
- **plot\_name**: `< ... >`
- **color**. 这只是一个关键字。
- **color\_index**（可选）：颜色索引（如果有的话）。这只是一个颜色索引。也就是说，要取代成交量默认为绿色的颜色，应该使用 `color_index = 1`。

备注1: `color.0` 是 `color` 的同义词。因此路径 `volume.volume.color.0` 和 `volume.volume.color` 被视为相同。

备注2: 现在，不支持自定义区域填充颜色和透明度。

限制:

- 颜色只支持 `#RRGGBB` 格式。不要使用短格式的“`#RGB`”。
- 透明度在`[0..100]`范围内变化。100意味着完全不透明的。
- 厚度是一个整数。

## 研究选项

格式: `indicator_name.options.option_name`

- **indicator\_name**: `< ... >`
- **options**: 关键字
- **option\_name**: 你想分配的选项名称。支持的值是：
  - **showStudyArguments**: boolean, 控制标题中的参数可见性
  - **showLastValue**: boolean, 控制价格标签的可见性

例子: `volume.options.showStudyArguments` , `volume.options.showLastValue`

## 默认精度

1.6版本开始，您可以使用 `name.precision` 格式更改研究的默认精度。例: `"average true range.precision": 8`

# 形状与覆盖

使用createMultipoint图形(points, options, callback)可以创建50多个不同的图形。 这些全部包含在下表中，默认属性可以使用 overrides 来改变。

图形	背景颜色	背景透明度	粗体	边框颜色	颜色	绘图边框	填充
text	#9BBED5	70	FALSE	#667B8B	#667B8B	FALSE	FAI
anchored_text	#9BBED5	70	FALSE	#667B8B	#667B8B	FALSE	FAI
note	#FFFFFF	0	FALSE				
anchored_note	#FFFFFF	0	FALSE				
callout	#991515		FALSE	#991515	#FFFFFF		
balloon	#ffece			#8c8c8c	#667b8b		

图形	背景颜色	边框颜色	颜色	字体	字体大小	文本	透明度	字体粗细
arrow_up			#787878	Verdana	20	text		
arrow_down			#787878	Verdana	20	text		
arrow_left			#787878	Verdana	20	text		
arrow_right			#787878	Verdana	20	text		
price_label	#ffffff	#8c8c8c	#667b8b	Arial	11		30	bold
flag								

图形	背景颜色	粗体	颜色	填充背景	字体	字体大小
xabcd_pattern	#CC2895	FALSE	#CC2895	TRUE	Verdana	12
abcd_pattern		FALSE	#009B00		Verdana	12
triangle_pattern	#9528CC	FALSE	#9528FF	TRUE	Verdana	12
3divers_pattern	#9528CC	FALSE	#9528FF	TRUE	Verdana	12
head_and_shoulders	#45A82F	FALSE	#45682F	TRUE	Verdana	12
cypher_pattern	#CC2895	FALSE	#CC2895	TRUE	Verdana	12

图形	背景颜色	颜色	级别	向左延伸	向右延伸	填充背景
elliott_impulse_wave		#3d85c6	7			
elliott_triangle_wave		#ff9800	7			
elliott_triple_combo		#6aa84f	7			
elliott_correction		#3d85c6	7			
elliott_double_combo		#6aa84f	7			
cyclic_lines						
time_cycles	#6AA84F					TRUE
sine_line						
rectangle	#153899	#153899				TRUE
rotated_rectangle	#8e7cc3	#9800ff				TRUE
ellipse	#999915	#999915				TRUE
triangle	#991515	#991515				TRUE
polyline	#153899					TRUE
curve	#153899			FALSE	FALSE	FALSE
double_curve	#153899			FALSE	FALSE	FALSE
arc	#999915	#999915				TRUE

图形	背景颜色	<b>bold</b>	<b>extendLeft</b>	<b>extendRight</b>	填充背景	<b>font</b>
vertical_line						
horizontal_line		TRUE				Verdana
horizontal_ray		TRUE				Verdana
trend_line		FALSE	FALSE	FALSE		Verdana
trend_angle		TRUE	FALSE	FALSE		Verdana
arrow		FALSE	FALSE	FALSE		Verdana
ray		FALSE	FALSE	TRUE		Verdana
extended		FALSE	TRUE	TRUE		Verdana
parallel_channel	#b4a7d6		FALSE	FALSE	TRUE	
disjoint_angle	#6AA84F	FALSE	FALSE	FALSE	TRUE	Verdana
flat_bottom	#153899	FALSE	FALSE	FALSE	TRUE	Verdana
fib_spiral						

图形	填充背景	<b>transparency</b>	<b>style</b>	<b>median.visible</b>	<b>median</b>
pitchfork	TRUE	80	0	TRUE	#A50
schiff_pitchfork_modified	TRUE	80	1	TRUE	#A50
schiff_pitchfork	TRUE	80	3	TRUE	#A50
inside_pitchfork	TRUE	80	2	TRUE	#A50
pitchfan	TRUE	80		TRUE	#A50

图形	填充背景	<b>arcsBackground.</b> 填充背景	<b>arcsBackground.transparency</b>	
gannbox_square	FALSE	TRUE	50	

图形	<b>showLabels</b>	<b>font</b>	填充背景	<b>transparency</b>	<b>level1.visible</b>
gannbox_fan	TRUE	Verdana	TRUE	80	TRUE

图形	<b>color</b>	<b>linewidth</b>	<b>linestyle</b>	<b>font</b>	<b>showTopLabels</b>
gannbox	#153899	1	0	Verdana	TRUE
fib_speed_resist_fan		1	0	Verdana	TRUE

图形	showCoeffs	showPrices	font	填充背景	transpa
fib_retracement	TRUE	TRUE	Verdana	TRUE	80
fib_trend_ext	TRUE	TRUE	Verdana	TRUE	80
fib_timezone			Verdana	FALSE	80
fib_trend_time	TRUE		Verdana	TRUE	80
fib_circles	TRUE		Verdana	TRUE	80
fib_speed_resist_arcs	TRUE		Verdana	TRUE	80
fib_wedge	TRUE		Verdana	TRUE	80
fib_channel	TRUE	TRUE	Verdana	TRUE	80

图形	边框颜色	drawBorder	填充背景	fillLabelBackground
date_range	#667B8B	FALSE	TRUE	TRUE
price_range	#667B8B	FALSE	TRUE	TRUE
date_and_price_range	#667B8B	FALSE	TRUE	TRUE

图形	边框颜色	drawBorder	填充背景	fillLabelBackground	font
long_position	#667B8B	FALSE	TRUE	TRUE	Verdana
short_position	#667B8B	FALSE	TRUE	TRUE	Verdana

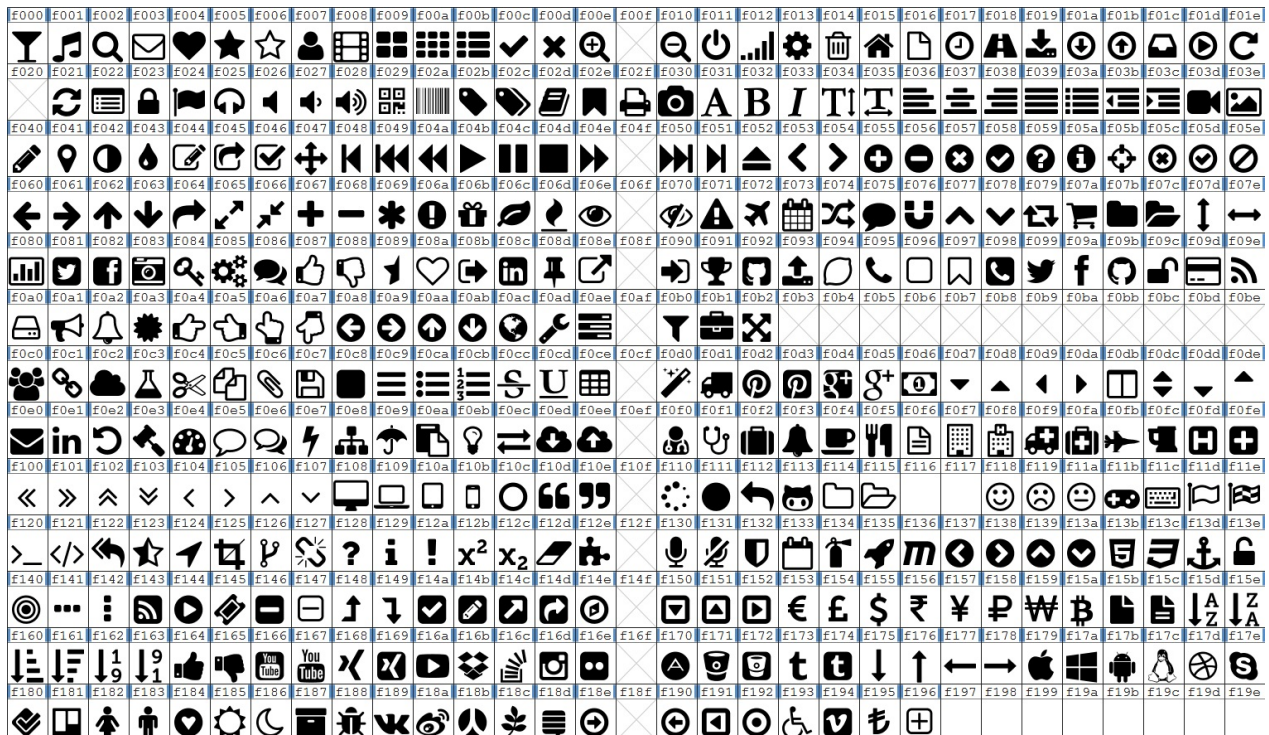
图形	showCoeffs	font	填充背景	transparency	color1	color2
projection	TRUE	Verdana	TRUE	80	#008000	#FF0000

图形	linecolor	linewidth	centersColor	failureBackground	failureText
forecast	#1c73db	2	#202020	#e74545	#ffffff

图形	averageHL	variance	transparency	candleStyle.upColor	car
ghost_feed	20	50	50	#6BA583	#D

图形	color	size	angle (rad)	scale	icon*
icon	#3d85c6	40	1,571	1	0x263A

\* 图标可以是以下值之一:



图形	color	flipped	mirrored	mode
bars_pattern	#5091CC	FALSE	FALSE	0

一些属性的可能值:

- linestyle : [0 (solid), 1 (dotted), 2 (dashed), 3 (large dashed)]
- linewidth : [1, 2, 3, 4]
- horzLabelsAlign : ["center", "left", "right"]
- vertLabelsAlign : ["top", "middle", "bottom"]
- leftEnd , rightEnd : [0 (Normal), 1 (Arrow)]
- bars\_pattern - mode : [0 (HL Bars), 1 (Line-Close), 2 (OC Bars), 3 (Line-Open), 4 (Line-High), 5 (Line-Low), 6 (Line-HL/2)]



## 订阅

Subscription对象由[Chart Methods](#)返回。使用此对象，您可以订阅图表事件并取消订阅。所以有两种方法：

### subscribe(object, method, singleshot)

1. `object` 是一个上下文对象，用作 `method` 函数的 `this` 指针。如果你不需要上下文，可以使用 `null`。
2. `method` 是事件发生时要调用的方法
3. `singleshot` 是一个可选的参数。设置为 `true`，当事件第一次发生时自动取消订阅。

### unsubscribe(object, method)

Use the same `object` and `method` which you used in `subscribe` function to unsubscribe from the event.

使用 `unsubscribe` 方法中使用的 `object` 和 `method` 来取消订阅事件。

### unsubscribeAll(object)

使用 `unsubscribe` 方法中使用的 `object` 和 `method` 来取消订阅所有事件。

# 交易原语

交易原语是一种低级机制，旨在为您提供对交易原语行为的最详细控制。

## 通用数据

### 属性

您可以为交易行对象的某些属性使用特殊值 `inherit` 这将使图书馆使用该属性的出厂默认值。您可以开启 `trading_options` 功能来显示交易界面。

您不能控制特定对象的可见性 - 在图表属性窗口的交易选项卡中可以为用户提供头寸、订单和交易指令的通用属性。

### 颜色和字体

您可以使用以下字符串格式设置颜色：

- 1. `"#RGB"`
- 2. `"#RRGGBB"`
- 3. `"rgb(red, green, blue)"`
- 4. `"rgba(red, green, blue, alpha)"`

您可以使用以下字符串格式设置字体：`<bold> <italic> <size>pt <family>`。颜色和字体字符串将自动解析，以确定GUI控件的值。

### 线条样式

支持以下线条样式：

Style	Value
Solid	0
Dotted	1
Dashed	2

### 回调

1. 您可以通过 `this` 关键字在回调中访问API对象
2. 您可以将两个参数传递给回调注册函数 - 在这种情况下，第一个参数是一个对象，它将作为第一个参数传递给回调函数（请参阅示例）。
3. 如果未设置回调，则相应的按钮将被隐藏（影响 `onReverse` , `onClose` 和 `onCancel` 的回调).

## 在K线上做标记

图表库支持在K线上显示标记。这里是一个例子：



每个标记可以有一个颜色，大小，标签和弹出消息。如果它声明支持它们，则从后端请求标记。标记旨在让您能够显示附加到K线的一些事件。这里有一些例子：

- 新闻
- 一些比较特殊的K线配置
- 拆分/股息
- 等等

# 委托

委托对象由一些 [Trading Terminal](#) 方法返回，并且需要使用 [Account Manager](#)。使用这个对象，你可以更新和更新一个事件。

## subscribe(object, member)

订阅事件。

1. `object` 是 `member` 的所有者，它可以为 `null`
2. `member` 是对象的一种方法

## unsubscribe(object, member)

取消订阅事件 使用与 `subscribe` 函数中使用的相同的对象和成员来取消订阅事件。

# WatchedValue

WatchedValue对象由一些Trading Terminal方法返回。使用这个对象，你可以获得/设置值，并在值改变时得到通知。

## value()

返回当前值。

## setValue(value)

设置新价值。

## subscribe(callback, options)

1. `callback` 值被改变时被调用的函数
2. `options` 具有以下属性的对象：
  - i. `once` - 如果是`true`，回调将只执行一次
  - ii. `callWithLast` - 如果它是`true`，回调将被执行以前的值（如果可用）

## unsubscribe(callback)

使用您在 `subscribe` 函数中使用的相同函数来取消订阅更新。