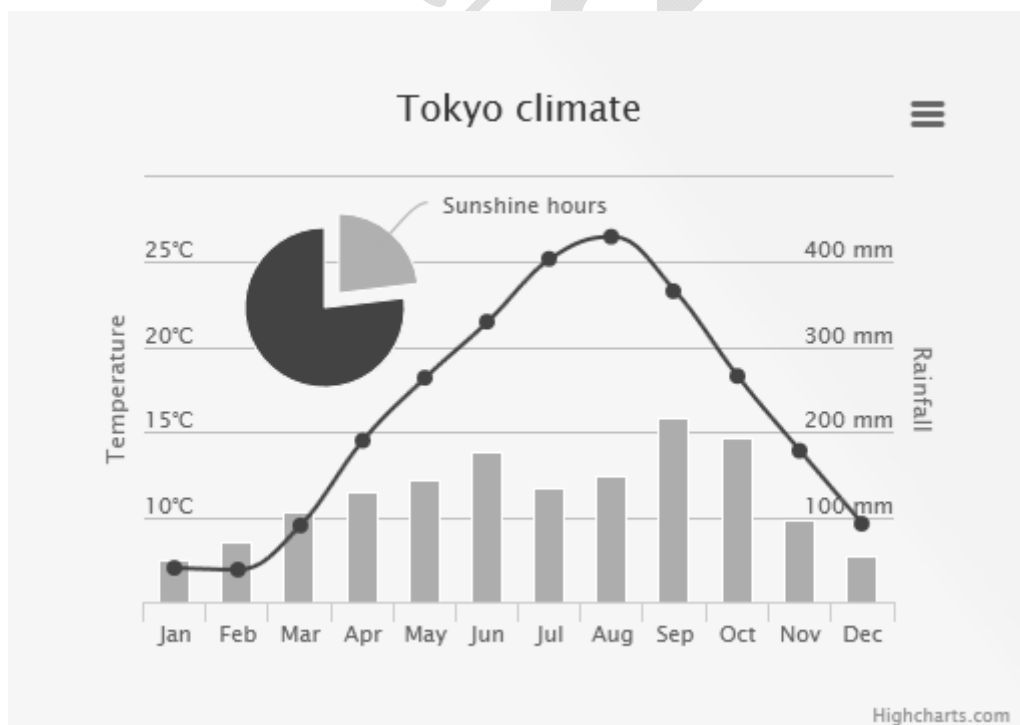


# 网页图表 Highcharts

## 实践教程基础篇

(内部资料 v1.0)



大学霸

[www.daxueba.net](http://www.daxueba.net)



---

# 前言

随着 Web 的应用的发展，用户对页面的精美程度要求越来越高。在数据呈现方面，逐步从传统的单调的表格方式过渡精美的图表方式。由于浏览器在图形绘制的短板，使得构建图表变得非常困难。为了满足网页开发者的需要，很多图表插件应运而生。

Highcharts 是国际知名的一款网页图表插件。该插件使用 Javascript 实现，容易获得 Web 浏览器各种支持。同时，它采用模版化的设计方式，用户只需要写少量的代码，就可以实现各种精美的图表。这样，可以节省开发者的大量时间。同时，该插件提供丰富的 API，允许用户对图表进行各种定制，以满足不同的应用需求。

本教程是国内第一本 Highcharts 专向教程。本教程作为基础篇，着重讲解 Highcharts 图表的构成方式，全面讲解图表各个元素的设置方法。读者可以借助本教程，实现各种各样的图表，以满足网站和 Web 应用需求。

## 1.学习所需的系统和设备

本教程在编写时基于以下操作系统和开发环境：

- ☐ Windows 7 操作系统
- ☐ Visual Studio 2012
- ☐ IE 11、FireFox 37

## 2.学习建议

大家学习之前，可以致信到 xxxxxxxxxxxx，获取相关的资料和软件。如果大家在学习过程遇到问题，也可以将问题发送到该邮箱。我们尽可能给大家解决。



## 目 录

第 1 章 认识 Highcharts.....	1
1.1 Highcharts 概述.....	1
1.1.1 下载 Highcharts.....	1
1.2.2 Highcharts 的图表类型.....	3
1.2 第一个实例.....	9
1.3 图表构成.....	10
1.3.1 界面构成.....	10
1.3.2 代码构成.....	11
1.4 商业授权和定制开发.....	11
1.4.1 商业授权.....	11
1.4.2 定制开发.....	12
第 2 章 图表区.....	14
2.1 图表区的构成.....	14
2.2 外层图表区.....	14
2.2.1 外层图表区的基本设置.....	14
2.2.2 外层图表区的边框.....	16
2.2.3 外层图表区的范围.....	17
2.2.4 外层图表区和图表内容的间距.....	18
2.2.5 外层图表区阴影.....	20
2.2.6 图表区样式.....	21
2.3 绘图区.....	22
2.3.1 绘图区的边框.....	22
2.3.2 绘图区的背景.....	23
2.4 图表缩放.....	24
2.4.1 图表缩放类型.....	25
2.4.2 填充选择区.....	26
2.4.3 重置按钮.....	26
2.5 图表平移.....	30
2.5.1 启用图表平移.....	30
2.5.2 设置图表平移方式.....	30
第 3 章 辅助元素.....	32
3.1 标题/副标题.....	32
3.1.1 标题和副标题的构成.....	32
3.1.2 标题的样式.....	33
3.1.3 标题的布局方式.....	34
3.2 版权信息.....	36

3.2.1	启用版权信息 .....	37
3.2.2	设置版权信息内容 .....	37
3.2.3	设置版权信息位置和样式 .....	37
3.3	标签组 .....	38
3.3.1	标签组的结构 .....	38
3.3.2	构建标签 .....	39
3.3.3	设置标签的样式 .....	39
3.4	载入动画 .....	40
3.4.1	显示载入动画 .....	40
3.4.2	本地化载入动画 .....	41
3.4.3	设置动画效果 .....	42
第 4 章	图例 .....	44
4.1	图例的构成 .....	44
4.2	图例区 .....	44
4.2.1	图例区基本设置 .....	45
4.2.2	图例区的布局 .....	47
4.2.2	图例区边框 .....	50
4.3	标题 .....	51
4.4	图例项目 .....	52
4.4.1	图例项目的构成 .....	52
4.4.2	图例项目的基本设置 .....	52
4.4.3	图例符号 .....	54
4.4.4	图例标签 .....	55
4.4.5	图例项目的布局 .....	57
4.4.6	图例项目的状态 .....	59
4.5	图例的导航 .....	60
第 5 章	坐标轴 .....	62
5.1	坐标轴的基本构成 .....	62
5.1.1	构建坐标轴 .....	62
5.1.2	坐标轴组 .....	64
5.1.3	动态添加/删除坐标轴 .....	67
5.1.4	坐标轴类型 .....	70
5.2	轴线 .....	72
5.2.1	坐标轴线 .....	72
5.2.2	标题 .....	75
5.3	刻度 .....	79
5.3.1	设定刻度值 .....	79
5.3.2	主刻度线 .....	82
5.3.3	次刻度线 .....	85
5.3.4	刻度标签 .....	86
5.3.5	时间日期型坐标轴刻度 .....	89

5.3.6	标签空间的节省 .....	94
5.3.7	间隔部分 .....	98
5.4	网格线 .....	100
5.4.1	主网格线 .....	100
5.4.2	次网格线 .....	101
5.5	数据标志线 .....	102
5.5.1	标志线 .....	102
5.5.2	标志线标签 .....	105
5.6	数据条带 .....	106
5.6.1	条带 .....	106
5.6.2	条带标签 .....	109
第 6 章	数据列和数据节点 .....	112
6.1	数据列 .....	112
6.1.1	数据列的配置 .....	112
6.1.2	数据列的构成 .....	113
6.1.3	基本配置 .....	113
6.1.4	关联图例 .....	117
6.1.5	关联坐标轴 .....	121
6.1.6	设置阈值 .....	123
6.1.7	数据列裁剪 .....	123
6.1.8	数据列事件 .....	124
6.2	关联鼠标 .....	124
6.2.1	禁止鼠标跟随 .....	125
6.2.2	禁止黏性跟随 .....	126
6.2.3	设置悬浮状态 .....	126
6.3	数据列区域 .....	129
6.4	数据节点 .....	130
6.4.1	定义节点 .....	130
6.4.2	节点标记 .....	134
6.4.3	节点标签 .....	139
6.4.4	节点状态 .....	144
6.4.5	节点事件 .....	145
第 7 章	提示框 .....	147
7.1	提示框构成 .....	147
7.1.1	提示框简介 .....	147
7.1.2	页眉 .....	149
7.1.3	节点信息 .....	150
7.1.4	页脚 .....	151
7.1.5	十字准线 .....	152
7.1.5	整体设置 .....	154
7.2	提示框外观 .....	155

7.2.1 外观 .....	155
7.2.2 动画效果 .....	157





## 第 1 章 认识 Highcharts

Highcharts 是国际知名的一款图表插件。它完全使用 Javascript 编写实现。其结构清晰，使用简单。开发人员可以很轻松地构建出常见的各种图表类型。本章将简要介绍 Highcharts 的特点，并实现第一个 Highcharts 图表。

### 1.1 Highcharts 概述

为了更好地学习 Highcharts 的使用，我们首先了解如何获取 Highcharts 插件和 Highcharts 所支持的图表类型。

#### 1.1.1 下载 Highcharts

Highcharts 官网提供了完整的 Javascript 脚本和范例程序。开发者都可以免费获取这些资源。下面讲解如何下载 Highcharts。

(1) 在浏览器打开官网 <http://www.highcharts.com/>，如图 1.1 所示。



图 1.1 Highcharts 官方首页

(2) 单击 Highcharts 页面下的 Download 按钮，进入 Highcharts 的下载页面，如图 1.2 所示。

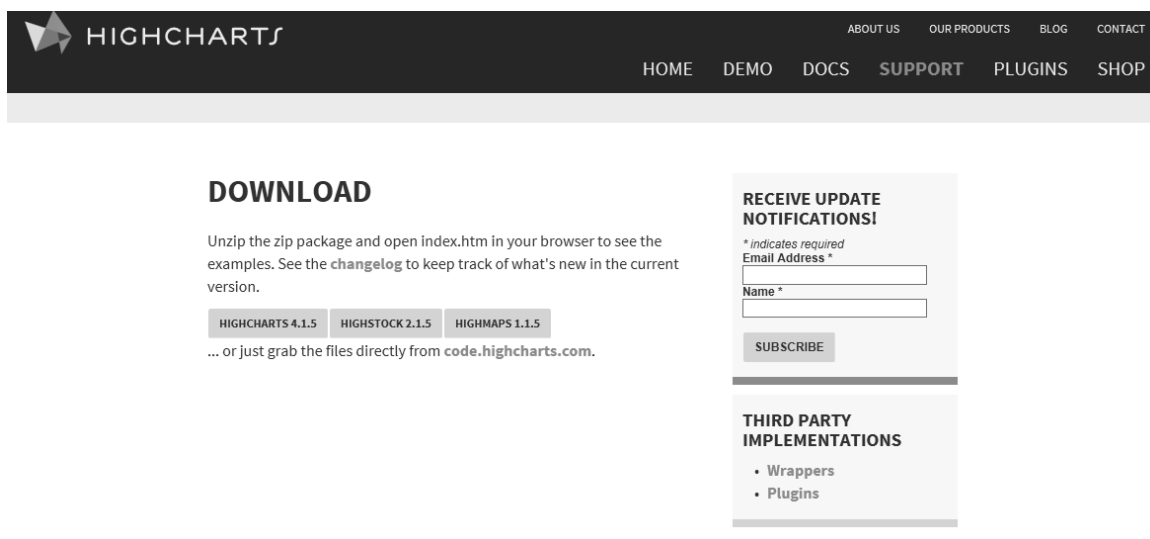


图 1.2 Highcharts 下载页面

(3) 该页面提供两种两种下载：普通下载和定制下载。这里选择普通下载模式，直接单击 HIGHCHARTS 4.1.5 按钮，下载保存文件 Highcharts-4.1.5.zip。解压该文件后如图 1.3 所示。

api	2015/4/13 14:56	文件夹
examples	2015/4/13 14:56	文件夹
exporting-server	2015/4/13 14:56	文件夹
gfx	2015/4/13 14:56	文件夹
graphics	2015/4/13 14:56	文件夹
js	2015/4/13 14:56	文件夹
index.htm	2015/4/13 14:56	HTML 文档

图 1.3 Highcharts 文件结构

这些文件夹依次保存着 Highcharts 各种重要文件。

- ❑ api 文件夹中保存着离线的 API 说明文档；
- ❑ examples 文件夹中保存着 Highcharts 官方提供的各种范例程序；
- ❑ exporting-server 文件夹保存着图表导出功能实现的服务器端代码；
- ❑ gfx 文件夹保存这 VML 功能所需要的图形文件；
- ❑ graphics 文件夹保存着范例程序所使用的图形文件；

js 目录中保存着 Highcharts 最重要的 js 文件，如图 1.4 所示。它里面包含几个文件夹，下面依次讲解讲解：










 adapters	2015/4/13 14:56	文件夹	
 modules	2015/4/13 14:56	文件夹	
 themes	2015/4/13 14:56	文件夹	
 highcharts.js	2015/4/13 14:54	JScript Script 文件	159 KB
 highcharts.src.js	2015/4/13 14:56	JScript Script 文件	488 KB
 highcharts-3d.js	2015/4/13 14:55	JScript Script 文件	19 KB
 highcharts-3d.src.js	2015/4/13 14:56	JScript Script 文件	36 KB
 highcharts-more.js	2015/4/13 14:55	JScript Script 文件	24 KB
 highcharts-more.src.js	2015/4/13 14:56	JScript Script 文件	65 KB

图 1.4 js 目录结构

当前目录下的 Javascript 脚本文件分为两类。直接以.js 结尾的脚本文件是最常使用的文件。这些文件在使用的时候需要依赖 jQuery 框架。以.src.js 结尾的文件是源码文件，供开发者查阅。下面依次讲解其中的几个文件夹。

- ❑ adapters 文件夹保存着 Highcharts 适配 Monntools、Prototype 框架的脚本文件；
- ❑ modules 文件夹保存着 Highcharts 的一些模块功能脚本；
- ❑ themes 文件夹保存着 Highcharts 图标主题脚本。

## 1.2.2 Highcharts 的图表类型

Highcharts 支持的大量的图标类型。用户可以通过范例文件查看这些图标的类型。双击 index.html 文件，可以打开示例页面。在该页面列出十大类示例，如下所示：

注意：由于网页中引用了 Google API 的托管代码，所以查看示例的时候，最好使用 VPN 联网查看。否则，会造成页面无法打开的问题。

（1）Line Charts（折线图），如图 1.5 所示。

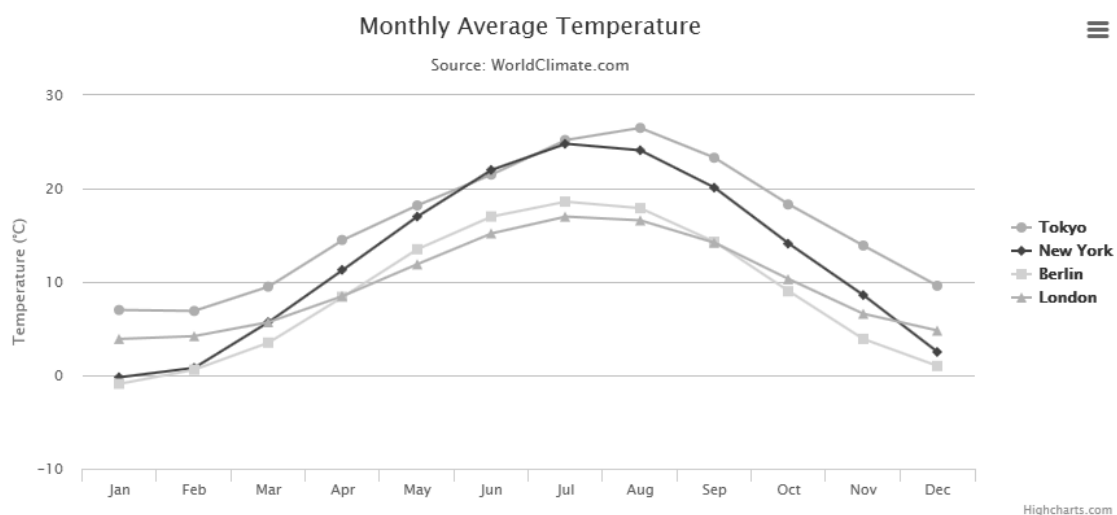


图 1.5 折线图

（2）Area charts（面积图），如图 1.6 所示。

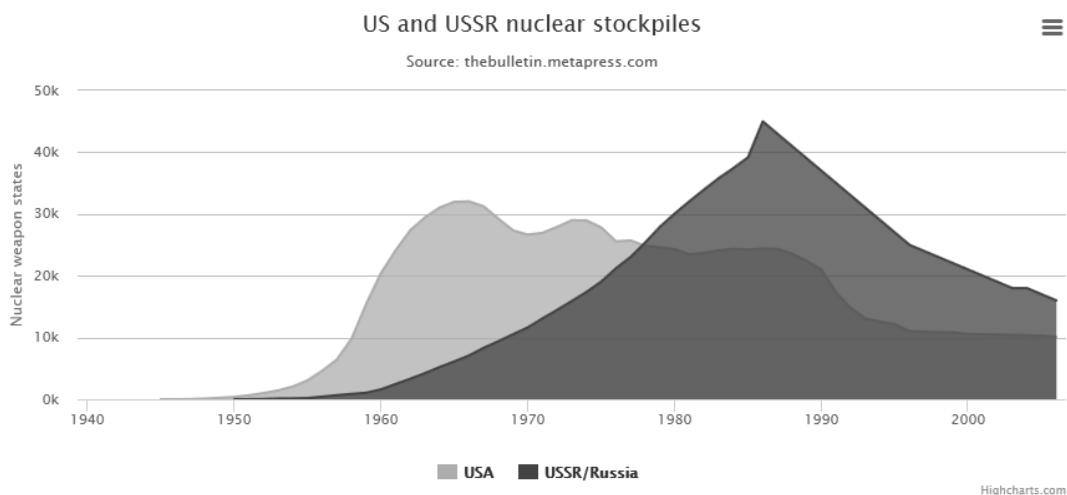


图 1.6 面积图

(3) Column and bar charts（柱形图和条形图），如图 1.7 和 1.8 所示。

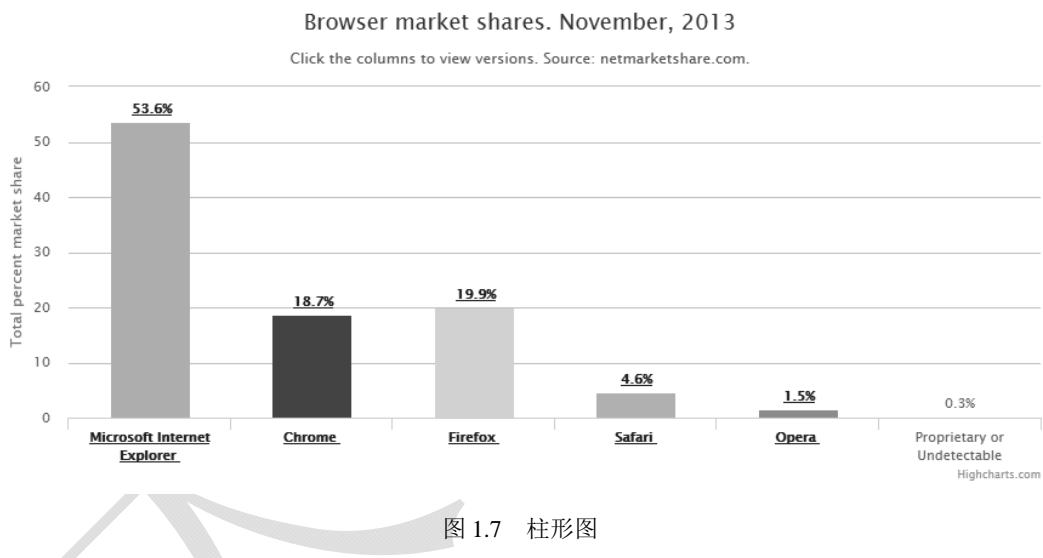


图 1.7 柱形图

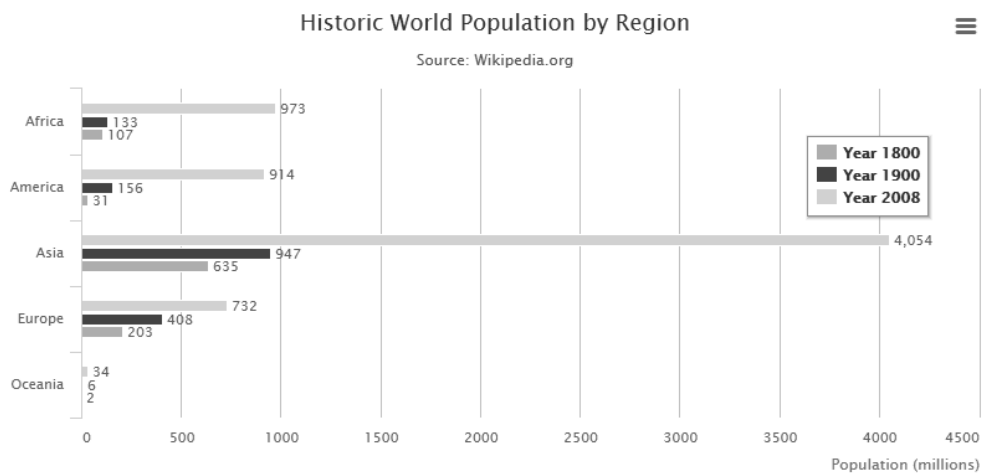


图 1.8 条形图

（4）Pie charts（饼图），如图 1.9 所示。

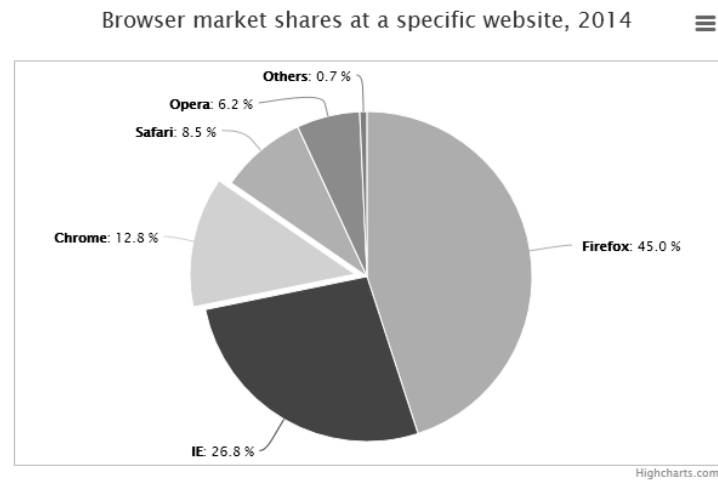


图 1.9 饼图

（5）Scatter and bubble charts（散点图和气泡图），如图 1.10 和 1.11 所示。

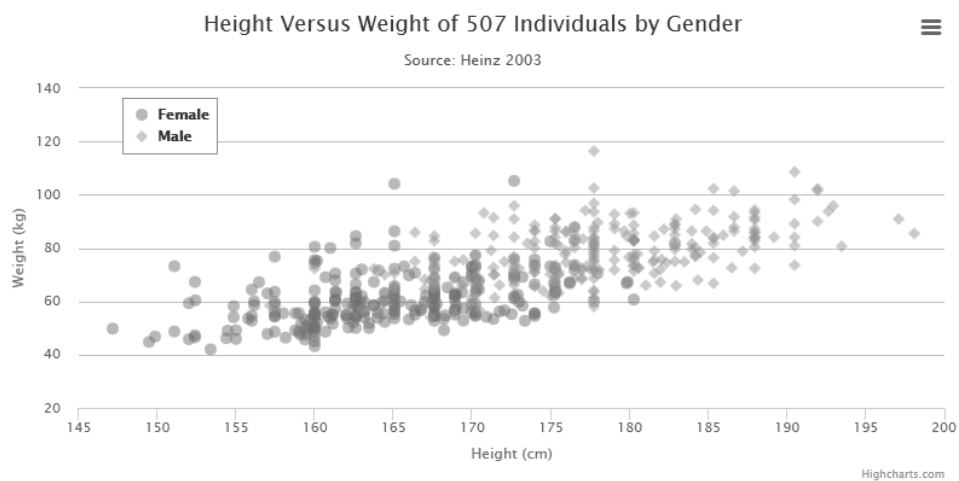


图 1.10 散点图

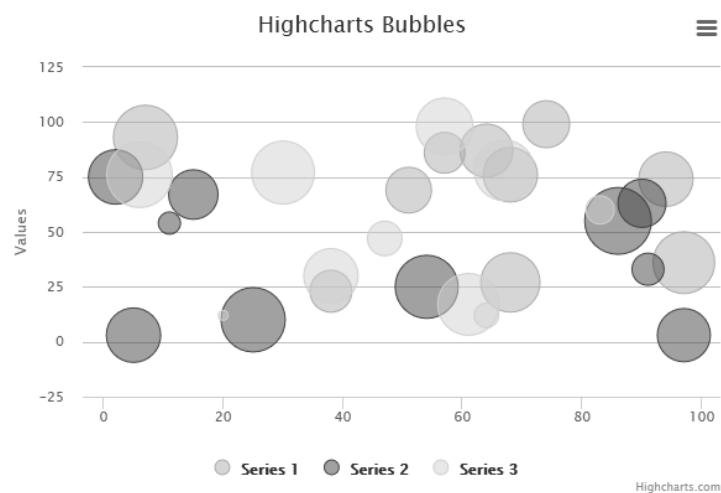


图 1.11 气泡图

(6) Dynamic charts（动态图）是特殊一类图表。他显示的数据是动态变化，如图 1.12 所示。

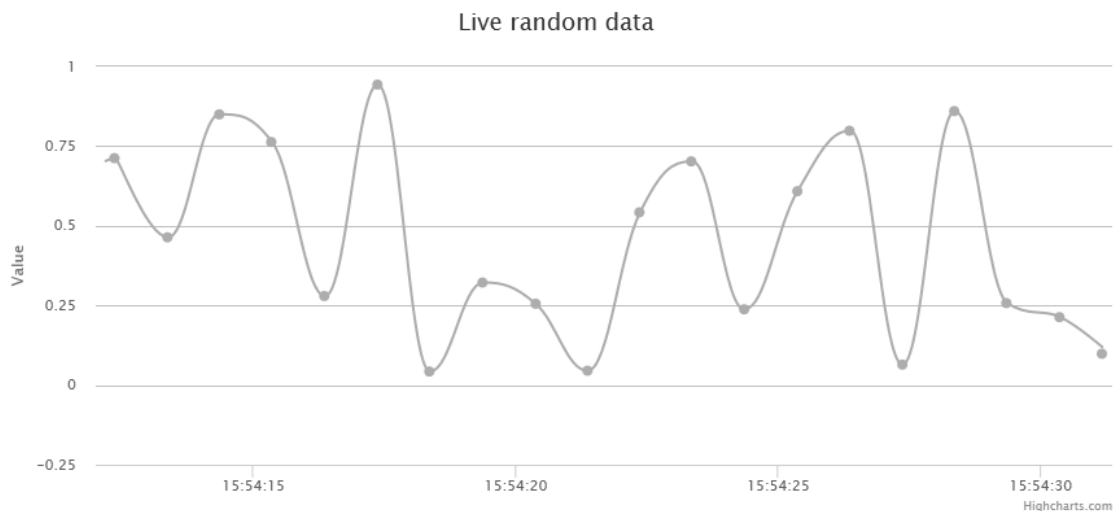


图 1.12 动态图

(7) Combinations（组合图）是将前面类型的图标进行组合显示在一个区域内，如图 1.13 所示。

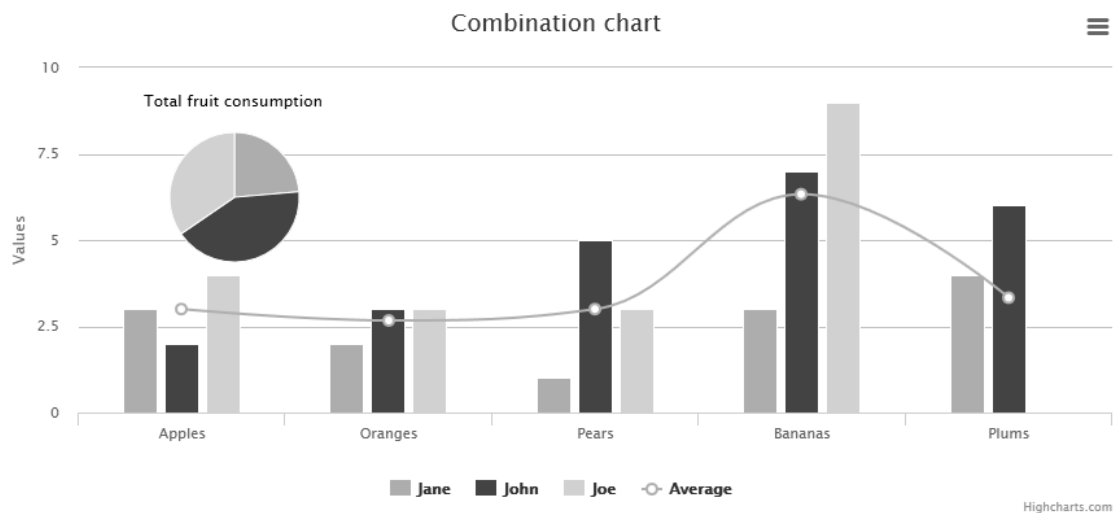


图 1.13 组合图

(8) 3D charts（3D 图表）是在基础图标的基础上，以 3D 效果进行显示，如图 1.14 所示。

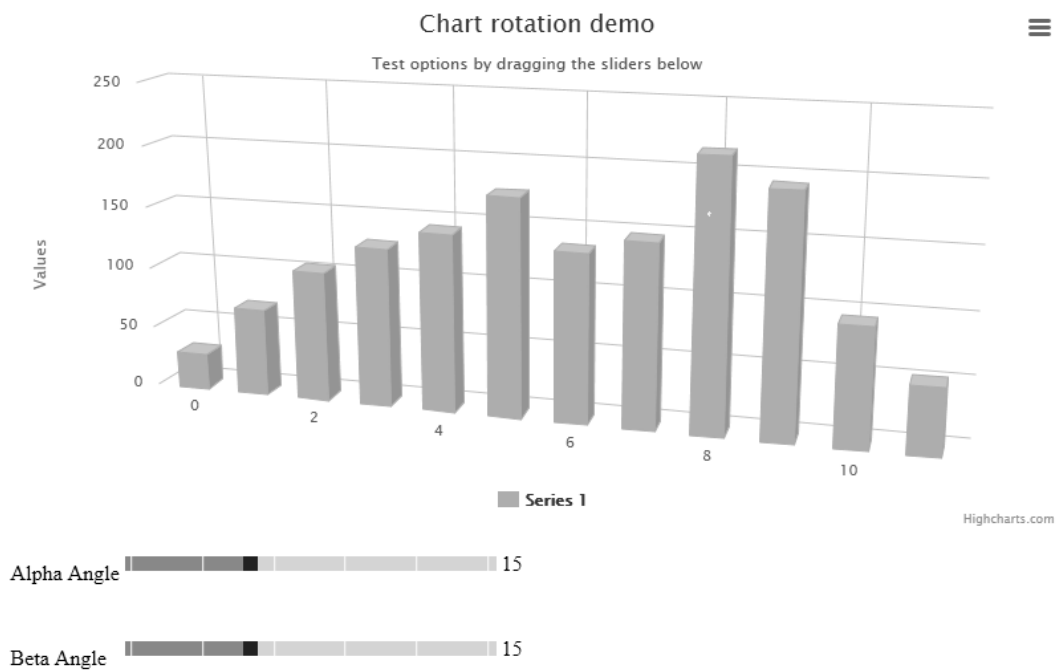


图 1.14 3D 图

(9) Gauges（仪表盘）也是一类特殊的图标。它以类似速度计的方式显示数据，如图 1.15 所示。

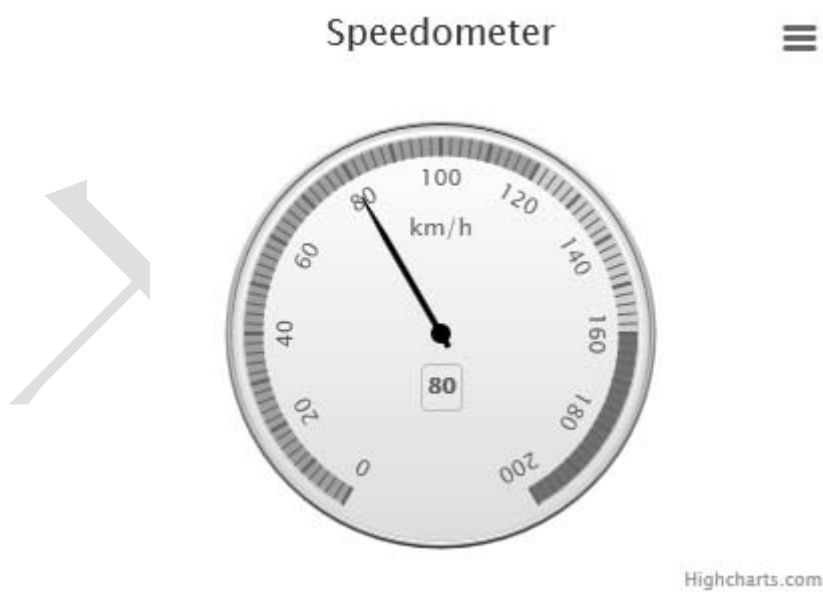


图 1.15 仪表盘

(10) Heat maps（热图）如图 1.16 所示。



图 1.16 热图

(11) More chat types（其他图）包含一些不常用的图标类型，如图 1.17 所示的雷达图。

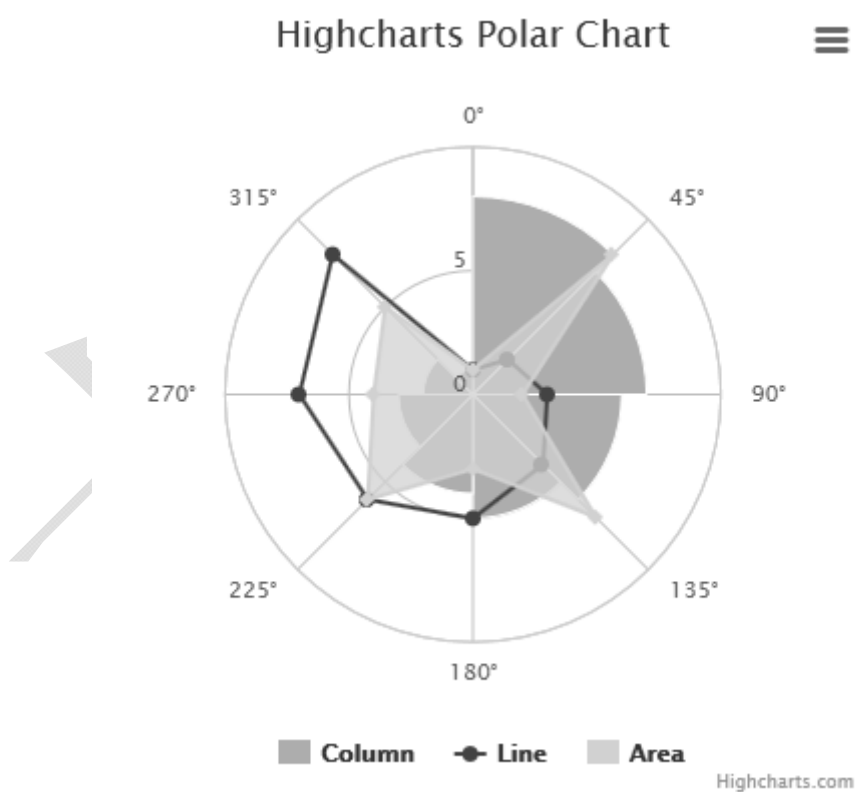


图 1.17 雷达图



## 1.2 第一个实例

下面我们来实现本书的第一个 Highcharts 实例。

【实例 1-1】下面来制作北京连续一周最高温度折线图。操作过程如下：

（1）新建一个网页文件，命名为 Hello.html。同时，将 title 设置为 Hello Highcharts。代码如下：

```
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta charset="utf-8" />
  <title>Hello Highcharts</title>
</head>
<body>
</body>
</html>
```

（2）在网页中添加一个 div，设置 id 为 container。

```
<div id="container"></div>
```

（3）设置该 div 的样式，代码如下：

```
<style type="text/css">
  #container {
    width:500px;
    height:300px;
    border:1px solid #000;
    padding :20px;
    margin:10px;
  }
</style>
```

（4）引入 jQuery 脚本和 Highcharts 脚本，代码如下：

```
<script src="jquery.js"></script>
<script src="highcharts.js"></script>
```

注意：这里需要将这两个脚本文件放到 Hello.html 所在的目录。

（5）添加图表绘制代码。代码如下：

```
<script type="text/javascript">
  $(document).ready(function () {
    var options = {
      chart: {
        type: 'line'
      },
      title: {
        text: '北京一周最高温度'
      },
      subtitle: {
        text: '2015.02.08--2015.02.14'
      },
      series: [{
        name: '最高温度',
        data: [7, 11, 8, 7, 9, 9, 9]
      }]
    };
  });
</script>
```

```
$('#container').highcharts(options);  
});  
</script>
```

保存以上文件，运行结果如图 1.18 所示。

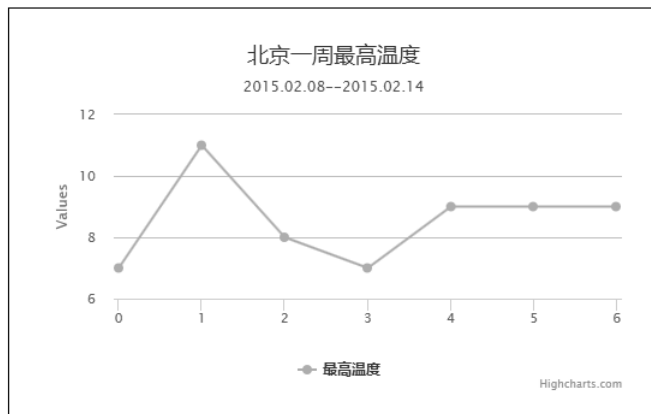


图 1.18 第一个实例

## 1.3 图表构成

为了方便大家更好了解 Highcharts 图表，这里从界面和代码两个角度讲解 Highcharts 图表的实现方式。

### 1.3.1 界面构成

在第一个实例中，我们实现了一个图表。作为一个图表，通常是由图表区、标题、绘图区、坐标轴、图例/数据列等几个部分构成，如图 1.19 所示。在 Highcharts 中，每个部分都由一个或者多个组件构成。其中，黑色方框括住的部分就是图表区。而由坐标轴围起来的部分就是绘图区。后面将依次讲解这几个部分的实现方式。这里，大家只要知道每个部分的名称即可。

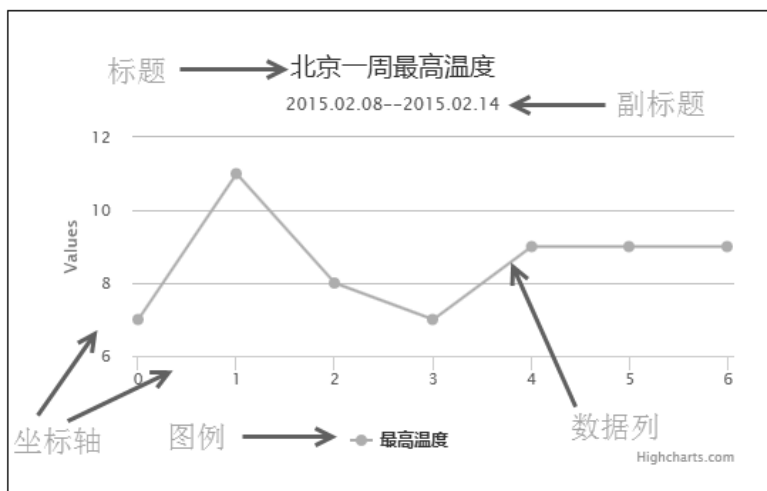


图 1.19 图表构成

### 1.3.2 代码构成

虽然我们在第一个实例中并没有编写多少代码，但图 1.19 却展现丰富的内容。这充分的体现了 Highcharts 使用的便捷性。实现 Highcharts 图表实际分为两个步骤，依次讲解。

#### 1. 图表配置项对象

Highcharts 核心主体就是图表配置项对象。该对象包含了图表的各类数据和配置信息。每个部分往往都是由更小的配置项对象组成。用户只要按照规范的格式，填写对应的数据和配置值，就可以。整个过程就像搭建积木一样。下面对第一个实例的配置项对象做简要介绍。

```
var options = {  
  chart: {                                //图表基本信息  
    type: 'line'                          //指定图表类型  
  },  
  title: {                                //设置图表标题  
    text: '北京一周最高温度'             //指定标题内容  
  },  
  subtitle: {                             //设置图表副标题  
    text: '2015.02.08--2015.02.14'        //设置副标题内容  
  },  
  series: [{                              //定义数据列  
    name: '最高温度',                     //指定数据列的标题  
    data: [7, 11, 8, 7, 9, 9, 9]          //指定数据节点  
  ]  
};
```

这块内容也可以使用传统 Javascript 逐项赋值的方式实现。从维护角度考虑，推荐使用以上方式实现。

#### 2. 在容器中绘制图表

在指定好配置项对象后，用户就可以使用 Highcharts 插件提供的 `highcharts` 方法在指定的容器中绘制图表。需要的代码只需要以下一行。

```
$('#container').highcharts(options);
```

只需要以上两个步骤，就可以在现有的网页中绘制出图 1.19 所示的图表了。用户对图表进行扩展，只需要修改图表配置对象即可。

## 1.4 商业授权和定制开发

在实际 Highcharts 开发过程中，开发者往往会面对授权问题和复杂需求问题。这里针对这两个方面最简要讲解，以帮助开发者更好的使用 Highcharts。

### 1.4.1 商业授权

Highcharts 是一个非常好的图表插件。在使用的时候，它针对个人和非商业应用是全部免费的。对于商业开发，开发者需要购买相应的商业授权。在国内，开发者可以通过 Highcharts 官方授权的 Highcharts 中文网（[hcharts.cn](http://hcharts.cn)）购买商业授权。

该网站是国内最权威的 Highcharts 技术网站。它提供 Highcharts 各项服务，如商业授权、定制、咨询等服务。在浏览器中输入网址 <http://www.hcharts.cn/service/license.php>，就可以进入该网站的商业授权网页，如图 1.20 所示。



图 1.20 商业授权页面

按照网页提示，就可以申请购买商业授权。

## 1.4.2 定制开发

在 Highcharts 中，各类图表的实现采用模版化机制。用户只需要极少的设置，就可以配置精美的图表。但实际开发中，开发者经常面临各种更为复杂的客户需求。这个时候，使用 Highcharts 提供各种配置项往往很难实现。遇到此类问题，用户可以通过购买定制服务，来解决使用中遇到的难题。

Highcharts 中文网提供一流的技术咨询和定制服务，用户只需要进入官网的图表定制服务页面（如图 1.21），就可以申请响应的服务。

**Highcharts中文网** 首页 中文论坛 在线演示 中文教程 API文档 产品中心 支持服务 关于我们

开放CDN服务 商业授权 图表定制服务<sup>beta</sup> 更多服务

### 您的图表，我来帮忙

专业的团队为你解决 Highcharts 图表各种疑难问题及各种图表需求

#### 服务范围及收费标准

服务名称	描述	收费标准（元）*
技术支持	一对一帮您解决Highcharts、Highstock图表的疑难问题	100-500 / 次
Highcharts技术支持套餐*	为期一个月的技术支持	2500 / 月
	为期一年的技术支持服务	25000 / 年
图表定制	帮助您实现需要的图表效果（动态语言仅限 php、java、asp.net）	500-3000 / 张图表
图表解决方案	提供综合图表解决方案	邮件联系

注意事项：  
收费标准只是一个大概的范围，具体价格需要对你的需求进行评估后确定；  
如果你的项目是商业用途的，请先购买Highcharts商用License，详见License价格总览。

分享

图 1.21 定制服务

## 第 2 章 图表区

图表区是图表的基本区域。所有的数据和图形都是绘制在图表区中。从图形绘制范围来分，图表区域分为外层图表区和绘图区。本章将详细讲解图表区的设置和创建。

### 2.1 图表区的构成

为了方便管理，Highcharts 将所有的图表元素都绘制在一个 Box 区域内，如图 2.1 所示。这个区域被称为图表区。由于图表中最重要的数据就是图表图形，所以图表图形所在的区域又构成一个独立的区域，称为绘图区，如图 2.1 所示。

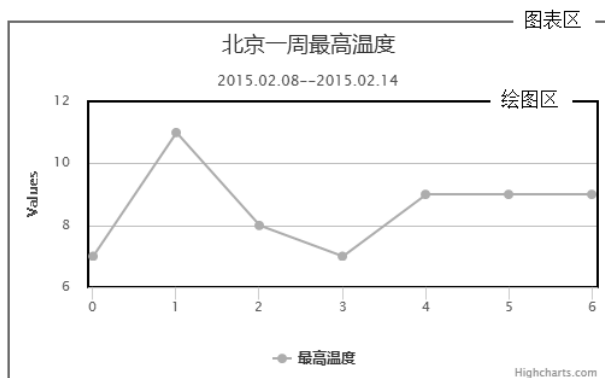


图 2.1 图表区构成

从图 2.1 中可以看到，整个图表区由坐标轴构成的矩形分为两个部分：一部分是坐标轴围成的绘图区，一部分是图表区和绘图区之间的部分，称为外层图表区。下面依次讲解这两个部分。

### 2.2 外层图表区

外层图表区是图表区和绘图区之间的部分。标题、图例这类图表元素往往都分布在这个区域中。本节将详细讲解外层图表区的设置。

#### 2.2.1 外层图表区的基本设置

外层图表区基本设置主要包括图表类型、图表区所在的 HTML 元素、外层图表区的宽高等属性。下面依次讲解这些属性。

##### 1. 图表类型 type

在 Highchart 中，每种图表都是由 chart 组件实现的。所以，要得到不同类型的图表，必须指定图表类型。图表类型由 type 配置项指定。语法形式如下：

type:string

其中，string 是 Highcharts 指定的图表类型字符串。其值可以为 area、arearange、areaspline、areasplinerange、bar、boxplot、bubble、column、columnrange、errorbar、funnel、gauge、heatmap、line、pie、pyramid、scatter、series、solidgauge、spline、waterfall。如果不指定，默认就是 line 类型。

在实例 1-1 中指定了图表类型为 line（折线）类型。

type: 'line'

## 2.图表区所在 HTML 元素 renderTo

在脚本中，也可以通过配置项 renderTo 直接指定图表绘制在网页中哪个容器中。语法形式如下：

renderTo: String

其中，String 是网页元素的 id。

在实例 1-1 中，直接使用了 jQuery 的选择器，所以没有指定该选项。

## 3.指定外层图表区大小

在 Highcharts 中，可以通过配置项 width、height 指定外层图表区的宽和高。语法形式如下：

width: Number1

height: Number2

其中，Number1、Number2 分别是图表区的宽高值。单位为像素。

【实例 2-1: chartheightandwidth】修改实例 1-1 的代码，将图表区宽度设定 300px，高度设定为 200px。代码如下：

```
chart: {
  type: 'line',
  height: 200,           //指定高度
  width: 300,           //指定宽度
  borderWidth: 1
},
```

执行后，效果如图 2.2 所示。

注意：为了显示图表区大小，这里设置了图表区的边框宽度为 1px。

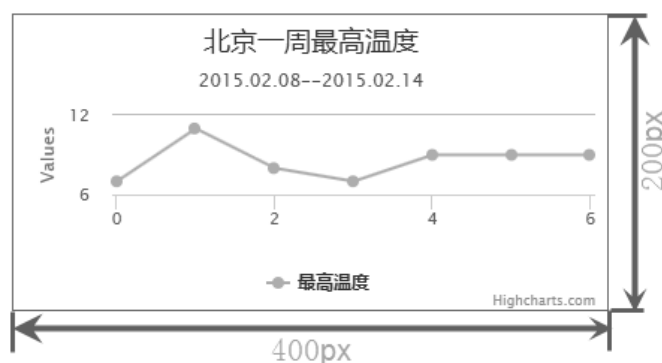


图 2.2 设置图表区的长和宽

如果不设定图表区的长宽值，Highcharts 会根据图表区中的元素自动计算。如果图表区包含的元素高度为 0，则将 Height 设置为 400px。

## 4.指定外层图表区背景颜色

为了美化图表显示效果，可以通过配置项 backgroundColor 来指定图表区的背景颜色。语法如下：

backgroundColor: Color

其中，Color 是使用单引号括起来的颜色值。默认值是#FFFFFF。

【实例 2-2: chartbackgroundColor】修改实例 2-1 的代码，设定图表区的背景颜色为#FFFF00。代码如下：

```
chart: {  
    type: 'line',  
    height: 200,  
    width: 400,  
    borderWidth: 1,  
    backgroundColor: '#FFFF00' //设定背景颜色  
},
```

执行后，效果如图 2.3 所示。



图 2.3 设置图表区背景颜色

## 2.2.2 外层图表区的边框

在 Highcharts 中，允许为外层图表区设置边框。相关的配置项包括宽度 `borderWidth`、颜色 `borderColor`、边框圆角半径 `borderRadius`。其语法如下：

```
borderWidth: Number1  
borderColor: Color  
borderRadius: Number2
```

其中，Number1 表示边框的宽度，默认值为 0，表示没有边框；Color 是表示颜色的字符串；Number2 表示边框的圆角半径。Number1 和 Number2 的单位均为像素 px。

【实例 2-3: chartborder】下面修改实例 1-1 的代码，为图表区添加边框。需要修改的代码如下：

```
chart: {  
    type: 'line',  
    borderWidth: 3, //设置边框的宽度  
    borderColor: '#000000', //设置边框的颜色  
    borderRadius: 10 //设置边框的圆角  
},
```

执行后，效果如图 2.4 所示。



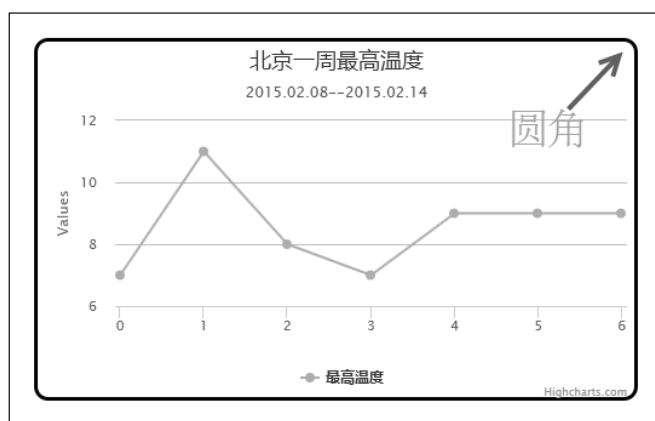


图 2.4 设置图表区的边框

### 2.2.3 外层图表区的范围

外层图表区位于图表区内部，绘图区外部，如图 2.5 的浅绿色部分所示。通常，外层图表区的范围由 Highcharts 自动计算。

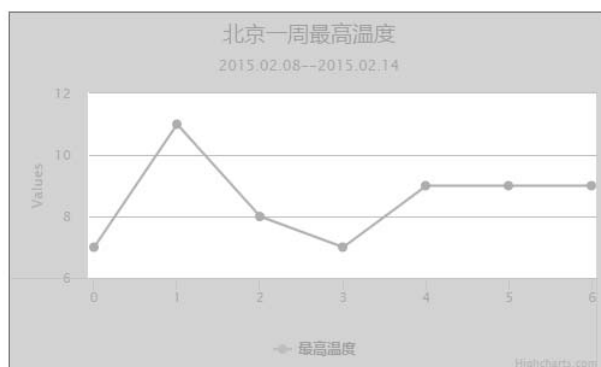


图 2.5 图表区和绘图区的间距

用户也可以通过 chart 组件的配置项 `marginBottom`、`marginLeft`、`marginRight`、`marginTop` 属性来分别设置两个区域的下边距、左边距、右边距和上边距。语法如下：

```
marginBottom:Number1
marginLeft:Number2
marginRight:Number3
marginTop:Number4
```

其中，Number1、Number2、Number3、Number4 分别表示各个边距值。单位为 px。这四个配置项可以同时使用，也可以分开使用。

【实例 2-4: chartmargin】修改实例 1-1 的图表区和绘图区下间距，将其设置为 2。修改代码如下：

```
chart: {
  type: 'line',
  borderWidth: 1,
  marginBottom: 2,           //设置下边距
  marginLeft: 30,          //设置左边距
  marginRight: 30,         //设置右边距
```

```
marginTop:20
```

```
//设置上边距
```

```
},
```

执行后，效果如图 2.6 所示。从运行结果中可以看到，当边距设置不当，会造成部分图表内容显示不完整，甚至丢失。例如，x 坐标轴的刻度没有显示。

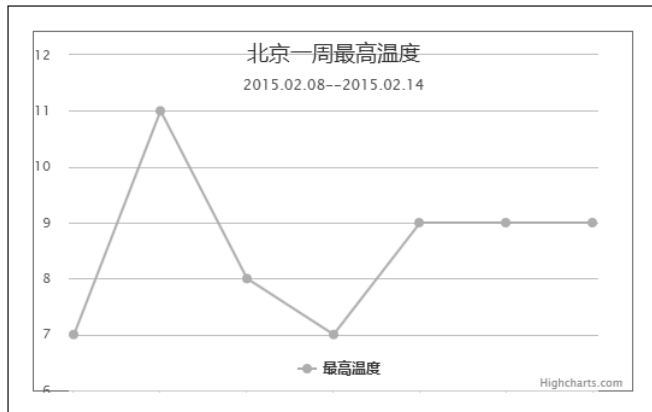


图 2.6 修改图表区和绘图区间距

为了简化操作，Highcharts 提供一个配置项 `margin`，可以一次性对四个边距直接设置。语法如下：

`margin:Array`

其中，`Array` 是一个数组，用来设置图表区和绘图区的上边距、右边距、下边距和左边距。使用该属性后，实例 2-4 的代码就可以简化为：

```
var options = {
  chart: {
    type: 'line',
    borderWidth: 1,
    margin:[2,30,30,20] //依次设置上边距、右边距、下边距、左边距
  },
}
```

## 2.2.4 外层图表区和图表内容的间距

为了美观，图表中的内容和图表区之间有一定的间距。如图 2.7 所示，绿色的部分为两者之间的间距。对于这部分间距，`chart` 组件提供了对应的配置项来设置。例如，配置项 `spacingBottom` 用来设置下端间距；配置项 `spacingLeft` 用来设置左侧间距；配置项 `spacingRight` 用来设置右侧间距；配置项 `spacingTop` 用来设置顶部间距。

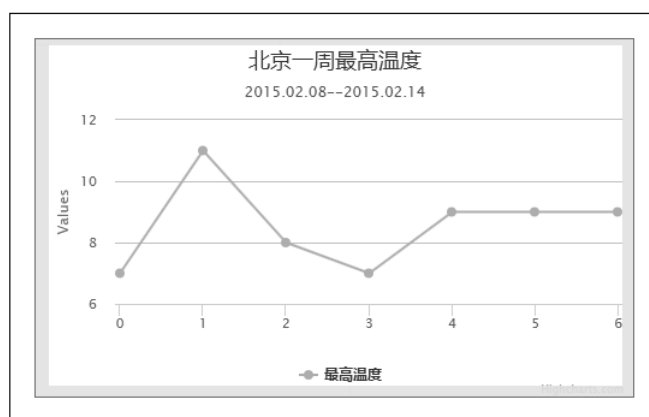


图 2.7 图表区和图表内容的间距

这四个配置项的语法如下：

spacingBottom: Number1

spacingLeft: Number2

spacingRight: Number3

spacingTop: Number4

其中，Number1 参数用来设置底部间距，默认值为 15；Number2 参数用来设置左侧间距，默认值为 10；Number3 参数用来设置右侧间距，默认值为 10；Number4 参数用来设置顶部间距，默认值为 10。

【实例 2-5: spacing】下面取消实例 1-1 中，图表内容和图表区的间距。修改代码如下：

```
chart: {
  type: 'line',
  borderWidth: 1,
  spacingBottom: 0, //设置底部间距
  spacingLeft: 0, //设置左侧间距
  spacingRight: 0, //设置右侧间距
  spacingTop: 0, //设置顶部间距
},
```

执行后，效果如图 2.8 所示。从图中可以发现，取消间距后，图例和标题紧贴外层图表区边框。

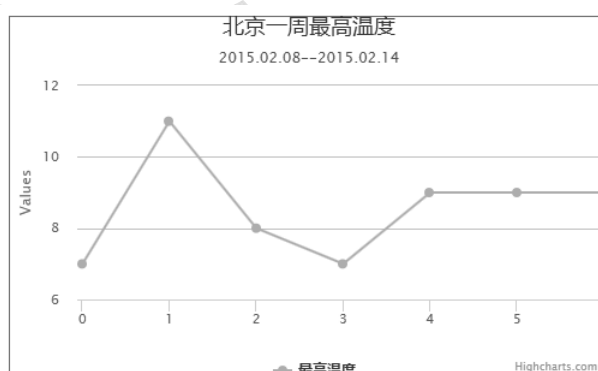


图 2.8 取消图表区和图表内容间距

为了方便设置，chart 组件提供一个配置项 spacing，用来同时设置四个间距。语法形式如下：

spacing: Array

其中，Array 参数是由四个数值构成的数组。其中每个数值依次表示顶部、右侧、底部、左侧间距值。使用属性 spacing 可以简化实例 2-5 的代码。简化后的代码如下：

```
chart: {
  type: 'line',
  borderWidth: 1,
  spacing:[0,0,0,0]           //依次设置四个间距值
},
```

间距值也可以设置为负数，代码如下：

```
spacing:[-10,-10,-10,-10]
```

这个时候，会造成部分图表内容显示不完整，如图 2.9 所示。标题和图例都没有完全显示。

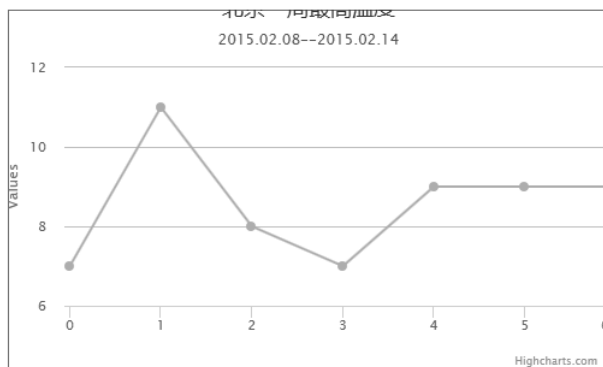


图 2.9 间距值为负值

注意：版权信息部分不受间距影响。

## 2.2.5 外层图表区阴影

为了增加立体感，chart 组件提供阴影配置项。其语法如下：

```
shadow: Boolean|Object
```

其中，shadow 的参数可以是布尔值 true/false，也可以是参数对象。

（1）当为布尔值的时候，可以使用 true 和 false 表示是否有属性。

【实例 2-6: shadow】下面为实例 1-1 添加阴影效果。修改代码如下：

```
chart: {
  type: 'line',
  shadow:true           //启用阴影效果
},
```

执行后，效果如图 2.10 所示。在图表区的右侧和底端均出现灰色的阴影效果。

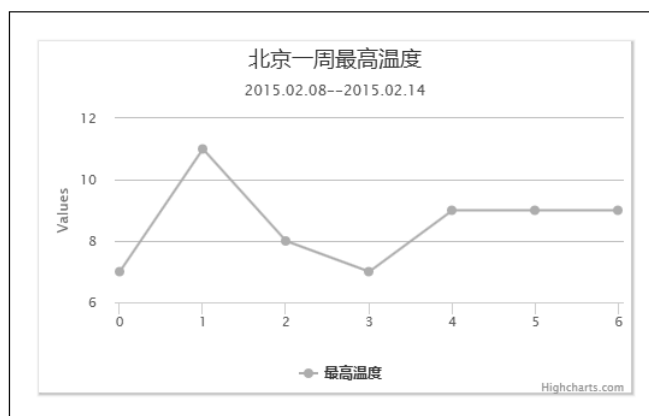


图 2.10 图表阴影

(2) 用户也可以采用对象的方式添加阴影效果。语法形式如下：

```
{
  color:Color,
  setX:Number1,
  setY:Number2,
  opacity:Number3,
  width:Number4
}
```

其中，配置项 `color` 表示阴影的颜色；配置项 `setX` 表示阴影在 x 轴上的偏移距离；配置项 `setY` 表示阴影在 y 轴上的偏移距离；配置项 `opacity` 表示阴影的透明度；配置项 `width` 设置阴影的宽度。

【实例 2-7: shadowObject】使用对象的方式重新实现实例 2-6 的阴影效果。修改代码如下：

```
chart: {
  type: 'line',
  shadow: {
    color: 'grey',           //设置阴影颜色
    offsetX: 1,             //设置 x 轴方向的偏移距离
    offsetY: 1,             //设置 y 轴方向的偏移距离
    opacity: 0.5,           //设置透明度
    width: 2                 //设置阴影宽度
  }
},
```

## 2.2.6 图表区样式

为了便于用户制作出更美观的图表，`chart` 组件提供两个配置项，用来设置图表所在容器 `div` 的 CSS 样式。下面依次讲解这两个属性。

### 1. 内部样式 `style`

使用配置项 `style`，可以直接在 Javascript 脚本中指定所使用的样式。其语法形式如下：

```
style: CSSObject
```

其中，属性 `CSSObject` 是 CSS 对象，由 CSS 属性和值构成。其默认值如下：

```
{"fontFamily": "\"Lucida Grande\", \"Lucida Sans Unicode\", Verdana, Arial, Helvetica, sans-serif",
  "fontSize": "12px"}
```

## 2. 外部样式属性 className

为了方便用户管理 CSS 脚本，chart 组件提供了外部样式配置项 className。其语法如下：

className: String

其中，参数 String 表示 CSS 类选择器名。具体使用方式，请参考 CSS 相关书籍。

## 2.3 绘图区

绘图区是 Highcharts 图表的最重要区域。所有的图表图形都绘制在该区域中。整个绘图区由背景和边框两大部分组成。本节详细讲解绘图区的设置。

### 2.3.1 绘图区的边框

在 Highcharts 的默认设置中，只有坐标轴 x 和 y 可以显示绘图区的左边界和下边界。为了方便用户识别区域，也可以指定绘图区的边框。这样，可以更明确体现图表图形的范围。

#### 1. 绘图区边框宽度 plotBorderWidth

配置项 plotBorderWidth 用来设置绘图区的边框宽度。默认宽度为 0，所以在前面的实例中都看不到边框线。其语法如下：

plotBorderWidth: Number

其中，Number 参数用来指定边框线的宽度，单位为 px。

【实例 2-8：plotBorderWidth】下面我们为实例 1-1 的图表绘图区添加宽度为 1px 的边框线。修改的代码如下：

```
chart: {  
  type: 'line',  
  borderWidth: 1,  
  plotBorderWidth: 1           //设置边框线宽度为 1  
},
```

执行后，效果如图 2.11 所示。在图中，只标出了绘图区上边缘和右边缘的线。实际绘图区的左边缘和下边缘也有类似的边框线，只是被坐标轴覆盖，所以无法显示。当用户设置的 borderWidth 值较大时，就可以显示出左侧边框和底端边框。

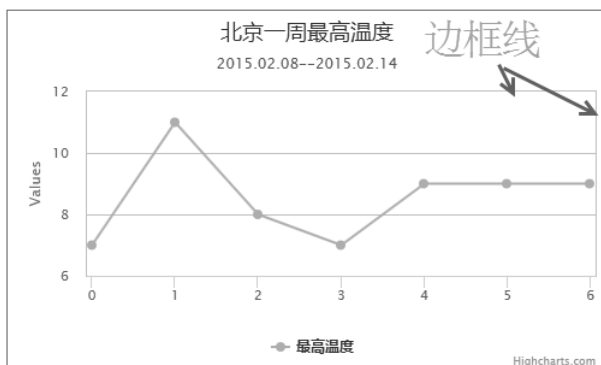


图 2.11 绘图区的边框线

#### 2. 边框线的颜色 plotBorderColor

从图 2.11 中可以看到，边框线已经显示出来，并且具有一定的颜色。绘图区边框线默认的颜色为 #C0C0C0。开发者也可以使用配置项 `plotBorderColor` 设定。其语法形式如下：

`plotBorderColor:Color`

其中，Color 参数用来指定边框线的颜色。

【实例 2-9: `plotBorderColor`】修改实例 2-8 的代码，把绘图区边框的颜色设置为黑色。修改代码如下：

```
chart: {  
    type: 'line',  
    borderWidth: 1,  
    plotBorderWidth: 2,  
    plotBorderColor: '#000000'           //设置边框线颜色为#000000  
},
```

执行后，效果如图 2.12 所示。这里为了让边框线颜色更明显，将边框线的宽度设置为 2px。这时，就可以明显看出绘图区四个边缘的边框线。

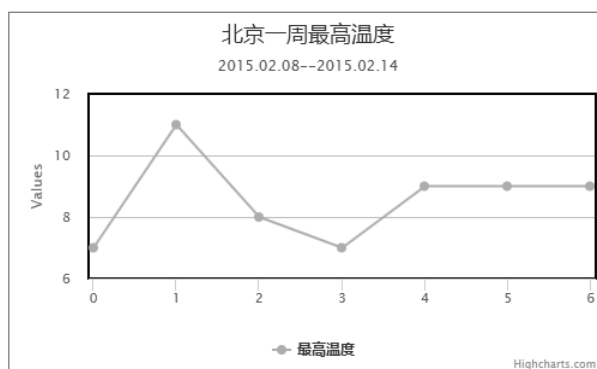


图 2.12 设置为黑色的边框线

## 2.3.2 绘图区的背景

绘图区是图表图形展现的最重要的舞台。通过设置合理的背景，可以帮助用户更好的识别数据。Highcharts 提供背景颜色和背景图案两种方式来美化绘图区。

### 1. 绘图区背景颜色 `plotBackgroundColor`

为了明确绘图区的范围，不仅可以绘制边框线，还可以为绘图区设置背景颜色。这时需要使用配置项 `plotBackgroundColor`。其语法形式如下：

`plotBackgroundColor:Color`

其中，Color 用来指定绘图区的颜色值。

【实例 2-10: `plotBackgroundColor`】修改实例 1-1，为绘图区添加灰色背景。修改代码如下：

```
chart: {  
    type: 'line',  
    borderWidth: 1,  
    plotBackgroundColor: '#C0C0C0'       //设置绘图区背景色  
},
```

执行后，效果如图 2.13 所示。从图中可以看到，绘图区已经被填充灰色。在这里，还可以指定渐变色。我们将在后面的章节中进行讲解。

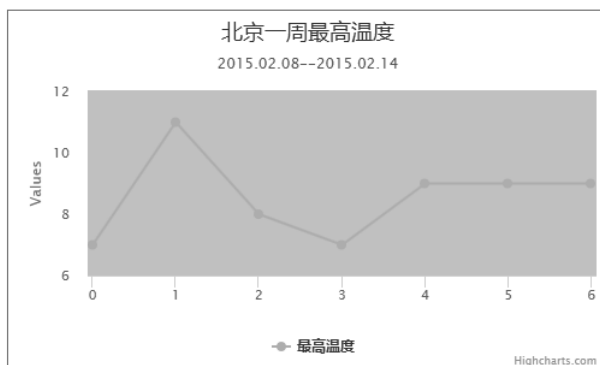


图 2.13 绘图区背景色

## 2. 绘图区背景图片 plotBackgroundImage

在 chart 组件中，用户不仅可以通过设置背景色美化绘图区，还可以为绘图区设置背景图片。这时需要通过属性 plotBackgroundImage 设置。其语法如下：

plotBackgroundImage:String

其中，String 参数指定图片文件的 URL 地址。

【实例 2-11: plotBackgroundImage】修改实例 2-10，将绘图区的背景改成渐变色图片 back.png。修改代码如下：

```
chart: {
  type: 'line',
  borderWidth: 1,
  plotBackgroundImage: 'back.png' //设置背景图片
},
```

执行后，效果如图 2.14 所示。从图中可以发现，背景图片被拉伸直至和绘图区重合。所以，使用背景图片时需要注意背景图片变形问题，避免影响美观。

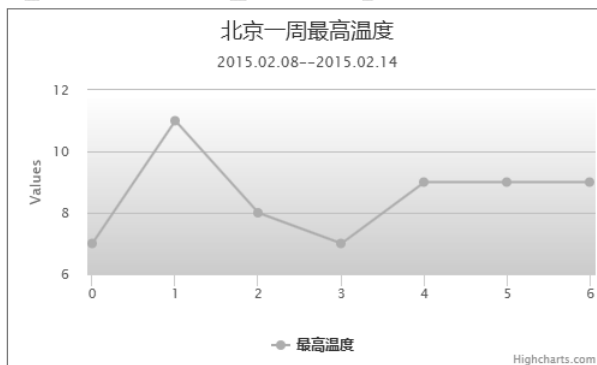


图 2.14 设置绘图区的背景图片

## 2.4 图表缩放

Highcharts 为图表提供缩放功能。为图表添加缩放功能后，当图表中的数据较多时，可以放大局部数据，便于用户查看数据细节。本节将讲解图表缩放功能的设置。



## 2.4.1 图表缩放类型

Highcharts 提供了三种缩放方式。这三种方式都是通过配置项 `zoomType` 来设置的。语法如下：

`zoomType`: String

其中，参数 String 指定缩放的方式。其值有以下几种：

- ☐ None: 表示没有缩放。这是默认值；
- ☐ x: 表示沿 x 轴方向缩放；
- ☐ y: 表示沿 y 轴方向缩放；
- ☐ xy: 表示沿 x、y 轴同时缩放。

【实例 2-12: `zoomType`】下面为实例 1-1 的图表添加 x 轴方向的缩放。修改代码如下：

```
chart: {  
    type: 'line',  
    borderWidth: 1,  
    zoomType: 'x'  
},  
//设置 x 轴缩放
```

执行后，在绘图区中，使用鼠标沿着 x 轴方向拖动，会产生一个选择区域，如图 2.15 所示。松开鼠标后，图表会沿 x 轴方向放大，并多显示一个 **Reset zoom** 按钮，运行效果如图 2.16 所示。当单击 **Reset zoom** 按钮，将恢复原始的大小。

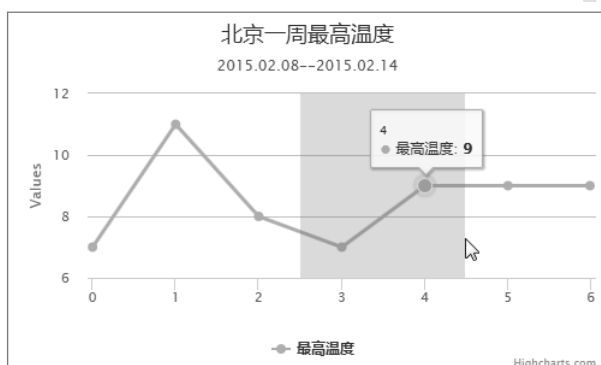


图 2.15 沿着 x 轴方向拖动鼠标

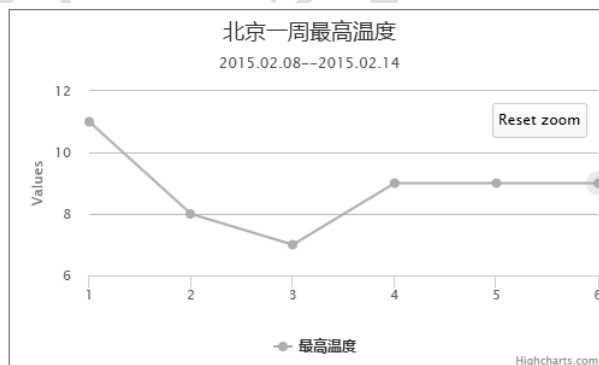


图 2.16 x 轴缩放后的图表

当参数 `zoomType` 设置为 `y` 或者 `xy` 后，拖动的方式有所不同，产生的选择区域也不同，分别如图 2.17 和 2.18 所示。

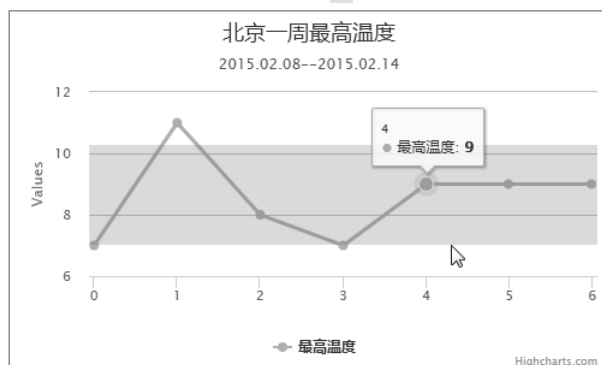


图 2.17 沿着 y 轴方向拖动鼠标

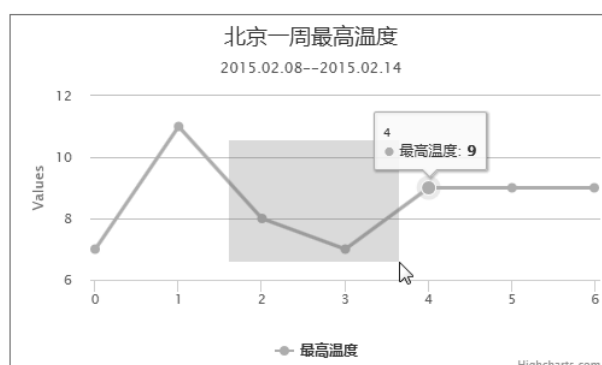


图 2.18 同时沿着 x、y 轴方向拖动鼠标

## 2.4.2 填充选择区

在缩放的时候，为了便于用户识别选择的范围，选择区域会被自动填充。通过配置项 `selectionMarkerFill`，用户可以修改填充的颜色和透明度。语法如下：

`selectionMarkerFill: Color`

其中，参数 `Color` 指定填充颜色的 RGB 值和透明度。默认值为 `'rgba(69,114,167,0.25)'`。

【实例 2-13: `selectionMarkerFill`】修改实例 2-12 的默认填充颜色的透明度。修改后的代码如下：

```
chart: {  
  type: 'line',  
  borderWidth: 1,  
  selectionMarkerFill: 'rgba(69,114,167,0.90)', //设定填充颜色  
  zoomType: 'x'  
},
```

执行后，效果如图 2.19 所示。从图中可以看出，填充颜色并没有发生变化，但透明度降低了。

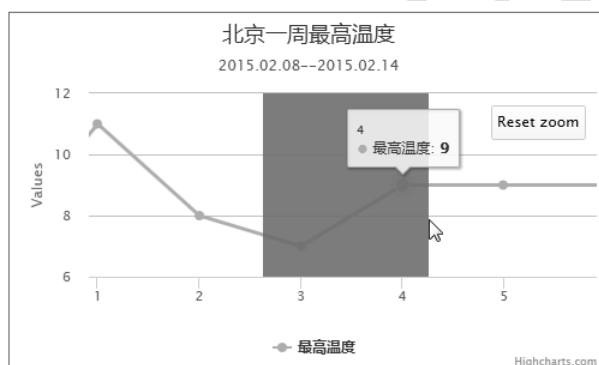


图 2.19 设置填充颜色

## 2.4.3 重置按钮

当图表缩放后，图表会增加一个显示元素——`Reset zoom` 按钮，如图 2.20 所示。当用户的鼠标移动到该按钮上，还会弹出一个提示框，提示用户单击该按钮后，回到 1:1 的比例。该按钮由 `chart` 的子组件 `resetZoomButton` 实现。下面详细重置按钮的各项设置。

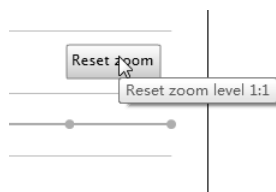


图 2.20 重置按钮

### 1.本地化 `Reset zoom` 按钮

`Reset zoom` 按钮的名称和提示框内容都是英文的。这不太符合我们国内的需要。用户需要将其本地化。`Highcharts` 提供一个本地化组件 `lang` 来实现基础元素的本地化工作。其语法如下：

```
Highcharts.setOptions({  
  lang: {  
    //配置项
```

```
    }
```

```
  });
```

在这里对 Reset zoom 按钮需要使用到两个配置项 `resetZoom` 和 `resetZoomTitle`。其中，`resetZoom` 用来设置按钮的文本内容；`resetZoomTitle` 用来设置提示框的文本内容。其语法如下：

```
resetZoom: String1
```

```
resetZoomTitle: String2
```

其中，String1 表示本地化的按钮文本内容；String2 表示本地化的提示框文本内容。

【实例 2-14: `resetZoomlang`】下面将 Reset zoom 按钮实现本地化。添加代码如下：

```
Highcharts.setOptions({
  lang: {
    resetZoom: '原始大小',           //本地化按钮文本
    resetZoomTitle: '回到初始状态 1:1' //本地化提示框文本
  }
});
```

执行后，效果如图 2.21 所示。

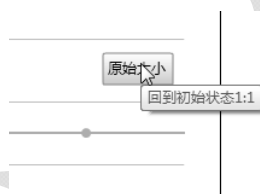


图 2.21 本地化 Reset zoom 按钮

## 2. 设置位置 position

配置项 `position` 用来指定 Reset zoom 按钮显示的位置。其语法如下：

```
position:Object
```

其中，参数 `Object` 是一个位置对象，用来指定该按钮的对齐方式和位置坐标。该 `Object` 的结构如下所示：

```
{
  align: String1,
  verticalAlign:String2,
  x:Number1,
  y:Number2
}
```

- ❑ 配置项 `align` 用来指定水平对齐方式。String1 的值可以为 `left`、`center`、`right`，默认值为 `right`；
- ❑ 配置项 `verticalAlign` 用来指定垂直对齐方式。String2 的值可以为 `top`、`middle`、`bottom`，默认值为 `top`；
- ❑ 配置项 `x` 用来指定水平偏移坐标值；
- ❑ 配置项 `y` 用来指定垂直偏移坐标值。

由于涉及坐标轴的确定和坐标偏移，所以这里详细讲解绘制过程中，按钮位置的计算方式。主要包括以下三步：

(1) 根据配置项 `align` 确定 x 坐标轴的位置，如图 2.22 所示。

注意：根据对齐方式不同，x 轴的起始点不同。



图 2.22 水平对齐方式下的 x 轴位置

(2) 根据配置项 `verticalAlign` 确定 y 坐标轴的位置，如图 2.23 所示。注意，根据对齐方式不同，y 轴的起始点不同。

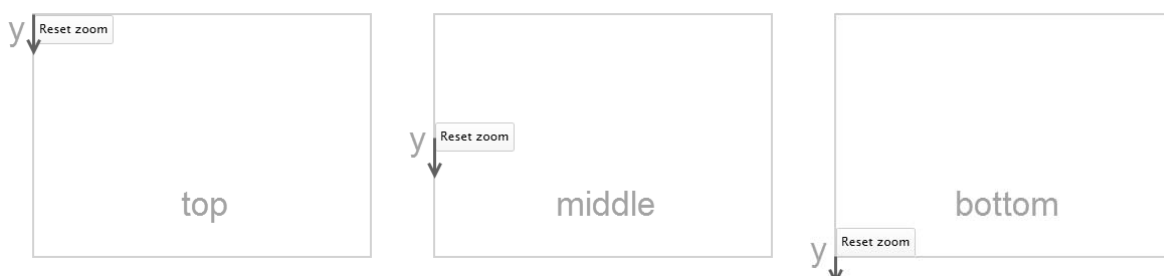


图 2.23 垂直对齐方式下的 y 轴位置

(3) 确定 x、y 坐标轴后，根据配置项 `x`、`y` 确定按钮的偏移位置。如果值为正的，则顺着坐标轴方向偏移；否则，反向偏移。

对于 `{align:center,verticalAlign:middle,x:40,y:20}` 来说，其 `Reset zoom` 按钮位置如图 2.24 所示。

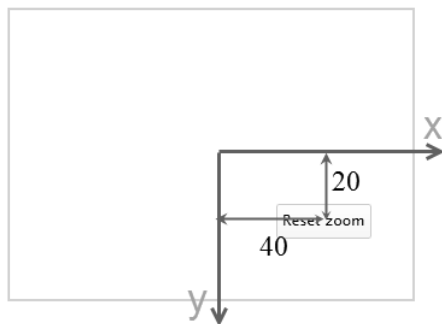


图 2.24 示例位置

【实例 2-15: `resetZoomButton`】下面不采用默认设置，重新设置 `Reset zoom` 按钮的位置。修改代码如下：

```
chart: {
  type: 'line',
  borderWidth: 1,
  zoomType: 'x',
  resetZoomButton: {
    position: {
      align: 'left',
```

```

        verticalAlign: 'bottom',
        x: 20,
        y:-50
    }
}

```

执行后，效果如图 2.25 所示。

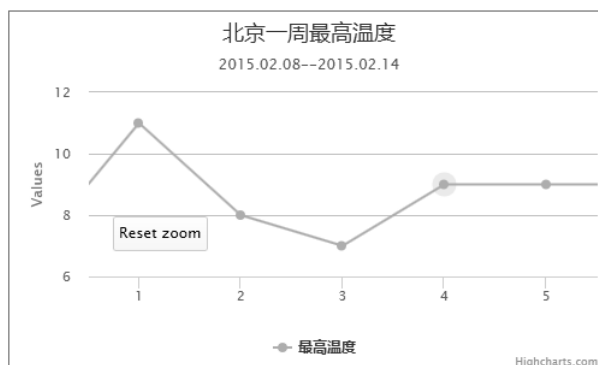


图 2.25 修改 Reset zoom 按钮位置

## 2. 相对位置 relativeTo

设置按钮的对齐方式时，实际还涉及对齐方式所参考的标准。上面讲解的都是基于绘图区的。Highcharts 还允许用户基于图表区进行对齐。这时候需要使用配置项 `relativeTo`。其语法如下：

`relativeTo: String`

其中，参数 `String` 用来指定参考标准。其值可以为 `'plot'`、`'chart'` 中的一个。默认是 `'plot'`。

【实例 2-16: `relativeTo`】修改实例 2-15 的代码，将参考标准从默认的 `plot`，改为 `chart`。修改代码如下：

```

chart: {
    type: 'line',
    borderWidth: 1,
    zoomType: 'x',
    resetZoomButton: {
        relativeTo: 'chart',           //修改参考标准
        position: {
            align: 'left',
            verticalAlign: 'bottom',
            x: 20,
            y: -50
        }
    }
}

```

执行后，效果如图 2.26 所示。从图中可以明显看出，`Reset zoom` 按钮由原有的绘图区的位置移动到绘图区的下边。

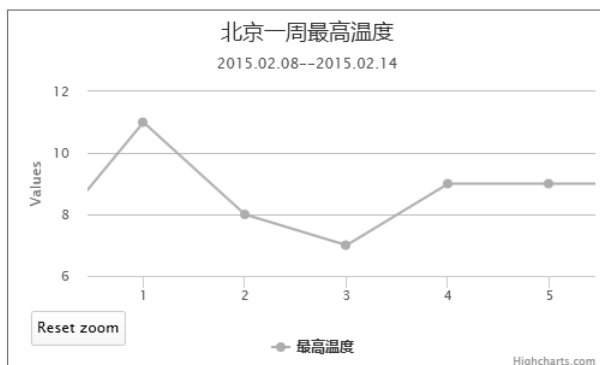


图 2.26 修改参考标准

### 3. 设置按钮样式配置项 theme

用户除了可以设置复原按钮的位置，还可以进一步对按钮的样式进行设定。这时，需要使用属性 theme。其语法如下：

theme:Object

其中，参数 Object 用来设置按钮的各项属性。

## 2.5 图表平移

当图表缩放后，图表往往会超出绘图区的显示范围，影响查看完整的数据。为了解决这个问题，Highcharts 提供了图表平移功能。下面讲解使用平移功能。

### 2.5.1 启用图表平移

平移功能默认是禁用的。启用该功能，需要设置配置项 panning。其语法如下：

panning:Boolean

其中，参数 Boolean 是布尔值，确认是否启用平移功能。当为 true 时，启用该功能；当为 false 时，禁用该功能。默认值是 false。

### 2.5.2 设置图表平移方式

由于缩放和平移都是通过鼠标操作，所以需要设置一个按键来实现缩放和平移操作的切换。这时候，需要使用配置项 panKey。其语法如下：

panKey:String

其中，参数 String 表示按键的名称。常用的按键名称包括 'ctrl'、'alt'、'shift'，分别表示 Ctrl、Alt、Shift 键。

【实例 2-17: panKey】为实例 2-16 添加图表平移功能。修改代码如下：

```
chart: {
  type: 'line',
  borderWidth: 1,
  zoomType: 'x',
  panning: true,           //启用平移功能
```

```
panKey:'ctrl',                                //设置功能切换键
resetZoomButton: {
  relativeTo: 'chart',
  position: {
    align: 'left',
    verticalAlign: 'bottom',
    x: 20,
    y: -50
  }
},
},
```

执行后，效果如图 2.27 所示。当用户按住 Ctrl 键并按下鼠标后，鼠标指针就变成移动图表。这时，用户就可以左右移动图表，查看完整的图表数据。

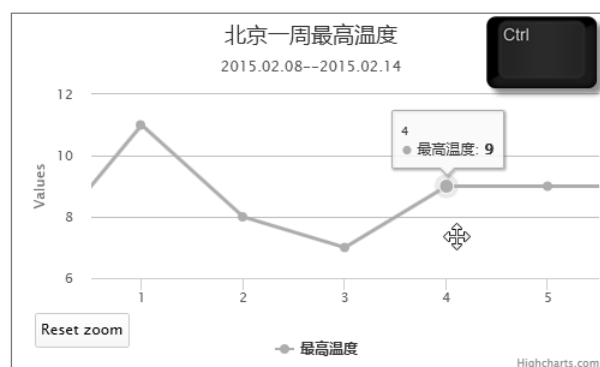


图 2.27 启用平移功能

## 第 3 章 辅助元素

辅助元素图表的非必要元素，如标题、版权信息、标签、载入动态。它们不和图表数据发生关联，只是额外说明一些基本信息。合理的使用这些部分，可以补充数据的不足。本章将详细讲解几种常见的辅助元素。

### 3.1 标题/副标题

为了说明图表展现的数据，Highcharts 为每个图表提供标题 `title` 和副标题 `subtitle` 两个组件。本节将详细讲解这两个组件的使用方式。

#### 3.1.1 标题和副标题的构成

在实例 1-1 中，我们已经接触到标题的构建了。其中构建标题的代码如下所示：

```
title: {  
  text: '北京一周最高温度'  
},
```

其中，`title` 标记为图表构建一个标题；配置项 `text` 表示标题的内容。其语法如下：

```
text:String
```

其中，参数 `String` 指定标题的文本内容。默认值是 'Chart title'。如果不要想要标题，必须将设置为 `null`。

副标题 `subtitle` 的构成方式类似于标题 `title`。其语法形式如下：

```
subtitle:{  
  text:String  
}
```

`subtitle` 组件也具有配置项 `text`。该配置项的默认值为空。如果不设置配置项 `text`，则不显示副标题。

【实例 3-1：title】修改实例 1-1 的源代码如下：

```
chart: {  
  type: 'line'  
},  
subtitle: {  
  text:"  
},  
series: [{  
  name: '最高温度',  
  data: [7, 11, 8, 7, 9, 9, 9]  
}]
```

执行代码，运行结果如图 3.1 所示。虽然在代码中没有设置 `title` 组件，但图表仍然显示一个默认标题，标题内容为 `Chart title`。同时，虽然设置副标题 `subtitle`，但标题内容为空，所以没有副标题。



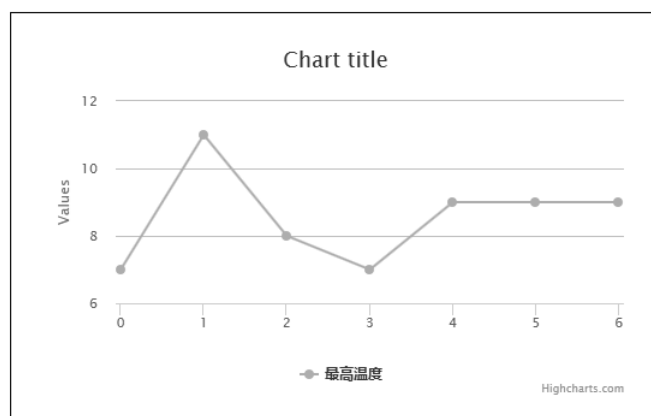


图 3.1 修改后的图表

### 3.1.2 标题的样式

标题的内容为文本形式。Highcharts 默认对一些基本 HTML 标签提供支持，如<b>、<br/>、<em>、<i>、<strong>。在文本中可以直接使用这几个基本标签。

【实例 3-2: titletext】为实例 1-1 的标题加粗，修改代码如下：

```
title: {  
  text: '<b>北京一周最高温度</b>' //使用加粗标签  
},
```

执行后，效果如图 3.2 所示。从图中可以明显看出，标题被加粗了。

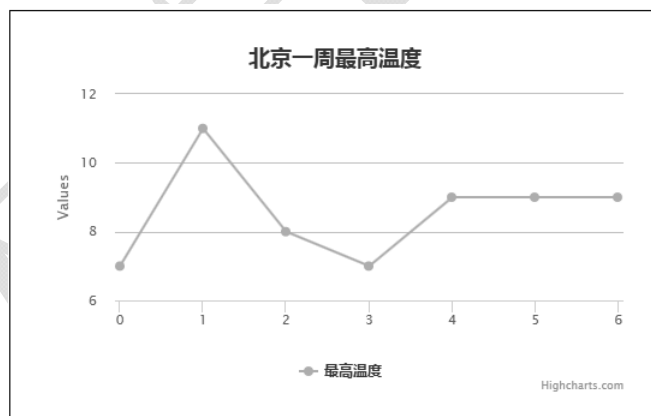


图 3.2 加粗的标题

虽然这五个标签功能常用，但是功能有限。为了扩展功能，title 和 subtitle 都提供以下两种方式。

#### 1.HTML 渲染 useHTML

设置配置项 useHTML 后，可以允许在标题文本中使用其他 HTML 标签。其语法如下：

useHTML:Boolean

其中，参数 Boolean 为布尔值，可以为 true 和 false。默认值为 false，表示不支持使用其他的 HTML 标签。

#### 2.设置标题样式 style

用户也可以指定标题文本所使用的 CSS 样式。其语法形式如下：

style:CSSObject

其中，参数 CSSObject 为 CSS 样式对象。默认值为{"color": "#555555"}。

### 3.1.3 标题的布局方式

由于标题位于图表区内部，往往还在绘图区的上面。所以，标题的布局依赖于这两者。下面详细讲解标题的布局方式。

#### 1.标题相对图表区的布局

由于标题区在图表区内部，所以标题可以以图表区作为标准，进行对齐。这时，需要使用 title/subtitle 的配置项 align 和 verticalAlign。其语法如下所示：

align:String1

verticalAlign:String2

其中，参数 String1 表示水平对齐方式，值包括 left、center、right，默认值为 center；参数 String2 表示垂直对齐方式，值包括 top、middle、bottom。

【实例 3-3: titlealign】设置实例 1-1 的标题在左下角显示。修改代码如下：

```
title: {
  text: '北京一周最高温度',
  align: 'left',           //设置水平左对齐
  verticalAlign:'bottom'  //设置垂直底对齐
},
```

执行后，效果如图 3.3 所示。

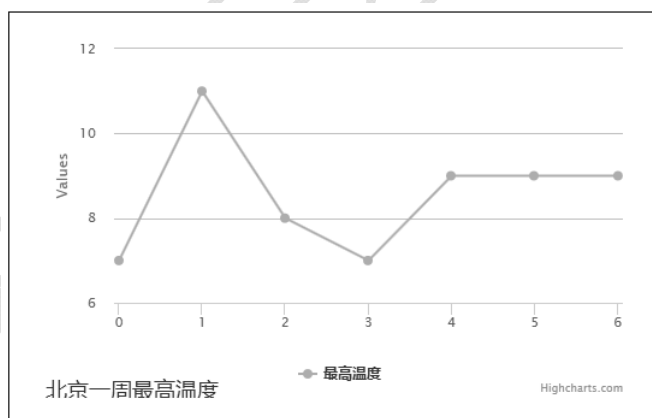


图 3.3 设置标题的对齐方式

#### 2.标题相对绘图区的布局

标题和绘图区都位于绘图区内部，标题可以浮动在绘图区上，也可以不浮动。用户可以通过配置项 floating 设置是否浮动。其语法如下：

floating:Boolean

其中，参数 Boolean 的值包括 true 和 false。默认值为 false

【实例 3-4: titlefloating】设置实例 1-1 的标题为浮动方式，修改代码如下：

```
title: {
  text: '北京一周最高温度',
  floating:true           //设置浮动方式
},
```

执行代码，效果如图 3.4 所示。从图中可以看到，标题进入了绘图区中。

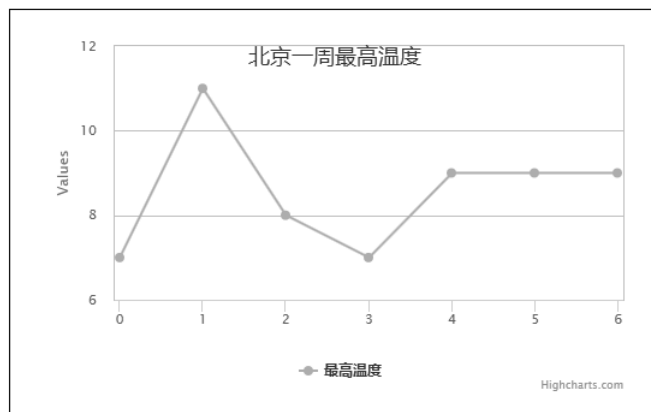


图 3.4 设置标题的浮动方式

如果设置了配置项 `verticalAlign` 的值，默认启用浮动方式。`subtitle` 也可以设置浮动方式。如果图表同时包含标题和副标题，建议两者浮动方式统一，并且只在 `title` 中设置。否则，会造成两个标题重合。当标题不设置浮动，可以通过配置项 `margin` 设置标题和绘图区的间距。其语法如下：

`margin: Number`

其中，参数 `Number` 表示标题和绘图区的间距，单位是 `px`。默认值为 15。如果存在副标题，则间距为 `Number+15`。

【实例 3-5: `titlemargin`】下面将实例 1-1 中的标题和绘图区间距进行修改，修改代码如下：

```
title: {
  text: '北京一周最高温度',
  margin: 100 //设置间距为 100
},
subtitle: {
  text: '2015.02.08--2015.02.14'
},
```

执行后，效果如图 3.5 所示。从图中可以看出，标题距离绘图区的距离是 `100px+15px`。其中，`15px` 是副标题和绘图区的默认间距。

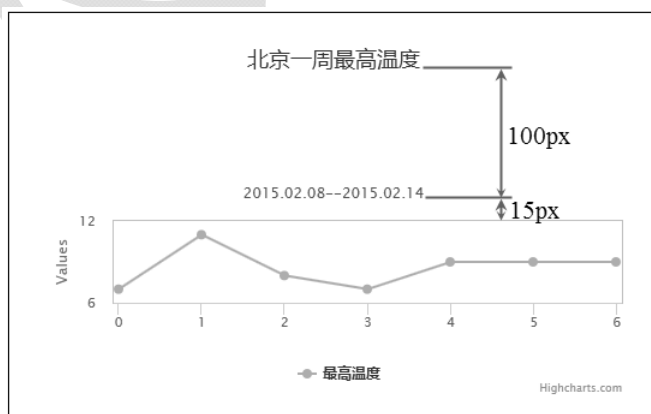


图 3.5 设置标题和绘图区间距

### 3.标题的偏移位置

除了使用 `align` 和 `verticalAlign` 对齐外，标题和副标题还可以基于对齐方式进行进一步的偏移。这一点与重置按钮 `resetZoomButton` 类似。它也通过横向偏移配置项 `x` 和纵向偏移配置项 `y` 实现。其语法如下：

```
x:Number1  
y:Number2
```

其中，参数 `Number1` 表示横行偏移距离的值，单位为 `px`，默认值为 `0`；参数 `Number2` 表示纵向偏移距离的值，单位为 `px`。

【实例 3-6: `titlexy`】下面将实例 1-1 的标题位置进行偏移，修改代码如下：

```
title: {  
  text: '北京一周最高温度',  
  align: 'left',           //设置横向对齐方式  
  x:100                   //设置横向偏移距离  
},
```

执行后，效果如图 3.6 所示。为了方便观察偏移位置，这里设置显示图表边框线。图表区与内部的图表元素本来左侧默认有 `10px` 的间隔。设置标题横向左对齐后，并设置偏移 `100px` 后，标题距离图表区的左边框距离为  $100+10=110\text{px}$ 。

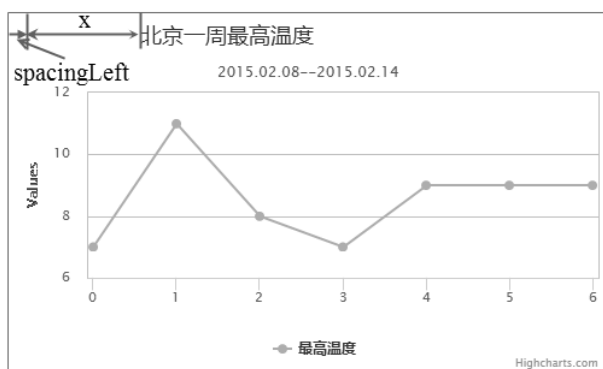


图 3.6 标题的偏移位置

由于副标题的设置效果与标题的设置效果等同，这里不再讲解。

## 3.2 版权信息

版权信息可以帮助浏览者明确信息发布者和权利所有者。通常版权信息为与右下角。我们前面实例中右下角，都有 `Highcharts.com` 的版权信息比较，如图 3.7 所示。

Highcharts.com

图 3.7 版权信息

`Highcharts` 图表的版权信息是由组件 `credits` 实现的。其语法如下：

```
credits: {
```

```
//配置项
```

```
}
```

下面讲解常见的版权信息配置项。

### 3.2.1 启用版权信息

默认情况下，显示版权信息功能是开启的。但很多时候，为了节省图表空间，需要禁用该功能。这时需要使用设置配置项 `enable`。其语法如下：

`enabled: Boolean`

其中，`enabled` 的值类型为布尔类型。如果值为 `true`，则显示版权信息；如果为 `false`，则不显示版权信息。默认值为 `true`。如果要禁用该功能，只需要将该配置项的值设置为 `false` 即可。

【实例 3-7: `creditsenabled`】下面禁止图表的版权信息显示。修改代码如下：

```
credits: {  
    enabled:false           //禁止显示版权信息  
}
```

### 3.2.2 设置版权信息内容

默认图表中的版权信息内容是 `Highcharts.com`，而单击后，会跳转到 `Highcharts` 的官网 `http://www.highcharts.com/`。对于开发者来说，这都是需要修改。用户可以通过配置项 `text` 和 `href` 来指定自己需要的版权信息。其语法如下：

`text: String1`

`href: String2`

其中，参数 `String1` 指定版权信息的文本内容，默认值为 `Highcharts.com`；参数 `String2` 指定版权信息的超链接网址，默认值为 `http://www.highcharts.com/`。

【实例 3-8: `textandhref`】下面修改图表默认的版权信息，修改代码如下：

```
credits: {  
    text: '大学霸',           //设置版权信息文本  
    href:'http://daxueba.net' //设置版权信息的超链接  
}
```

执行后，运行结果如图 3.8 所示。从图中看到，版权信息已经修改为代码所设置的信息了。



图 3.8 定制版权信息

### 3.2.3 设置版权信息位置和样式

除了能设置版权信息内容外，用户还可以设置版权信息显示的位置和各种样式。这时，需要使用配置项 `position` 和 `style`。其语法如下：

position: Object

style: CSSObject

其中，参数 Object 指定版权信息显示的位置信息，可以包含的项如下：

- ☐ align: 表示水平对齐方式，值可以为 left、center、right。默认值为 right。
- ☐ verticalAlign: 表示垂直对齐方式，值可以为 top、center、bottom。默认值为 bottom。
- ☐ x: 表示水平偏移位置，单位为 px。默认值为-10。
- ☐ y: 表示垂直偏移位置，单位为 px。默认值为-5。

所以，参数 Object 的默认值为

```
{
  align:'right',
  verticalAlign:'bottom',
  x:-10,
  y:-5
}
```

参数 CSSObject 指定版权信息的 CSS 样式。默认值如下：

```
{
  cursor: 'pointer',
  color: '#909090',
  fontSize: '10px'
}
```

### 3.3 标签组

在图表的大部分元素都提供了标签功能。但很多时候，我们需要额外说明一些信息。这个时候借助原有的图表元素的标签功能就不是很方便。Highcharts 为用户提供了标签组功能。使用该功能可以在图表区的任意位置添加一个或者多个标签，如图 3.9 所示。该图表中在副标题前面增加一个标签，用以说明数据列展现的年份。

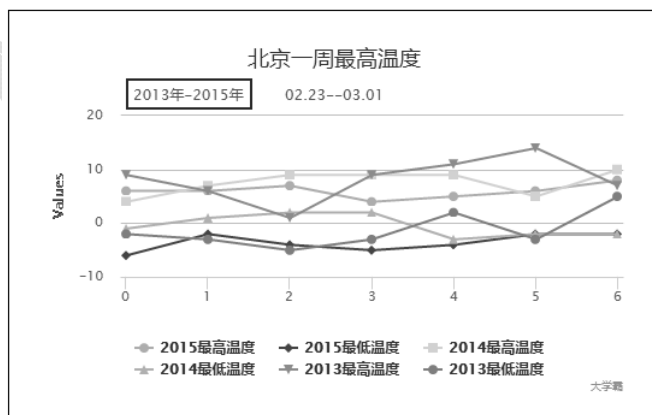


图 3.9 标签组的应用

#### 3.3.1 标签组的结构

在 Highcharts 中，标签组使用 labels 组件实现。其语法形式如下：

```
labels:{
    //相关配置项
}
```

由于标签组不隶属于任何图表元素，所以它不作为其他图表元素的子项，而直接包含在图表选项中。

### 3.3.2 构建标签

顾名思义，标签组是由一组标签构成。在使用的时候，用户可以定义一个，也可以定义多个标签。标签组中标签使用 items 组件构成。其语法如下：

```
items: [{
    //配置项
},
{
    //配置项
}
..... //可以是很多项
]
```

其中，每个标签都由一个花括号括起来；多个标签使用逗号分隔，然后用[]括起来。

构建标签，主要使用配置项 html 和 style 来设定标签内容和位置。其语法如下：

```
html:String1
style:CSSObject
```

其中，参数 String 指定标签内容，该值可以是 HTML 代码，也可以是纯文本内容。参数 CSSObject 指定标签的显示位置，包含 left 和 top 配置项。其语法如下：

```
left:Number1
top:Number2
```

其中，参数 Number1 表示标签的水平方向偏移距离，单位为 px，默认值为 0；参数 Number2 表示标签的垂直方向偏移距离，单位为 px，默认值为 0。

注意：必须设置 style，包括其中两个配置项 left 和 top 中的一个，这样才能显示标签。否则，光有配置项 html 是无法显示标签的。

【实例 3-9：items】下面实现图 3.9 的标签效果。修改代码如下：

```
labels: {
    items: [{
        html: '2013 年-2015 年', //添加标签
        style: { //设置标签内容
            left: 10, //设置标签位置
            top: -25 //设置水平位置
        } //设置垂直位置
    }]
}
```

执行后，效果如图 3.9 所示。从图中可以看到，标签默认以绘图区的左上角作为原点。配置项 x 和 y 都是相对该点计算的。

### 3.3.3 设置标签的样式

在组件 items 中，可以通过配置项 html 指定 HTML 脚本的方式来设定每个标签的文本样式。为了

设置方便，Highcharts 还为组件 labels 提供配置项 style 来设定所有的的标签样式。其语法如下：

style:CSSObject

其中，参数 CSSObject 用来指定所有标签共同的 CSS 样式。其默认值为{color: '#3E576F'}。

用户可以根据自己的需要添加特定的样式。

## 3.4 载入动画

载入动画是一个过渡动画。当图表加载数据需要时间较长，而无法展现图表数据，可以使用加载动画。其效果如图 3.10 所示。加载动画位于绘图区，会覆盖绘图区原有内容，并中心显示提示内容 Loading...。

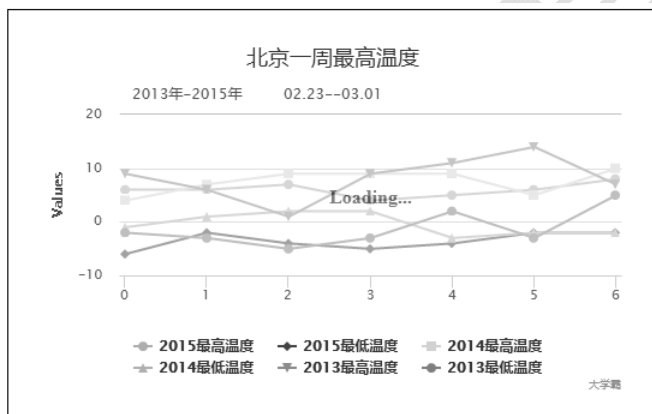


图 3.10 载入动画

### 3.4.1 显示载入动画

输入动画和前面的图表元素不同，它并不会自动显示。它需要使用图表对象来显式调用。这时需要使用方法 showloading()。其语法如下：

chart.showloading(String str)

其中，chart 必须是图表对象；参数 str 指定组件 loading 的配置项。该方法返回值为 null。

载入动画除了在方法 showloading 的参数指定，还可以在图表选项中指定。其语法形式如下：

```
loading:{
  //相关配置项
}
```

【实例 3-10: showloading】下面在图表中显示载入动画，修改代码如下：

```
<script type="text/javascript">
  $(document).ready(function () {
    var options = {
      chart: {
        type: 'line',
        zoomType: 'x'
      },
      title: {
        text: '北京一周最高温度'
      },
      series: [{
```



```

        name: '2015 最高温度',
        data: [6, 6, 7, 4, 5, 6, 8]
    }
    ],
    credits: {
        text: '大学霸',
        href: 'http://daxueba.net'
    }
});
var chart;
$('#container').highcharts(options);
chart = $('#container').highcharts();
chart.showLoading();

});
</script>

```

//定义图表变量  
 //创建图表动画  
 //获取图表对象  
 //显示载入动画

执行后，效果如图 3.11 所示。

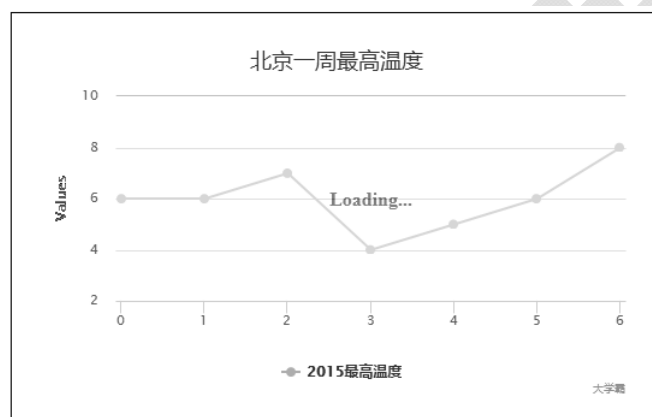


图 3.11 显示载入动画

载入动画并不会自动消失。如果取消载入动画，需要再使用图表对象的 `hideloading()` 方法。其语法如下：

```
chart.hideloading()
```

其中，`chart` 表示图表对象；该方法没有参数，返回值为 `null`。

### 3.4.2 本地化载入动画

从图 3.11 中可以看到，载入动画默认的提示内容是 `Loading...`。对于国内使用者，这并不友好。用户可以对提示内容本地化。需要使用到组件 `lang` 的配置项 `loading`。其语法如下：

```
loading: String
```

其中，参数 `String` 用来指定载入动画显示的提示内容。默认值为 `loading...`。

【实例 3-11: customloading】下面对载入动画的提示内容实现本地化，将其修改为“数据载入中...”。修改代码如下：

```

Highcharts.setOptions({
    lang: {
        loading: '数据载入中...'
    }
});
//设置载入动画的提示内容

```

```
}
});
```

执行后，效果如图 3.12 所示。

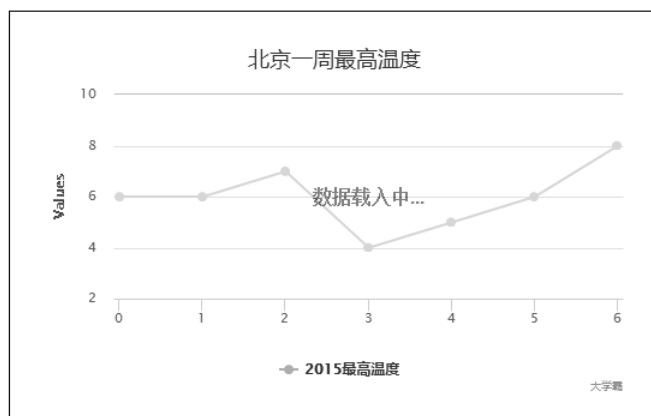


图 3.12 本地化载入动画

注意：对载入动画实施本地化，并不是设置组件 loading 的配置项，而是组件 lang 中设置。

### 3.4.3 设置动画效果

为了使载入动画更符合当前图表的需要，用户可以借助 Highcharts 提供的配置项对动画进行定制。下面依次讲解常用的几种定制方式。

#### 1. 设置动画区域的样式

载入动画的区域覆盖了绘图区。这样，显示载入动画的时候，就可以遮挡绘图区的显示，起到提示用户的作用。该区域可以使用组件 loading 的配置项 style 进行设置。其语法如下：

style: CSSObject

其中，参数 CSSObject 指定载入动画区域的 CSS 样式。默认值如下：

```
{
  position: 'absolute',
  backgroundColor: 'white',
  opacity: 0.5,
  textAlign: 'center'
}
```

#### 2. 设置提示内容样式

虽然提示内容支持一定的 HTML 的标签，但更好的方式是使用配置项 labelStyle。其语法如下：

labelStyle: CSSObject

其中，参数 CSSObject 指定动画提示内容的 CSS 样式。默认值如下：

```
{
  "fontWeight": "bold",
  "position": "relative",
  "top": "45%"
}
```

#### 3. 设置动画效果

除了设置各种 CSS 样式外，用户还可以设置动画的淡入、淡出效果的持续时间。这时，需要使用配

置项 `showDuration` 和 `hideDuration`。其语法如下：

`showDuration: Number1`

`hideDuration: Number2`

其中，参数 `Number1` 指定淡入动画持续时间，单位为毫秒 `ms`，默认值为 100；参数 `Number2` 指定淡出动画持续时间，单位为毫秒 `ms`，默认值为 100。

【实例 3-12: loading】下面重新设定载入动画的淡入效果持续时间，修改代码如下：

```
loading: {  
    showDuration: 1000,           //设置淡入效果持续时间  
    hideDuration: 1000           //设置淡出效果持续时间  
}
```

注意：淡入效果只有在显示载入动画时候才有效；反之，淡出效果只在隐藏载入动画时有效。