

taylor@ichartjs.com

ichartjs open source technology team

2013/6/26



# ichartjs 入门实用教程

本教程适用于 v1.2

<http://www.ichartjs.com>

## 变更说明

版本	更改日期	更改人	更新说明
1.0	2012/12/15	taylor	初始版本
1.1	2013/03/12	taylor	新增高级图表组件章节 详见: <a href="#">changelog</a>
1.2	2013/06/26	taylor	新增常用 api 章节 详见: <a href="#">changelog</a>

## 目录

1. 前言 .....	6
1.1. 文档说明 .....	6
1.2. 文档约定 .....	6
1.3. 图表术语 .....	6
1.4. 感谢语 .....	7
2. 简介 .....	8
2.1. what's the ichartjs? .....	8
2.2. 特点 .....	8
2.3. 应用场景 .....	8
3. 部署与结构 .....	9
3.1. 一步部署 .....	9
3.2. 组件结构分析 .....	9
3.3. 快速入门向导 .....	10
3.3.1. 说明 .....	10
3.3.2. 预览 .....	10
3.3.3. 代码 .....	10
4. 基础篇 .....	12
4.1. \$符号的含义 .....	12
4.2. 数据格式的定义 .....	12
4.2.1. 单一数据源 .....	12
4.2.2. 多值数据源 .....	12
4.3. 大小、内边距及边框 .....	13
4.3.1. 设置图表的大小 .....	13
4.3.2. 设置图表的内边距 .....	13
4.3.3. 图表的边框 .....	13
4.4. 对齐方式与偏移量 .....	14
4.4.1. 设置组件的大小 .....	14
4.4.2. 设置图表的对齐方式 .....	14
4.4.3. 设置图表的偏移量 .....	14
4.5. 阴影效果 .....	14
4.6. 背景色与渐变 .....	15
4.6.1. 设置背景色 .....	15
4.6.2. 设置渐变 .....	15
4.7. 字体样式 .....	15
4.8. 文字旋转 .....	16
4.9. 图表的标题 .....	16
4.10. 图表的脚注 .....	17
4.11. label 配置项的含义 .....	17
4.12. 过渡动画效果 .....	17
4.13. 图例 .....	18
4.13.1. 开启图例功能 .....	18

4.13.2.	图例的对齐方式.....	18
4.13.3.	图例的行列布局.....	19
4.13.4.	图例的行高.....	19
4.13.5.	图例符号的设置.....	20
4.13.6.	文字颜色与符号图形颜色一致.....	20
4.14.	提示框.....	20
4.14.1.	开启提示框功能.....	20
4.14.2.	固定位置与跟随位置.....	20
4.14.3.	提示框的渐入效果.....	21
4.15.	子配置项.....	21
4.16.	坐标系.....	21
4.16.1.	坐标系大小.....	21
4.16.2.	坐标轴线.....	22
4.16.3.	值轴.....	22
4.16.4.	自定义刻度文本.....	23
4.16.5.	有效坐标区域.....	23
4.16.6.	网格.....	24
4.16.7.	隔行变色.....	24
4.16.8.	自定义 3D 坐标系样式.....	25
5.	图表篇.....	27
5.1.	2D 饼状图.....	27
5.1.1.	数据源格式.....	27
5.1.2.	应用场景.....	27
5.1.3.	起始角度.....	27
5.1.4.	半径的设置.....	27
5.1.5.	扇形属性设置.....	27
5.1.6.	迷你 Label.....	28
5.1.7.	事件.....	28
5.1.8.	分离间距.....	28
5.2.	3D 饼状图.....	29
5.2.1.	数据源格式.....	29
5.2.2.	应用场景.....	29
5.2.3.	Z-轴旋转.....	29
5.2.4.	厚度.....	29
5.2.5.	分离间距.....	29
5.3.	2D 环形图.....	30
5.3.1.	数据源格式.....	30
5.3.2.	应用场景.....	30
5.3.3.	环形的宽度.....	30
5.3.4.	环形中心文字.....	30
5.3.5.	分离间距.....	30
5.4.	条形图.....	31
5.4.1.	2D 条形图.....	31
5.4.2.	数据源格式.....	31

5.4.3.	应用场景.....	31
5.4.4.	条形区域属性设置.....	31
5.4.5.	设置条形的高度.....	31
5.4.6.	事件.....	31
5.5.	2D 簇状条形图.....	32
5.5.1.	数据源格式.....	32
5.5.2.	应用场景.....	32
5.6.	2D(百分比)堆积条形图(news) .....	32
5.6.1.	数据源格式.....	32
5.6.2.	应用场景.....	32
5.6.3.	应用为百分比堆积图.....	32
5.7.	2D 柱形图.....	33
5.7.1.	数据源格式.....	33
5.7.2.	应用场景.....	33
5.7.3.	柱形区域属性设置.....	33
5.7.4.	设置柱形的宽度.....	33
5.7.5.	事件.....	33
5.8.	2D 簇状柱形图.....	34
5.8.1.	数据源格式.....	34
5.8.2.	应用场景.....	34
5.9.	2D(百分比)堆积柱形图 .....	34
5.9.1.	数据源格式.....	34
5.9.2.	应用场景.....	34
5.9.3.	应用为百分比堆积图.....	34
5.10.	3D 柱形图.....	35
5.10.1.	数据源格式.....	35
5.10.2.	应用场景.....	35
5.10.3.	3D 旋转 .....	35
5.11.	3D 簇状柱形图.....	35
5.11.1.	数据源格式.....	35
5.11.2.	应用场景.....	35
5.11.3.	组内柱间距.....	35
5.12.	3D(百分比)堆积柱形图 .....	35
5.12.1.	数据源格式.....	35
5.12.2.	应用场景.....	35
5.12.3.	应用为百分比堆积图.....	35
5.13.	2D 折线图.....	36
5.13.1.	数据源格式.....	36
5.13.2.	应用场景.....	36
5.13.3.	线段属性设置.....	36
5.13.4.	设置线段数据点的大小与样式.....	36
5.13.5.	平滑曲线.....	36
5.14.	2D 面积图.....	37
5.14.1.	数据源格式.....	37

5.14.2. 应用场景.....	37
5.14.3. 设置区域的透明度.....	37
6. 常用 api 详解.....	38
6.1. draw.....	38
6.1.1. 简介.....	38
6.1.2. 参数.....	38
6.1.3. 返回值.....	38
6.2. eventOff.....	38
6.2.1. 简介.....	38
6.2.2. 参数.....	38
6.2.3. 返回值.....	38
6.3. eventOn.....	38
6.3.1. 简介.....	38
6.3.2. 参数.....	38
6.3.3. 返回值.....	38
6.4. getCoordinate.....	38
6.4.1. 简介.....	38
6.4.2. 参数.....	38
6.4.3. 返回值.....	38
6.5. load.....	39
6.5.1. 简介.....	39
6.5.2. 参数.....	39
6.5.3. 返回值.....	39
6.6. resize.....	39
6.6.1. 简介.....	39
6.6.2. 参数.....	39
6.6.3. 返回值.....	39
7. 高级图表组件.....	40
7.1. 自定义图表组件.....	40
7.1.1. 应用场景.....	40
7.1.2. 绘图接口.....	40
7.1.3. plugin 方法.....	40
7.1.4. 示例-绘制 X 轴平均线.....	41
7.1.5. 示例-绘制说明文本.....	42
7.2. 组合图.....	43
7.2.1. 应用场景.....	43
7.2.2. 主图与子图.....	43
7.2.3. 组合原理.....	43
7.2.4. 共用坐标系.....	43
7.2.5. 在线示例.....	44

# 1. 前言

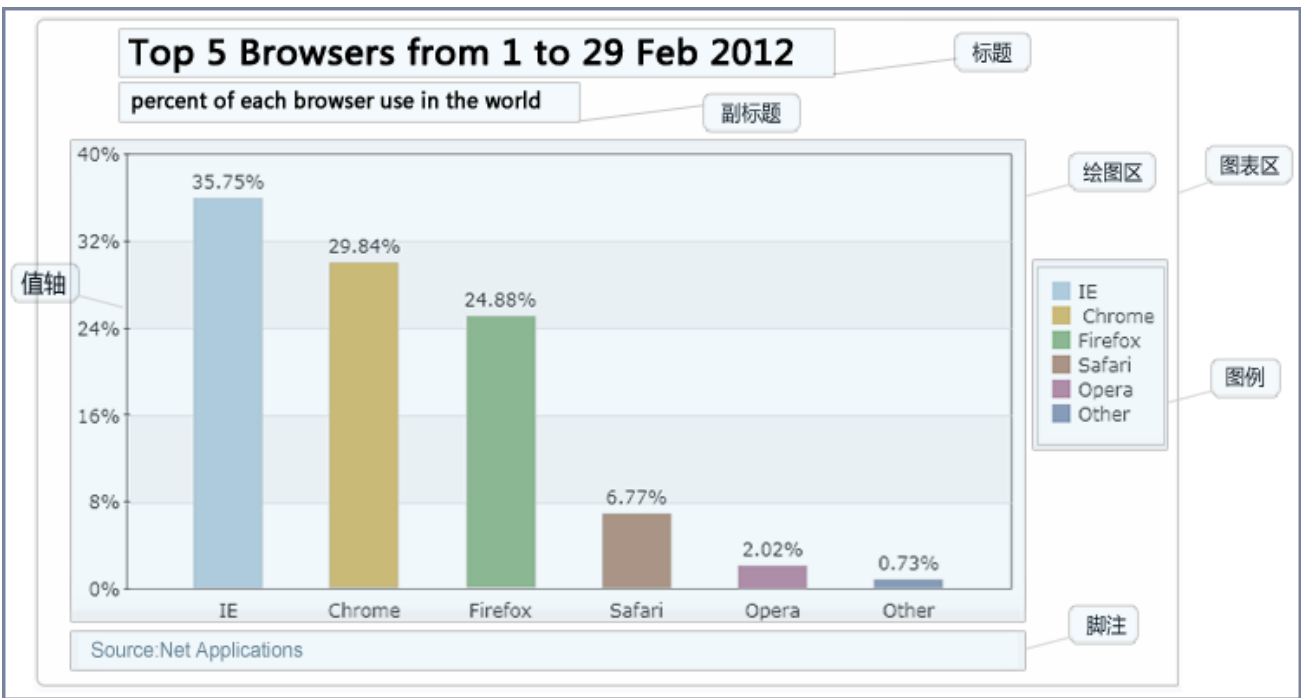
## 1.1. 文档说明

本文档作为快速入门的向导，在本文档中所用到的配置属性，并不是图表中的全部属性，要配合 `ichartjs` API 进行阅读才能达到事半功倍的效果。阅读完本文档可以基本掌握 `ichartjs` 的大致功能以及使用方式，要想达到随心所欲的境界还是要多多进行实例练习，组件中提供了大量的实例是学习 `ichartjs` 的好途径。本文档适用于 `ichartjs.v1.2` 正式版本。如果您在阅读过程中有任何的意见和建议请发送邮件给我们：[taylor@ichartjs.com](mailto:taylor@ichartjs.com)。

## 1.2. 文档约定

- a) 文档中，默认的情况下都已经导入了 `ichart-1.2.min.js`。
- b) 文档中，示例代码默认是在 `$(function(){ ... })` 中。且初始化之后，均调用了 `draw()` 方法。
- c) 文档中，配置中的 `render : 'canvasDiv'` 表示渲染目标是一个 `id` 为 `canvasDiv` 的 `div`。
- d) 文档中，数值未指明单位时，默认单位均为像素 (`px`)。
- e) 文档中，除特殊说明，基础篇中的代码适用于所有图表类型，示例中均以 `Column2D` 做示范。

## 1.3. 图表术语



**图表区：**整个图表的背景区域，其他组成部分都绘制在图表区中，如图表标题、绘图区、图例、垂直轴、水平轴、脚注以及网格线等。

**绘图区：**图表中的核心组成部分，包括数据的展现形式和坐标系等。

**标题：**用于显示图表的主题。

**副标题：**用于对图表的主题进一步补充。

**图例：**用于表示图表中的数据系列的名称或者分类而指定的图案或颜色。

**坐标轴：**分为垂直轴和水平轴，对于不同方向的图表这两个轴的含义不同，对于像柱形图这样的图表，

垂直轴用于确定图表中垂直坐标轴的最小和最大刻度值。水平轴用于显示文本标签。而对于像条形图这样的水平图表则水平确定图表中垂直坐标轴的最小和最大刻度值。垂直轴用于显示文本标签。有刻度值的轴又称为值轴。

**脚注:**位于页面的底部，可以作为文档某处内容的注释。一般用于说明数据的来源。

#### 1.4. 感谢语

在编写本文档时得到了广大网友的热心帮助，指出了很多编写中的不当之处。在后期校对过程中，众多热心网友协助完成了校对工作。在这里要向他们表示衷心的感谢。

同时，`ichart.js` 是一款刚刚起步的开源项目。我们坚信有了你的参与我们才能继续发展壮大。你有多种参与我们的方式：代码与文档的编写，`bug` 的提交与修复，组件的使用与推广等等。你的任何一个建议就是对我们莫大的支持。你的关注就是我们前进的动力。期待您参与到我们中来。



## 2. 简介

### 2.1. what's the ichartjs?

ichartjs 是一款基于 HTML5 的图形库。使用纯 javascript 语言，利用 HTML5 的 canvas 标签绘制各式图形。ichartjs 致力于为 WEB 应用提供简单、直观、可交互的体验级图表组件。是 WEB 图表展示方面的解决方案。如果你正在开发 HTML5 的应用，ichartjs 正好适合你。ichartjs 目前支持饼图、折线图、区域图、柱形图、条形图。ichartjs 是基于 Apache License 2.0 协议的开源项目。目前最新的版本是 2013 年 06 月 26 日发布的 v1.2 版本。v1.2 版本新增了堆积柱形图、百分比堆积柱形图、组合图。

### 2.2. 特点

ichartjs 是一款基于 HTML5 的图形库。所以天生具有以下特点：

- 轻量级代码
- 运行速度快
- 跨平台
- 基于事件的交互性
- 自定义绘图接口
- 灵活的 Web 视觉效果

### 2.3. 应用场景

ichartjs 是一款基于 HTML5 的图形库。它是基于 web 的图形库，所以它的运行环境就是支持 HTML5 的浏览器，如今，主流的浏览器已经可以很好的支持 HTML5 了。ichartjs 致力于为企业应用提供简单、直观、可交互的体验级图表组件。并且提供一个跨平台的统一视图。所以很适应于应用具有多平台，多终端的 web 应用。适用于 iPhone 与 android 的手机平台。

### 3. 部署与结构

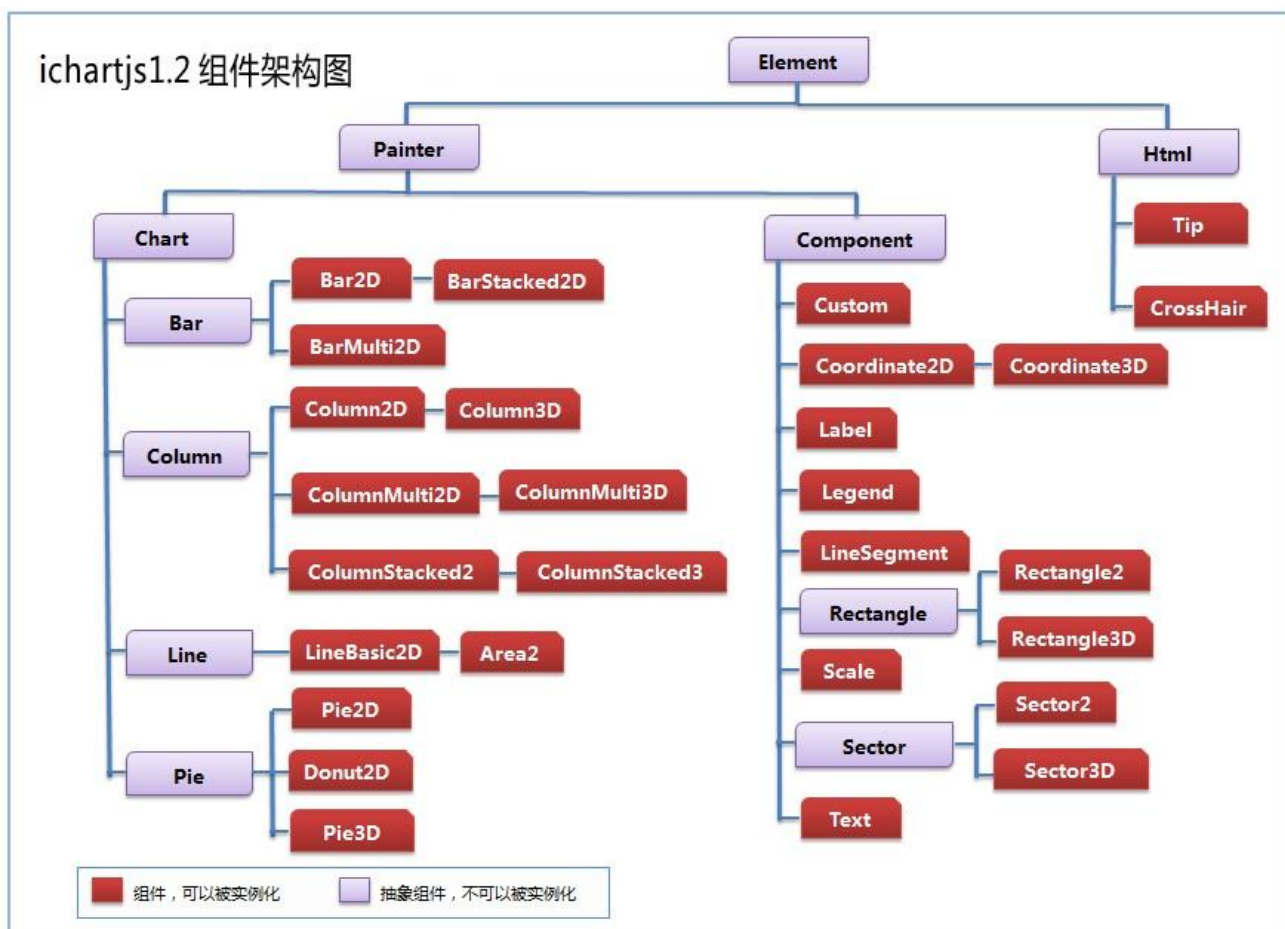
#### 3.1. 一步部署

ichartjs 本质上就是一个 js 库，所以只要将其引入你的页面即可。代码如下：

```
<script type="text/javascript" src="ichart-1.2.min.js"></script>
```

引入 ichart.1.2.min.js 就完成了搭建 ichartjs 运行环境的操作。

#### 3.2. 组件结构分析



- ◆ **Element**: 组件顶级父类，定义了配置功能特性。
- ◆ **Painter**: 画图的抽象类，定义了画图功能特性。
- ◆ **Chart**: 图表的父类，定义了图表的基本属性以及事件的特性。
- ◆ **Component**: 图表组件的父类，定义了具有画图以及响应事件的特性。
- ◆ **Html**: 所有以 html 方式构建的组件的父类，定义了基本 css 属性。且具有 Transition 动画特性。

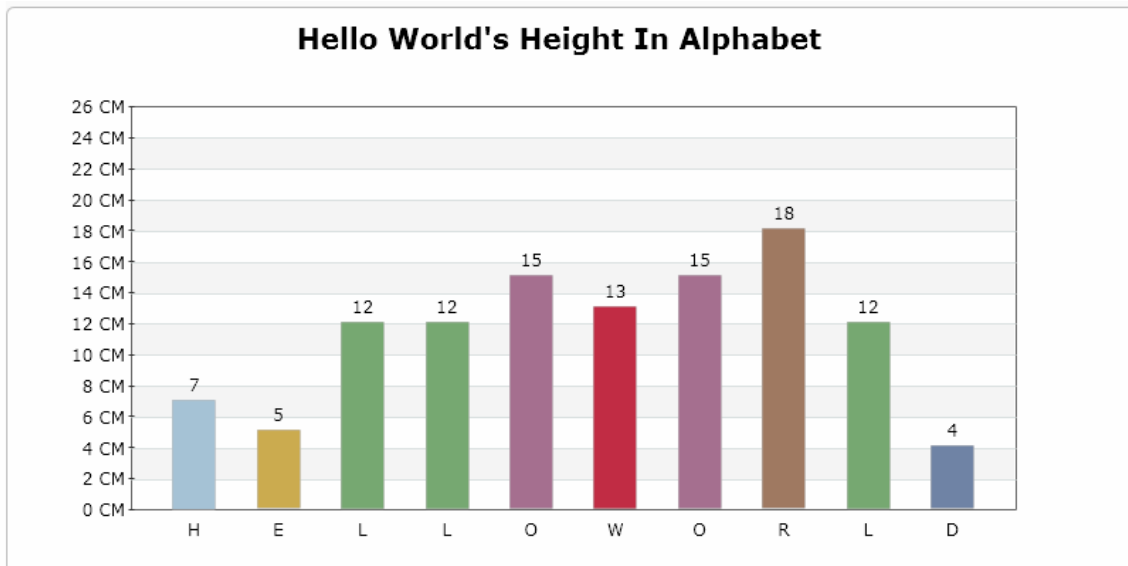
以上所有继承 Chart 的组件均为图表组件，继承 Component 的组件均为图表服务的小部件，继承 Html 的则属于利用 Html 模拟一些效果的部件。

### 3.3. 快速入门向导

#### 3.3.1. 说明

在这节中，我们利用 2D 柱形图来展示 Hello World 的各个字母在字母表中的位置的情况。我们将位置抽象为其高度。来以我们的方式来展示一个不一样的 Hello World。

#### 3.3.2. 预览



#### 3.3.3. 代码

导入 ichart.js Code

```
<script type="text/javascript" src="ichart-1.2.min.js"></script>
```

#### Javascript 代码

//定义数据源

```
var data = [  
  {name : 'H',value : 7,color:'#a5c2d5'},  
  {name : 'E',value : 5,color:'#cbab4f'},  
  {name : 'L',value : 12,color:'#76a871'},  
  {name : 'L',value : 12,color:'#76a871'},  
  {name : 'O',value : 15,color:'#a56f8f'},  
  {name : 'W',value : 13,color:'#c12c44'},  
  {name : 'O',value : 15,color:'#a56f8f'},  
  {name : 'R',value : 18,color:'#9f7961'},  
  {name : 'L',value : 12,color:'#76a871'},  
  {name : 'D',value : 4,color:'#6f83a5'}  
];  
$(function){  
  new iChart.Column2D({  
    render : 'canvasDiv', //渲染目标  
    data: data, //绑定数据
```

```

    title :{ text:'Hello World\'s Height In Alphabet'}, //设置标题
    width : 800, //图表宽度
    height : 400, //图表高度
    coordinate:{ //设置坐标系
        scale:[{
            position:'left', //左边值轴
            start_scale:0, //起始刻度值
            end_scale:26, //结束刻度值
            scale_space:2, //刻度值间距
            listeners:{
                parseText:function(t,x,y){
                    return {text:t+" CM"}//个性化值轴
                }
            }
        }]
    }
    }).draw();//调用 draw 方法进行绘图
});

```

Html 代码

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

在你的页面中敲入上述代码，之后用浏览器打开这个页面，就是出现开头预览的效果。

## 4. 基础篇

### 4.1. \$符号的含义

\$符号是 iChart 对象的简写，同时它是一个便捷的函数，使你的 Dom 元素加载之后再运行代码。作用类似 body 里的 onload 函数。如果你使用了其他的 js 库也使用了\$，比如：jQuery。则在使用的时候直接用 iChart 代替\$即可。

### 4.2. 数据格式的定义

ichart.js 注重于图表的展示，所以数据源统一采用 json 对象方式。对于其他格式请自行转换为 json 格式。数据源种类分为单一数据源与集合多值数据源，单一数据源的值为单一的数值，而集合多值数据源为数值集合。不同图表利用不同的数据源。

#### 4.2.1. 单一数据源

单一的数据源格式为[ {}, {}, {} ];

其中每个对象配置项如下：

name:数据项名称。

value:数据项值，可能值为数字或者数组。

color:数据项(扇形)的颜色。

line\_width(可选):用于折线图，表示线条的宽度。

例如：

代码片段：

```
var data = [
  {name : 'IE6',value : 20,color:'#a5c2d5'},
  {name : ' IE7',value : 15,color:'#cbab4f'},
  {name : ' IE8',value : 25,color:'#76a871'},
  {name : ' IE9',value : 30,color:'#9f7961'},
  {name : ' IE10',value : 10,color:'#a56f8f'}
];
```

#### 4.2.2. 多值数据源

多值数据源拥有两个与数据相关的属性，data, labels。与单一的数据源相比多了一个 labels 属性，并且 data 的 value 是数组的。我们没有将这两个属性整合到一起，因为你一旦理解这两个属性的作用。写起来是相当的容易的。

data 的格式范例：

代码片段：

```
var data = [
  {
    name : 'DPS01A',
    value:[45,52,54,74,90,84],
    color:'#1f7e92'
  },
```

```

    {
      name : 'DPS01B',
      value:[60,80,105,125,108,120],
      color:'#2b7f39'
    }
  ];

```

labels 的格式范例：

代码片段：

```
var labels= ["一月", "二月", "三月", "四月", "五月", "六月"];
```

可见 data 的 value 的长度是要与 labels 一致的，这说明 labels 是 value 对应的标签项。

### 4.3. 大小、内边距及边框

#### 4.3.1. 设置图表的大小

在创建图表时，设置图表宽度为 800px，高度为 400px 的代码如下：

代码片段：

```
new iChart.Column2D({
  width : 800,
  height : 400
});
```

如果在手客户端需要自适应屏幕，则设置 fit:true 即可。

#### 4.3.2. 设置图表的内边距

在创建图表时，设置图表内边距为 10px 的代码如下：

代码片段：

```
new iChart.Column2D({
  padding : 10
});
```

#### 4.3.3. 图表的边框

设置图表的边框，此配置同时适用于图表组件中的边框设置。设置边框的属性如下：

enable :是否开启边框特性

color :边框颜色

width : 边框宽度

radius :边框圆角值

在创建图表时，设置图表边框的代码如下：

代码片段：

```
new iChart.Column2D({
  border : {
    enable : false,
    color : '#BCBCBC',
    width : 1,
    radius : 5
  }
});
```

```
}  
});
```

#### 4.4. 对齐方式与偏移量

##### 4.4.1. 设置组件的大小

##### 4.4.2. 设置图表的对齐方式

设置主图在图表中的位置。

在创建图表时，设置图表对齐的代码如下：

代码片段：

```
new iChart.Column2D({  
    align : 'left'  
});
```

图表的对齐边界是设置内边距之后的。如果想要达到很精确的位置。你应该配合位置偏移量使用。

##### 4.4.3. 设置图表的偏移量

设置主图的坐标偏移量，有两个关键属性：

offsetx: x 轴方向的偏移量，正数向右偏移，负数向左偏移。

offsety: y 轴方向的偏移量，正数向下偏移，负数向上偏移。

如果想要达到很精确的位置。你应该配合位置偏移量使用。

在创建图表时，设置图表偏移量的代码如下：

代码片段：

```
new iChart.Column2D({  
    offsetx : 10 ,  
    offsety : 10  
});
```

#### 4.5. 阴影效果

阴影效果是图表组件共有的属性，但一般我们主要对图表进行设置。设置阴影主要有以下几个属性：

shadow: 是否启用阴影效果。

shadow\_blur: 阴影效果的模糊值。

shadow\_color: 阴影颜色。

shadow\_offsetx: 阴影 x 轴偏移量，正数向右偏移，负数向左偏移。

shadow\_offsety: 阴影 y 轴偏移量，正数向下偏移，负数向上偏移。

启用图表的阴影效果或多或少会影响图表的渲染时间，当你对渲染速度没有苛刻的要求的时候请放心使用。

在创建图表时，设置图表阴影的代码如下：

代码片段：

```
new iChart.Column2D({  
    shadow : true,  
    shadow_blur : 4,
```

```
shadow_color : '#111111',
shadow_offsetx : 2 ,
shadow_offsety : 2
});
```

## 4.6. 背景色与渐变

### 4.6.1. 设置背景色

背景色是一个基本的属性，图表中的组件都拥有这个属性，图表中的颜色属性支持：rgb, rgba, 颜色名称的颜色(如:gray)和十六进制值的颜色（比如 #ff0000）。

在创建图表时，设置图表背景色的代码如下：

代码片段：

```
new iChart.Column2D({
    background_color : '#EFEFEF'
});
```

### 4.6.2. 设置渐变

渐变是一个很常见的配色技巧，组件中也提供了便捷设置渐变的属性：

gradient：是否开启渐变，默认 false。

color\_factor：颜色渐变因子。取值越大渐变效果越大，颜色变化是基于背景颜色的。取值范围 {0.01 - 0.5}。默认 0.15。

gradient\_mode：渐变模式，有以下几个可选值：

- 'LinearGradientUpDown': 上-下
- 'LinearGradientDownUp': 下-上
- 'LinearGradientLeftRight': 左-右
- 'LinearGradientRightLeft': 右-左
- 'RadialGradientOutIn': 外-内
- 'RadialGradientInOut': 内-外

其中，饼图只有外-内，内-外模式选项。

在创建图表时，设置图表背景渐变的代码如下：

代码片段：

```
new iChart.Column2D({
    gradient : true,
    color_factor:0.2
});
```

## 4.7. 字体样式

设置字体的属性如下：

font：同 css 的 font-family。

fontsize：同 css 的 font-size。

fontweight：同 css 的 font-weight。

color：同 css 的 color。



在创建图表时，设置图表标题字体的代码如下：

代码片段：

```
new iChart.Column2D({
  title: {
    text: '2012 年 2 月份市场浏览器占用情况',
    font: 'sans-serif',
    fontsize: 14,
    fontweight: 600,
    color: '#ffffff'
  }
});
```

#### 4. 8. 文字旋转

调整文字的方向在文本过长，数据多，空间狭小的情况下使用。

在创建图表时，设置 x 轴 label 的文字向下 45 度的代码如下：

代码片段：

```
new iChart.Column2D({
  label: {
    rotate: -45,
    textBaseline: 'middle',
    textAlign: 'right'
  }
});
```

代码含义为，以右侧中部为原点，逆时针方向旋转 45 度。

#### 4. 9. 图表的标题

标题是图表必不可少的一部分，我们提供主标题与副标题。并且副标题必须在主标题存在的情况下而存在。图表的标题与副标题设置方式一致。其主要配置属性如下：

title：主标题配置项，你可以提供字符串配置项或者对象配置项，若给出的是字符串，则会用默认的字体颜色等配置渲染标题，若给出的是一个 iChart.Text 对象的配置项，则用对象配置项进行渲染。

subtitle：副标题配置项，与主标题一致，你也可以提供字符串配置项或者对象配置项。

在创建图表时，设置图表标题的代码如下：

代码片段：

```
new iChart.Column2D({
  title: '2012 年 2 月份市场浏览器占用情况',
  sub title: 'IE 依然领先于其他浏览器'
});
```

当对标题进行左对齐或者右对齐时，可能达不到你想要的效果，这时通过偏移量来对其进行调整。

代码如下：

代码片段：

```
new iChart.Column2D({
  title: {
```

```

        text : '2012 年 2 月份市场浏览器占用情况',
        offsetx:100,
        textAlgin : ' left'
    },
    sub title: {
        text : 'IE 依然领先于其他浏览器',
        offsetx:100
    },
    title_algin : ' left'
});

```

#### 4. 10. 图表的脚注

脚注有时候会给你的图表带来意想不到的效果。是对图表信息的一个注释。一般会将数据的来源以脚注的方式展示出来。其主要配置属性如下：

footnote：脚注配置项，你可以提供字符串配置项或者对象配置项，若给出的是字符串，则会用默认的字体颜色等配置渲染标题，若给出的是一个 iChart.Text 对象的配置项，则用对象配置项进行渲染。。

脚注的设置方式与标题一致。请参考 [图表的标题。](#)

#### 4. 11. label 配置项的含义

label 的含义是标签，具有提示性的文字都可以称作为标签。ichartjs 标签是一个 iChart.Text 对象。大部分图表本身与 sub\_option 选项的子组件均有 label 配置项，都是设置 label 文本的配置项。只是设置的文本不同而已。具体子配置项请参考 iChart.Text 的 api 手册。图表中各个组件中 label 配置含义如下：

Bar：配置左侧标签文本。

Column：配置底部标签文本。

Line：配置底部标签文本。如果 scale 中配置了。则以 scale 中为准。

Pie：无。

Scale：配置轴标签文本。

LineSegment：配置折线交点上方的标签文本。设置为 false 则禁用。

Sector：此组件中有两个含义：当设置 mini\_label 为 true 时，并且扇形角度在 mini\_label\_threshold\_angle 之上。则此配置项为 iChart.Text 对象。配置的为扇形图中心区域的文本。否则，配置项为扇形图外侧标签对象。为 iChart.Label 对象。设置为 false 则禁用。

Rectangle：配置柱形的标签文本。设置为 false 禁用。

#### 4. 12. 过渡动画效果

动画效果往往给人带来意想不到的效果，组件也提供一些简单的过渡动画，其属性如下：

animation：是否开启过渡动画效果。

animation\_timing\_function：过渡动画效果函数。

duration\_animation\_duration：过渡动画过程所用的时间，单位毫秒。

渲染动画对于手持客户端来说，目前由于硬件的原因运行效果还不够流畅，时间上会有一些延迟。

但是对于 PC 来讲则没有任何问题。同时注意开启阴影效果会对动画造成一定的延迟。

在创建图表时，设置图表动画的代码如下：

代码片段：

```
new iChart.Column2D({
    animation: true
});
```

#### 4. 13. 图例

##### 4. 13. 1. 开启图例功能

图例的配置项通过属性 `legend` 来进行设置，配置项中有一个 `enable` 属性表示是否开启图例功能，仅当 `enable=true` 时才启用图例，并根据配置进行渲染。设置图例的代码如下：

代码片段：

```
new iChart.Column2D({
    legend: {
        enable : true
        ...
    }
});
```

##### 4. 13. 2. 图例的对齐方式

图例位置可以围绕在绘图区的周围。有两个属性定位其位置：

`align`：控制其水平方向的对齐方式。

`valign`：控制其垂直方向的对齐方式。

通过上面两个属性的组合可以配置出图例的对齐方式。其中 `{center/middle}` 这个组合是无效的。

以 `(align/valign)` 各种组合的位置如下：



设置图例在图表的右居中对齐的代码如下：


代码片段:

```
new iChart.Column2D({
  legend: {
    enable : true,
    align : 'right',
    valign: 'middle'
  }
});
```


#### 4. 13. 3. 图例的行列布局

图例的行列布局一般为左对齐或者右对齐选用一行布局方式，而上对齐或者下对齐选用一行对齐方式，当数据比较多时则进行多行多列布局形式。

设置图例一行布局方式的代码如下：

代码片段:	效果预览
<pre>new iChart.Column2D({   legend: {     enable : true,     row : 'max',     column: 1   } });</pre>	

设置图例一行布局方式的代码如下：

代码片段:	效果预览
<pre>new iChart.Column2D({   legend: {     enable : true,     row : 1,     column: 'max'   } });</pre>	

#### 4. 13. 4. 图例的行高

当你设置图例的文字之后可能会出现，多行重叠的问题，这时你可以通过属性 `line_height` 来设置行高。以避免多行重叠。

设置图例行距为 20px 的代码如下：

代码片段:

```
new iChart.Column2D({
  legend: {
    enable : true
    line_height: 20
  }
});
```

#### 4.13.5. 图例符号的设置

图例项的符号配置由以下几个属性决定：

sign: 符号的形状。

sign\_size: 符号的大小。

sign\_space: 符号与文字的间距。

设置图例符号的代码如下：

代码片段：

```
new iChart.Column2D({
  legend: {
    enable : true,
    sign : 'round',
    sign_size : 10,
    sign_space : 6
  }
});
```

#### 4.13.6. 文字颜色与符号图形颜色一致

配置项 text\_with\_sign\_color 标识文字是否与符号颜色一致，默认为 false。此配置要考虑你的符号图形的颜色与背景色的颜色差，以免由于颜色相近造成文字模糊不清。

### 4.14. 提示框

#### 4.14.1. 开启提示框功能

在图表中的空间有限，能表现的数据量也有限，这样，在一些特殊的地方，数据就可以以提示框的形式展示出来。图例的配置项通过属性 tip 来进行设置，配置项中有一个 enable 属性表示是否开启提示框功能，仅当 enable=true 时才启用提示框，并根据配置进行渲染。设置提示框的代码如下：

代码片段：

```
new iChart.Column2D({
  tip : {
    enable : true
    ...
  }
});
```

#### 4.14.2. 固定位置与跟随位置

提示框提供两种位置模式，固定位置则无论鼠标进入组件的位置，提示框则总是以固定的位置展示。

跟随位置模式下，提示框则会根据鼠标的位置变化而变化。

设置提示框的位置为跟随模式的代码如下：

代码片段：

```
new iChart.Column2D({
  tip : {
    enable : true
```

```
        showType: 'follow'
    }
});
```

#### 4.14.3. 提示框的渐入效果

提示框的渐入渐出效果是由 css3 的 transition 完成的，所以这个效果需要你的浏览器支持 transition。好在现在主流的浏览器都已经支持该属性了，所以你不必为这个问题担忧。关于此效果的配置项如下：

animation：是否开启动画效果。

fade\_duration：渐入渐出过程所用的时间。单位毫秒。

设置提示框渐入渐出的效果代码如下：

代码片段：

```
new iChart.Column2D({
    tip: {
        enable: true
        animation: true,
        fade_duration: 500
    }
});
```

#### 4.15. 子配置项

每个图表都是由不同形状的小组件构成，比如：饼图是由扇形构成。柱形图、条形图是由矩形构成。折线图是由线段构成。每个子组件均是一个对象。ichart.js 中设置子组件统一用 sub\_option 配置项。而每个配置项的具体内容，需要了解子组件的具体类型。建议对照各个组件具体的 api 进行配置。设置子配置项的代码如下：

代码片段：

```
new iChart.Column2D({
    sub_option: {

    }
});
```

#### 4.16. 坐标系

坐标轴是图表中必不可少的一部分，设计一个符合条件且又美观的坐标轴就需要熟悉每一个属性的含义。coordinate 是配置坐标轴的属性。是一个 iChart.Coordinate2D/ iChart.Coordinate3D 的对象。其中除了 Pie 图没有坐标系之外。其他图表均有坐标系的配置项。

##### 4.16.1. 坐标系大小

坐标系的宽度与高度可以设置为固定的像素值，也可以设置为百分比字符串。

设置坐标系大小的代码如下：

代码片段：

```
new iChart.Column2D({
  coordinate: {
    width: 600,
    height: '75%'
  }
});
```

#### 4.16.2. 坐标轴线

坐标系中有四个边，不同的图表中有着不同的意义和作用。组件中提供对四边轴的设置。axis 是配置四边轴线的属性。其子属性如下：

enable: 是否设置其值。

width: 轴四边的宽度，如果是数值，则代表四周均为此值，若为数组，则长度必须为 4。其 4 个值依次代表上-右-下-左。

设置轴线的代码如下：

代码片段：

```
new iChart.Column2D({
  coordinate: {
    axis: {
      enable: true,
      width: [0,0,2,2],
      color: '#333333'
    }
  }
});
```

#### 4.16.3. 值轴

值轴是图表中很重要部分，为图表提供数值参考线。组件中配置值轴的是 iChart.Scale 对象，由属性 scale 设置。其是一个数组类型或者对象类型的属性。一般来讲，一个值轴就满足需求了，但是，特殊情况下，可能会有多个值轴。这样就配置数组类型的 scale。其主要子属性如下：

position: 刻度的位置。此值一般会由图表中属性 scaleAlign 覆盖。

start\_scale: 起始刻度值。

end\_scale: 结束刻度值。

scale\_space: 刻度间距值。

scale2grid: 是否按照刻度画网格参考线。

scale\_enable: 是否显示刻度线。

scale\_size: 刻度线的大小(线条宽度)。

scale\_width: 刻度线的长度。

scale\_color: 刻度线颜色。

scaleAlign: 刻度线对齐方式。

labels: 自定义的刻度文本，一般作为非值轴的轴文本的设置。

设置刻度的代码如下：

代码片段:

```
new iChart.Column2D({
  coordinate: {
    scale: {
      position: 'left',
      start_scale: 0,
      end_scale: 26,
      scale_space: 2
    }
  }
});
```

#### 4.16.4. 自定义刻度文本

默认的刻度的文本都是数值，但是一般我们都要给单位。我们利用 `parseText` 事件返回自定义的刻度文本。参数中可以设置自定义的位置。

设置自定义的刻度代码如下：

代码片段:

```
new iChart.Column2D({
  coordinate: {
    scale: {
      position: 'left',
      start_scale: 0,
      end_scale: 26,
      scale_space: 2,
      listeners: {
        parseText: function(t,x,y) {
          return {text:t+" CM"}
        }
      }
    }
  }
});
```

#### 4.16.5. 有效坐标区域

在特殊情况下我们的图表可能不是完全按照坐标轴的大小设置的。这样组件提供自定义有效坐标区域的功能，主要有两个属性：

`valid_width`: 设置有效的坐标区域宽度，必须要小于坐标宽度。

`valid_height`: 设置有效的坐标区域高度，必须要小于坐标高度。

设置有效坐标区域的代码如下：

代码片段:

```
new iChart.Column2D({
  coordinate: {
    valid_width: 600
  }
});
```



```
    }  
  });
```

#### 4.16.6. 网格

网格是图表中很常用的元素，自定义的网格可以使你的图表看起来更和谐，组件网格通过属性 `grids` 来设置。`grids` 中有两个属性：

`horizontal`：水平网格配置项。

`vertical`：垂直网格配置项。

这两个属性的具体配置如下：

`way`：网格设置方式，可选值 `[share_alike, given_value]`。

`value`：根据属性 `way` 值的不同有着不同的意义，当 `way` 取值 `share_alike` 时表示网格间隙的数量。间隙的数量=网格线数量-1。当 `way` 取值 `given_value` 时表示网格间隙的距离。间隙的数量=坐标轴宽度/网格间隙的距离。

设置有 12 个间隙 (13 个网格线) 的网格代码如下：

代码片段：

```
new iChart.Column2D({  
  coordinate: {  
    grids: {  
      vertical: {  
        way: 'share_alike',  
        value: 12  
      }  
    }  
  }  
});
```

设置每个间隙为 50px 的网格代码如下：

代码片段：

```
new iChart.Column2D({  
  coordinate: {  
    grids: {  
      vertical: {  
        way: 'given_value',  
        value: 50  
      }  
    }  
  }  
});
```

#### 4.16.7. 隔行变色

隔行变色是常用的设计手法，有助于提升图表的可读性。组件提供便捷设置隔行变色的方式。默认此配置项是开启的，但要注意只有配置了背景色的坐标系才会生效。

设置隔行变色的代码如下：

代码片段:

```
new iChart.Column2D({
  coordinate: {
    striped: true
  }
});
```

#### 4.16.8. 自定义 3D 坐标系样式

3D 坐标系的各个面都可以自定义样式，通过指定背景颜色来达到个性化的配置。在配置项 `coordinate` 中，`wall_style` 是指定样式的子配置项，其可以指定长度为 6 的样式对象，每个对象中有 2 个属性：

`color` : 背景墙颜色

`alpha` : 背景墙透明度

设置坐标系各个背景的代码如下：

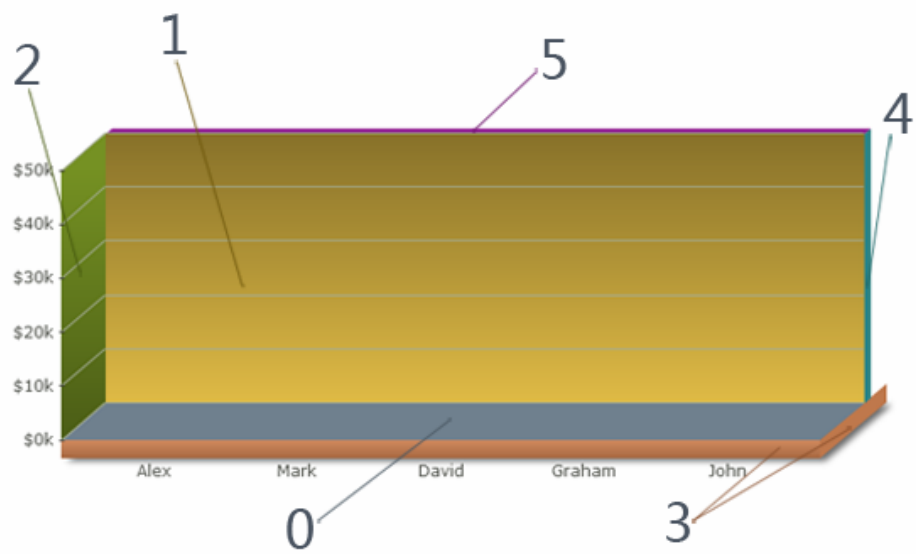
代码片段:

```
new iChart.Column3D({
  coordinate: {
    wall_style: [
      {color: '#333333'},
      {color: '#b2b2d3'},
      {color: '#a6a6cb'},
      {color: '#333333'},
      {color: '#74749b'},
      {color: '#a6a6cb'}
    ]
  }
});
```

预览(为了展示效果，将柱形隐藏):



其各个面与 `wall_style` 数组各个对象的对于关系如下图：



## 5. 图表篇

### 5.1. 2D 饼状图

#### 5.1.1. 数据源格式

饼状图的数据源格式为单一数据源型。其数据项中的 value 为数值。具体请参考：[数据源的定义-单值数据源](#)

#### 5.1.2. 应用场景

饼形图使用圆的扇形段来显示部分与整体的关系。饼形图对于突出显示比例非常有用。但是如果数据项非常的多，我们建议你选择其他的图形。

#### 5.1.3. 起始角度

默认第一个扇形的起始角度是 3 点钟方向，你可以通过配置项 offset\_angle 来设置。正值代表顺时针偏移，负值代表逆时针偏移。选择合适的角度能增加图表的可读性。

设置起始角度在 12 点钟方向的代码如下：

代码片段：

```
new iChart.Pie2D({
    offset_angle:-90
});
```

#### 5.1.4. 半径的设置

组件默认的半径会根据你图表的大小适应，但在特殊的时候，自己设置一个值是理想的选择。你可以通过配置项 radius 来设置。选择合适大小的半径值能使图表看起来更协调。

设置半径为 120px 的代码如下：

代码片段：

```
new iChart.Pie2D({
    radius:120
});
```

#### 5.1.5. 扇形属性设置

饼状图的由扇形组件构成，饼图中的扇形区域由 sub\_option 属性进行配置。扇形区域的详细的配置项可以通过 api 查询。

为扇形区域设置属性的代码如下：

代码片段：

```
new iChart.Pie2D({
    sub_option: {
        //...
    }
});
```

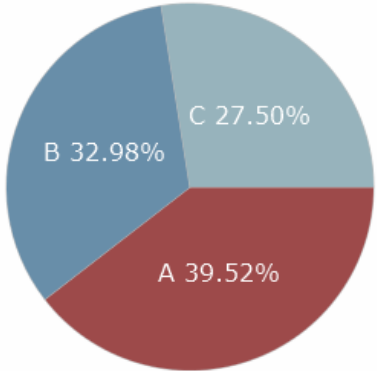
### 5.1.6. 迷你 Label

在数据较少的情况下，为了满足视觉效果，我们会将延伸在饼状图外侧的 label 去掉，将 Label 直接显示在饼图上。我们用配置项 `mini_label` 来开启这种表现的功能。有两个相关的配置项。

`mini_label`: 是否开启迷你 Label。

`mini_label_threshold_angle`: 开启迷你 Label 的最小角度。此配置项防止，由于角度太小，扇形区狭小，导致文本显示的问题。

开启迷你 Label 的代码如下：

代码片段:	效果预览
<pre>new iChart.Pie2D({   sub_option: {     mini_label : true,     mini_label_threshold_angle : 30 ,     label: {       fontsize:16,       color: '#FFFFFF'     }   } });</pre>	

### 5.1.7. 事件

饼状图的事件由其拥有的组件决定，饼图中有事件行为的组件有：扇形，图例，标签。所以设置事件均在相应的配置属性中进行设置。具体事件行为请参考组件 `api`。

为扇形区域设置单击事件的代码如下：

代码片段:
<pre>new iChart.Pie2D({   sub_option: {     listeners:{       click:function(c, e) {         //...       }     }   } });</pre>

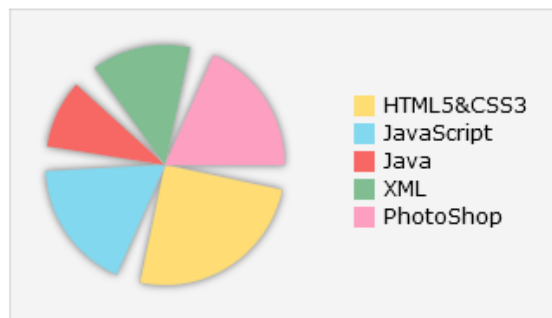
### 5.1.8. 分离间距

设置分离间距后，饼图中扇形处于分离状态，有着特殊的视觉效果。配置项 `separate_angle` 的取值范围为 (0-90)，单位为角度。注意，此角度是间距的总和，默认为 30。

设置分离间距的代码如下：

代码片段:	效果预览
-------	------

```
new iChart.Pie2D({
    separate_angle:60
});
```



## 5.2. 3D 饼状图

3D 饼状图除此了拥有 2D 饼状图的特征之外，还有其独有的设置。

### 5.2.1. 数据源格式

3D 饼状图的数据源格式为单一数据源型。其数据项中的 value 为数值。具体请参考：[数据源的定义-单值数据源](#)

### 5.2.2. 应用场景

3D 饼状图继承自 2D 饼状图。拥有 2D 的应用特点。而 3D 的效果具有强烈的视觉效果。

### 5.2.3. Z-轴旋转

3D 饼状图在 Z 轴的旋转角度值，默认为 45, 有效值为 (0-90)。

设置旋转角度值为 30 的代码如下：

代码片段：

```
new iChart.Pie3D({
    zRotate:30
});
```

### 5.2.4. 厚度

3D 饼状图的饼的厚度，默认为 30。

设置厚度为 50 的代码如下：

代码片段：

```
new iChart.Pie3D({
    zHeight:50
});
```

### 5.2.5. 分离间距

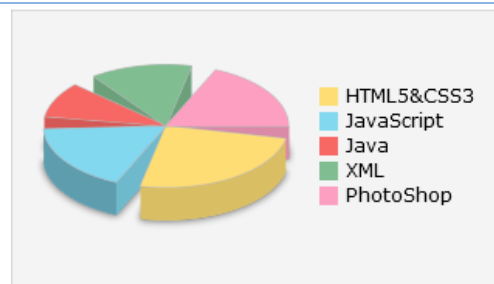
作用同 Pie2D 一致。

设置分离间距的代码如下：

代码片段：

效果预览

```
new iChart.Pie3D({
    separate_angle:60
});
```



### 5.3. 2D 环形图

#### 5.3.1. 数据源格式

环形图的数据源格式为单一数据源型。其数据项中的 value 为数值。具体请参考：[数据源的定义-单值数据源](#)

#### 5.3.2. 应用场景

目前环形图使用与饼状图一致，目前不支持多数据源。只是在表现形式上是饼状图中间有一个空心而已。但是可以利用这个空心，在上面写一些信息。

#### 5.3.3. 环形的宽度

设置不同环形的宽度可以有不同的视觉效果，这里提供两种设置方式。

固定宽度：设置一个固定的宽度值，单位 px。

百分比宽度：设置一个小数，按照半径的百分比设置环形宽度。

设置环形宽度为半径的 30%的代码如下：

代码片段：

```
new iChart.Donut2D({
    donutwidth:0.3
});
```

#### 5.3.4. 环形中心文字

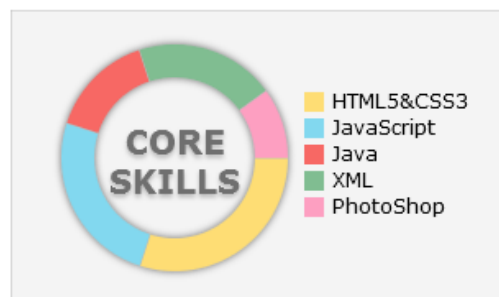
我们可以在环形中心来写文字，来表达一定的信息。利用 center 配置项来进行配置，此配置项是一个 iChart.Text 的对象。

设置环形中心文本的代码如下：

代码片段：

```
new iChart.Donut2D({
    center:{
        text:'CORE\nSKILLS',
        color:'#6f6f6f'
    }
});
```

效果预览



#### 5.3.5. 分离间距

环形图中的分离间距比 Pie2D 和 Pie3D 更常用一些。

设置分离间距的代码如下：

代码片段：	效果预览
<pre>new iChart.Donut2D ({     separate_angle:60 });</pre>	

## 5. 4. 条形图

### 5. 4. 1. 2D 条形图

### 5. 4. 2. 数据源格式

2D 条形图的数据源格式为单一数据源型。其数据项中的 value 为数值。具体请参考：[数据源的定义-单值数据源](#)

### 5. 4. 3. 应用场景

条形图用于展示具有类别关系，分布关系，时间关系的数据项。一般情况下均可用柱形图替换，但在轴标签过长的情况下，条形图是一个很好的选择。

### 5. 4. 4. 条形区域属性设置

条形图的由矩形组件构成，条形图中的矩形区域由 sub\_option 属性进行配置。矩形区域的详细的配置项可以通过 api 查询。

为矩形区域设置属性的代码如下：

代码片段：
<pre>new iChart.Bar2D({     sub_option: {         //...     } });</pre>

### 5. 4. 5. 设置条形的高度

默认情况下，组件会根据坐标系的高度来自动计算条形图中单个条形的高度值。你可以通过 bar\_height 属性来进行自定义高度，当然你设置的值的总和不能超过坐标系最大的高度。

设置单个条形的高度代码如下：

代码片段：
<pre>new iChart.Bar2D({     bar_height: 20 });</pre>

### 5. 4. 6. 事件

饼状图的事件由其拥有的组件决定，饼图中有事件行为的组件有：扇形，图例，标签。所以设置事



件均在相应的配置属性中进行设置。具体事件行为请参考组件 api。

为扇形区域设置单击事件的代码如下：

代码片段：

```
new iChart.Bar2D({
  sub_option: {
    listeners: {
      click: function(c, e) {
        //...
      }
    }
  }
});
```

## 5.5. 2D 簇状条形图

### 5.5.1. 数据源格式

2D 簇状条形图的数据源格式为多值数据源型。其数据项中的 value 为数组。具体请参考：[数据源的定义-多值数据源](#)

### 5.5.2. 应用场景

簇状条形图用于展示具有类别关系，分布关系，时间关系的数据项。簇状的特点是可以展现类别之间相关性。

## 5.6. 2D(百分比)堆积条形图 (news)

### 5.6.1. 数据源格式

2D(百分比)堆积条形图的数据源格式为多值数据源型。其数据项中的 value 为数组。具体请参考：[数据源的定义-多值数据源](#)

### 5.6.2. 应用场景

堆积条形图用于展示具有类别关系，分布关系，时间关系的数据项。堆积图的特点是可以比较相交于类别轴上的每一数值占总数值的大小。百分比堆积图则可以比较相交于类别轴上的每一数值占总数值的百分比。堆积条形图与堆积柱形图大多数情况下都可以通用。在轴标签过长的情况下，条形图是一个很好的选择。

### 5.6.3. 应用为百分比堆积图

在呈现单项数据百分比的时候，需要设置为百分比堆积图。

设置为百分比堆积图的代码如下：

代码片段：

效果预览

```
new iChart.BarStacked2D({
    percent : true
});
```



## 5.7. 2D 柱形图

### 5.7.1. 数据源格式

2D 柱形图的数据源格式为单一数据源型。其数据项中的 value 为数值。具体请参考：[数据源的定义-单值数据源](#)

### 5.7.2. 应用场景

柱形图用于展示具有类别关系，分布关系，时间关系的数据项。一般情况下均可用条形图替换，但在轴标签过长的情况下，条形图是一个很好的选择。

### 5.7.3. 柱形区域属性设置

柱形图的由矩形组件构成，柱形中的矩形区域由 sub\_option 属性进行配置。矩形区域的详细的配置项可以通过 api 查询。

为矩形区域设置属性的代码如下：

代码片段：

```
new iChart.Column2D({
    sub_option: {
        //...
    }
});
```

### 5.7.4. 设置柱形的宽度

默认情况下，组件会根据坐标系的宽度来自动计算条形图中单个柱形的宽度值。你可以通过 column\_width 属性来进行自定义宽度，当然你设置的值的总和不能超过坐标系最大的宽度。

设置单个柱形的宽度代码如下：

代码片段：

```
new iChart.Column2D({
    column_width: 20
});
```

### 5.7.5. 事件

饼状图的事件由其拥有的组件决定，饼图中有事件行为的组件有：扇形，图例，标签。所以设置事件均在相应的配置属性中进行设置。具体事件行为请参考组件 api。

为扇形区域设置单击事件的代码如下：

代码片段：

```
new iChart.Column2D({
  sub_option: {
    listeners: {
      click: function(c, e) {
        //...
      }
    }
  }
});
```

## 5.8. 2D 簇状柱形图

### 5.8.1. 数据源格式

2D 柱形图的数据源格式为多值数据源型。其数据项中的 value 为数组。具体请参考：[数据源的定义-多值数据源](#)

### 5.8.2. 应用场景

簇状柱形图用于展示具有类别关系，分布关系，时间关系的数据项。簇状的特点是可以展示组之间相关性与对比性。一般情况下均可用簇状条形图替换，但在轴标签过长的情况下，簇状条形图是一个很好的选择。

## 5.9. 2D(百分比)堆积柱形图

### 5.9.1. 数据源格式

2D(百分比)堆积柱形图的数据源格式为多值数据源型。其数据项中的 value 为数组。具体请参考：[数据源的定义-多值数据源](#)

### 5.9.2. 应用场景

堆积柱形图用于展示具有类别关系，分布关系，时间关系的数据项。堆积图的特点是可以比较相交于类别轴上的每一数值占总数值的大小。百分比堆积图则可以比较相交于类别轴上的每一数值占总数值的百分比。

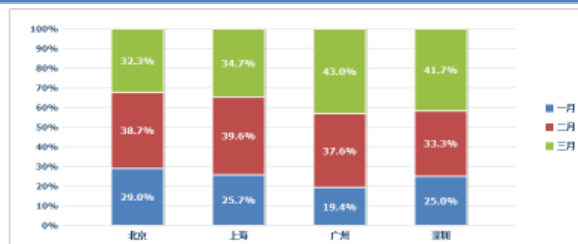
### 5.9.3. 应用为百分比堆积图

在呈现单项数据百分比的时候，需要设置为百分比堆积图。  
 设置为百分比堆积图的代码如下：

代码片段：

```
new iChart.ColumnStacked2D({
  percent : true
});
```

效果预览



## 5.10. 3D 柱形图

### 5.10.1. 数据源格式

2D 柱形图的数据源格式为单一数据源型。其数据项中的 value 为数值。具体请参考：[数据源的定义-单值数据源](#)

### 5.10.2. 应用场景

3D 柱形图继承自 2D 柱形图。拥有 2D 的应用特点。而 3D 的效果具有强烈的视觉效果。

### 5.10.3. 3D 旋转

我们提供两个属性对 3D 效果进行旋转设置。可以根据自己的视觉需要设置其值。

xAngle:x 轴方向旋转角度, 默认 60, 有效值 (0-90)

yAngle:y 轴方向旋转角度, 默认 20, 有效值 (0-90)

## 5.11. 3D 簇状柱形图

### 5.11.1. 数据源格式

2D 柱形图的数据源格式为多值数据源型。其数据项中的 value 为数组。具体请参考：[数据源的定义-多值数据源](#)

### 5.11.2. 应用场景

3D 簇状柱形图继承自 2D 簇状柱形图。拥有 2D 的特点。而 3D 的效果具有强烈的视觉效果。

### 5.11.3. 组内柱间距

3D 柱形图由于有 3D 的视觉效果, 所以在空间上会有遮挡的现象, 配置项 group\_factor 可以配置一个分组内的 3D 柱形之间的间距, 默认为 0.3。按照与其宽度的比例显示。

设置柱间距的代码如下:

代码片段:

```
new iChart.ColumnMulti3D ({  
    group_factor : 0.4  
});
```

## 5.12. 3D(百分比)堆积柱形图

### 5.12.1. 数据源格式

3D(百分比)堆积柱形图的数据源格式为多值数据源型。其数据项中的 value 为数组。具体请参考：[数据源的定义-多值数据源](#)

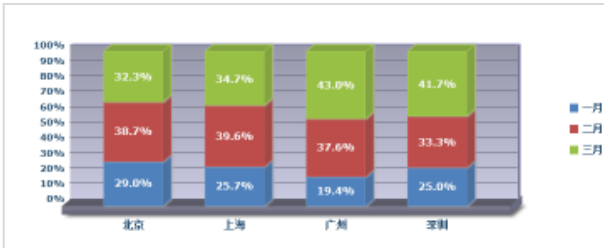
### 5.12.2. 应用场景

3D(百分比)堆积柱形图继承自 2D 堆积柱形图。拥有 2D 的特点。而 3D 的效果具有强烈的视觉效果。

### 5.12.3. 应用为百分比堆积图

在呈现单项数据百分比的时候, 需要设置为百分比堆积图。

设置为百分比堆积图的代码如下:

代码片段:	效果预览																				
<pre>new iChart.ColumnStacked3D({     percent : true });</pre>	 <table><caption>3D Stacked Bar Chart Data (Estimated)</caption><thead><tr><th>Category</th><th>一月 (%)</th><th>二月 (%)</th><th>三月 (%)</th></tr></thead><tbody><tr><td>北京</td><td>29.0%</td><td>38.7%</td><td>32.3%</td></tr><tr><td>上海</td><td>25.7%</td><td>39.6%</td><td>34.7%</td></tr><tr><td>广州</td><td>19.4%</td><td>37.6%</td><td>43.0%</td></tr><tr><td>深圳</td><td>25.0%</td><td>33.3%</td><td>41.7%</td></tr></tbody></table>	Category	一月 (%)	二月 (%)	三月 (%)	北京	29.0%	38.7%	32.3%	上海	25.7%	39.6%	34.7%	广州	19.4%	37.6%	43.0%	深圳	25.0%	33.3%	41.7%
Category	一月 (%)	二月 (%)	三月 (%)																		
北京	29.0%	38.7%	32.3%																		
上海	25.7%	39.6%	34.7%																		
广州	19.4%	37.6%	43.0%																		
深圳	25.0%	33.3%	41.7%																		

## 5.13. 2D 折线图

### 5.13.1. 数据源格式

2D 折线图的数据源格式为单一数据源型。其数据项中的 value 为数值。具体请参考：[数据源的定义-多值数据源](#)

### 5.13.2. 应用场景

折线图用于展示具有随时间或者类别变化数据的趋势。

### 5.13.3. 线段属性设置

折线图的由线段组件构成，折线图线段由 sub\_option 属性进行配置。线段的详细的配置项可以通过 api 查询。

为线段设置属性的代码如下：

代码片段:
<pre>new iChart.LineBasic2D ({     sub_option: {         //...     } });</pre>

### 5.13.4. 设置线段数据点的大小与样式

你可以设置适合自己图表风格的数据点大小与样式。属性如下：

point\_size : 数据顶点大小，默认为 6。

hollow : 是否为空心点，默认为空心。

设置单个条形的高度代码如下：

代码片段:
<pre>new iChart.LineBasic2D ({     sub_option: {         point_size : 10,         hollow : false     } });</pre>

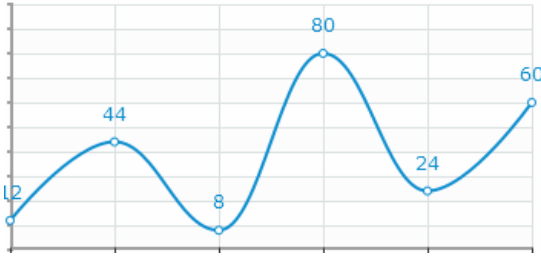
### 5.13.5. 平滑曲线

你可以设置适合自己图表风格的数据点大小与样式。属性如下：

point\_size : 数据顶点大小，默认为 6。

hollow : 是否为空心点，默认为空心。

设置直线图为平滑曲线的代码如下：

代码片段:	效果预览
<pre>new iChart.LineBasic2D ({   sub_option:{     smooth : true,     point_size:8   } });</pre>	

## 5. 14. 2D 面积图

### 5. 14. 1. 数据源格式

2D 面积图的数据源格式为单一数据源型。其数据项中的 value 为数值。具体请参考：[数据源的定义-多值数据源](#)

### 5. 14. 2. 应用场景

面积图与折线图类似，同样用于展示具有随时间或者类别变化数据的趋势。不过它们是在折线下的区域中显示区域颜色。这些色彩丰富的可视显示能更清楚地区分数据。

### 5. 14. 3. 设置区域的透明度

有时我们会有多个面积图汇集在一个图表中，这样重合的地方可能会被前面的面积图遮挡住，通过设置面积图的透明度可以解决这一问题。

为面积图设置透明度的代码如下：

代码片段:
<pre>new iChart.Area2D ({   area_opacity:0.2 });</pre>

## 6. 常用 api 详解

### 6.1. draw

#### 6.1.1. 简介

绘图方法。当实例化图表后，调用此方法，将触发图表开始绘制图形。

#### 6.1.2. 参数

无。

#### 6.1.3. 返回值

无。

### 6.2. eventOff

#### 6.2.1. 简介

关闭事件监听。调用此方法后，图表将忽略所有事件监听。图表将表现类似为一张静态图片。

#### 6.2.2. 参数

无。

#### 6.2.3. 返回值

无。

### 6.3. eventOn

#### 6.3.1. 简介

开启事件监听。调用此方法后，图表将开启所有事件监听。图表表现为有一定的交互能力。

#### 6.3.2. 参数

无。

#### 6.3.3. 返回值

无。

### 6.4. getCoordinate

#### 6.4.1. 简介

获取此图表的坐标系对象。饼图无此方法。

#### 6.4.2. 参数

无。

#### 6.4.3. 返回值

坐标系对象 `iChart.Coordinate2D`。

## 6.5. load

### 6.5.1. 简介

加载新的数据。调用此方法后，图表会自动调用 draw 绘制新图形。

### 6.5.2. 参数

data:数据，与原数据格式一致。

### 6.5.3. 返回值

无

## 6.6. resize

### 6.6.1. 简介

改变图表大小。调用此方法后，图表会自动调用 draw 绘制新图形。

### 6.6.2. 参数

width:宽度。

height:高度

### 6.6.3. 返回值

无



## 7. 高级图表组件

### 7.1. 自定义图表组件

#### 7.1.1. 应用场景

自定义图表组件(*iChart.Custom*)是一个非常非常有用的组件。用于在常规图上绘制个性化的内容。可以利用 *ichart.js* 封装的 *api* 或者调用原始的 *api* 绘制图形。

#### 7.1.2. 绘图接口

*iChart.Custom* 对象配置项 *drawFn* 默认是一个空函数，自定义绘图代码需要配置在此函数中。

构造自定义组件代码如下：

代码片段：

```
var custom = new iChart.Custom({
  drawFn:function(){
    //自定义绘图代码区
  }
});
```

#### 7.1.3. plugin 方法

图表中有一个方法 *plugin(Chart c)*，该方法会将传递进来的子图表 *c* 以插件的方式组合在主图中。插入自定义组件代码如下：

代码片段：

```
//构造主图
var column = new iChart.Column2D({
  //...
});
//构造自定义图表
var custom = new iChart.Custom({
  drawFn:function(){
    //....
  }
});
//将自定义组件插入主图
column.plugin(custom);
column.draw();
```

#### 7.1.4. 示例-绘制 X 轴平均线

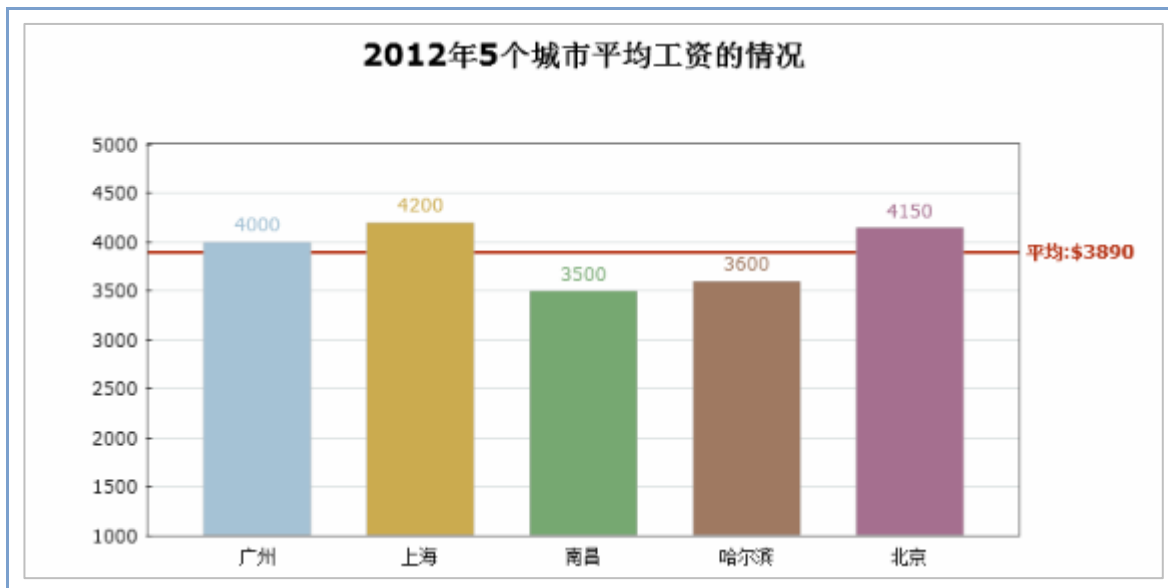
该图表是一个柱形图，展示了 2012 年 5 个城市平均工资的情况。利用自定义组件，在柱形图中绘制了一条醒目的工资平均线。[在线预览该示例](#)

绘图平均线的代码如下：

代码片段：

```
var chart = new iChart.Column2D({
    //...
});
/**
 *自定义组件
 */
chart.plugin(new iChart.Custom({
    drawFn:function(){
        /**
         *计算平均值的高度(坐标 Y 值)
         */
        var avg = chart.total/5,
            coo = chart.getCoordinate(),
            x = coo.get('originx'),
            W = coo.get('width'),
            S = coo.getScale('left'),
            H = coo.get('height'),
            h = (avg - S.start) * H / S.distance,
            y = chart.y + H - h;
        chart.target.line(x,y,x+W,y,2,'#b32c0d')
            .textAlign('start')
            .textBaseline('middle')
            .textFont('600 12px Verdana')
            .fillText('平均:$'+avg,x+W+5,y,false,'#b32c0d');
    }
}));
chart.draw();
```

效果预览：



#### 7.1.5. 示例-绘制说明文本

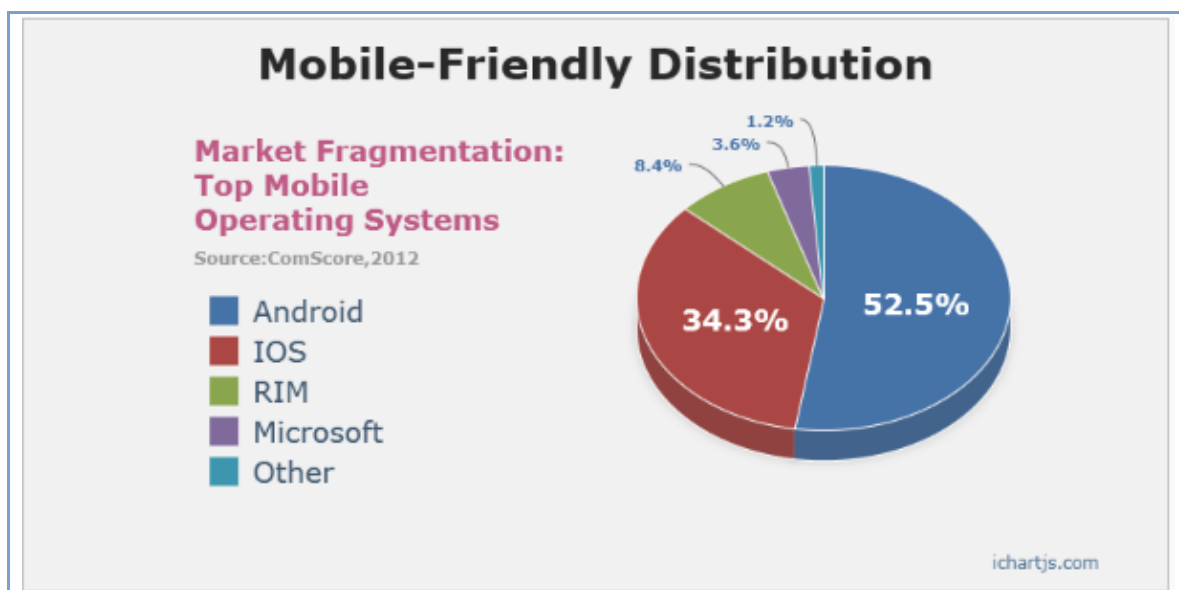
该图表是一个饼状图，展示了手机操作系统市场份额情况。利用自定义组件，在饼状图右侧绘制了一段说明文字，以便加强图表传达信息的力度。[在线预览该示例](#)

绘图说明文字的代码如下：

代码片段：

```
var chart = new iChart.Pie3D({
    //...
});
/**
 *自定义组件
 */
chart.plugin(new iChart.Custom({
    drawFn:function(){
        /**
         *在右侧的位置，渲染说明文字
         */
        chart.target.textAlign('start')
            .textBaseline('top')
            .textFont('600 20px Verdana')
            .fillText('Market Fragmentation:\nTop Mobile\nOperating Systems',120,80,false,'#be5985',false,24)
            .textFont('600 12px Verdana')
            .fillText('Source:ComScore,2012',120,160,false,'#999999');
    }
}));
chart.draw();
```

效果预览：



## 7.2. 组合图

### 7.2.1. 应用场景

组合图是将多个图表整合在一起，还可能共用一个坐标系。用于在有限视觉区域传达丰富的数据内容。其组合非常灵活，可以组合任意类型的图表。

### 7.2.2. 主图与子图

虽然组合图是各种图表组合在一起，但是也有主次之分。

主图：基础图表，渲染到 Dom 的图表。有 render 配置项。

子图：插件图表，渲染到主图中的图表。无 render 配置项。

### 7.2.3. 组合原理

图表中有一个方法 `plugin(Chart c)`，该方法会将传递进来的子图表以插件的方式组合在主图中。构造组合图的代码如下：

代码片段：

```
var chart = new iChart.Column2D({
  render: 'canvasDiv',
  //...
});
var pie = new iChart.Pie2D({
  //...
});
chart.plugin(pie);
```

### 7.2.4. 共用坐标系

多个图表共用一个坐标系是很常见的场景，共用一个坐标系可以加强对比的效果，而且也节省空间。组合图中共用坐标系的设置也很简单，只要将主图中的坐标系对象传递给子图即可。其中 `coo` 是坐标系对象的引用。

设置共用一个坐标系的代码如下：

代码片段:

```
var chart = new iChart.Column2D({
  render : 'canvasDiv',
  coordinate:{
    //...
  }
});
var line = new iChart.LineBasic2D({
  coordinate:chart.coo,
  //...
});
chart.plugin(line);
```

#### 7.2.5. 在线示例

- 1) [2D 柱形图+饼图](#)
- 2) [2D 柱形图+折线图](#)
- 3) [3D 柱形图+折线图](#)