



下载APP



## 37 | 实战（一）：如何使用图表库绘制常用数据图表？

2020-09-25 月影

跟月影学可视化

[进入课程 >](#)**讲述：月影**

时长 09:40 大小 8.87M



你好，我是月影。

图表是我们在可视化中展示数据常用的方式之一，常见的有柱状图、折线图、饼图等等，我们之前也都一一实现过。如果我让你总结一下图表的实现方法，你能说出来吗？总结不出来也没关系，这节课，我们就一起梳理一下实际项目中常用的制图方法。

实现图表有两种方式，一是使用现成的图表库，二是使用数据驱动框架，前者胜在简单易用，后者则更加灵活。所以，我们会用两节课的时间分别来讲，使用图表库和使用数据驱动框架的制图思路。



因为使用图表库更加简单，所以这一节课我们先来讲怎么使用它实现图表。另外，这节课的实践性比较强，我建议你跟着我的讲解一块儿动手去写，这样能更快地理解课程的内


容。

## 课前准备

说了这么多，我们今天到底会用哪些图表库呢？我们主要会使用我们比较熟悉的 SpriteJS 和 QCharts 来绘制图表。其他的图表库，例如更常用的 [ECharts](#)，它在图表实现上的原理和用法和我们今天讲的基本相同，所以学完了今天的内容，你完全可以根据它的教程文档去自学。

好了，确定了要使用的工具之后，我们就要配置和加载 SpriteJS 和 QCharts。具体怎么做呢？

最简单的方式是，我们直接通过 CDN，用 script 标签来加载 SpriteJS 和 QCharts 打包好的文件。

 复制代码

```
1 <script src="https://unpkg.com/spritejs/dist/spritejs.min.js"></script>
2 <script src="https://unpkg.com/@qcharts/core@1.0.25/dist/index.js"></script>
```


如果是完整的工程项目的话，你也可以使用 webpack 来打包和加载模块。这里有一个 [Quick Start](#)，你可以 fork 这个项目，按照其中的配置项来设置。当加载完成之后，我们就可以开始绘制基础的图表了。

## QCharts 图表的基本用法

QCharts 图表由图表（Chart）对象及其子元素构成。其中图表对象的子元素包含图形（Visual）和其他插件（Plugin）。图形是必选元素，其他的插件都是可选元素。


图表在构建的时候，需要传入一个 DOM 元素，这个元素可以是页面上任意一个块级元素，QCharts 用这个元素作为容器来创建 Canvas 画布，并其还会根据容器的大小来设置 Canvas 画布的大小。默认情况下，图表会根据画布大小来自动适配。

接下来，我们看一下具体的操作。首先，我们创建一个图表对象，设置它的容器设置成一个 id 为 app 的元素。

 复制代码

```
1 const { Chart, Line } = qcharts;
2 const chart = new Chart({
3   container: '#app'
4 });
```


创建了这个容器之后，我们就可以为它绑定数据，假设我们绑定一份销售数据。

 复制代码

```
1 const data = [
2   { date: '05-01', category: '图例一', sales: 15.2 },
3   { date: '05-02', category: '图例一', sales: 39.2 },
4   { date: '05-03', category: '图例一', sales: 31.2 },
5   { date: '05-04', category: '图例一', sales: 65.2 },
6   { date: '05-05', category: '图例一', sales: 55.2 },
7   { date: '05-06', category: '图例一', sales: 75.2 },
8   { date: '05-07', category: '图例一', sales: 95.2 },
9   { date: '05-08', category: '图例一', sales: 100 }
10 ];
11
12 chart.source(data, {
13   row: 'category',
14   value: 'sales',
15   text: 'date'
16 });
```

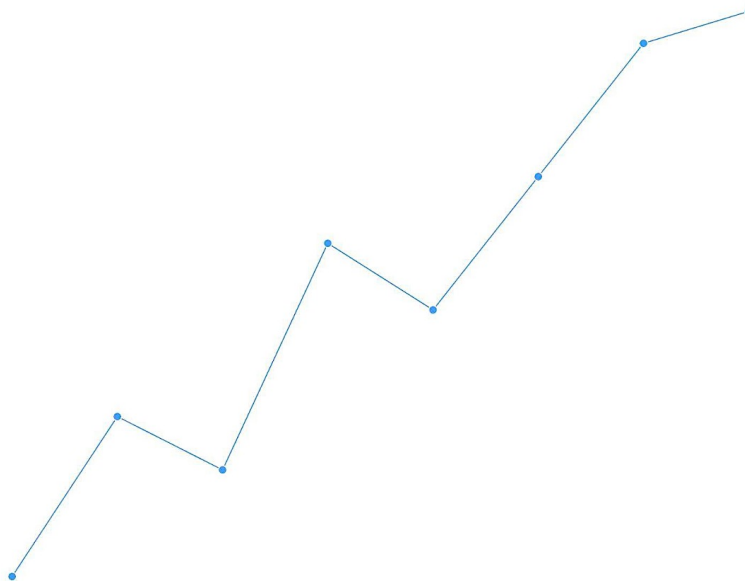
如上面代码所示，我们将数据内容与图表对象通过 `chart.source` 方法绑定起来。绑定数据的时候，我们可以指定数据的行（列）、数值和文本，这些设置会被图形（Visual）和其他插件使用。这里，我们将行设为 `category` 这个字段，数值设为 `sales` 字段，文本设为 `date` 字段。

设置之后，图表并没有马上显示出来。这是因为，我们还没有给图表指定图形。QCharts 支持多种图形对象，比如 `Line`、`Area`、`Bar` 等等。假设我们选择了 `Line` 对象，那我们只需要创建它，然后将它添加到 `chart` 对象的子元素中就可以了。

 复制代码

```
1 const line = new Line();
2 chart.append([line]);
```

这样，我们就让图形显示出来了，效果如下：



了解了 QCharts 图表的基本用法之后，我们一起进入实践环节吧。

## QCharts 绘制折线图、面积图、柱状图和饼图的方法

我们先来讲讲折线图、面积图、柱状图和饼图的实现方法，因为之前已经实现过很多次了，所以理解起来比较容易。

首先是折线图，它是可视化中最常用的图表之一。刚才我们用 Line 图形绘制了一条折线，但它还不是完整的折线图。完整的折线图，除了有图形以外，还要有表示数据的坐标轴、提示信息和图例，这些都需要在 QCharts 中由插件来完成。

接下来，我们就继续给前面的代码添加元素。首先，我们给它增加两个坐标轴，分别是底部和左侧的。

我们通过 Axis 类来创建 axis 对象，默认的坐标轴是在底部不用修改，再通过.style 改变它的样式属性，比如我们将"grid"设置为 false，这样画布上不会显示纵向的网格线。

```
1 const axisBottom = new Axis().style("grid", false);
```

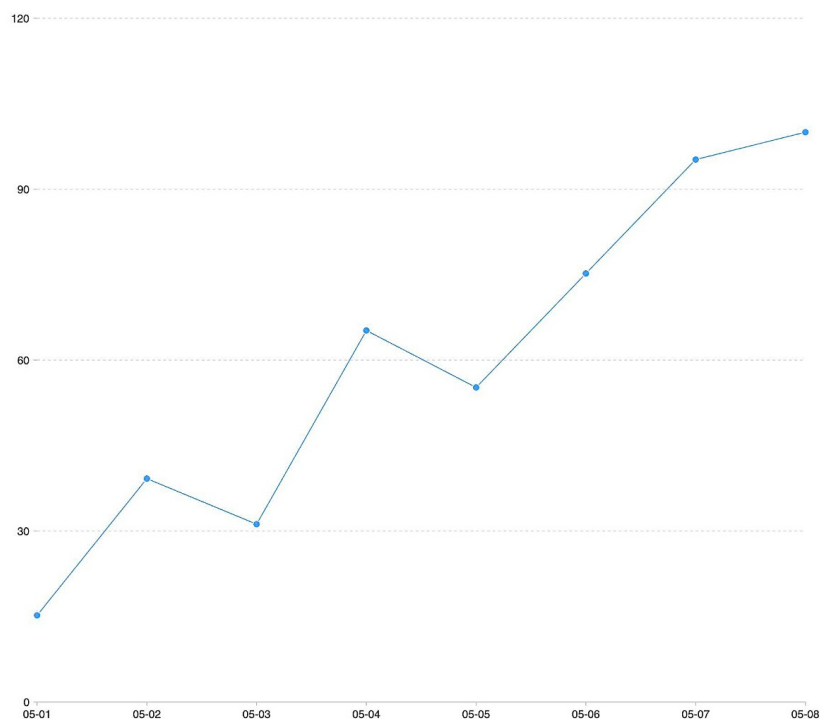
 复制代码

然后，我们同样创建一个 `axis` 对象，通过设置 `orient: "left"` 让它朝左，这样就成功创建了左侧的坐标轴。同样，我们也要把它样式中的 `"axis"` 属性设置为 `false`，这样画布上就不会显示坐标轴的实线了。

[复制代码](#)

```
1 const axisLeft = new Axis({ orient: "left" })
2   .style("axis", false);
```

最后，我们将两个坐标轴添加到 `chart` 对象的子元素中去，就可以将坐标轴显示出来。

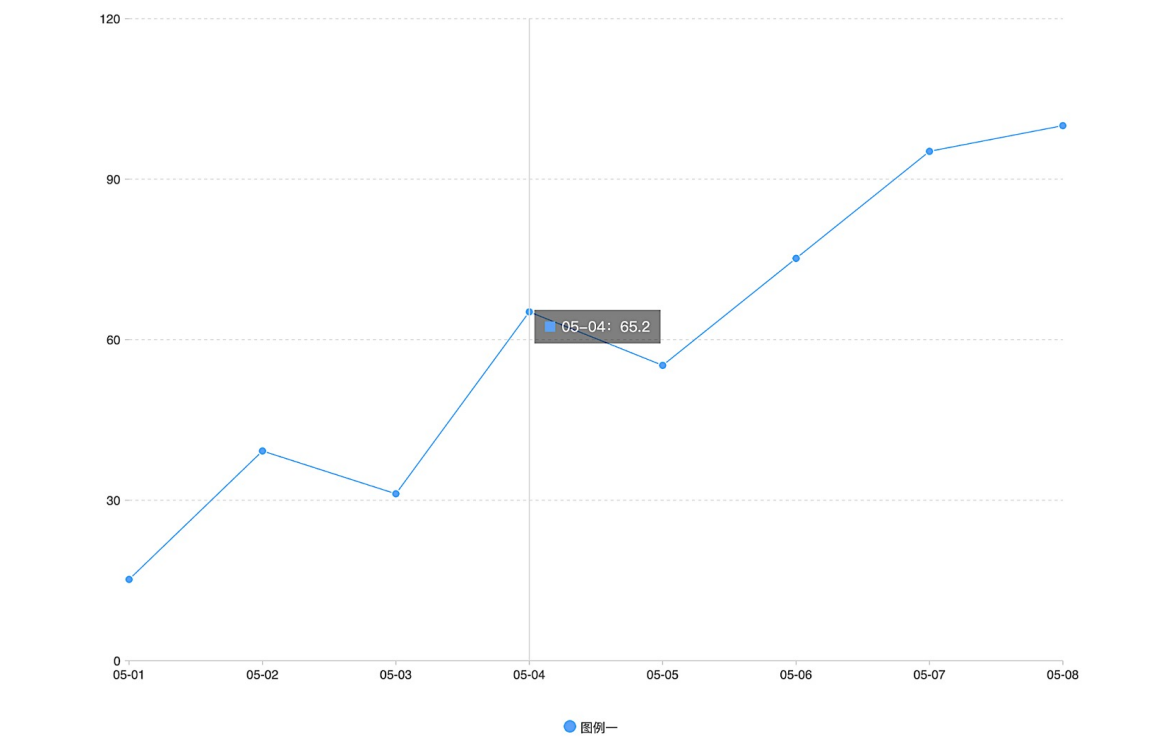


添加完坐标轴之后，我们再添加图例（`Legend`）和提示（`Tooltip`）。最简单的方式，是我们直接创建两个对象，然后将它们添加到 `charts` 对象的子元素中。

[复制代码](#)

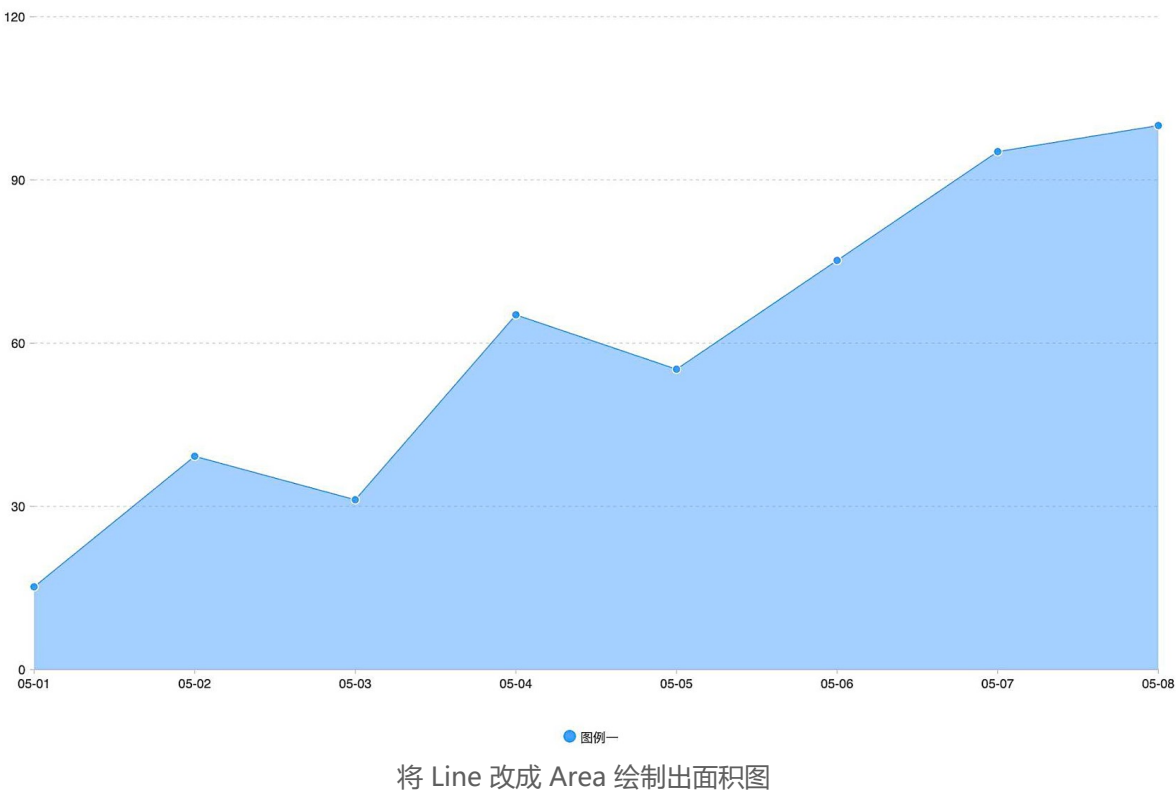
```
1 const legend = new Legend();
2 const tooltip = new Tooltip();
3 chart.append([line, axisBottom, axisLeft, legend, tooltip]);
```

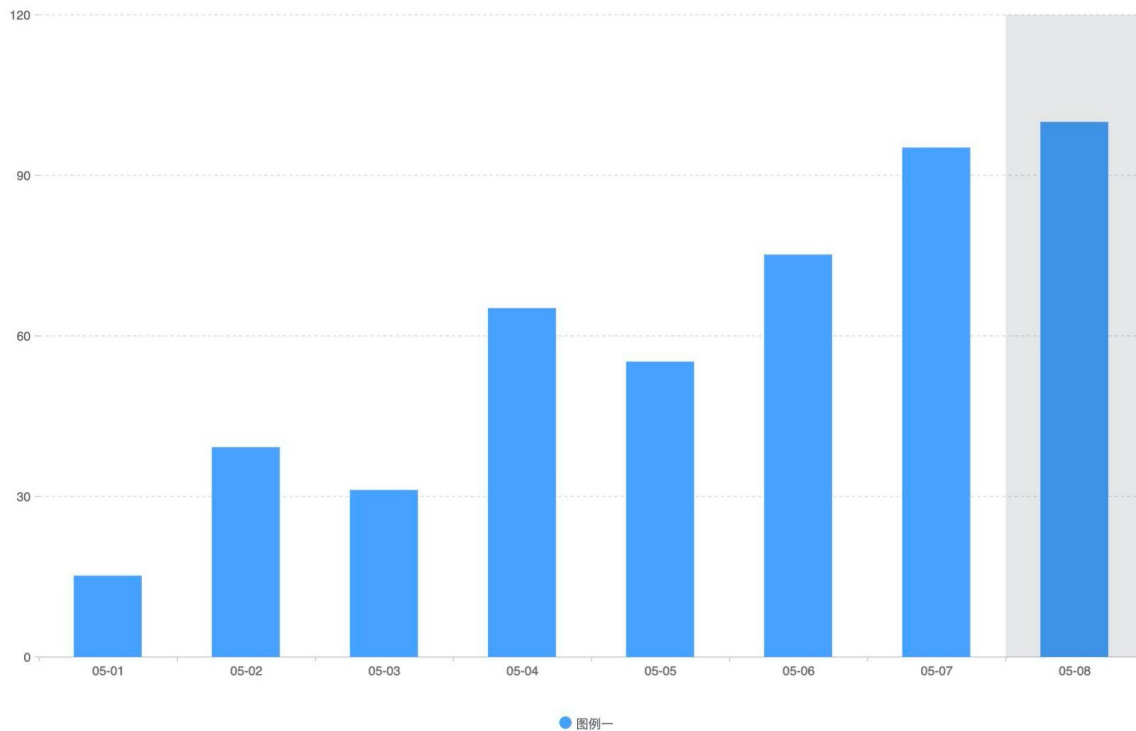
添加了图例和提示信息后的图表如下图所示：



接下来，我们再来说怎么用 QCharts 绘制面积图和柱状图。

学会了折线图的绘制方法，我们就可以用相同的思路非常简单地绘制其他图形了，比如说，我们可以简单将 Line 对象换成 Area 或 Bar 对象，这样就可以用同样的数据绘制出面积图或柱状图。这是为什么呢？





将 Line 改成 Area 绘制出柱状图

因为像折线图、面积图、柱状图这些表现形式不同的图表，它们能够接受同样格式的数据，只是想要侧重表达的信息不同而已。一般来说，折线图强调数据变化趋势，柱状图强调数据的量和差值，而面积图同时强调数据量和变化趋势。在实际项目中，我们要根据不同的需求选择不同的基本图形。

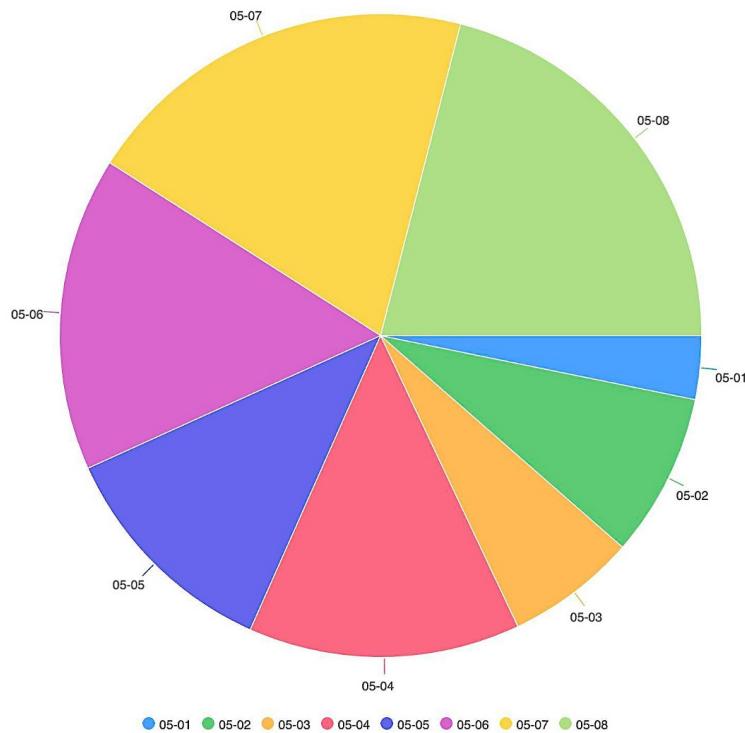
如果要强调整体和局部比例，我们还可以选择绘制饼图。与折线图、面积图、柱状图相比，饼图不用配置图例，因为图例会自动生成，而且饼图也不需要坐标轴，所以使用起来更加简单。

[复制代码](#)

```
1  const data = [  
2    { date: '05-01', sales: 15.2 },  
3    { date: '05-02', sales: 39.2 },  
4    { date: '05-03', sales: 31.2 },  
5    { date: '05-04', sales: 65.2 },  
6    { date: '05-05', sales: 55.2 },  
7    { date: '05-06', sales: 75.2 },  
8    { date: '05-07', sales: 95.2 },  
9    { date: '05-08', sales: 100 }  
10 ];  
11  
12 chart.source(data, {  
13   row: 'date',  
14   value: 'sales',  
15   text: 'date'
```

```
16 });  
17  
18 const pie = new Pie();  
19 const legend = new Legend();  
20 const tooltip = new Tooltip();  
21 chart.append([pie, legend, tooltip]);
```

上面的代码用同样的数据绘制了一个饼图。

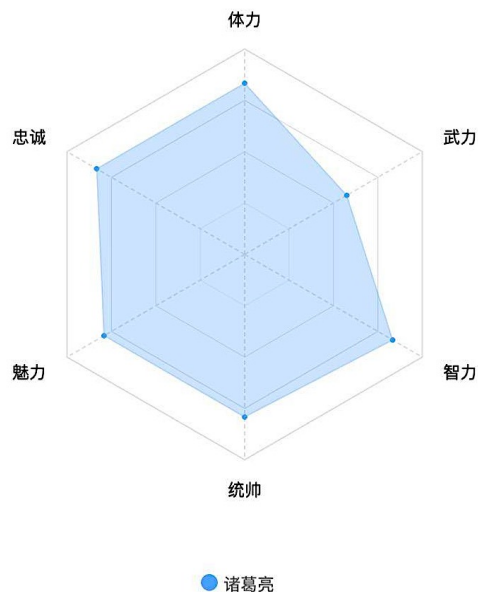


## QCharts 绘制雷达图、仪表盘和玉块图、南丁格尔玫瑰图

讲完了这些常见的基础图表，我们再来看几个比较有趣的图表。

首先是雷达图，它一般用来绘制多组固定数量的数据，可以比较直观地显示出这组数据的特点。我们经常会在游戏中看见它的应用，比如，下面这张图表就显示了三国武将诸葛亮的各方面数据。



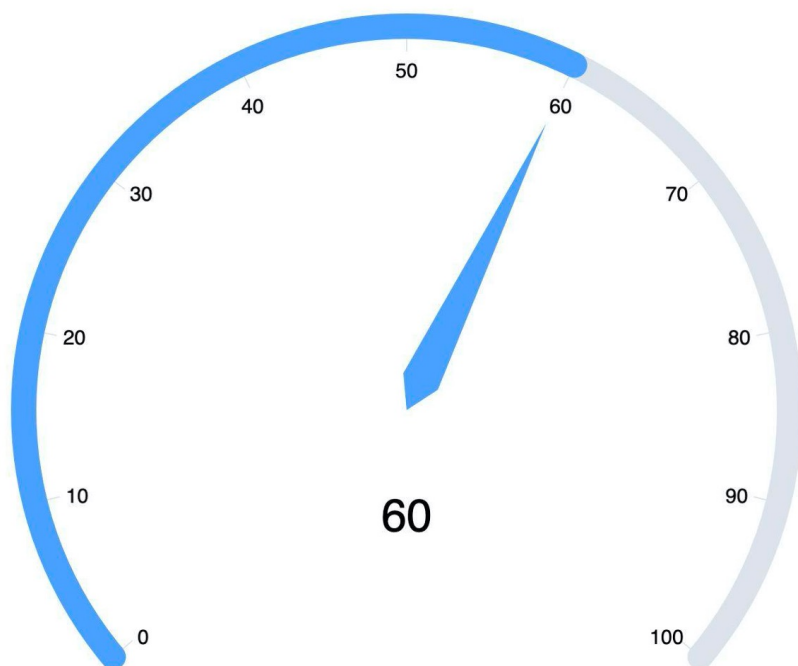


雷达图的实现代码如下：


复制代码

```
1  const data = [  
2    { date: '体力', category: '诸葛亮', sales: 100 },  
3    { date: '武力', category: '诸葛亮', sales: 69 },  
4    { date: '智力', category: '诸葛亮', sales: 100 },  
5    { date: '统帅', category: '诸葛亮', sales: 95 },  
6    { date: '魅力', category: '诸葛亮', sales: 95 },  
7    { date: '忠诚', category: '诸葛亮', sales: 100 },  
8  ];  
9  
10 chart.source(data, {  
11   row: 'category',  
12   value: 'sales',  
13   text: 'date'  
14 });  
15  
16 const radar = new Radar();  
17 radar.style('section', (d) => ({ opacity: 0.3 }));  
18  
19 const legend = new Legend({ align: ['center', 'bottom'] });  
20 const tooltip = new Tooltip();  
21 chart.append([radar, legend, tooltip]);
```

除此之外，还有一些其他类型的图表，可以用来展示特殊的信息。比较典型的有仪表盘，它可以显示某个变量的进度。



仪表盘实现代码比较简单，如下所示：

 复制代码

```
1  const gauge = new Gauge({
2    min: 0,
3    max: 100,
4    percent: 60,
5    lineWidth: 20,
6    tickStep: 10
7  });
8
9  gauge.style('title', { fontSize: 36 });
10
11 chart.append(gauge);
```

如果我们要显示多个变量的进度，还可以使用玉块图。



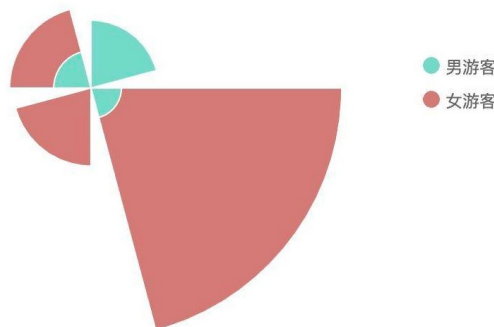
对应的代码也非常简单，如下所示：

复制代码

```
1  const data = [  
2    {  
3      type: '政法干部',  
4      count: 6654  
5    }, {  
6      type: '平安志愿者',  
7      count: 5341  
8    }, {  
9      type: '人民调解员',  
10     count: 3546  
11   }, {  
12     type: '心理咨询师',  
13     count: 4321  
14   }, {  
15     type: '法律工作者',  
16     count: 3923  
17   }, {  
18     type: '网格员',  
19     count: 5345  
20   }  
21 ].sort((a, b) => a.count - b.count);  
22  
23 chart.source(data, {  
24   row: 'type',  
25   text: 'type',  
26   value: 'count'  
27 });
```

```
28
29 const radialBar = new RadialBar({
30   min: 0,
31   max: 10000,
32   radius: 0.6,
33   innerRadius: 0.1,
34   lineWidth: 10
35 });
36
37 radialBar.style('arc', { lineCap: 'round' });
38
39 const legend = new Legend({
40   orient: 'vertical',
41   align: ['right', 'center'],
42 });
43
44 chart.append([radialBar, legend, new Tooltip()]);
```

最后是南丁格尔玫瑰图，它可以显示多维度的信息，比如下图就显示了男女游客在公园四个区域内的分布情况。南丁格尔玫瑰图的绘制思路和代码，我们在🔗第 35 节课已经说过，这里就不再多说了。如果你还不熟悉，可以再去回顾一下。



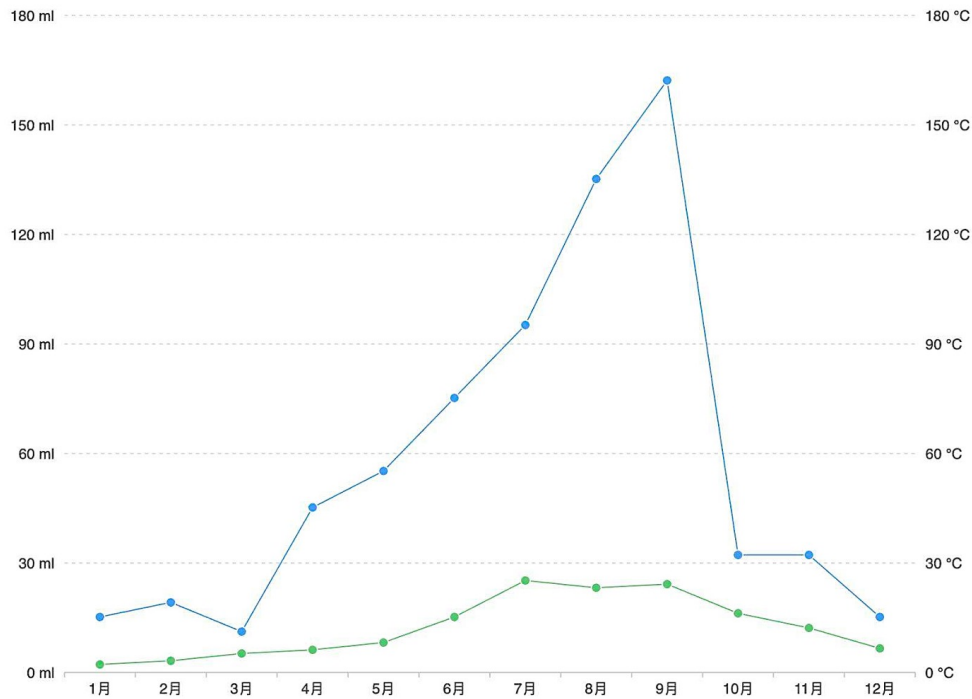
## QCharts 绘制图表组合

我们刚才讲的都是图表上绘制单一变量，那要想在图表上聚合多元变量，比如在一张天气图表上同时展示降水量、气温，我们可以同时绘制多个图形来表示。

我们可以分两种情况来讨论，最简单的情况是用相同的图形来绘制不同的变量。这个时候，我们可以直接通过不同 category 来绘制多个图形。比如，下面的数据就是用两组 category 分别绘制了两条折线。

[复制代码](#)

```
1  const data = [  
2    { date: "1月", category: "降水量", val: 15.2 },  
3    { date: "2月", category: "降水量", val: 19.2 },  
4    { date: "3月", category: "降水量", val: 11.2 },  
5    { date: "4月", category: "降水量", val: 45.2 },  
6    { date: "5月", category: "降水量", val: 55.2 },  
7    { date: "6月", category: "降水量", val: 75.2 },  
8    { date: "7月", category: "降水量", val: 95.2 },  
9    { date: "8月", category: "降水量", val: 135.2 },  
10   { date: "9月", category: "降水量", val: 162.2 },  
11   { date: "10月", category: "降水量", val: 32.2 },  
12   { date: "11月", category: "降水量", val: 32.2 },  
13   { date: "12月", category: "降水量", val: 15.2 },  
14  
15   { date: "1月", category: "气温", val: 2.2 },  
16   { date: "2月", category: "气温", val: 3.2 },  
17   { date: "3月", category: "气温", val: 5.2 },  
18   { date: "4月", category: "气温", val: 6.2 },  
19   { date: "5月", category: "气温", val: 8.2 },  
20   { date: "6月", category: "气温", val: 15.2 },  
21   { date: "7月", category: "气温", val: 25.2 },  
22   { date: "8月", category: "气温", val: 23.2 },  
23   { date: "9月", category: "气温", val: 24.2 },  
24   { date: "10月", category: "气温", val: 16.2 },  
25   { date: "11月", category: "气温", val: 12.2 },  
26   { date: "12月", category: "气温", val: 6.6 },  
27 ];  
28  
29 chart.source(data, {  
30   row: "category",  
31   value: "val",  
32   text: "date",  
33 });  
34  
35 const line = new Line({ axisGap: true });  
36 ...
```

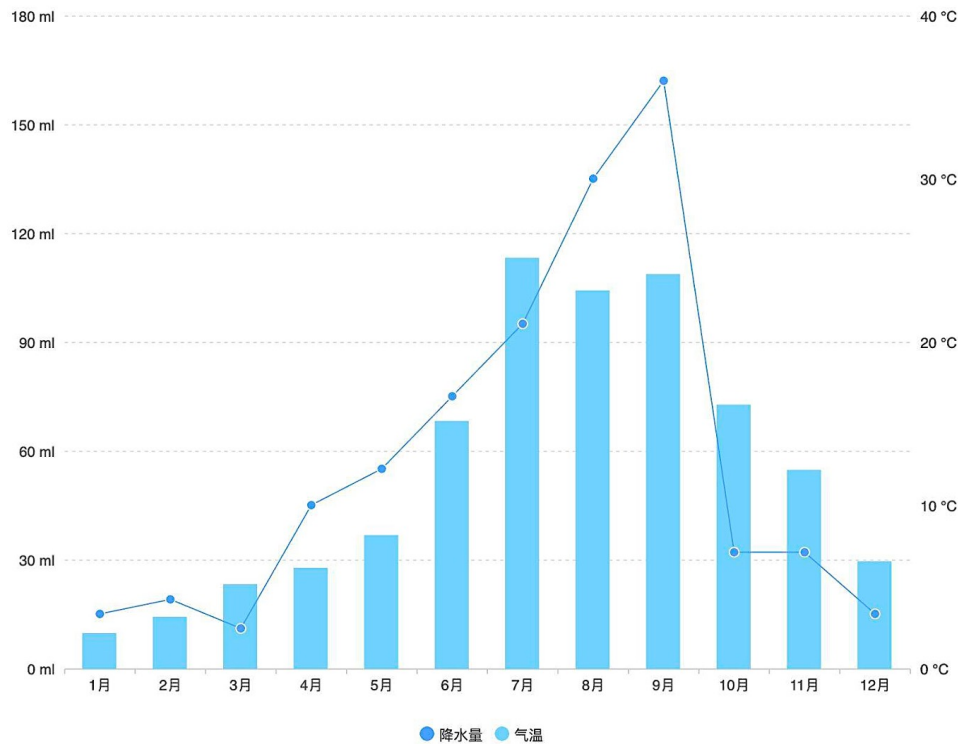


如果我们想用不同类型的图形来展示多个变量，在 QCharts 中，我们只需要创建多个不同的图形对象，然后把它们都添加到 chart 对象的子元素中去就可以了。

[复制代码](#)

```
1 const ds = chart.dataset;
2
3 const d1 = ds.selectRows("降水量");
4 const line = new Line({ axisGap: true })
5   .source(d1)
6   .style("point", { strokeColor: "#fff" });
7
8 const d2 = ds.selectRows("气温");
9 const bar = new Bar().source(d2);
10 bar.style("pillar", { fillColor: "#6CD3FF" });
```

如上面代码所示，我们先可以通过 `chart.dataset` 拿到通过 `source` 绑定给 `chart` 对象的数据集，然后，通过 `selectRows` 分别将降水量和气温数据过滤出来。接着，我们分别创建 `Line` 和 `Bar` 两个图形对象，再将降水量和气温数据分别绑定给它们。最后，我们将这两个对象同时添加到 `chart` 子元素列表里，就可以将两个不同类型的图形显示出来了，具体的效果如下：



## 要点总结

这节课，我们主要学习了 QCharts 图表库的使用。

QCharts 是基于 SpriteJS 的简单可视化图表库，我们通过它可以绘制各种类型的图表。一般来说，我们是先创建图表对象，然后绑定数据，接着添加图形对象以及其他的插件，包括图例和提示。通过将图形和插件以子元素的形式添加到图表对象上，就能把图表内容最终显示出来了。

我们可以很方便地根据数据特点和业务需要，用数据绘制折线图、面积图、柱状图、饼图、雷达图等图表，还可以绘制特殊的仪表盘和玉块图。另外，如果要显示多维数据，我们也可以用稍微复杂一些的南丁格尔玫瑰图。

最后，我们还可以把多维度变量聚合在一个图表中来显示不同的图形组合。具体操作是，我们先筛选数据，然后创建不同类型的图形对象，最后将它们都添加到图表对象的子元素中。

总的来说，我们今天讲的其实都是 QCharts 图表库，最基础、最常用的方法。QCharts 还提供了众多其他类型的图表，以及灵活操作图表样式的 API。如果你有兴趣继续钻研，可以通过我课后给出的参考链接进一步学习。

## 小试牛刀

你学会了使用不同图表来表达不同数据了吗？你可以试着使用 GitHub 仓库里的北京市天气数据和空气质量数据，实现一个温度、湿度、风速、空气质量的聚合图表吗？如果用来展示温度、湿度、风速和空气质量的图形都不相同就更好了。

这些常用的制图方法你都学会了吗？下节课，我们会接着来学，怎么使用数据驱动框架来表达不同数据，挑战才刚刚开始呢，我们下节课再见！

---

## 源码

课程代码详见 [🔗 GitHub 仓库](#)

## 推荐阅读

[🔗 QCharts 官网](#)

提建议



## 更多课程推荐

# 数据结构与算法之美

为工程师量身打造的数据结构与算法私教课

王争

前 Google 工程师



立省 ¥40

破 90000 订阅特惠，到手价 ¥89

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 36 | 设计（二）：如何理解可视化设计原则？

下一篇 38 | 实战（二）：如何使用数据驱动框架绘制常用数据图表？

## 精选留言

写留言

由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。